UNCLA

SECURITY C

AD-A256 038

(2)

OCUMENTATION PAGE

DTIC ELECTE SEP 2 9 1992 S A D

1a. REPORT Unclassified	1b. RESTRICTIVE MARKINGS
a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
a. NAME OF PERFORMING ORGANIZATION Amtec Engineering, Inc. / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION US Army Missile Command
c. ADDRESS (City, State, and ZIP Code) 3075 112th Ave NE, Suite 106 Bellevue, WA 98004	7b. ADDRESS (City, State, and ZIP Code) Bldg 7770 Redstone Arsenal, AL 35898-5244
a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAH01-90-C-0373
c. ADDRESS (City, State, and ZIP Code) 3701 North Fairfax Drive Arlington, VA 22203-1714	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

1 TITLE (Include Security Classification)
3D Navier-Stokes Flow Analysis for Shared and Distributed Memory MIMD Computers

2 PERSONAL AUTHOR(S)
Imlay, Scott T. and Soetrisno, Moeljo

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 5/15/90 TO 7/30/92	1992, September 15	118

6. SUPPLEMENTARY NOTATION

7 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	3D Navier-Stokes, Viscous-Flow, Compressible-Flow, Hypersonic Parallel, Shared Memory, Distributed Memory

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

Multiple instruction multiple data (MIMD) parallel computers were investigated as a means of obtainng the immense computer power required to analyze the flow about a complete aircraft configuration. A computational fluid dynamics computer program for solving the Navier-Stokes equations, with coupled nonequilibrium chemistry, was parallelized on three MIMD computer: the Intel Touchstone distributed memory computer, the Cray Y/MP super-computer, and the Silicon Graphics IRIS 4-D/380 shared memory computer. The program used an LU-SGS implicit algorithm. The parallelization was performed using permanent domain decomposition. The parallel efficiency and the main limitations to efficient parallelization were evaluated for a series of real life engineering problems. On the Intel Touchstone, the parallel efficiency decreased slowly as the number of processors was increased. The main deterrent to parallel performance was the latency time for inter-processor communication. On the Cray Y/MP, the parallel efficiency dropped rapidly as the number of processors was increased. The main deterrent to parallel performance was the reduction in vector length during domain decomposition.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. ABSTRACT cont.

On the Silicon Graphics IRIS, the parallel efficiency was high for fewer than four processor but decreased rapidly for greater than four processors. The main deterrents to parallel performance on the IRIS were memory contention, low bus bandwidth, and competition for CPU time with other processes.

P90120.04

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability [illegible]	
Dist	[illegible]
A-1	

Final Report

3D Navier-Stokes Analysis for Shared and Distributed Memory MIMD Computers

September 15, 1992

Sponsored by
Defense Advanced Research Projects Agency (DoD)
Defense Small Business Innovation Research Program

ARPA Order No. 6685
Issued by U.S. Army Missile Command Under
Contract # DAAH01-90-C-0373

Amtec Engineering, Inc.
3075 112th Ave N.E., Suite 106
Bellevue, WA 98004

Principal Investigator: Scott T. Imlay
Phone Number: (206) 827-3304

Effective Date of Contract: May 15, 1990
Contract Expiration Date: July 30, 1992
Reporting Period: May. 15, 1990 — July 30, 1991

DISCLAIMER

Contents

List of Figures **iii**

Nomenclature **v**

1 Introduction **1**

2 Phase II Tasks **4**

3 Model Code **7**

3.1 Distributed Memory MIMD Computer 8
3.2 Shared Memory Cray Y-MP 13
3.3 Shared Memory Silicon Graphics Iris 14

4 Navier-Stokes Equations with Chemistry **17**

4.1 Transport Equations 17
4.2 Thermodynamics 20
4.2.1 Perfect Gas 21
4.2.2 Equilibrium Air Curve Fits 21
4.2.3 Thermally Perfect Species - Thermal Equilibrium 22
4.2.4 Thermally Perfect Species - Thermal Nonequilibrium 24
4.3 Transport Properties 27
4.3.1 Viscosity 27
4.3.2 Conductivity 28
4.3.3 Diffusion coefficient 29
4.4 Boundary Conditions 29
4.4.1 Free-Slip Walls 29
4.4.2 No-Slip Walls 30
4.4.3 Rarefied Gas Slip Walls 31
4.4.4 Supersonic Inflow 33
4.4.5 Supersonic Outflow 33
4.4.6 Mixed Subsonic/Supersonic Outflow 34
4.4.7 Subsonic Inflow 34

5 Navier-Stokes Solution Procedure **37**

5.1 Internal Grid Cells 37
5.1.1 Steger and Warming Flux Function 38
5.1.2 Harten and Yee Flux Function 41
5.1.3 Diffusive Fluxes 44
5.1.4 Implicit Treatment — Point Implicit 45
5.1.5 Implicit Treatment — LU-SGS 48
5.2 Boundary Conditions 50

5.2.1 Free-Slip Walls 50
5.2.2 No-Slip Walls 50
5.2.3 Rarefied Gas Slip Walls 51
5.2.4 Supersonic Inflow 53
5.2.5 Supersonic Outflow 53

6 Parallel Navier-Stokes Implementation 54
6.1 Distributed Memory MIMD Computer 54
6.2 Shared Memory Cray Y-MP 56
6.3 Shared Memory Silicon Graphics Iris 57

7 Test Cases 61
7.1 RAE-2822 Airfoil 61
7.2 Unsteady flow past a cylinder 61
7.3 Mars penetrator with CO_2 chemistry model 63
7.4 Hydrogen-air shock induced combustion 63
7.5 Premixed hydrogen-air past a 10° ramp 66
7.6 Hydrogen-air oblique detonation ram accelerator 67
7.7 Methane-air shock induced combustion 69
7.8 Methane-air ram accelerator 73
7.9 Ideal Gas 3-D Ram Accelerator 82

8 Evaluation and Comparisons of Parallel Performance 86
8.1 Distributed Memory MIMD 88
8.2 Shared Memory Cray Y-MP 92
8.3 Shared Memory Silicon Graphics Iris 94

9 Conclusions 99

References 100

List of Figures

1 Domain Decomposition on Distributed Memory MIMD Computer . . 8
2 Solutions to model equation with domain decompositions 9
3 Convergence rates for model equation with domain decompositions . 9
4 Speed-up with Load Balancing Problem 11
5 Efficiency with Load Balancing Problem 11
6 Speed-up for Perfect Load Leveling 12
7 Efficiency for Perfect Load Leveling 13
8 Parallel performance of model code on the Cray Y-MP - calculated by comparing to one zone on one cpu 14
9 Parallel performance of model code on the Cray Y-MP - calculated by comparing to p zones on one cpu 15
10 Parallel Performance of the Model Equation on the Silicon graphics . 16
11 Finite Volume Mesh 20
12 Types of heat transfer to the wall 30
13 Finite Volume Cell 37
14 Mathematical characteristics of one-dimensional subsonic flow 39
15 Diagonal Wave Plane (where $i + j + k$ is constant) 59
16 HANA code subroutine hierarchical levels 59
17 Domain Decomposition 60
18 Interprocessor Communication 60
19 RAE-2822: Two-zone Mesh 62
20 RAE-2822: Pressure Contours 62
21 RAE-2822: Computed Pressure Coefficient vs. Experimental Data . 62
22 Streamlines - M=0.45 Past a Cylinder at Re=110,000 64
23 Geometry of the Mars Penetrator 65
24 Temperature contours for the Mars penetrator 65
25 Hydrogen-Air Shock Induced Combustion 66
26 Hydrogen-Air Past a 10°*Ramp* 67
27 $M = 6.7$: Temperature Contours 68
28 $M = 7.5$: Temperature Contours 68
29 $M = 8.0$: Temperature Contours 68
30 $M = 6.7$: Water Vapor Mass Fraction Contours 68
31 $M = 7.5$: Water Vapor Mass Fraction Contours 68
32 $M = 8.0$: Water Vapor Mass Fraction Contours 68
33 $M = 6.7$: Pressure Profile along Projectile Wall 69
34 $M = 6.7$: Pressure Profile along Tube Wall 70
35 Blunt Cylinder with different Chemistry Models 72
36 Rods different Diameters 73
37 Ram Accelerator Mesh and Geometry 74
38 $M_{CJ} = 0.71$ Ram Accelerator 75

39 $M_{CJ} = 0.76$ Ram Accelerator 76
40 $M_{CJ} = 0.83$ Ram Accelerator 76
41 $M_{CJ} = 0.87$ Ram Accelerator 77
42 $M_{CJ} = 0.92$ Ram Accelerator 77
43 $M_{CJ} = 0.95$ Ram Accelerator 78
44 $M_{CJ} = 0.99$ Ram Accelerator 78
45 $M_{CJ} = 1.04$ Ram Accelerator 79
46 $M_{CJ} = 1.11$ Ram Accelerator 79
47 $M_{CJ} = 1.14$ Ram Accelerator 80
48 Thrust Coefficient of the Ram Accelerator 81
49 Mesh and stream-tubes for Ideal Gas 3-D Ram Accelerator 82
50 Tube Wall Pressure Profiles at $0^o, 15^o, 45^o$ angles to the fin 83
51 Speed on Intel Touchstone iPSC/860 89
52 Efficiency on Intel Touchstone iPSC/860 89
53 iPSC/860: Distribution of CPU time when the amount of work/node is constant 90
54 iPSC/860: Distribution of CPU time for constant problem size . . . 91
55 Speedup on Cray Y-MP 92
56 Efficiency on Cray Y-MP 93
57 Cray Y-MP: Distribution of CPU time when the amount of work/node is constant 93
58 Cray Y-MP: Effects of Vector Length Reductions 94
59 Cray Y-MP: Distribution of CPU time for constant problem size . . 95
60 Speedup on SGI 4-D/380 96
61 Efficiency on SGI 4-D/380 96
62 SGI 4-D/380: Distribution of CPU time when the amount of work/node is constant 97
63 SGI 4-D/380: Distribution of CPU time for constant problem size . . 98

Nomenclature

Upper Case

A	flux Jacobian, $A = \frac{\partial F}{\partial U}$
A_b^s	a coefficient for Blottner's species viscosity equation
$A_{k_1} - A_{k_6}$	coefficients for temperature dependent thermal conductivity
$A_{\mu_1} - A_{\mu_6}$	coefficients for temperature dependent viscosity
$A^{\pm}$	Jacobians evaluated with negative/positive eigenvalues set to zero
$\hat{A}^{\pm}$	Jacobians of the F^+ and F^- flux-split flux functions
$\tilde{A}^{\pm}$	Yoon's approximate Jacobians
B_b^s	a coefficient for Blottner's species viscosity equation
C	temporary constant
C_b^s	a coefficient for Blottner's species viscosity equation
C_p	specific heat at constant pressure, J/mole K
C_v	specific heat at constant volume, J/mole K
$\bar{C}_v$	averaged specific heat at constant volume, J/mole K
C_{v_e}	specific heat at constant volume for electronic modes, J/mole K
C_{v_r}	specific heat at constant volume for rotational modes, J/mole K
C_{v_t}	specific heat at constant volume for translational modes, J/mole K
$C_{v_{tr}}$	specific heat at constant volume for translational and rotational modes, J/mole K
C_{v_v}	specific heat at constant volume for vibrational modes, J/mole K
$C_{v_{ve}}$	specific heat at constant volume for vibrational and electronic modes, J/mole K
$C_{\epsilon_1}, C_{\epsilon_2}$	constants in $k - \epsilon$ source terms
C_μ	constant in calculation of eddy viscosity from k and ϵ
$\mathcal{D}$	diffusion coefficient
$\mathcal{D}_s$	species diffusion coefficient
D_a	cell Damkohler number
E	total energy
E_v	vibrational/electronic energy per unit volume
$\mathcal{F}$	stretch factor for adapted mesh
$F(U)$	inviscid flux function
F_1, F_2, F_3	flux-vectors in x, y, and z directions
$\tilde{F}_{i+1/2,j,k}$	inviscid flux through surface dividing i,j,k and $i+1,j,k$ cells
$F^{\pm}(U)$	Steger & Warming split flux functions for positive/negative eigenvalues
F_1^d, F_2^d, F_3^d	diffusive fluxes in the x, y, and z directions
F_1^λ, F_2^λ, F_3^λ	Steger & Warming split flux functions for eigenvalues
G_k	term in $k - \epsilon$ source terms
H	total enthalpy, $H = (E + p)/\rho$
$K_{c,r}$	equilibrium constant for reaction r

M	molecular weight of mixture
M_s	molecular weight of species s
N	source terms
N_B	number of cells in shock capture band for bow shock adaption
N_C	total number of adaptable cells along the mesh line for bow shock adaption
NR	number of rcactions
NS	number of species
$\vec{P}$	flux vector
P_r	Prandtl number
P_{r_t}	turbulent Prandtl number
P_y	normal momentum flux
$\vec{P} \cdot \vec{S}$	flux through cell surface
$\vec{P} \cdot \vec{S}^{diff}$	viscous, heat conducting, and diffusing portion of flux through cell surface
$\vec{P} \cdot \vec{S}^{inv}$	inviscid portion of flux through cell surface
R	mixture gas constant, $R = \tilde{R}/M$
$\tilde{R}$	universal gas constant, = 8.31434 J/mole K
$\mathcal{R}$	matrix of eigenvectors fo the flux Jacobian matrix, A
$\mathcal{R}^{-1}$	inverse of $\mathcal{R}$
S	surface of a volume
$\mathcal{S}$	sign of a characteristic variable (Harten Yee flux)
$\vec{S}$	vector normal to a face of a cell with magnitude equal to the area of the face
S_{c_s}	species Schmidt number
S_{sp}	distance from the edge of the protected mesh to the shock, for bow shock adaption
T	translational temperature
T_{db}	temperature of distant body (for radiative heat transfer)
T_{iw}	temperature inside body
T_{ref}	reference temperature for the c_{p_s}, h_s, and s_s curve fit
T_s	temperature at edge of Knudsen layer
T_{ve}	vibrational/electronic temperature
T_w	wall temperature
$TV(U)$	total variation of U
U	conservative variables
Vol	volume
$\vec{V}$	velocity vector
X_s	mole fraction of species s
Y_s	chemical formula for species s
$[Y_s]$	molar concentration of species s in moles / cm^3

Lower Case

$a_{1s} - a_{7s}$	polynomial curve fit coefficients for c_{p_s}, h_s, and s_s in terms of T
$b_1 - b_6$	constants in the Landau-Teller expression for the vibrational relaxation time constant
b_{ls}	coefficient of the curve fits to $c_{v_{e,s}}$ and e_{ve_s}
c	speed of sound
c_{ch}	heat capacity of wall
c_p	specific heat at constant pressure, J/Kg K
c_s	mass fraction of species s, $c_s = \rho_s/\rho$
c_v	specific heat at constant volume, J/Kg K
$\bar{c}_v$	averaged specific heat at constant volume, J/Kg K
c_{v_e}	specific heat at constant volume for electronic modes, J/Kg K
c_{v_r}	specific heat at constant volume for rotational modes, J/Kg K
c_{v_t}	specific heat at constant volume for translational modes, J/Kg K
$c_{v_{tr}}$	specific heat at constant volume for translational and rotational modes, J/Kg K
c_{v_v}	specific heat at constant volume for vibrational modes, J/Kg K
$c_{v_{ve}}$	specific heat at constant volume for vibrational and electronic modes, J/Kg K
$d_1 - d_6$	coefficients of eigenvalue smoothing (Harten-Yee fluxes)
e	mass specific internal energy, J/Kg
$\breve{e}$	volume specific internal energy, J/m^3
e_0	variable in the approximate equation of state, $p = \rho(\bar{\gamma} - 1)/(e - e_0)$
e_e	mass specific electronic energy, J/Kg
e_r	mass specific rotational energy, J/Kg
e_t	mass specific translational internal energy, J/Kg
e_v	mass specific vibrational energy, J/Kg
e_{ve}	mass specific vibrational/electronic energy, J/Kg
e^*_{ve}	equilibrium value of mass specific vibrational/electronic energy, J/Kg
g_i	intermediate term in evaluation of Harten-Yee fluxes
h_s	enthalpy of species s
$\vec{i}$	unit normal vector in x-direction
$\vec{j}$	unit normal vector in y-direction
$\vec{k}$	unit normal vector in z-direction
k	turbulent kinetic energy
$\tilde{k}$	boltzmann constant $= 1.3805 \times 10^{-23}$
$k_{b,r}$	backward chemical reaction rate coefficient for reaction r
$k_{f,r}$	forward chemical reaction rate coefficient for reaction r
k_l	coefficient of laminar heat conduction
k_s	coefficient of heat conduction for solid wall

k_t	coefficient of turbulent heat conduction
k_{ve}	coefficient of laminar heat conduction for vibrational/electronic energy modes
m_i	mass of molecule for specie i
$\bar{m}_s$	average mass per molecule for mixture at edge of Knudsen layer
$\vec{n}$	unit normal vector to a surface
n_i^s	number density of specie i at edge of Knudsen layer
n_x, n_y, n_z	x, y, and z components of the unit normal vector
p	pressure
p^s	pressure at edge of Knudsen layer
p_{ρ_i}	partial derivative of pressure with respect to the density of specie i, with the other conservative variables held constant
p_{m_i}	partial derivative of pressure with respect to the i'th component of momentum, with the other conservative variables held constant
p_E	partial derivative of pressure with respect to the total energy, with the other conservative variables held constant
p_{E_v}	partial derivative of pressure with respect to E_v, with the other conservative variables held constant
q_i	conductive heat flux in the i'th direction
q_g	conductive heat flux from gas to wall
q_r	radiative heat flux from far off body to wall
q_s	conductive heat flux from within body to wall
q_{ve_i}	conductive heat flux of vibrational/electronic energy in the i'th direction
s_s	entropy of species s
t	time
t_{wall}	thickness of wall
u_1, u_2, u_3	components of velocity in x, y, and z directions
u'	a component of velocity tangent to the surface and orthogonal to w'
$v_{s_i}^d$	diffusive velocity in i'th direction for species s
v'	component of velocity normal to the surface
$w_1, w_2, \ldots$	mass source term for species 1, 2, ...
w'	a component of velocity tangent to the surface and orthogonal to u'
w_{ve}	source term for vibrational/electronic energy
x	first component of position (also x_1)
x_i	three component of position (also x, y, and z)
x'	a tangential component of position in surface oriented coordinate system
y	second component of position (also x_2)
y'	normal component of position in surface oriented coordinate system
z	third component of position (also x_3)
z'	a tangential component of position in surface oriented coordinate

system

Greek Letters

Γ	intermediate term in evaluation of Harten-Yee fluxes
Δ_s	initial cell height of the adapted mesh for bow shock fitting
Δt	time step
Λ	diagonal matrix with eigenvalues of A on diagonal
$\Lambda^{\pm}$	Λ with negative/positive eigenvalues set to zero
Φ	dissipation operator for Harten-Yee fluxes
Ψ	function used during evaluation of Harten-Yee fluxes
α_w	absorptivity of wall (for radiative heat transfer)
$\alpha_{i+1/2}$	characteristic variables, used for Harten-Yee fluxes
γ	thermodynamic variable in the expression, $c^2 = \frac{\gamma p}{\rho}$
$\tilde{\gamma}$	thermodynamic variable in the expression, $p = \rho(\tilde{\gamma} - 1)(\epsilon - \epsilon_0)$
$\delta(\,)$	change of variable () over time step
δ_{ij}	Kronecker delta
ϵ	turbulent kinetic energy dissipation rate
ϵ_w	emissivity of wall (for radiative heat transfer)
ε_m	term in eigenvalue smoothing for Harten-Yee fluxes
ζ	coordinate running in direction of increasing k index
η	coordinate running in direction of increasing j index
θ	wall accomodation coefficient
κ	partial derivative of the thermal eq. of state $p = p(\rho_1, \ldots, \rho_{NS}, \tilde{e})$ for thermal equilibrium, $\kappa = \frac{\partial p}{\partial \tilde{e}}\Big|_{\rho_s}$
$\tilde{\kappa}$	parameter in extrapolation of U to surface for MUSCL differencing ($\tilde{\kappa} = -1$: fully upwind, $\tilde{\kappa} = 1$: central)
κ_{tr}	partial derivative of the thermal eq. of state $p = p(\rho_1, \ldots, \rho_{NS}, \tilde{e}_{tr}, \tilde{e}_{ve})$ for thermal nonequilibrium, $\kappa_{tr} = \frac{\partial p}{\partial \tilde{e}_{tr}}\Big|_{\rho_s, \tilde{e}_{ve}}$
κ_{ve}	partial derivative of the thermal eq. of state $p = p(\rho_1, \ldots, \rho_{NS}, \tilde{e}_{tr}, \tilde{e}_{ve})$ for thermal nonequilibrium, $\kappa_{ve} = \frac{\partial p}{\partial \tilde{e}_{ve}}\Big|_{\rho_s, \tilde{e}_{tr}}$
$\Lambda_m(A)$	m'th eigenvalue of matrix A
μ	molecular viscosity of mixture
μ_t	turbulent viscosity of mixture
μ_k	coefficient of viscosity for k transport equation
μ_ϵ	coefficient of viscosity for ϵ transport equation
$\nu'_{s,r}$	forward stoichiometric coefficients for species s and reaction r
$\nu''_{s,r}$	backward stoichiometric coefficients for species s and reaction r
ξ	coordinate running in direction of increasing i index
π	universal constant (= 3.1415927)
ρ	mass density of mixture

ρ_s	mass density of species s
$\bar{\rho}_{i+1/2}$	arithmetical averaged density
σ	Stefan-Boltzmann constant ($= 5.67032 \times 10^{-8}$)
$\sigma_{i+1/2}$	intermediate term for Harten-Yee fluxes
σ_k, σ_ϵ	constants for $k - \epsilon$ turbulence model
τ_{ij}	stress tensor
τ_{LT}	Landau-Teller time constant for vibrational relaxation
ϕ	parameter in extrapolation of U to surface for MUSCL differencing ($\phi = 0$: first order, $\phi = 1$: second order)
χ_s	partial derivative of thermal equation of state, $\chi_s = \frac{\partial p}{\partial \rho_s}\Big\|_{\rho_{t \neq s}, \tilde{e}_{tr}, \tilde{e}_{ve}}$

Mathematical Symbols

$\sum_{s=1}^{N}$	summation over index s from 1 to N
$\prod_{s=1}^{N}$	product over index s from 1 to N

Subscripts

i, j, k	average value of variable in cell i,j,k

Superscripts

n	value of variable at current time step (time step where solution is known)
$n+1$	value of variable at next time step (time step where solution is being sought)
s	value of variable at edge of Knudsen layer

1 Introduction

Computational fluid dynamics (CFD) is becoming a major element in the aerodynamic design and analysis of full aircraft and aircraft components. As computers and solution algorithms become even faster, numerical analysis will gradually replace wind-tunnel testing in most design procedures in the aerospace industry. The emphasis on wind tunnel testing will shift from parametric testing of candidate design configurations to validation of CFD numerical analyses and verification of final designs. CFD analysis can potentially generate design data much faster and at substantially less cost than by using wind-tunnel testing. CFD also offers the capability of numerically simulating flow fields that can not be (or are extremely difficult to be) achieved in wind tunnels. Since hypervelocity high-temperature flows are very difficult to achieve in wind tunnels, CFD analysis is of critical importance in the design of hypersonic aircraft such as the National Aerospace Plane (NASP).

CFD analyses require immense computer resources. The future success of CFD analysis depends greatly upon the development of faster computers and better solution algorithms. Solutions of the 3D Navier-Stokes equations (modeling the viscous flow of compressible fluids) run for hours on present supercomputers (such as the Cray-X-MP, Cray-II, and the Cyber 205) for the analysis of flow about relatively simple aircraft components. Peterson [1] projects that computers capable of well over one teraflops (one trillion floating-point operations per second) and having at least one billion words of memory will be necessary for accurately simulating whole aircraft turbulent flow fields using current algorithms. The estimates are even more awesome when additional complexities are introduced into the equations modeling the flow field: multi-species chemical reactions, multiple phases, sub-grid scale turbulence models, and direct simulation of turbulence. The (theoretical) technological limit of the Single-Instruction-Single-Data (SISD) and the Single-Instruction-Multiple-Data (SIMD) computer architectures currently used on most supercomputers, however, is only about one gigaflop (one billion floating-point operations per second) [2]. New computer architectures must be utilized to achieve the substantial increases in computing speed required for the desired CFD applications.

The only clear path to achieving substantial increases in computing speeds is the implementation of Multiple-Instruction-Multiple-Data (MIMD) computer architectures — that is parallel computers. Numerous first generation parallel computers have been built and/or are presently available. These include the eight-cpu Cray-Y-MP, ILLIAC-IV, Denelcor's HEP computer, Alliant's FX/8, Intel's Touchstone (iPSC/860), and Silicon Graphics Iris. Unfortunately, the current algorithms have been developed primarily for SISD machines with some use of vectorization. These algorithms will not be able to utilize the full capabilities of parallel computers efficiently.

There is an urgent need to develop parallel algorithms for CFD flow analysis codes to take advantage of parallel computers, but development of parallel algorithms is not a straight forward process. One of the problems with the use of parallel computers is that their architectures differ in many respects. The number of CPU's can vary from 2 to tens of thousands and have greatly differing computing capability. The memory may be globally shared by all CPU's or each CPU may have its own dedicated memory. The method by which the CPU's communicate is also a variable. Each CPU may have a dedicated path to all of the memory, a switching network that connects the CPU's to memory, a bus that provides a common path connecting all the CPU's with the memory, or other methods as well. The above architectural aspects of parallel computers, and the fact that these architectures are changing in time, make the development of parallel algorithms difficult.

The goal of this work is to develop a flow analysis procedure for solving the three-dimensional Navier-Stokes equations. The flow analysis procedure is capable of simulating three-dimensional viscous hypersonic flows over complex aerodynamic bodies, including the effects of finite-rate chemical reactions. Specific applications could include hypersonic vehicles like the National Aerospace Plane (NASP), SDI interceptors, as well as other conventional aircraft flow fields. The flow analysis procedure utilizes efficient parallel algorithms for efficient computing on three parallel computer: a CRAY Y-MP with 8 processors, a distributed memory Intel iPSC860 computer with 128 processors, and a Silicon Graphics Iris 4-D with 8 processors.

The Navier-Stokes equations, with additional equations for thermochemical nonequilibrium, provide an accurate mathematical model of the flow of gases over most aerodynamic bodies at all speeds from low subsonic to hypersonic. If engineers could solve these flow equations accurately and timely (like within an hour) for partial or full aircraft configurations, the aircraft design process would be revolutionized. Engineers could rapidly evaluate candidate configurations, and explore radically different designs quickly and inexpensively. If numerical flow solutions could be performed in much less than one hour, then software could be developed to perform automated optimization of aircraft aerodynamic designs. The results would be more fuel-efficient and higher-performance aircraft that cost less to design as compared to today's aircraft.

During the Phase I work several candidate parallel algorithms were developed and implemented into a prototype 3D Navier-Stokes code. The algorithms were developed on an Encore Multimax computer [3] with 10 NS32332 32-bit microprocessors, each capable of executing 2 million instructions per second (mips), yielding a total of 20 MIPS (millions of instructions per second) and approximately 3 MFLOPS (millions of floating-point operations per second) for the Linpack double precision benchmark. After incorporating the algorithms into prototype computer codes, the codes were applied to a computationally demanding flow field calculation on the Multimax.

During the current work (Phase II) the most promising parallel algorithm has been refined and optimized. The algorithm is incorporated into a production code capable of simulating hypersonic flows over complete aircraft with complex geometry. The final result of Phases II and III will be a useful CFD flow analysis code that efficiently utilizes the parallel processing capability of MIMD computers with large numbers of processors and can be applied to the most computationally demanding flow field problems.

This report is broken into several sections. The section following this introduction, section 2, discusses the specific tasks performed during this contract. Section 3 discusses the model code developed to experiment with parallel processing before beginning parallelizing the more complex Navier-Stokes code. Section 4 discusses the governing equations and boundary condition solved by the Navier-Stokes code and section 5 discusses the solution procedure used by the Navier-Stokes code. Section 6 discusses the parallelization of the Navier-Stokes code on three parallel computers. Section 7 discusses the the suite of test cases chosen for this contract. Section 8 discusses the analysis of the parallel performance of the Navier-Stokes code and, finally, section 9 gives conclusions.

2 Phase II Tasks

This section contains a brief description of the tasks completed during this contract. Details of the research and results are discussed in following sections.

TASK 1. Develop a Parallel Implicit Algorithm for the Model Equations for the Silicon Graphics Iris.

A model code was developed that simulates the operation count, memory accesses, memory storage requirements, and the ratio of serial/parallel code of the Navier-Stokes equations. Since the model code uses the parallel implicit algorithm to solve a simple set of Laplace equations, instead of the much more complicated Navier-Stokes equations, the development time is much shorter and modifying the algorithm is much more convenient. The model code allowed us to study the behavior of the parallel implicit algorithm on the Silicon Graphics Iris 4-D such as data communication overhead, optimum subzone size, and potential speedup. Using this model code the parallel implicit algorithm was optimized for execution speed. The model code was written in Fortran 77. See section 3 for discussion and results.

Task 2. Develop a Parallel Implicit Algorithm for the Model Equations for a Distributed-Memory MIMD Computer.

As in TASK 1 a model code was developed to study the behavior of the parallel implicit algorithm on a distributed-memory MIMD computer. The distributed-memory MIMD computers available at the beginning of this contract were examined and the Intel Touchstone iPSC860 computer system was chosen. Time was obtained on the 128 processor iPSC860 at NASA's Numerical Aerodynamic Simulator (NAS) to perform the programming of this and latter tasks.

Task 3. Develop a Parallel Implicit Algorithm for the Model Equations for the Cray Y-MP Computer.

As in TASK 1 a model code was developed to study the behavior of the parallel implicit algorithm on a Cray Y-MP supercomputer. The code was parallelized using macrotasking. The Cray Y-MP at NASA's Numerical Aerodynamic Simulator (NAS) at Moffett Field, California was used. An important part of the effort was to compare the cost/performance of various MIMD computers with current supercomputers.

Task 4. Evaluate Ada, C, and Mixed Fortran77/C Languages.

The use of Ada, C and mixing Fortran77 with C, instead of Fortran77 was evaluated. It was decided that the alternative languages provided no benefit significant enough to warrant reprogramming the existing production Navier-Stokes program. Fortran 77 was used for all of the remaining tasks.

Task 5. Implement a Parallel Implicit Algorithm into the Zonal Navier-Stokes code for the Silicon Graphics Iris.

In the first four tasks we learned how the parallel implicit algorithm should be implemented into the full zonal Navier-Stokes code. With this information, in this task the coding was performed for execution on the Silicon Graphics Iris 4-D computers.

The governing equations and solver for the Navier-Stokes code are described in section 4 and 5, respectively. This code is also used in the following five tasks.

Task 6. Calculate Flow Problems on the Silicon Graphics Iris.

In this task a suite of six to ten *real-life* engineering calculations was performed on the Silicon Graphics Iris. These cases include simple and complex two and three-dimensional viscous flows. Comparisons where made to experimental data. The parallel efficiency of the algorithm was measured over a wide range of processors. These problems include subsonic, transonic, and supersonic flows, over a range of Reynold's number, and with various types of boundary conditions. The choice of test cases is described in section 7.

Task 7. Implement a Parallel Implicit Algorithm into the Zonal Navier-Stokes code for a Distributed-Memory MIMD Computer.

The parallel implicit algorithm was implemented in the zonal Navier-Stokes code for execution on a distributed-memory MIMD computer. The distributed-memory MIMD computers available at the beginning of this contract were examined and the Intel Touchstone iPSC860 computer system was chosen. Time was obtained on the 128 processor iPSC860 at NASA's Numerical Aerodynamic Simulator (NAS) to perform the programming of this task and the calculations of Task 8.

Task 8. Calculate Flow Problems on the Distributed-Memory MIMD Computer.

As in Task 6, we ran the same suite of *real-life* engineering flow problems to evaluate the performance of the zonal Navier-Stokes code on the distributed-memory MIMD computer.

Task 9. Implement a Parallel Implicit Algorithms into the Zonal Navier-Stokes code for the Cray Y-MP.

In order to get comparisons with current supercomputer speeds, we implemented the parallel implicit algorithm into the zonal Navier-Stokes code for execution on the CRAY Y-MP.

Task 10. Calculate Flow Problems on the Cray Y-MP Computer.

The suite of *real-life* engineering flow problems was run on the CRAY Y-MP with the number of processors varying from 1 to 8.

Task 11. Evaluate and Compare Results.

In this task we evaluated and compared the results from the previous tasks. In particular, the parallel speedup and efficiency of selected flow problem on each computer was compared. Trends were identified. Effects of flow conditions, computer architecture, algorithm characteristics, and any other significant items were analyzed and documented in section 8.

Task 12. Deliver Software to DARPA.

The zonal Navier-Stokes software that is developed in this contract will have use by others in the DARPA research community for application to flow field computations. There may also be interest in the DARPA research community to analyze and study the actual coding in the actual coding in the zonal Navier-Stokes code in order to better understand the parallel implicit algorithm. The use of the code are be documented in a user's manual. The software is being delivered to DARPA.

Task 13. Reporting, Documentation, and Reviews.

Results of this work were documented in semi-annual technical reports and in this final technical report. R&D status reports and funds expenditure charts were delivered monthly. In order to enhance the communication between DARPA and Amtec, oral reviews were made periodically at the DARPA office in Arlington, Virginia.

3 Model Code

A model code was developed to simulate the operation count, memory accesses, memory storage requirements, and the ratio of serial to parallel code in the Navier-Stokes code. This development was under tasks 1 through 4 as described in section 2. Since the model code uses a parallel implicit algorithm to solve a simple set of Laplace equations, instead of the much more complicated Navier-Stokes equations, the development time is much shorter and modifying the algorithm is much more convenient. The model code is written to match the structure of the Navier-Stokes code as closely as possible.

The model code solves the Laplace's equation ($\nabla^2\phi = 0$) on a three-dimensional multiple-zone Cartesian grid. The internal points are differenced using standard second differences

$$\begin{gathered}\frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{\Delta x^2} + \\ \frac{\phi_{i,j+1,k} - 2\phi_{i,j,k} + \phi_{i,j-1,k}}{\Delta y^2} + \frac{\phi_{i,j,k+1} - 2\phi_{i,j,k} + \phi_{i,j,k-1}}{\Delta z^2} = 0\end{gathered} \tag{1}$$

Dirichlet boundary conditions are used with ϕ set to unity on some of the faces of the zones and zero on the other faces. The result, when equation 1 is applied to all cells in the grid, is a large sparse linear system of algebraic equations. These equations are solved using point Gauss-Seidel relaxation. This relaxation scheme is modified to be a lower-upper (LU) approximate factorization as is done for the Navier-Stokes solver (see section 5). The procedure uses a diagonal wavefront algorithm with which the inner most loop is over all points on the $i + j + k$ constant plane. All points on this plane are independent of the other points on the plane so that they may all be updated simultaneously. This allows the code to vectorize (or parallelize) over the inner loop. The Navier-Stokes solver also uses a diagonal wavefront algorithm.

The model code and the Navier-Stokes code are multiple-zone codes which use multiple i, j, k ordered grids patched together at common boundaries. The hierarchical structure of the codes reflect their multiple-zone nature. The main routine is a driver routine which calls subroutines to operate on zones. The main operations are "perform one iteration for a zone" and "transfer data between zones". This model fits nicely with the data partitioning and interprocessor communication approach required on parallel computers.

Development of the model code was completed on the Cray Y-MP, the Intel Touchstone iPSC/860 and the Silicon Graphics Iris 4-D machines. The following subsections present results and specific implementation of the domain decomposition due to different computer architectures on the distributed memory MIMD computer (Intel Touchstone iPSC/860), the Cray Y-MP, and the SGI machine.

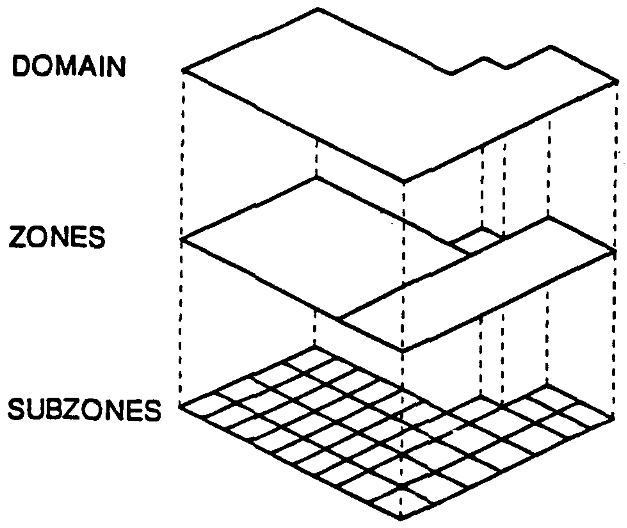


Figure 1: Domain Decomposition on Distributed Memory MIMD Computer

3.1 Distributed Memory MIMD Computer

The code was parallelized on the distributed memory computer using a permanent domain decomposition technique. The basic idea of domain decomposition is to break up the overall domain into subdomains (subzones) which are assigned to separate processors. In this case, the number of subzones is equal to the number of processors used in the calculations. A schematic diagram of the domain decomposition of a simple two-dimensional geometry is shown in Figure 1. The size of the subzones must be small enough to fit within the available local memory of the processor attached to that subzone. Calculations are driven using a global time step. Subzones are connected by means of interzone patches which provide communication between adjacent subzones. This communication between the subzones on different nodes is done by copying the data from the sending subzone to an intermediate array on the same processor. This array is then sent to an intermediate array on the receiving processor and subsequently used as boundary conditions in the corresponding subzone on the receiving processor.

However, this domain decomposition technique changes the relaxation procedure somewhat because data is lagged at the boundaries of the zones. We studied the effect of this modification on the convergence rate by running a $40 \times 40 \times 40$ calculation using 1 through 25 processors on the iPSC/860 computer (i.e., 1 to 25 zones). Figure 2 shows the results of the model code with number of zones varying from 1 to 25. Figure 3 shows the corresponding convergence rates. It is seen that subdivision of the zones reduces the convergence rate slightly as the number of zones increases. However, the effect seems to be small and therefore, can be ignored especially when the gain in speed up is significant.

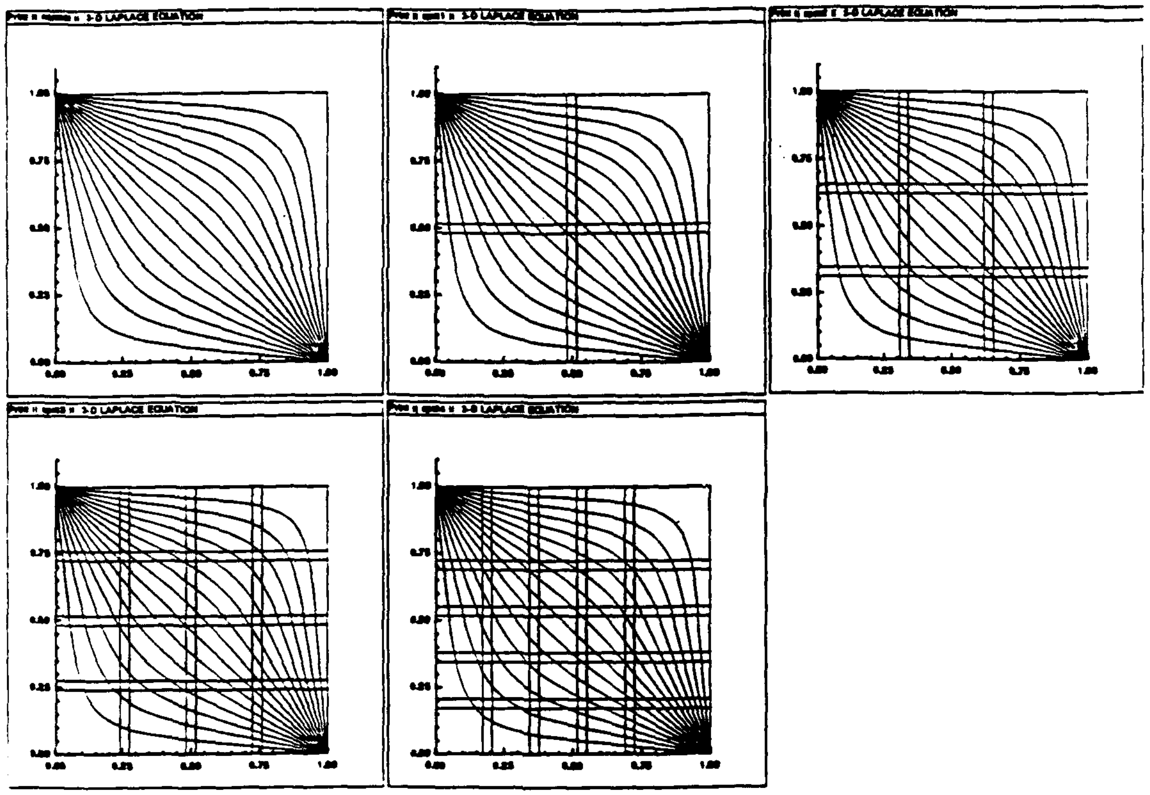


Figure 2: Solutions to model equation with domain decompositions

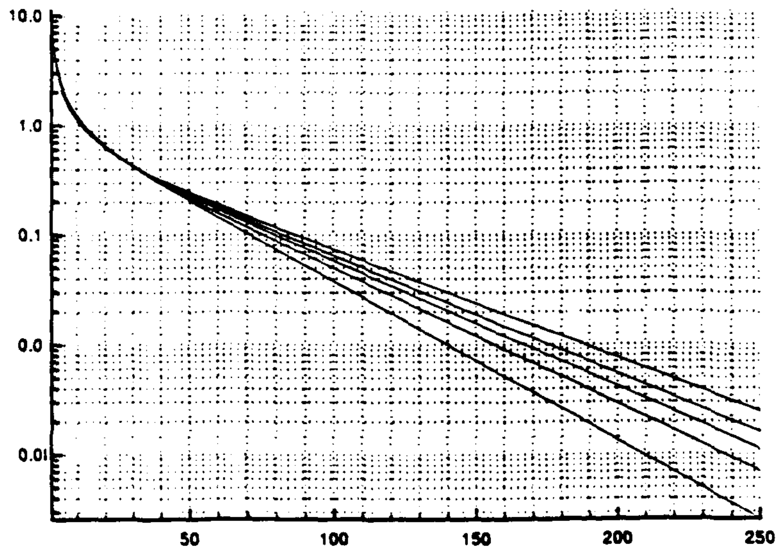


Figure 3: Convergence rates for model equation with domain decompositions

Processors	Time (seconds)	Speed-Up	Efficiency	MFlops
1	1763.5	1.00	100	1.70
4	440.1	4.00	100	6.81
9	196.0	9.00	100	15.30
16	110.5	15.96	100	27.14
25	73.6	23.96	96	40.75
32	59.6	29.59	92	50.32
36	53.0	33.27	92	56.59
49	41.8	42.19	86	71.75
64	32.9	53.60	84	91.16

Table 1: Results of Model Code Calculations on Intel iPSC/860 MIMD Computer.

The Laplace code was run numerous times to obtain timings on the iPSC/860 parallel computer. The results, which were obtained using the vector option on the Intel Portland Group compiler (if77), are shown in Table 1. Note that the efficiencies are very high for this problem. The MFlops were calculated from the measured time on the iPSC/860 and the number of floating point operations measured by the Cray hardware performance monitor for the Cray Y/MP version of the model code. The main contributors to the reduction in efficiency with increasing number of processors are sequential code, load leveling, and interprocessor communication overhead.

Sequential code are sections of the computer program that cannot be parallelized. The time required to execute the sequential portion of the code does not decrease with increasing number of processors and therefore becomes increasingly significant as the total run time decreases with the addition of more processors. Even though the algorithm is designed to be perfectly parallel (i.e., no sequential portion whatsoever), there are redundant initializations which are performed by each processor and thus can be thought of as sequential code. Unfortunately, the overall amount of work performed by the Laplace code is small and hence the work performed on this very small portion of sequential code is a significant percentage of the total work.

The second contribution to the reduction in efficiency is poor load leveling. Load leveling is making the amount of work performed by all processors equal. If the amount of work in not equal then some processors will waste CPU time waiting for other processors to finish, and hence the overall performance is dictated by the slowest processor (or processor with the most work). The model code automatically subdivides the grid into relatively equal subzones. If the subzones are all of the same size, the amount of work performed by the processors is nearly equal and the load is leveled. Unfortunately, the solution domain cannot always be subdivided into equally sized subzones. To elucidate this effect, we have performed calculations on four different grid sizes. The original speed-up and efficiencies are shown in Figures 4 and 5.

The lack of load leveling can be factored out to obtain the potential speed-up and

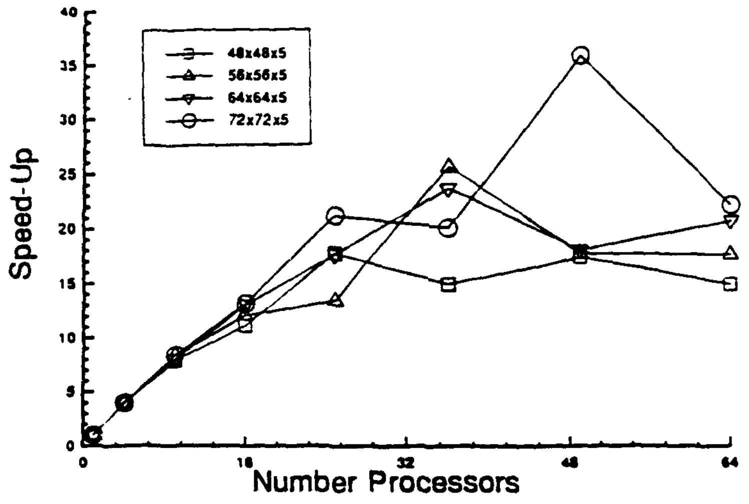


Figure 4: Speed-up with Load Balancing Problem

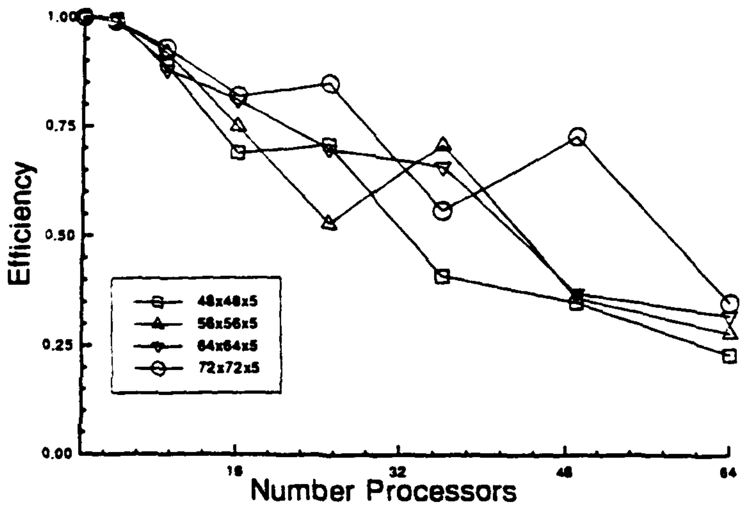


Figure 5: Efficiency with Load Balancing Problem

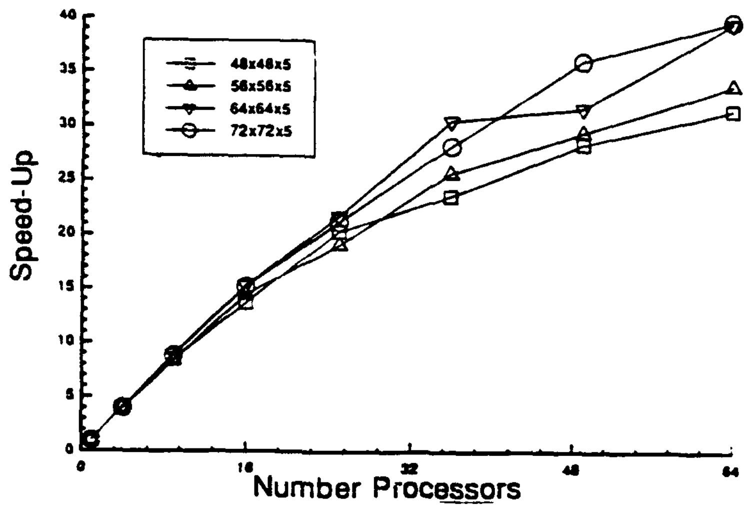


Figure 6: Speed-up for Perfect Load Leveling

efficiency when the work load of each processors are equal,

$$\text{efficiency} = \frac{\text{average zone size}}{\text{maximum zone size}}.$$

The adjusted speedup and efficiency, with the effect of load leveling factored out, is shown in Figures 6 and 7. The adjusted efficiency with 64 processors varies from 48% for a 48x48x5 grid to 62% for a 72x72x5 grid. The remaining reduction in efficiency is due to sequential code and interprocess communication overhead. Note that for smaller problems, the overall parallel performance is shown to be degraded. Therefore, to maximize a parallel performance, one must run the largest problem possible on each of the processor. For small problems, use a small number of processors and for large problems use a larger number of processors. However, if wall-clock-time is the primary interest, the more processors used in the calculation the shorter the time. For the problems considered, there was never a case where increasing the number of processors increased the run time.

The final contribution to the reduction in efficiency is interprocessor communication overhead. Interprocessor communication adds an overhead which is highly dependent on the parallel computer being used. For the iPSC/860 the communication rate between processors is very fast so the main overhead is in establishing the communication link. The iPSC/860 therefore favors the passing of a few large sections of data rather than many small sections of data. For a given size problem, the interprocessor communication overhead increases with the number of processors for two reasons: the total amount of data transferred increases and the number of interzone communication links increases (more communication startup overhead). Interprocessor communication overhead is believed to be the biggest contributor to the

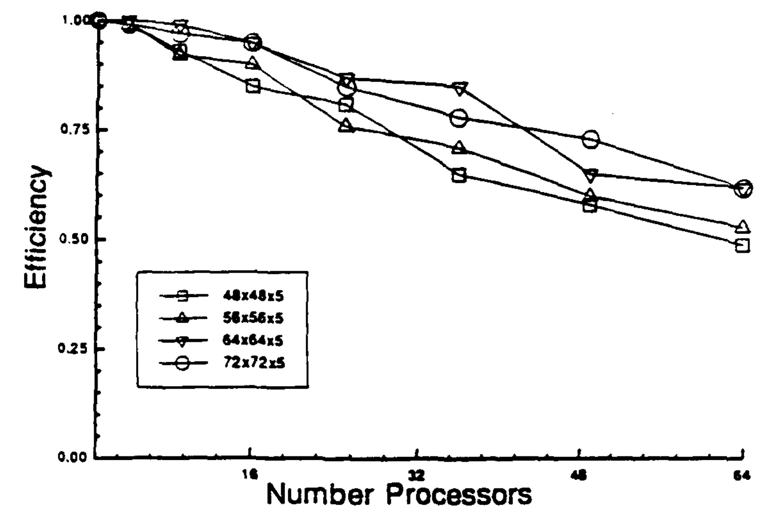


Figure 7: Efficiency for Perfect Load Leveling

reduction in parallel efficiency in Figures 6 and and 7.

3.2 Shared Memory Cray Y-MP

The procedure for parallelizing on the Cray Y-MP is similar to the procedure used for the distributed memory MIMD computer. Both use domain decomposition, where the overall domain (original zones) is broken up into subdomains (new zones) which are assigned to separate processors. This subdivision of zones changes the relaxation procedure because data is lagged at the boundaries of the zones. The effect of these lagged boundaries on the convergence rate was shown to be small for the iPSC/860 version, and since the Cray Y-MP has only 8 processors this effect is negligible. The parallel performance was measured for macrotasked runs with 1 to 8 processors. It is important to note that these runs were performed in the dedicated (one user) mode, since there are no tools to measure the performance of macrotasked jobs in the multiuser mode. The resulting speedup and parallel efficiency are shown in Figure 8. The efficiency varies from 100% for one cpu to 23% for 8 cpu's. This is a very rapid drop in efficiency compared to the results for the Intel Touchs one computer. In fact, the speedup in Figure 8 shows that it is actually slower to run with 6 or 8 processors than to run with 4 processors.

It was concluded that the poor parallel performance of the model code on the Cray Y-MP is due to a reduction in vector length with zonal subdivision. The Cray Y-MP cpu's are highly dependent on vectorization to obtain their good performance. The domain decomposition used to parallelize the code reduces the vector lengths. To test this hypothesis we ran the model code on one cpu with the grid divided into subzones as it is when run in parallel. The speedup and efficiency are then calculated

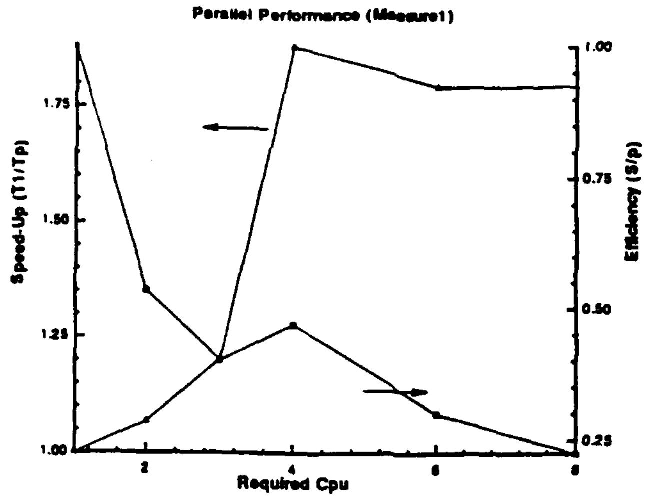


Figure 8: Parallel performance of model code on the Cray Y-MP - calculated by comparing to one zone on one cpu

by comparing to the single cpu results with an equivalent number of subzones. The results are shown in Figure 9, where p is the number of subzones. Using this comparison, the parallel efficiency varies between 98% and 126%. Clearly, the reduction of vector length due to the division of the solution domain into subzones is the primary cause of the low performance in Figure 8, not the fact that multiple cpu's are used.

3.3 Shared Memory Silicon Graphics Iris

Most of the subroutines developed for Cray Y-MP and distributed memory MIMD computers were used to develop a parallel version of the model equations on the Silicon Graphics Iris 4-D with 8 cpu's. The SGI is a shared memory MIMD computer. The parallelization is done by generating threads that run on the different processors. The DO loop is the basic work unit which is split into concurrent threads. Since the compiler on this machine is limited only to DO loop parallelization, the original thread is the master, and it controls the execution of all other threads. By splitting the DO loops surrounding the A-level in the driver routine, the code is executed in parallel. The C$DOACROSS compiler directive is used for the DO loop. The subzone number is defined as the LASTLOCAL variable where the final value is important. However, the iteration number, time step, and convergence rate are defined to be shared. Data for each subzone is isolated and can only be accessed by one thread, thus preventing memory access collisions.

Based on the $i-$ and $j-$direction divisions specified by the user, each zone is divided automatically into subzones. One of the processors must perform the job management and i/o in addition to number crunching. This adds work to one pro-

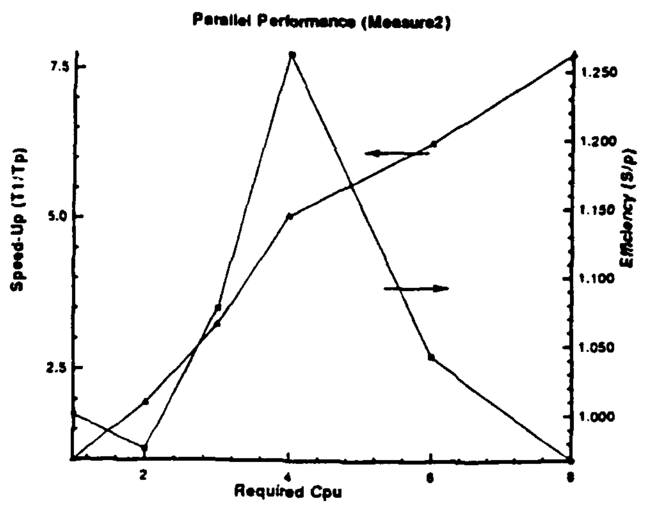


Figure 9: Parallel performance of model code on the Cray Y-MP - calculated by comparing to *p zones on one cpu*

cessor and could effect the load leveling for small problems, but it is insignificant for most problems. The interzone communication is done by copying boundary condition data into an intermediate array. Data in this array is locked until all processors finish the time step. This synchronizes the calculation so that no processor may start on the next time step until all processors are finished with the current time step. If the subdivision of the zones has a load balancing problem, or if other processes delay the execution of one thread. this synchronization will significantly reduce the parallel efficiency. However, synchronization is necessary to prevent memory collision which could easily happen in a shared memory system. After the time step is finished in all subzones. data in the intermediate arrays are available to complete the transfer of data between the subzones.

The model code was run numerous times to obtain timings on the parallel Silicon graphics computer. The solution was the same as that obtained on the iPSC/860, Figure 2. The parallel speed up and efficiency results are shown in Figures 10. The test case consists of $40 \times 40 \times 40$ mesh points, similar to the one used for timings on the Cray Y-MP and Intel Touchstone iPSC/860 machines. It is important to note that the runs were performed in a multi-user mode. Since the maximum number of cpus on the SGI machine is 8, the effects of the domain decomposition on the convergence rate is negligible. Unlike the Cray Y-MP, however, the reduction in vector length from the subdomain decomposition does not significantly affect the overall performance of the code. The parallel speed up and parallel efficiency are excellent for up to 4 cpu's. The significant drop in efficiency of the code for more than 4 cpus is due to memory contention, bus bandwidth limitations, and cpu-time competition from other users on

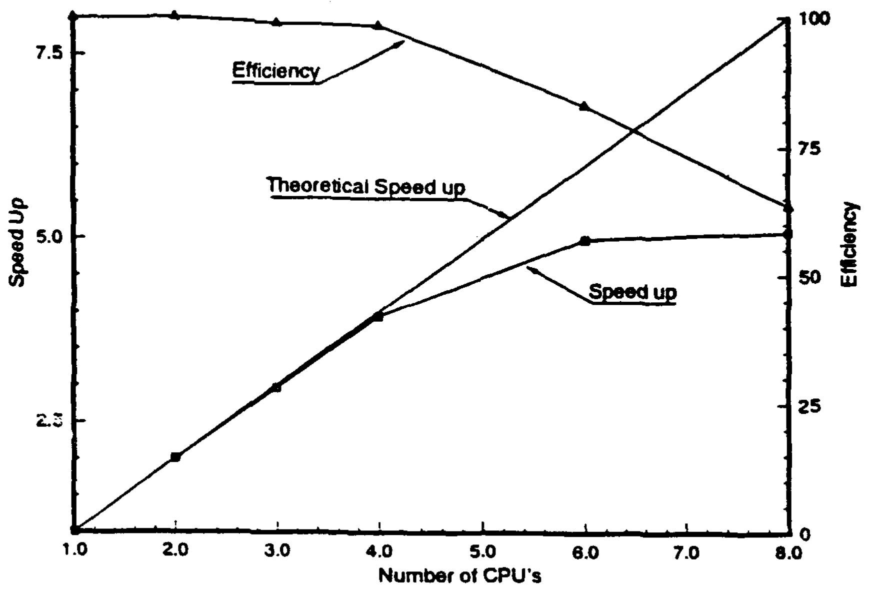


Figure 10: Parallel Performance of the Model Equation on the Silicon graphics

the system.

4 Navier-Stokes Equations with Chemistry

4.1 Transport Equations

Tasks 5 through 11 of this contract (see section 2) were performed using Amtec's existing three-dimensional Navier-Stokes code. This code solves the Reynolds-averaged Navier-Stokes equations with user specified chemistry, thermal nonequilibrium, and a two-equation turbulence model. These equations are given below in integral form [4].

$$\frac{\partial}{\partial t}\iiint_{Vol} U\,dV + \iint_S \vec{P}\cdot\vec{n}\,dS = \iiint_{Vol} N\,dV \tag{2}$$

where

$$\vec{P} = F_1\vec{i} + F_2\vec{j} + F_3\vec{k}$$

and

$$U = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_s \\ \vdots \\ \rho_{NS} \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ E \\ \rho e_{ve} \\ \rho k \\ \rho\epsilon \end{pmatrix}$$

$$F_i = \begin{pmatrix} \rho_1 u_i + \rho_1 v^d_{1i} \\ \vdots \\ \rho_s u_i + \rho_s v^d_{si} \\ \vdots \\ \rho_{NS} u_i + \rho_{NS} v^d_{NSi} \\ \rho u_i u_1 + p\delta_{1i} + \tau_{i1} \\ \rho u_i u_2 + p\delta_{2i} + \tau_{i2} \\ \rho u_i u_3 + p\delta_{3i} + \tau_{i3} \\ (E+p)u_i + u_j\tau_{ij} + q_i \\ \rho e_{ve} u_i + q_{ve_i} \\ \rho k u_i - \mu_k \frac{\partial k}{\partial x_i} \\ \rho\epsilon u_i - \mu_\epsilon \frac{\partial \epsilon}{\partial x_i} \end{pmatrix}$$

$$N = \begin{pmatrix} w_1 \\ \vdots \\ w_s \\ \vdots \\ w_{NS} \\ 0 \\ 0 \\ 0 \\ 0 \\ w_{ve} \\ G_k - \rho\epsilon \\ C_{\epsilon_1}\frac{\epsilon G_k}{k} - C_{\epsilon_2}\frac{\rho\epsilon^2}{k} \end{pmatrix}$$

$$\begin{aligned} E &= \rho\left(e + \frac{1}{2}u_j u_j\right) = \rho H - p \\ \tau_{ij} &= -(\mu + \mu_t)\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right] \\ q_i &= -(k_l + k_t)\frac{\partial T}{\partial x_i} \\ q_{ve_i} &= -(k_{ve} + k_t)\frac{\partial T_{ve}}{\partial x_i} \\ \rho_s v_{si}^d &= -\rho \mathcal{D}_s \frac{\partial X_s}{\partial x_i} \end{aligned}$$

Here the standard summation convention (sum over repeated indices) is followed and δ_{ij} is the Kronecker delta function ($\delta_{ij} = 1$ when $i = j$ and $\delta_{ij} = 0$ otherwise). The subscript s is the number of the chemical species being considered.

The w_1 through w_s are the chemical species source terms. For a set of NR reactions defined by

$$\sum_{s=1}^{NS} \nu'_{s,r} Y_s \overset{k_{f,r}}{\rightleftharpoons} \sum_{s=1}^{NS} \nu''_{s,r} Y_s$$

they are calculated from the equation

$$w_i = \frac{1}{M_i}\frac{d[Y_i]}{dt}$$

where

$$\frac{d[Y_i]}{dt} = \sum_{r=1}^{NR}\left(\nu''_{i,r} - \nu'_{i,r}\right) k_{f,r}\left\{\prod_{s=1}^{NS}[Y_s]^{\nu'_{s,r}} - \frac{1}{K_{c,r}}\prod_{s=1}^{NS}[Y_s]^{\nu''_{s,r}}\right\}$$

The forward reaction rate, $k_{f,r}$, is calculated from the Arrhenius equation using user supplied coefficients. The equilibrium constant, $K_{c,r}$, is calculated from user supplied curve fits to the species enthalpy and entropy.

The vibrational-electronic energy source term is given by the Landau-Teller formula,

$$w_{ve} = \rho \frac{e^*_{ve} - e_{ve}}{\tau_{LT}} \tag{3}$$

where the time constant τ_{LT} is given by the Landau-Teller expression,

$$\tau_{LT} = \frac{b_1 T^{b_2} \exp\left((b_3/T)^{b_4}\right)}{p(1 - b_5 e^{-b_6/T})} \tag{4}$$

and the equilibrium vibrational energy, e^*_{ve} is evaluated in terms of the translation-rotational temperature, T, using the curve fits described in section 4.2. Equations 3 and 4 are a simple relaxation of the mixture vibration-electronic energy toward equilibrium.

For turbulent flows there are two options for turbulence models: the Baldwin-Lomax algebraic model [5] and the $k - \epsilon$ two-equation model [6]. When the $k - \epsilon$ turbulence model is chosen, the last two transport equations in equations 2 are solved. The pertinent constants and relations for this model are as follows.

$$\begin{aligned}
\mu_k &= \mu + \frac{\mu_t}{\sigma_k} \\
\mu_\epsilon &= \mu + \frac{\mu_t}{\sigma_\epsilon} \\
C_{\epsilon_1} &= 1.44 \\
C_{\epsilon_2} &= 1.92 \\
\sigma_k &= 1.0 \\
\sigma_\epsilon &= 1.3 \\
G_k &= \mu_t \left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right) \frac{\partial u_i}{\partial x_k} \\
\mu_t &= C_\mu \frac{\rho k^2}{\epsilon} \\
C_\mu &= 0.09 \\
k_t &= \frac{\mu_t c_p}{P_{r_t}}
\end{aligned}$$

The values of C_{ϵ_1}, C_{ϵ_2}, σ_k, σ_ϵ, and C_μ may be specified by the user. The above values are the defaults.

Physically Equation 1 represents a very simple idea: the time rate of change of mass, momentum, and energy within an arbitrarily chosen volume, V, is equal to the apparent flux of these quantities inward through the surface, S, surrounding the volume plus the production of these quantities within the volume. The finite volume

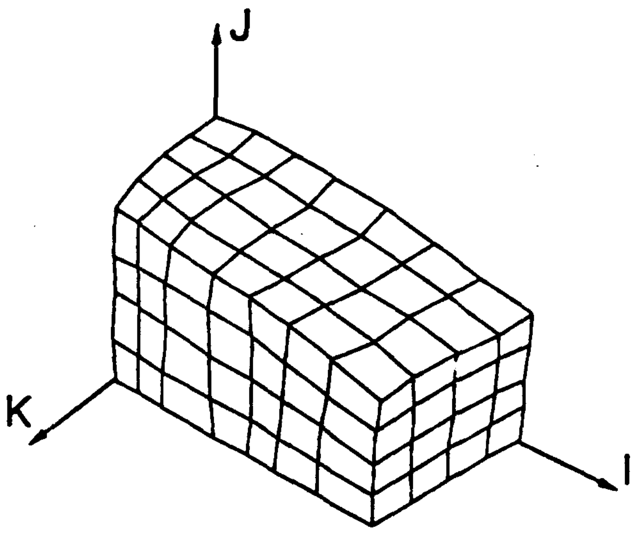


Figure 11: Finite Volume Mesh

method consists of breaking the flow field up into a large number of finite volume cells, as shown in Figure 11, and applying the integral equations directly to each volume.

4.2 Thermodynamics

The equations in the previous section must be supplemented by an equation of state to calculate p and T from ρ and e. The code offers three options for equations of state: a single species perfect gas, Tannehill's curve fits for real air, or a mixture of thermally perfect gases. The latter option must be used if multiple species or thermal nonequilibrium is considered.

In addition to calculating p and T, it is necessary to calculate five thermodynamic variables which are needed to calculate the numerical fluxes. The first two are γ and $\bar{\gamma}$. For perfect gas flows γ and $\bar{\gamma}$ are identical and are equal to the ratio of specific heats. $\gamma = \frac{c_p}{c_v}$. For real gases, however, the ratio of specific heats no longer has any physical significance and the quantities γ and $\bar{\gamma}$ are defined so as to simplify the calculation of numerical fluxes. In particular we use Vinokur's definition [7].

$$\begin{aligned} \gamma &= \frac{\rho c^2}{p} \\ \bar{\gamma} &= 1 + \frac{p}{\rho(e - e_0)} \end{aligned}$$

For numerical fluxes based on Roe's flux function Vinokur has defined the alternative variables κ and χ which are the partial derivatives of pressure p with respect to internal energy per unit volume $\tilde{e}$ and density ρ.

$$\kappa = \left.\frac{\partial p}{\partial \tilde{e}}\right|_{\rho}$$
$$\chi = \left.\frac{\partial p}{\partial \rho}\right|_{\tilde{e}}$$

The final state variable is the mean specific heat $\bar{c}_v = \frac{e-e_0}{T}$.

4.2.1 Perfect Gas

The perfect gas equation of state is expressed with the following equations:

$$p = \rho(\gamma - 1)e \quad (5)$$
$$T = \frac{e}{c_v} \quad (6)$$

The equation of state can be altered for different gases by specifying the ratio of specific heats, γ, and the gas constant, R. The gas constant for a gas of a given molecular weight, M, is calculated from the universal gas constant $\tilde{R} = 8.31434$J/mole K using the formula $R = \tilde{R}/M$. The specific heat, c_v is calculated using $c_v = \frac{R}{\gamma-1}$.

The remaining five needed state variables are given by the following formulas.

$$\begin{aligned}
\gamma &= \text{specified} \\
\bar{\gamma} &= \gamma \\
\kappa &= \gamma - 1 \\
\chi &= 0 \\
\bar{c}_v &= c_v = \frac{R}{\gamma - 1}
\end{aligned}$$

4.2.2 Equilibrium Air Curve Fits

The second option for equation of state is Tannehill's curve fits for real air. With this option the code calls a subroutine given in reference [8] to calculate p and T from ρ and e for each finite-volume cell and each time step. This routine contains curve fits for p, T, and c in terms of ρ and e obtained from detailed calculations of air in thermochemical equilibrium. The curves are valid up to 25,000K.

The remaining five needed state variables must be calculated from the outputs of the equilibrium air subroutine. In addition to pressure p and temperature T, the subroutine also returns the speed of sound, c. With this, and the fact that $e_0 = 0$ we can calculate three variables directly from their definitions.

$$\gamma = \frac{\rho c^2}{p}$$

$$\begin{aligned}\bar{\gamma} &= 1+\frac{p}{\rho e}\\ \bar{c}_v &= \frac{e}{T}\end{aligned}$$

The pressure derivatives, κ and χ cannot be directly calculated using data returned from the subroutine. Some numerical differentiation is required.

To evaluate κ and χ, start with the chain rule

$$\delta p = \left(\frac{\partial p}{\partial \tilde{e}}\right)_\rho \delta\tilde{e} + \left(\frac{\partial p}{\partial \rho}\right)_{\tilde{e}} \delta\rho = \kappa\delta\tilde{e} + \chi\delta\rho$$

and the expression for speed of sound in terms of κ and χ (equation 5 in [7])

$$c^2 = \chi + \kappa h$$

Eliminating χ from the two equations and solving for κ yields

$$\begin{aligned}\kappa &= \frac{\delta p - c^2\delta\rho}{\delta\tilde{e} - h\delta\rho}\\ &= \frac{p(\rho+\delta\rho,\tilde{e}+\delta\tilde{e}) - p(\rho,\tilde{e}) - c^2\delta\rho}{\delta\tilde{e} - h\delta\rho}\end{aligned}$$

This expression may be used to calculate κ from any combination of $\delta\rho$ and $\delta\tilde{e}$. After some experimentation, $\delta\tilde{e} = 2h\delta\rho$ was chosen since it seems to avoid most spurious results near discontinuities in the curve fits. The other pressure derivative, χ, is then calculated from $\chi = c^2 - \kappa h$.

4.2.3 Thermally Perfect Species - Thermal Equilibrium

The third option for equation of state is for thermally perfect species in either thermal equilibrium or thermal non-equilibrium. This option requires curve fits in terms of temperature for the specific heat at constant pressure, c_p, the enthalpy, h, and the entropy, s, for each species. These curve fits are expressed as polynomials for up to three user specified temperature ranges.

$$\begin{aligned}\frac{c_{p_s}}{R} &= a_{1_s} + a_{2_s}T + a_{3_s}T^2 + a_{4_s}T^3 + a_{5_s}T^4\\ \frac{h_s}{RT} &= a_{1_s} + \frac{a_{2_s}}{2}T + \frac{a_{3_s}}{3}T^2 + \frac{a_{4_s}}{4}T^3 + \frac{a_{5_s}}{5}T^4 + \frac{a_{6_s}}{T}\\ \frac{s_s}{R} &= a_{1_s}\log T + a_{2_s}T + \frac{a_{3_s}}{2}T^2 + \frac{a_{4_s}}{3}T^3 + \frac{a_{5_s}}{4}T^4 + a_{7_s}\end{aligned} \tag{7}$$

The temperature ranges chosen for most of the cases considered in this research effort are 300K to 1000K, 1000K to 6000K, and 6000K to 15000K. If the temperature exceeds the maximum temperature of the upper range or the minimum temperature of the lower range, the specific heat is assumed to be constant at the value calculated for the closest temperature within the valid temperature range. The out of range enthalpy and entropy curves are then calculated by appropriate integrations of c_p, and matching of values at the edge of the valid temperature range.

For thermal equilibrium an iteration is necessary to obtain the temperature, T, from the species densities, ρ_i, and internal energy, e, using the above curve fits. The formula for internal energy in terms of temnerature for a given set of species densities is obtained from the definition of enthalpy, $h = e + p/\rho$, the thermal equation of state, $p_i = \frac{\rho_i}{M_i}\tilde{R}T$, and the the curve fit for enthalpy from equation 7. The resulting expression,

$$e = \sum_{s=1}^{NS} \frac{\rho_s}{\rho}\frac{\tilde{R}}{M_s}\left(a_{6s} + \sum_{l=1}^{5}\frac{a_{ls}}{l}T^l - T\right) \tag{8}$$

is a nonlinear equation for T which must be solved iteratively. A Newton iteration is used to approximately solve this equation for each cell on each time step. The temperature from the previous time step is used as an initial condition for the Newton iteration.

For the evaluation of the Steger and Warming fluxes γ, $\bar{c_v}$, and $\bar{\gamma}$ are needed. The speed of sound used for the calculation of γ is the frozen chemistry speed of sound defined by

$$c^2 = \left(\frac{\partial p}{\partial \rho}\right)_{s,c_i}$$

where $c_i = \rho_i/\rho$ is the mass fraction of species i. Using this and previously given definitions, the resulting expressions for a thermally perfect mixture are

$$\begin{aligned} \gamma &= 1 + \frac{R}{c_v} \\ \bar{c_v} &= \frac{e}{T} \\ \bar{\gamma} &= 1 + \frac{R}{\bar{c_v}} \end{aligned}$$

Here

$$\begin{aligned} R &= \sum_{s=1}^{NS} c_s \frac{\tilde{R}}{M_s} \\ c_v &= \sum_{s=1}^{NS} c_s C_{v_s} \end{aligned}$$

Note that the expressions for γ and $\bar{\gamma}$ are very similar except that the actual specific heat, c_v, is used for γ while the mean specific heat, $\bar{c_v}$ is used for $\bar{\gamma}$.

The pressure derivatives, κ and χ are needed for the Harten Yee flux function. In this case there must actually be a χ defined for each species.

$$\chi_s = \left(\frac{\partial p}{\partial \rho_s}\right)_{e,\rho_{i\neq s}}$$

The expressions for the pressure derivatives for the mixture of thermally perfect species is

$$\begin{aligned} \chi_s &= \frac{\tilde{R}}{M_s}T - \frac{R}{c_v}e_s \\ \kappa &= \frac{R}{c_v} \end{aligned}$$

In the above expressions R and c_v are the mixture values defined previously.

4.2.4 Thermally Perfect Species - Thermal Nonequilibrium

For thermal nonequilibrium we must consider the detailed contributions to the internal energy. It is generally assumed [9, 10, 11, 12] that the internal energy is the sum of *independent* energies from translational motion, rotation motion, electronic excitation, and vibrational excitation.

$$e = e_t + e_r + e_e + e_v$$

In general, each of these components may have an independent temperature associated with it whereas in equilibrium all of the temperatures are the same. For most calculations it is reasonable to assume that the rotational mode is in equilibrium with the translational mode [10, 11] since they both equilibriate very fast in comparison to the vibrational mode. This allows these two energies to be considered, for all practical purposes, as one energy, $e_{tr} = e_t + e_r$, which is described by the temperature T. Likewise, the electronic and vibrational energy levels may be considered to be in equilibrium and are described by the combined energy $e_{ve} = e_e + e_v$ and the temperature T_v. Furthermore, it is assumed that the vibrational/electronic modes of all species are in equilibrium with one another and with the electron translational energy mode at the temperature T_v.

The goal of this section is to describe how the temperatures T and T_v are calculated from the species densities ρ_s, total internal energy e, and vibrational/electronic energy e_{ve}. To do this the functional form of the specific heats for the two modes in terms of their respective temperatures is needed. Since the energy modes are independent we may write for the case of equilibrium

$$c_{v_s}(T) = c_{v_{tr,s}}(T) + c_{v_{ve,s}}(T)$$

The variation of c_{v_s} is easily obtained for each species from the curve fits of equation 7.

$$\frac{c_{v_s}}{R} = a_1 - 1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$$

Since the code is generally used to analyze flows of gas at high temperatures, the rotational modes of diatomic or polyatomic gases are assumed to be fully excited. The species specific heat for the translational and rotational modes is therefore independent of temperature and given by

$$\frac{c_{v_{tr,s}}}{R} = 1.5 + d \tag{9}$$

where d is the number of rotational degrees of freedom. For monatomic molecules $d = 0$, for diatomic molecules $d = 2$, and for non-inline polyatomic molecules $d = 3$. The remaining contributions to specific heat are from vibrational and electronic excitation.

$$\frac{c_{v_{ve,s}}}{R} = a_1 - 2.5 - d + a_2T + a_3T^2 + a_4T^3 + a_5T^4$$

This approach to calculating $c_{v_{ve,s}}$ differs considerable from the approach taken by Candler [12] and Park [11]. They both used approximate formulas for $c_{v_v,s}$ and $c_{v_e,s}$ based on fundamental physics. However, their approach is more complex and should yield no better accuracy than the approach taken here. Furthermore, the current approach has the advantage of keeping the physics used to calculate the equilibrium constants for the chemical reactions consistent with the physics used to calculate the temperatures. Both Candler and Park use simple curve fits for the backward reaction rates which is less accurate and less general than calculating the backward reaction rates from the equilibrium constants and the equilibrium constants from the thermodynamics as is done in this investigation.

Once the variation of the specific heat $c_{v_{ve,s}}$ with T_{ve} is known we may obtain the variation of e_{ve} with T_{ve} for a species by integration.

$$\begin{aligned}\frac{e_{ve_s}}{R} &= \int_{T_0}^{T_{ve}} c_{v_{ve,s}}(T)dT + e_{ve_s}(T_0) \\ &= b_{6s} + (a_{1s} - 2.5 - d_s)\,T_{ve} + \frac{a_{2s}}{2}T_{ve}^2 + \frac{a_{3s}}{3}T_{ve}^3 + \frac{a_{4s}}{4}T_{ve}^4 + \frac{a_{5s}}{5}T_{ve}^5 \qquad (10) \\ & \qquad (11)\end{aligned}$$

where T_0 is some temperature, within the temperature range for the curve, at which the value of e_{ve_s} is known and

$$b_{6s} = e_{ve_s}(T_0) - \left[(a_{1s} - 2.5 - d_s)\,T_0 + \frac{a_{2s}}{2}T_0^2 + \frac{a_{3s}}{3}T_0^3 + \frac{a_{4s}}{4}T_0^4 + \frac{a_{5s}}{5}T_0^5\right] \tag{12}$$

As mentioned previously the curve fits are generally broken into three separate polynomials for three temperature ranges. The constant, b_{6s}, is first calculated using the curve for the temperature range containing T_{ref}, the reference temperature, where

$e_{ve_s} = 0$. The calculation is performed using equation 12 with $T_0 = T_{ref}$ and $e_{ve_s} = 0$. The constant b_{6_s} is then calculated for neighboring curves, using equation 12 by requiring that e_{ve_s} be continuous at the boundary between two curves.

Equation 10 is an expression for the species vibrational energy e_{ve_s} in terms of the vibrational temperature T_{ve}. After making the appropriate sums the expression for the mixture vibrational energy e_{ve} in terms of the vibrational temperature T_{ve} is

$$e_{ve} = \sum_{s=1}^{NS} \frac{\rho_s}{\rho} \frac{\tilde{R}}{M_s} \left[b_{6_s} + \sum_{l=1}^{5} \frac{a_{ls}}{l} T_{ve}^{l} - (2.5 + d_s) T_{ve} \right] \tag{13}$$

This expression is a nonlinear equation for T_{ve} which must be solved iteratively. A Newton iteration is used to approximately solve this equation for each cell on each time step. The vibrational temperature from the previous time step is used as an initial condition for the Newton iteration.

The equation for the species translation/rotational energy $e_{tr,s}$ in terms of translational/rotational temperature T is obtained by integrating equation 9 for $c_{v_{tr}}$ and requiring that, when in equilibrium, $\frac{e_{tr,s}}{R} + \frac{e_{ve,s}}{R} + 1$ give the original curve fit, equation 7. The result is

$$\frac{e_{tr,s}}{R} = a_{6_s} - b_{6_s} + (1.5 + d_s) T \tag{14}$$

When summed appropriately this expression gives the following equation

$$e - e_{ve} = \sum_{s=1}^{NS} \frac{\rho_s}{\rho} \frac{\tilde{R}}{M_s} [a_{6_s} - b_{6_s} + (1.5 + d_s) T] \tag{15}$$

This is a linear algebraic equation which may be solved directly for T.

The speed of sound used to evaluate γ is the frozen-chemistry, frozen vibrational-electronic energy speed of sound defined by

$$c^2 = \left(\frac{\partial p}{\partial \rho} \right)_{s, c_i, e_v}$$

Using this and previously given definitions, the resulting expression for a thermally perfect mixture in thermal nonequilibrium are

$$\gamma = \tilde{\gamma} = 1 + \frac{R}{c_{v_{tr}}}$$

Note that $c_{v_{tr}}$ is the mixture translational-rotational specific heat. This value is independent of temperature but depends on the species concentrations.

With the addition of thermal nonequilibrium there are now two κ's; one for each independent internal energy.

$$\kappa_{tr} = \left(\frac{\partial p}{\partial \tilde{e}_{tr}} \right)_{\tilde{e}_{ve}, \rho_i}$$
$$\kappa_v = \left(\frac{\partial p}{\partial \tilde{e}_{ve}} \right)_{\tilde{e}_{tr}, \rho_i}$$

As before, there is a χ defined for each species.

$$\chi_s = \left(\frac{\partial p}{\partial \rho_s}\right)_{\tilde{e}_{tr},\tilde{e}_v,\rho_{t\neq s}}$$

The expressions for the pressure derivatives for the mixture of thermally perfect species is

$$\begin{aligned}
\chi_s &= \frac{\tilde{R}}{M_s}T - \frac{R}{c_{v_{tr}}}e_{tr_s} \\
\kappa_{tr} &= \frac{R}{c_{v_{tr}}} \\
\kappa_v &= 0
\end{aligned}$$

The pressure derivative with respect to the vibrational-electronic energy, $\tilde{e}_{ve}$, is zero because the pressure is not a function of the vibrational-electronic temperature, T_v. If the translational energy of the free electrons is in equilibrium with the vibrational-electronic energy of the other species rather than their translational-rotational internal energy, the pressure will depend on T_v and the above expressions will be more complicated.

4.3 Transport Properties

The transport properties are the viscosity μ, the conductivity k, and the binary diffusion coefficient $\mathcal{D}$. These quantities are calculated within the code using various user specified methods. These methods are summarized below.

4.3.1 Viscosity

The viscosity is calculated using one of three methods. The first method is to assume that the viscosity depends only on temperature and that the dependence may be approximated by the formula

$$\mu = \frac{A_{\mu_1}T^{A_{\mu_2}} + A_{\mu_3}T + A_{\mu_4}}{A_{\mu_5}T + A_{\mu_6}}.$$

The famous Sutherland's law is obtained by setting

$$\begin{aligned}
A_{\mu_1} &= 1.451 \times 10^{-6} \\
A_{\mu_2} &= 1.5 \\
A_{\mu_3} &= 0 \\
A_{\mu_4} &= 0 \\
A_{\mu_5} &= 1 \\
A_{\mu_6} &= 110.
\end{aligned}$$

The second method for calculating the viscosity is the equilibrium air curve fits of Tannehill [13]. The third and final method for calculating viscosity is to use Blottner's formula for the viscosity of individual species

$$\mu_s = 0.1 \exp\left[\left(A_s^b \ln T + B_s^b\right) \ln T + C_s^b\right],$$

and then use Wilke's mixing formula to obtain the mixture viscosity. The Wilke's semiempirical mixing rule is

$$\mu = \sum_s \frac{X_s \mu_s}{\phi_s}$$

where X_s is the mole fraction of species s and

$$\phi_s = \sum_m X_m \left[1 + \sqrt{\frac{\mu_s}{\mu_m}} \left(\frac{M_m}{M_s}\right)^{1/4}\right]^2 \left[8\left(1 + \frac{M_s}{M_m}\right)\right]^{-1/2}. \tag{16}$$

This latter method was used by Candler for thermochemical nonequilibrium calculations [12].

4.3.2 Conductivity

Conductivity is calculated using one of four methods. The first is to assume a Prandtl number P_r and calculate the conductivity from

$$k = \frac{\mu c_p}{P_r}.$$

This method is always used for turbulent flows. A second is to assume that the conductivity is a function of temperature only and use the same sort of formula as used for viscosity,

$$k_l = \frac{A_{k_1} T^{A_{k_2}} + A_{k_3} T + A_{k_4}}{A_{k_5} T + A_{k_6}}.$$

The third method for calculating the conductivity is the equilibrium air curve fits of Tannehill [13]. The final option for calculating conductivity is to use the Eucken formula to calculate the species conductivity from the species viscosity,

$$\begin{aligned} k_s &= \mu_s \left(\frac{5}{2} c_{v_{t,s}} + c_{v_{r,s}} + c_{v_{ve,s}}\right) \\ k_{ve_s} &= \mu_s c_{v_{ve,s}} \end{aligned}$$

and then determine the mixture conductivity using Wilke's formula,

$$\begin{aligned} k_l &= \sum_s \frac{X_s k_s}{\phi_s} \\ k_{ve} &= \sum_s \frac{X_s k_{ve_s}}{\phi_s} \end{aligned}$$

where ϕ_s is given in equation 16.

4.3.3 Diffusion coefficient

The species diffusion coefficients are calculated using an expression from Lee [10].

$$\mathcal{D}_s = \frac{\frac{M_s}{M}(1 - c_s)\mathcal{D}}{1 - X_s}$$

The diffusion coefficient $\mathcal{D}$ is calculated from a specified Schmidt number S_c.

$$\mathcal{D} = \frac{\mu}{\rho S_c}$$

The default Schmidt number is 0.5. The diffusion coefficient for ions is generally assumed to be double the neutral species diffusion coefficient because of the existence of an electric field. This is accomplished in the code by specifying a different Schmidt number for each species ($S_{c_s} = 0.5$ for neutral species, $S_{c_s} = 0.25$ for ions). The species diffusion coefficient is then given by

$$\mathcal{D}_s = \frac{\frac{M_s}{M}(1 - c_s)}{1 - X_s} \frac{\mu}{\rho S_{c_s}}$$

The electron diffusion coefficient is often treated specially [10], but here it is calculated like any other species.

4.4 Boundary Conditions

The boundary conditions are an extremely important aspect of the problem formulation. At each boundary of the flow domain boundary conditions must be applied based on the physical nature of the boundary. The code has options for several types of boundary conditions. These boundary conditions are discussed below.

4.4.1 Free-Slip Walls

When an inviscid analysis is being performed, or when a known stream surface is chosen as a boundary of the computed flow domain, a free-slip boundary condition is specified. The boundary condition is simply that there is no flow through the wall. This means that the component of velocity normal to the surface is zero and that the component of velocity tangent to the surface is allowed to change as required by the governing equation. The presence of a free slip wall also affects the pressure and temperature fields. If the wall is curved, a non-zero pressure gradient is needed normal to the wall to turn the flow in the direction that the wall is curving. This pressure gradient may be calculated from the normal momentum equation at the wall and used to set the wall pressure. However, in most analyses, the viscous nature of the flow near actual walls is generally so significant that a no-slip boundary condition is necessary. The free-slip boundary condition then should be used only along stream surfaces, such as planes of symmetry, which are not curved. The gradients of pressure and temperature normal to free-slip walls are, therefore, assumed to be zero.

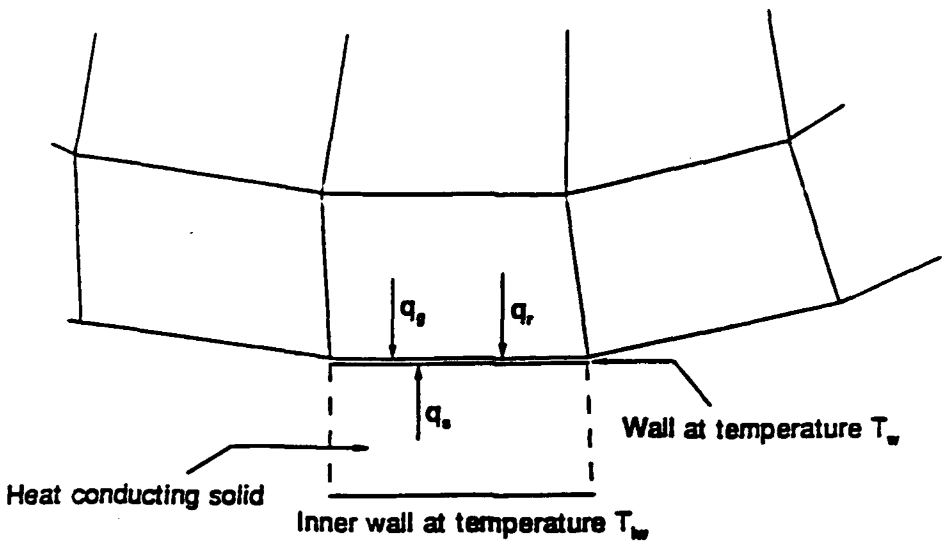


Figure 12: Types of heat transfer to the wall

4.4.2 No-Slip Walls

For flows of gases at all but the lowest densities the velocity at walls is essentially zero. Furthermore, for high Reynold's number flows, the normal pressure gradient is approximately zero deep within the boundary layer. This leaves the specification of boundary conditions for translational-rotational temperature, vibrational-electronic temperature, species concentrations, turbulent kinetic energy, and turbulent energy dissipation rate at the wall.

The wall temperatures may either be specified (isothermal) or determined from an analysis of the various heat transfers to and from the wall. For all but adiabatic walls the vibrational-electronic energies are assumed to in equilibrium with the translational-rotational energies at the wall so that $T_{ve} = T$. For adiabatic walls the gradient of the vibrational energy is zero, $\frac{\partial T_{ve}}{\partial y'} = 0$, where y' is the distance normal to the wall.

The possible forms of heat transfer to the surface of the wall are shown in Figure 12. In general they include heat conduction to the surface from the gas, heat conduction to the surface from within the solid wall, and radiation of heat away from the surface. The first law of thermodynamics requires that the sum of the heat fluxes into the surface be zero.

$$q_g + q_s + q_r = 0 \tag{17}$$

By writting q_g, q_s, and q_r in terms of the wall temperature and temperature gradient, equation 17 may be used to determine the wall temperature.

The simple adiabatic wall boundary condition is obtained when the wall is a nonradiating insulator. In this case $q_s = q_r = q_g = 0$. The heat conduction within the gas is given by Fourier's law of heat conduction, $q_g = -k\frac{\partial T}{\partial y'}$, for which the heat conduction is directly proportional to the normal gradient of temperature. If $q_g = 0$ the normal gradient of temperature is zero.

When the wall is not a perfect thermal insulator, or the wall emits or absorbs thermal radiation, the boundary condition for the wall temperature becomes more complex. Wall heat conduction can get quite complex, depending on the wall geometry and the properties of the wall material. Likewise, the radiative heat transfer to the wall can also be very complex depending on the geometry of the surface, temperatures of the gas, and composition of the gas. A complete treatment of these phenomena is beyond the scope of this research effort. However, simple approximations for the wall heat conduction and radiative heat transfer are included for the estimation of hypersonic vehicle wall temperatures. The wall heat conduction is assumed to be through a thin skin of thickness t_{wall}, uniform conductivity k_s, and no heat capacity. Furthermore, the temperature at the inner surface of the skin, T_{iw}, is assumed to be constant. The expression for the wall heat flux per unit area is then

$$q_s = -k_s \frac{T_w - T_{iw}}{t_{wall}} \tag{18}$$

The wall radiative heat transfer is assumed to be between the wall and some distant body of temperature T_{db}. Also, it is assumed that there is no interaction of the radiation with the gas. The radiative heat flux per unit area is then

$$q_r = -\sigma \left[\epsilon_w T_w^4 - \alpha_w T_{db}^4\right] \tag{19}$$

Here σ is the Stefan Boltzmann constant, ϵ_w is the emissivity of the wall, and α_w is the absorptivity of the wall. This expression excludes gas to wall radiative heat transfer which can contribute significantly to the surface heat flux balance.

Substituting the expressions for q_g, q_s, and q_r into equation 17 we get the following equation.

$$k \left.\frac{\partial T}{\partial y'}\right|_w = -k_s \frac{T_w - T_{iw}}{t_{wall}} - \sigma \left[\epsilon_w T_w^4 - \alpha_w T_{db}^4\right] \tag{20}$$

This is a differential expression for the wall temperature T_w which is coupled to the temperature field through the normal temperature derivative $\left.\frac{\partial T}{\partial y'}\right|_w$. The numerical treatment of this equation is described elsewhere in this report.

For flows with species transport, the walls are assumed to be non-catalytic so that $\frac{\partial c_i}{\partial y'} = 0$. The validity of this assumption will depend on the nature of the wall material and the degree to which the flow is out of equilibrium.

4.4.3 Rarefied Gas Slip Walls

The rarefied gas, slip wall boundary condition is included to enable analysis of hypersonic vehicles at higher altitudes than would otherwise be possible. At the high altitudes and hypersonic velocities at which many aerospace vehicles are expected to operate, the Knudsen numbers get large and the continuum assumption begins to break down. At the edge of the continuum limit, the continuum assumption fails first in a region next to the wall having a thickness on the order of the local molecular mean-free path. In this layer, called the Knudsen layer, the Navier-Stokes equations no longer apply. A rigorous treatment of the flow within the Knudsen layer would require the solution of the Boltzmann equation. However, such a procedure is beyond the scope of this work, and would likely be very expensive.

If the Knudsen layer is thin the flow field may be obtained by solving the equations of continuum flow, the Navier-Stokes equations in this case, with modified boundary conditions [14, 15, 16, 17]. In particular, a jump or slip in the wall values of species concentrations, pressure, velocity, and temperature must be allowed. The code uses simplified versions of the surface slip equations derived by Gupta, Scott, and Moss [17].

The jump relations are used for two purposes: to obtain boundary conditions at the edge of the continuum flow domain, and to obtain wall values of pressure, temperature, etc. for force integrations and thermal load estimations. For example, the pressure jump is not actually needed for the boundary condition but is needed to determine the wall pressure for force integrations. Conversely, the temperature jump is needed for the boundary condition when either the wall temperature is specified or a radiating and/or conducting wall is considered. The slip velocity relation is always used. The boundary conditions are actually the conditions at the edge of the Knudsen layer and will be indicated by the subscript s.

To remain consistent with the available no-slip boundary conditions, the rarefied-slip boundary is assumed to be non-catalytic. Therefore

$$\frac{\partial c_i}{\partial y'} = 0$$

Likewise, the pressure gradient at the edge of the Knudsen layer is assumed to be zero. The reasoning is the same as for the no-slip wall.

For the slip relations we define a local coordinate system with y' normal to the surface and x' and z' tangent to the surface. The velocity slip relationship is obtained from equations 40 and 41 in reference [17] by assuming that the derivatives along the surface are small compared to normal derivatives. The result is

$$\begin{aligned} u'_s &= \left\{ \sqrt{\frac{\pi}{2}} \frac{2-\theta}{\theta} \left[\frac{\mu}{\sqrt{kT}} \frac{\partial u'}{\partial y'} \right]_s \right\} / \sum_{i=1}^{NS} n_i^s \sqrt{m_i} && (21) \\ v'_s &= 0 && (22) \\ w'_s &= \left\{ \sqrt{\frac{\pi}{2}} \frac{2-\theta}{\theta} \left[\frac{\mu}{\sqrt{kT}} \frac{\partial w'}{\partial y'} \right]_s \right\} / \sum_{i=1}^{NS} n_i^s \sqrt{m_i} && (23) \end{aligned}$$

The normal velocity, v'_s is of course zero because the wall is impermeable.

The temperature boundary condition is different depending on whether the wall is adiabatic or otherwise. If it is adiabatic the total energy flux to the wall must be zero.

$$q_{g_s} = \left[k\frac{\partial T}{\partial y'} + \mu\left(u'\frac{\partial u'}{\partial y'} + w'\frac{\partial w'}{\partial y'}\right)\right]_s = 0 \tag{24}$$

Since the tangential velocities are no longer zero at the boundaries, the work term is not zero and the temperature derivative is not zero.

If the wall is not adiabatic, the following temperature jump equation, obtained from equation 43b of reference [17] is used.

$$\begin{aligned}\frac{T_s}{T_w} = & \left[\frac{1}{2}\left(\frac{P_y}{p^s}+1\right)\sum_{i=1}^{NS}\sqrt{\frac{2\tilde{k}T_s}{m_i}\frac{\bar{m}_s}{m_i}c_i^s}\right] / \\ & \left[-\sqrt{\pi}\frac{2-\theta}{\theta}\left(\frac{1}{2}\frac{k}{p}\frac{\partial T}{\partial y'}\right)+\frac{1}{4}\left(3\frac{P_y}{p^s}+1\right)\sum_{i=1}^{NS}\sqrt{\frac{2\tilde{k}T_s}{m_i}\frac{\bar{m}_s}{m_i}c_i^s}\right]\end{aligned} \tag{25}$$

Here P_y is the normal momentum flux, $P_y = p^s + \tau_{yy}$, which is approximately equal to the pressure, p^s, if the Knudsen Layer is thin. When the wall temperature T_w is specified, the above equation is used directly as an expression for the boundary temperature T_s. When the wall is radiating and/or conducting, the above equation is used in conjunction with the energy balance equation, equation 17. This equation is used with q_g given by equation 24, q_s given by equation 18, and q_r given by equation 19. The resulting expression is

$$-k_s\frac{T_w - T_{iw}}{t_{wall}} - \sigma\left[\epsilon_w T_w^4 - \alpha_w T_{db}^4\right] - \left[k\frac{\partial T}{\partial y'} + \mu\left(u'\frac{\partial u'}{\partial y'} + w'\frac{\partial w'}{\partial y'}\right)\right]_s = 0 \tag{26}$$

Equations 25 and 26 are two nonlinear, differential expressions for T_s and T_w. Together, they constitute the temperature boundary condition for rarefied-gas, radiating and/or conducting walls.

4.4.4 Supersonic Inflow

The supersonic inflow boundary condition requires specification of all variables by the user. This boundary condition is only appropriate where the Mach number, based on the velocity at the boundary, is greater than one.

4.4.5 Supersonic Outflow

Along a supersonic outflow boundary nothing may be specified by the user. The flow variables along such a boundary are completely determined by the upstream flow. Strictly speaking, this boundary condition is only appropriate when the Mach number, based on the velocity normal to the boundary, is greater than one. However, its use is acceptable for Mach numbers less than one when the low velocity is due to a thin boundary layer along a nearby wall.

4.4.6 Mixed Subsonic/Supersonic Outflow

The treatment of subsonic outflow boundary conditions is guided by the theory of characteristics. For a subsonic flow at the exit the $u' - c$ characteristic propagates information upstream from the boundary cell to the interior cell. In this case, one variable must be specified at the boundary cell. Currently, a constant static pressure is specified at the outflow boundary.

$$\delta p_B = 0$$

The remaining variables in the boundary cell are calculated using the four downstream running characteristic equations (thermal equilibrium and no chemistry). These equations, written in delta form, are

$$\begin{aligned}
\delta \rho_B + \frac{1}{c^2}\delta p_B &= -\frac{u'_I \Delta t|\vec{S}|}{Vol_I}\left[\rho_B - \rho_I + \frac{1}{c^2}(p_B - p_I)\right] = R_1 \\
\delta p_B + \rho c \delta u'_B &= -\frac{(u'+c)\,\Delta t|\vec{S}|}{Vol_I}\left[p_B - p_I + \rho c(u'_B - u'_I)\right] = R_2 \\
\delta v'_B &= \frac{u'_I \Delta t|\vec{S}|}{Vol_I}\left[v'_B - v'_I\right] = R_3 \\
\delta w'_B &= \frac{u'_I \Delta t|\vec{S}|}{Vol_I}\left[w'_B - w'_I\right] = R_4.
\end{aligned}$$

The above equations are five linear algebraic equations in the five unknowns $\delta\rho_B$, $\delta u'_B$, $\delta v'_B$, $\delta w'_B$, and δp_B. This system is solved directly and the boundary cell solution is updated.

When there is a secondary energy equation, for vibrational-electronic energy, the vibrational energy is simply extrapolated. The same is true of the species mass fractions and turbulence quantities (k and ϵ). This is consistent with the theory of characteristics since the vibrational energy and species transport equations corresponds to a u' characteristic.

4.4.7 Subsonic Inflow

For subsonic inflow the stagnation pressure, stagnation temperature, and flow direction are specified. These quantities are related to the static pressure and static temperature by the following equations:

$$\begin{aligned}
\frac{p}{p_t} &= \left[1 - \frac{\gamma-1}{\gamma+1}\left(\frac{V_{Tot}}{a_*}\right)^2\right]^{\frac{\gamma}{\gamma-1}} \\
\frac{\rho}{\rho_t} &= \left[1 - \frac{\gamma-1}{\gamma+1}\left(\frac{V_{Tot}}{a_*}\right)^2\right]^{\frac{1}{\gamma-1}}
\end{aligned}$$

$$\begin{aligned}\frac{u}{V_{Tot}} &= \left(\frac{u}{V_{Tot}}\right)_0 = \text{specified}\\ \frac{v}{V_{Tot}} &= \left(\frac{v}{V_{Tot}}\right)_0 = \text{specified}\\ \frac{w}{V_{Tot}} &= \left(\frac{w}{V_{Tot}}\right)_0 = \text{specified}\end{aligned} \qquad (27)$$

The first two equations above are simply the isentropic relations written in terms of the total velocity, V_{Tot}, and the speed of sound at a sonic throat, a_*. The speed of sound at a sonic throat is calculated from the specified stagnation temperature.

$$(a_*)^2 = \frac{2R}{\gamma+1}\frac{p_t}{\rho_t}$$

Equations 27 are a system of five equations in five unknowns: p, ρ, u, v, and w, but one of the last three equations is redundant. To complete the system another equation is needed. This is to be expected since there is an upwind running characteristic carrying information out of the flowfield interior to the inflow boundary.

The last equation to close the system is the characteristic relation carrying information upstream to the inflow boundary.

$$\frac{\delta p}{\delta t} - \rho c\frac{\delta u'}{\delta t} = (u'-c)\left[\frac{\delta p}{\delta x} - \rho c\frac{\delta u'}{\delta x}\right]$$

This equation is forward differenced in time.

$$\delta p_B - \rho c\delta u'_B = \frac{(u'-c)\,\Delta t}{\Delta x}\left[p_I - p_B - \rho c(u'_I - u'_B)\right]^n \qquad (28)$$

The subscripts I and B indicate the first interior cell and the inflow boundary cell, respectively. The prefix δ indicates the forward in time difference of the variable following it.

The number of unknowns is reduced to three if the isentropic relations, the first two of Equations 27 are written in terms of u'. This is done using the relation

$$V_{Tot}^2 = bu'^2 \qquad (29)$$

where

$$\begin{aligned} b &= \frac{1}{1+(u'/V_{Tot})^2 - (u/V_{Tot})^2 - (v/V_{Tot})^2 - (w/V_{Tot})^2}\\ &= \text{constant.}\end{aligned}$$

The modified isentropic relations are

$$\begin{aligned}\frac{p}{p_t} &= \left[1 - \frac{\gamma-1}{\gamma+1}b\left(\frac{u'}{a_*}\right)^2\right]^{\frac{\gamma}{\gamma-1}}\\ \frac{\rho}{\rho_t} &= \left[1 - \frac{\gamma-1}{\gamma+1}b\left(\frac{u'}{a_*}\right)^2\right]^{\frac{1}{\gamma-1}}.\end{aligned} \qquad (30)$$

The modified isentropic relations, Equations 30, and the discrete form of the upstream running characteristic relation, Equation 28 are three algebraic relations in three unknowns.

The isentropic relation for pressure, the first of Equation 30, may be placed in delta law form by considering incremental changes in the variables p and u'.

$$\delta p_B = p_t \frac{2b\gamma}{\gamma+1} \frac{u'}{a_*^2} \left[1 - \frac{\gamma-1}{\gamma+1} b \left(\frac{u'}{a_*} \right)^2 \right]^{\frac{1}{\gamma-1}} \delta u'_B \tag{31}$$

This equation and Equation 28 are solved for u'. The pressure and density are then obtained from the isentropic relations, Equations 30. The velocities are also calculated from u'_B using Equation 29 and the last three of Equations 27. The specification of the flow within the subsonic inflow boundary cell is then complete for flow without chemistry.

For flow with chemistry, the mass fractions of the species are also specified at the inflow boundary. The species densities are then calculated from the specifies species mass fractions and the calculated total density. At subsonic inflow boundaries the vibrational/electronic modes of internal energy are assumed to be in equilibrium with the translational/rotational modes. Therefore the vibrational temperature is set equal to the translational temperature and the energies are calculated accordingly.

When the two-equation turbulence model is used, the turbulent kinetic energy k *and dissipation rate* ϵ *are specified by the user.*

5 Navier-Stokes Solution Procedure

The Navier-Stokes equations are solved using an LU-SGS implicit finite-volume method which is based on work by Yoon[18, 19]. This type of algorithm has proven to be a robust and efficient relaxation procedure for steady state flow calculations. The algorithm used in this investigation is a combination of the algorithm presented by Peery and Imlay[20] and Yoon's algorithm. A detailed description of the algorithm is given in the following subsections.

5.1 Internal Grid Cells

An individual finite volume cell, with indices i, j, and k, is shown in Figure 13. Applying the integral equations in this volume gives

$$\frac{d}{dt}(U_{i,j,k}Vol_{i,j,k}) = -(D_i\vec{P}\cdot\vec{S} + D_j\vec{P}\cdot\vec{S} + D_k\vec{P}\cdot\vec{S}) + N_{i,j,k}$$

where $U_{i,j,k}$ is the mean value of U in cell i, j, k and $D_i\vec{P}\cdot\vec{S}$, for example, represents the difference of the fluxes through opposing faces of the cell.

The time derivative is approximated using backward in time differencing:

$$\frac{Vol_{i,j,k}}{\Delta t}\delta U_{i,j,k} = -(D_i\vec{P}\cdot\vec{S} + D_j\vec{P}\cdot\vec{S} + D_k\vec{P}\cdot\vec{S}) + N_{i,j,k} \tag{32}$$

where

$$\delta U_{i,j,k} = U^{n+1}_{i,j,k} - U^{n}_{i,j,k}$$

For the approximation of the flux through a surface the inviscid and diffusion terms of the flux vector are considered separately.

$$\vec{P}\cdot\vec{S} = \vec{P}\cdot\vec{S}^{inv} + \vec{P}\cdot\vec{S}^{diff}$$

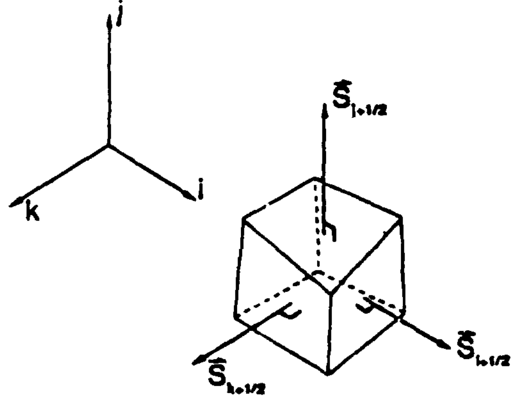


Figure 13: Finite Volume Cell

These terms are then evaluated in a manner consistent with the predominant nature of the equations in the limit as $Re \to \infty$ (hyperbolic) and $Re \to 0$ (parabolic); i.e., upwind differencing for the inviscid terms and central differencing for the diffusion (viscous stress and heat flux) terms.

The evaluation of the inviscid terms is based on the flux splitting in combination with upwind-biased MUSCL-like differencing [21]. The code currently supports two flux functions: the flux-vector-splitting of Steger and Warming[22] and the TVD fluxes of Yee[23]. The diffusion terms are evaluated using standard central differences [24]. These flux functions are described in detail in the following subsections.

The following subsections use the notation $F(U) = \vec{P} \cdot \vec{S}^{inv}$ = a function of U. This function is defined in equations at the beginning of section 4. The actual flux through a surface is given a tilde, $\tilde{F}$, to distinguish it from the function.

5.1.1 Steger and Warming Flux Function

Flux-vector-splitting was developed by Steger and Warming [22] and Van Leer [25] as a way to upwind difference the Euler equations in regions of subsonic flow. If the flow normal to the surface of a cell is supersonic, the theory of characteristics tells us that the flow field at the surface depends only on the solution upstream of the surface. In this case, the flux at the surface may be simply evaluated using the solution in the cell upstream of the surface.

$$\tilde{F}_{i+1/2,j,k} = \begin{cases} F\left(U_{i,j,k}\right) & \text{if } v' > c \\ F\left(U_{i+1,j,k}\right) & \text{if } v' < c \end{cases} \tag{33}$$

If the flow is subsonic however, the flux through the $i + 1/2$ surface depends on the solution both upstream and downstream of the surface. Then the simple upwinding applicable to supersonic flows, equation 33, is no longer appropriate. The term upwind differencing must now be defined to mean differencing in a manner appropriate with the mathematical characteristics of the flow. As shown in Figure 14, there are characteristics running both to the right and left in subsonic flow and upwind differencing must therefore use backward differencing on some of the flux and forward differencing on the rest of the flux.

In flux-vector-splitting methods, the flux function is written as

$$F(U) = F^{+}(U) + F^{-}(U) \tag{34}$$

so that the flux-split Jacobians, $\hat{A}^{\pm} = \frac{\partial F^{\pm}}{\partial U}$, have eigenvalues satisfying

$$\begin{aligned} \lambda_m\left(\hat{A}^{+}\right) &> 0 \qquad \text{for all } m \\ \lambda_m\left(\hat{A}^{-}\right) &< 0 \qquad \text{for all } m \end{aligned} \tag{35}$$

Flux splitting has therefore divided a flux which cannot in general be upwinded into two fluxes which separately can be upwinded. With first order upwinding the flux

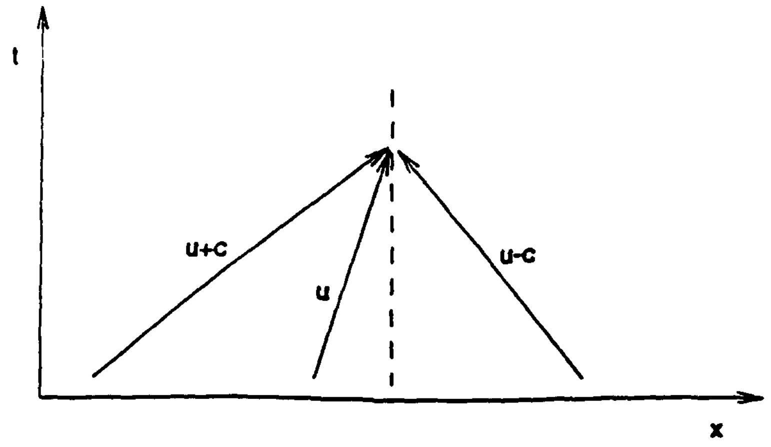


Figure 14: Mathematical characteristics of one-dimensional subsonic flow

through the $i + 1/2$ surface is

$$\tilde{F}_{i+1/2,j,k} = F^{+}\left(U_{i,j,k}\right) + F^{-}\left(U_{i+1,j,k}\right)$$

Higher order accurate methods are obtained by combined extrapolation and interpolation of the solution to the surface from nearby cells.

$$\begin{aligned} U^{-} &= U_{i,j,k} + \frac{\phi}{4}\left[(1-\tilde{\kappa})\left(U_{i,j,k} - U_{i-1,j,k}\right) + (1+\tilde{\kappa})\left(U_{i+1,j,k} - U_{i,j,k}\right)\right] && (36) \\ U^{+} &= U_{i+1,j,k} + \frac{\phi}{4}\left[(1-\tilde{\kappa})\left(U_{i+1,j,k} - U_{i+2,j,k}\right) + (1+\tilde{\kappa})\left(U_{i,j,k} - U_{i+1,j,k}\right)\right] && (37) \end{aligned}$$

Here ϕ varies the differencing between first order, $\phi = 0$, and second order, $\phi = 1$. The second parameter, $\tilde{\kappa}$, varies the differencing between fully upwind, $\tilde{\kappa} = -1$, and central, $\tilde{\kappa} = 1$. The second order flux is then

$$\tilde{F}_{i+1/2,j,k} = F^{+}\left(U^{-}\right) + F^{-}\left(U^{+}\right)$$

This form of differencing is called MUSCL differencing.

Equations 34 and 35 do not uniquely define the split fluxes $F^{\pm}(U)$. In fact, there are many flux-splittings that have been proposed [22, 25, 26]. The two most common flux-splittings are those of Steger and Warming [22] and Van Leer [25]. Both splittings were originally developed for ideal gases and were later extended for use with real gases [7, 27]. The code uses Steger and Warming flux-vector-splitting with a simple extension to multiple species transport and thermal nonequilibrium.

Steger and Warming [22] developed their splitting based on the homogeneity property, $F = AU$, of the flux vector with a thermally perfect gas. They wrote the flux

Jacobian as

$$A = A^+ + A^-$$

where

$$A^{\pm} = \mathcal{R}^{-1}\Lambda^{\pm}\mathcal{R}$$

and $\Lambda^{\pm}$ is the diagonal matrix with elements

$$\lambda_m^{\pm} = \frac{1}{2}\left(\lambda_m \pm |\lambda_m|\right)$$

Here the matrices Λ, $\mathcal{R}$, and $\mathcal{R}^{-1}$ are the diagonal matrix of eigenvalues, matrix of eigenvectors, and its inverse, for the flux Jacobian A. Therefore, A^+ and A^- are the flux Jacobian matrix with the negative and positive eigenvalues set to zero. The split fluxes are $F^{\pm} = A^{\pm}U$. This process may be performed numerically for each face on each time step, but it requires less computer time if the matrix multiplies are not required.

To eliminate the matrix multiplies, the contribution to the flux from each eigenvalue must be considered separately. There are three unique eigenvalues: $\lambda_1 = v'$, $\lambda_2 = v' + c$, and $\lambda_3 = v' - c$. The λ_1 eigenvalue is repeated so that, for flows without species transport or thermal nonequilibrium, $\Lambda = diag\,\{\lambda_1, \lambda_2, \lambda_1, \lambda_1, \lambda_3\}$. The separate contributions from each eigenvalue are obtained by defining diagonal matrices with all but the desired eigenvalue set to zero.

$$\begin{aligned}\Lambda_1 &= diag\,\{\lambda_1, 0, \lambda_1, \lambda_1, 0\}\\ \Lambda_2 &= diag\,\{0, \lambda_2, 0, 0, 0\}\\ \Lambda_3 &= diag\,\{0, 0, 0, 0, \lambda_3\}\end{aligned}$$

The fluxes are then $F_m^{\lambda} = \mathcal{R}^{-1}\Lambda_m\mathcal{R}U$. Performing the above operations for thermally perfect gas without species transport or thermal nonequilibrium gives.

$$F_1^{\lambda} = \frac{\lambda_1\rho(\gamma-1)}{\gamma}\begin{bmatrix} 1 \\ u \\ v \\ w \\ \frac{1}{2}\left(u^2+v^2+w^2\right)+\epsilon-\frac{c^2}{\gamma(\gamma-1)}\end{bmatrix} \tag{38}$$

$$F_2^{\lambda} = \frac{\lambda_2\rho}{2\gamma}\begin{bmatrix} 1 \\ u+cn_x \\ v+cn_y \\ w+cn_z \\ H+cu'\end{bmatrix} \tag{39}$$

$$F_3^{\lambda} = \frac{\lambda_3\rho}{2\gamma}\begin{bmatrix} 1 \\ u-cn_x \\ v-cn_y \\ w-cn_z \\ H-cu'\end{bmatrix} \tag{40}$$

Vinokur [7] used an alternative approach to derive the Steger and Warming fluxes and found that the above formulas also apply to real gases, provided that γ is defined $\gamma = \frac{\rho c^2}{p}$.

The split fluxes, $F^{\pm}$, are obtained from the F_m^{λ} by splitting according to the sign of the eigenvalue. For $-c < v' < 0$ we therefore have

$$F^+ = F_2^{\lambda} \qquad \text{and} \qquad F^- = F_1^{\lambda} + F_3^{\lambda}.$$

These fluxes are relatively inexpensive to calculate compared with the formulation requiring numerical matrix multiplies.

The Steger and Warming fluxes defined by equations 38 through 40 may be simply extended to handle species transport and thermal nonequilibrium. First, evaluate the fluxes with species transport equations combined into a single continuity equation having flux, F_{mass}. Then

$$F_s^+ = \frac{\rho_s^-}{\rho^-} F_{mass}^+$$
$$F_s^- = \frac{\rho_s^+}{\rho^+} F_{mass}^-$$

The same approach is used for the vibrational/electronic energy flux,

$$F_{e_v}^+ = \frac{(\rho e_v)^-}{\rho^-} F_{mass}^+$$
$$F_{e_v}^- = \frac{(\rho e_v)^+}{\rho^+} F_{mass}^-$$

and for the fluxes for the $k - \epsilon$ turbulence model. Using this approach, the additional cost of computing the fluxes for additional species transport or energy equations is minimal.

5.1.2 Harten and Yee Flux Function

In the early 1980's Yee, Warming, and Harten [23], and others [28, 29, 30] presented upwind biased schemes for nonlinear scalar equations for which the *total variation* of the solution always diminishes as the solution proceeds.

$$TV\left(U^{n+1}\right) \leq TV\left(U^n\right)$$

where

$$TV(U) = \sum_{i=-\infty}^{i=\infty} |U_{i+1} - U_i|$$

Schemes which are total variation diminishing (TVD) are guaranteed not to generate spurious oscillations and are therefore very robust for applications involving shock waves. When the same scheme is applied to nonlinear systems it is no longer TVD but it still exhibits excellent accuracy and robustness for problems involving strong shock waves. A finite-volume version of Harten and Yee's scheme is included in the code. The scheme is obtained by using the Harten-Yee flux function described in this section.

The Harten-Yee flux function is written as a central difference flux plus a dissipation term.

$$F_{i+1/2,j,k} = \frac{1}{2}\left[F_{i+1,j,k} + F_{i,j,k} + (\mathcal{R}_A\Phi_A)_{i+1/2,j,k}\right] \tag{41}$$

The dissipation operators Φ_A are defined as follows.

$$(\Phi_A)_{i+1/2} = g_i + g_{i+1} - \Psi\left(\Lambda_{i+1/2} + \Gamma_{i+1/2}\right)\alpha_{i+1/2} \tag{42}$$

where

$$\alpha_{i+1/2} = (\mathcal{R}_A)^{-1}_{i+1/2}(U_{i+1,j,k} - U_{i,j,k}) \tag{43}$$

and, in the standard, form $\Psi_{lm}(z) = |z_{lm}|$, where z is any matrix. A modified form of $\Psi(z)$ will be discussed latter in this section.

A key to the definition of the Harten-Yee flux is the eigensystem of the flux Jacobian matrix, $A = \frac{\partial F}{\partial U}$. The components of the eigensystem are the diagonal matrix of eigenvalues Λ, the matrix of right eigenvectors $\mathcal{R}$, and its inverse $\mathcal{R}^{-1}$. Expressions for these are given in appendix A. In the above equations Φ is the dissipation term which will reduce the scheme to Roe's first-order upwind scheme [31] in regions of steep gradients. The definition of the flux is completed by the relations

$$\begin{aligned} g_i &= \mathcal{S}\max\left[0, \min\left(\sigma_{i+1/2}|\alpha_{i+1/2}|, \mathcal{S}\,\sigma_{i-1/2}\alpha_{i-1/2}\right)\right] \\ \mathcal{S} &= sign(\alpha_{i+1/2}) \\ \sigma_{i+1/2} &= \frac{1}{2}\Psi\left(\Lambda_{i+1/2}\right) \end{aligned}$$

and

$$\Gamma_{i+1/2} = \begin{cases} \frac{g_{i+1}-g_i}{\alpha_{i+1/2}} & \text{if } \alpha_{i+1/2} \neq 0 \\ 0 & \text{if } \alpha_{i+1/2} = 0 \end{cases}$$

To avoid excessive roundoff error, $\Gamma_{i+1/2}$ in the above equation is actually set to zero any time $\alpha_{i+1/2}$ is near zero ($-10^{-22} < \alpha_{i+1/2} < 10^{-22}$).

This scheme is theoretically second-order accurate in space. Simply stated, the scheme is a central difference scheme where, for regions of smooth solutions, the dissipation term is turned off. Conversely, for regions with steep gradients such as shock waves, it reduces to a first order scheme to avoid aphysical oscillations in the solution. The first order scheme to which it reduces is the flux-difference splitting of Roe.

Since the Harten-Yee flux function is based on Roe's flux-difference splitting, it shares certain abnormalities with Roe's scheme. In particular, Roe's scheme does *not* necessarily satisfy the entropy inequality and may therefore yield aphysical solutions such as expansion shock waves [28, 29, 26]. A related phenomena, with serious ramifications, is the stagnation line Carbuncle described in [33]. The Carbuncle is an aphysical inward or outward bowing of the bow shock wave near the stagnation line on a supersonic blunt body. It occasionally occurs when an unmodified Roe or Harten-Yee scheme is used for hypersonic blunt body calculations. It is eliminated by modifying the function $\Psi(z)$ so that it never becomes zero. We use

$$\Psi_{mm}(z) = \frac{1}{2}\left(|z_{mm}| + \sqrt{z_{mm}^2 + \varepsilon_m}\right)$$

Far away from zero each element of Ψ approaches the absolute value of the corresponding element of z, as it should. When z_{mm} approaches zero, however, Ψ_{mm} approaches the positive number ε_m.

The values of ε_m are calculated based on user defined constants and the type of characteristics equation which corresponds to λ_m. If the characteristic transports entropy

$$\varepsilon_m = \left[d_1 + d_4\left|\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{p_{i+1,j,k} + 2p_{i,j,k} + p_{i-1,j,k}}\right|(|v'| + c)\right]$$

If the characteristic transports acoustic waves

$$\varepsilon_m = \left[d_2 + d_5\left|\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{p_{i+1,j,k} + 2p_{i,j,k} + p_{i-1,j,k}}\right|(|v'| + c)\right]$$

Finally, if the characteristic transports momentum components tangent to the surface

$$\varepsilon_m = \left[d_3 + d_6\left|\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{p_{i+1,j,k} + 2p_{i,j,k} + p_{i-1,j,k}}\right|(|v'| + c)\right]$$

Different constants were provided for each type of characteristic because, for a given problem, constants which are appropriate for one type of characteristic equation may be excessive for another type. For example, a blunt body problem may require a fairly large value of d_2 to avoid a Carbuncle but a relatively small value for d_3 to avoid excessive smearing of the boundary layer.

The cpu time for the matrix multiplies in equations 41 and 43 would normally increase quadratically with the number of transport equations. For example, the matrix multiplies would require more than four times as much cpu time for the Wray model (7 species, 11 transport equations) than for no chemistry (1 species, 5 transport equations). In this code, however, the matrix multiplies are performed analytically whenever possible and the cpu time increases nearly linearly with the number of added species.

To evaluate the variables at the cell surfaces arithmetic averaging of the cell centered values is used. This kind of averaging has the advantage of computational simplicity and can be easily extended for problems in thermochemical nonequilibrium.

Other kinds of averaging can also be used, but they are usually more complicated. Arithmetic averaging takes the form

$$\bar{\rho}_{i+1/2} = \frac{1}{2}\left(\rho_i + \rho_{i+1}\right)$$

The same procedure is used to calculated $\bar{u}$, $\bar{v}$, $\bar{w}$, and $\bar{a}$ at the surface.

5.1.3 Diffusive Fluxes

The diffusive fluxes are the contributions of the viscous stresses, heat conduction, and species diffusion to the surface flux, $\vec{P}\cdot\vec{S}$ in equation 32. These terms are obtained from

$$\vec{P}\cdot\vec{S}^{diff} = F_i^d n_i \left|\vec{S}\right|$$

where

$$F_i^d = \begin{pmatrix} +\rho_1 v_{1i}^d \\ \vdots \\ +\rho_s v_{si}^d \\ \vdots \\ +\rho_{NS} v_{NSi}^d \\ +\tau_{i1} \\ +\tau_{i2} \\ +\tau_{i3} \\ +u_j\tau_{ij} + q_i \\ q_{ve_i} \\ -\mu_k\frac{\partial k}{\partial x_i} \\ -\mu_\epsilon\frac{\partial \epsilon}{\partial x_i} \end{pmatrix}$$

and

$$\begin{aligned}
\tau_{ij} &= -\left(\mu + \mu_t\right)\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right] \\
q_i &= -\left(k_l + k_t\right)\frac{\partial T}{\partial x_i} \\
q_{ve_i} &= -\left(k_{ve} + k_t\right)\frac{\partial T_{ve}}{\partial x_i} \\
\rho_s v_{si}^d &= -\rho\mathcal{D}_s\frac{\partial X_s}{\partial x_i} \\
\mathcal{D}_s &= \frac{\frac{M_s}{M}(1 - c_s)\mathcal{D}}{1 - X_s}
\end{aligned}$$

The derivatives are approximated using standard central differences [24] centered at the cell face.

The calculation of the derivatives is the same regardless of the dependent variable, so we will present formulas for temperature T only. First the derivatives are estimated with respect to the ξ, η, ζ coordinates where ξ, η, and ζ are the coordinates running in the increasing i, j, and k direction respectively. The estimation of the temperature derivatives are given below for the $i+1/2, j, k$ surface.

$$\begin{aligned}
\frac{\partial T}{\partial \xi} &\approx T_{i+1,j,k} - T_{i,j,k} \\
\frac{\partial T}{\partial \eta} &\approx \frac{1}{2}\left[T_{i+1,j+1,k} - T_{i+1,j-1,k} + T_{i,j+1,k} - T_{i,j-1,k}\right] \\
\frac{\partial T}{\partial \zeta} &\approx \frac{1}{2}\left[T_{i+1,j,k+1} - T_{i+1,j,k-1} + T_{i,j,k+1} - T_{i,j,k-1}\right]
\end{aligned}$$

The treatment of the $j+1/2$ and $k+1/2$ surfaces is very similar. Then a transformation is performed to obtain the derivatives with respect to the x, y, z coordinates.

The transformation of the derivatives is performed as follows.

$$\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial T}{\partial \xi} \\ \frac{\partial T}{\partial \eta} \\ \frac{\partial T}{\partial \zeta} \end{bmatrix}$$

When the 3x3 matrix above is inverted,

$$\begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}^{-1},$$

the elements of the resulting matrix may be approximately evaluated in terms of surface area projections and volumes.

5.1.4 Implicit Treatment — Point Implicit

The flow equations with finite-rate chemistry are stiff (difficult to solve) because of the wide range of time scales involved. Certain reactions may have time constants orders of magnitude smaller than any flow characteristic time ($\tau = L/V$). As a result, explicit methods are unstable at any reasonable time step and implicit methods are required. Unfortunately, our experience is that implicit methods are also adversely affected by the chemical stiffness and that unacceptably small time steps must be taken. Add this to the fact that the chemical species transport equations dramatically increase the cost of an implicit time-step and you have a very inefficient scheme.

One standard way to overcome the stiffness of the chemical source terms is to treat the chemical source terms implicitly and the cell fluxes explicitly. The discrete flux balance is then

$$\frac{Vol_{i,j,k}}{\Delta t}\delta U_{i,j,k} = -\left(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S} + D_k\,\vec{P}\cdot\vec{S}\right)^n + N_{i,j,k}^{n+1} \tag{44}$$

where n is the current time step at which the solution is known and $n+1$ is the new time step at which the solution is being calculated. It is standard to linearize the implicit source term

$$N_{i,j,k}^{n+1} \approx N_{i,j,k}^{n} + \left(\frac{\partial N}{\partial U}\right)_{i,j,k}^{n} \delta U_{i,j,k}$$

so that the discrete flux balance equation becomes

$$\left[\frac{Vol}{\Delta t} - \left(\frac{\partial N}{\partial U}\right)^n\right]_{i,j,k} \delta U_{i,j,k} = -\left(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S} + D_k\,\vec{P}\cdot\vec{S}\right)^n + N_{i,j,k}^{n} = -R_{i,j,k}^{n} \tag{45}$$

The Jacobian of the source terms is an $N \times N$ matrix, where N is the number of transport equations. Equation 45 therefore represents a coupled system of N linear algebraic equations which must be solved for each finite-volume cell on each time step. The resulting method is called a ***block point implicit*** method since it requires the solutions of linear systems with $N \times N$ block matrices. If the number of species is large, the solutions of these systems can become the most expensive part of the procedure.

The cost of the block point implicit method may be reduced somewhat by taking advantage of the natural sparseness within the source Jacobian matrix. In particular, four rows of this matrix, those corresponding to the momentum and total energy equations, are always zero. For example, for laminar thermochemical nonequilibrium flows

$$\frac{\partial N}{\partial U} = \begin{bmatrix} \frac{\partial w_1}{\partial \rho_1} & \cdots & \frac{\partial w_1}{\partial \rho_{NS}} & \frac{\partial w_1}{\partial m_1} & \frac{\partial w_1}{\partial m_2} & \frac{\partial w_1}{\partial m_3} & \frac{\partial w_1}{\partial E} & \frac{\partial w_1}{\partial \epsilon_v} \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial w_{NS}}{\partial \rho_1} & \cdots & \frac{\partial w_{NS}}{\partial \rho_{NS}} & \frac{\partial w_{NS}}{\partial m_1} & \frac{\partial w_{NS}}{\partial m_2} & \frac{\partial w_{NS}}{\partial m_3} & \frac{\partial w_{NS}}{\partial E} & \frac{\partial w_{NS}}{\partial \epsilon_v} \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial w_{ve}}{\partial \rho_1} & \cdots & \frac{\partial w_{ve}}{\partial \rho_{NS}} & \frac{\partial w_{ve}}{\partial m_1} & \frac{\partial w_{ve}}{\partial m_2} & \frac{\partial w_{ve}}{\partial m_3} & \frac{\partial w_{ve}}{\partial E} & \frac{\partial w_{ve}}{\partial \epsilon_v} \end{bmatrix} \tag{46}$$

Utilizing the sparseness of this matrix, the cost of solving the linear system of equation 45 is little more than the cost of solving an $(NS+1)x(NS+1)$ linear system.

The cost of solving the linear system may be further reduced by using atom conservation equations to eliminate additional lines of the matrix. For example, Park and Yoon [34] used the conservation of oxygen and nitrogen atoms to reduce the size of the linear system for non-ionized air by two equations. This reduced the size of the system from $6x6$ to $4x4$, thereby reducing the computational effort for the system solution by two thirds: from 191 floating point operations to 62 floating point operations. This approach to matrix reduction could, with some difficulty, be extended to more complex reactions such as Parks ionizing air reaction model [11] or Jachimowski's hydrogen air combustion model [35]. Unfortunately, the savings is not as great for models with a large number of species (ionizing air) as it is for models with few species (non-ionizing air). Furthermore, this approach is very difficult to implement with general user-specified chemistry. The code therefore uses a different, more general, approach to reducing the cost of solving the linear system.

Two things have been done to improve the efficiency of the block point implicit method. The first is to automatically reduce the reaction rates when they are excessive. The second is to reduce the cost of the system solution by replacing the source Jacobian matrix with a diagonal approximation to the source Jacobian matrix.

The stiffness is greatly reduced by limiting the reaction rates so that the cell Damkohler number does not get too large. The cell Damkohler number, D_a, is the ratio of the characteristic flow time (time for flow to cross cell) to the characteristic reaction time (time for reaction to be $1/e$ completed).

$$D_a = \frac{\tau_f}{\tau_r}$$

If D_a is much larger than unity the model is attempting to complete the reaction in less time than it takes for the flow to cross the cell. In steady state flows this means that the model will attempt to complete the reaction in less than a cell width. We cannot accurately calculate any flow feature in less than 2 cell widths so there is no advantage in letting the reactions progress that fast. In cases where the reaction rates calculated from the Arrhenius equations give D_a greater than a user specified value, $D_{a_{max}}$, we simply reset the reaction rates so that D_a equals the $D_{a_{max}}$. The backward rates are modified by the same factor as the forward rates so that the equilibrium solution is unchanged. Numerical studies have shown that there is no effect on the accuracy of the final solution if $D_{a_{max}} > 10$.

The second modification of the block point implicit method is to replace the true Jacobian of the species source terms with an approximate diagonal Jacobian [55]. With a diagonal Jacobian, the solution of the linear system given by equation 45 reduces to N scalar divisions; very inexpensive compared to the solution of even the reduced block linear system. Two approximate forms for the source Jacobian were studied. The first approach uses the spectral radius of the true source Jacobian as the diagonal term of the approximate source Jacobian. Finding the eigenvalues of $\frac{\partial N}{\partial U}$ can be difficult so we actually use the L2-norm of the matrix. This gives a conservative estimate of the spectral radius.

The above approach is equivalent to rescaling the time step for the species transport equations so that the fastest reaction is explicitly stable. This is stable but may converge slowly since, when fast reactions are present, it can dramatically slow down the convection of mass. This is particularly true when one reaction is much faster than the others but does not involve any of the dominant species. In these cases it is much more efficient to rescale the time steps for each species independently.

The second approach bases the diagonal term for each species continuity equation on only the reactions affecting that species. This is done by replacing the diagonal elements of the source Jacobian matrix by the L2-norm of the elements along the row. This approach converges quicker than the uniform rescaling but it does not insure atom conservation in the unsteady solution. Atoms *are* conserved in the steady state solution however. This latter approach is currently implemented here.

5.1.5 Implicit Treatment — LU-SGS

The flux vector is a function of the solution in nearby cells. Consider only the $i+1/2$ surface.

$$\vec{P}\cdot\vec{S}_{i+1/2} = f(U_{i-1,j,k}, U_{i,j,k}, U_{i+1,j,k}, U_{i+2,j,k}) \tag{47}$$

The flux is evaluated at the new (unknown) time level and linearized using Yoon's approximate Jacobians.

$$\vec{P}\cdot\vec{S}^{\,n+1}_{i+1/2} \approx \vec{P}\cdot\vec{S}^{\,n}_{i+1/2} + A^{+}_{i+1/2,j,k}\,\delta U_{i,j,k} + A^{-}_{i+1/2,j,k}\,\delta U_{i+1,j,k} \tag{48}$$

where

$$\begin{aligned}\tilde{A}^{+}_{i+1/2,j,k} &= \frac{1}{2}\left(A_{i,j,k} + \rho(A_{i,j,k})\right),\\ \tilde{A}^{-}_{i+1/2,j,k} &= \frac{1}{2}\left(A_{i+1,j,k} - \rho(A_{i+1,j,k})\right),\end{aligned}$$

A is the flux Jacobian, and $\rho(A)$ is the spectral radius (maximum eigenvalue) of A. Substituting equation 48 and the corresponding flux for the $j-$ and $k-$direction faces into the discrete conservation equation, equation 32, yields the implicit finite-volume equation. This equation may be written as three steps:

1. Calculate the residuals for each cell using an explicit flux balance.

$$R_{i,j,k} = -(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S} + D_k\,\vec{P}\cdot\vec{S})^n + N^n_{i,j,k} \tag{49}$$

   The residuals will approach zero as the solution approaches steady state.

2. Calculate the change in the solution by solving the block-linear system of algebraic equations. The rows of this system are the block-linear algebraic relations resulting from the implicit flux balance applied to each cell.

$$\begin{aligned}&\frac{Vol_{i,j,k}}{\Delta t}\delta U_{i,j,k} - \frac{\partial N_{i,j,k}}{\partial U_{i,j,k}}\delta U_{i,j,k}\\ +\;& D_i\left[A^{+}_{i+1/2,j,k}\,\delta U_{i,j,k} + A^{-}_{i+1/2,j,k}\,\delta U_{i+1,j,k}\right]\\ +\;& D_j\left[B^{+}_{i,j+1/2,k}\,\delta U_{i,j,k} + B^{-}_{i,j+1/2,k}\,\delta U_{i,j+1,k}\right]\\ +\;& D_k\left[C^{+}_{i,j,k+1/2}\,\delta U_{i,j,k} + C^{-}_{i,j,k+1/2}\,\delta U_{i,j,k+1}\right] = R_{i,j,k}\end{aligned} \tag{50}$$

Because of the choice of Jacobians, this resulting block matrix multiplying $\delta U_{i,j,k}$ is approximately diagonal when the source Jacobian, $\frac{\partial N_{i,j,k}}{\partial U_{i,j,k}}$, is zero.

3. Update the solution using

$$U_{i,j,k}^{n+1} = U_{i,j,k}^{n} + r\delta U_{i,j,k}$$

where r is a relaxation factor. The code starts with a user specified r near one and updates the solution. If the solution at time level $n+1$ results in negative pressures or temperatures the relaxation factor is halved and the solution at $n+1$ is recalculated. This process is continued until all pressures and temperatures are non-zero or r reaches a specified minimum.

The above algorithm consists of three steps which must be done on each time step. The first step is an explicit step which calculates the residual from the local variation of the solution. The second step implicitly calculates the change in the solution according to global variations in the residual. The last step updates the solution.

The second step is the most difficult because it requires the solution of a large sparse system of linear equations. The LU-SGS algorithm approximately solves this system using two sweeps of a point Gauss-Seidel relaxation. Assume the iteration sweeps first in the direction of increasing i, j, k indices and then in the direction of decreasing i, j, k indices. The increasing i, j, k iteration is performed by applying the equation

$$\begin{aligned}&\left[\left(\frac{Vol}{\Delta t} + r_A + r_B + r_C\right) I - \left(\frac{\partial N}{\partial U}\right)_{i,j,k}\right] \delta U_{i,j,k}^{*} \\ &- A_{i-1/2,j,k}^{+}\delta U_{i-1,j,k}^{*} - B_{i,j-1/2,k}^{+}\delta U_{i,j-1,k}^{*} - C_{i,j,k-1/2}^{+}\delta U_{i,j,k-1}^{*} = R_{i,j,k} \qquad (51)\end{aligned}$$

at each internal cell. The decreasing i, j, k iteration is performed by applying the equation

$$\begin{aligned}&\left[\left(\frac{Vol}{\Delta t} + r_A + r_B + r_C\right) I - \left(\frac{\partial N}{\partial U}\right)_{i,j,k}\right] \delta U_{i,j,k} \\ &+ A_{i+1/2,j,k}^{-}\delta U_{i+1,j,k} + B_{i,j+1/2,k}^{-}\delta U_{i,j+1,k} + C_{i,j,k+1/2}^{-}\delta U_{i,j,k+1} \\ &- A_{i-1/2,j,k}^{+}\delta U_{i-1,j,k}^{*} - B_{i,j-1/2,k}^{+}\delta U_{i,j-1,k}^{*} - C_{i,j,k-1/2}^{+}\delta U_{i,j,k-1}^{*} = R_{i,j,k} \qquad (52)\end{aligned}$$

This scheme may be written as an approximate lower-upper (LU) factorization by subtracting equation 51 from equation 52 and rearranging to remove $\delta U_{i,j,k}^{*}$ from the left hand side.

$$\left[\left(\frac{Vol}{\Delta t} + r_A + r_B + r_C\right) I - \left(\frac{\partial N}{\partial U}\right)_{i,j,k}\right] \delta U_{i,j,k}$$

$$+ \quad A^{-}_{i+1/2,j,k}\delta U_{i+1,j,k} + B^{-}_{i,j+1/2,k}\delta U_{i,j+1,k} + C^{-}_{i,j,k+1/2}\delta U_{i,j,k+1}$$
$$= \quad \left[\left(\frac{Vol}{\Delta t} + r_A + r_B + r_C\right) I - \left(\frac{\partial N}{\partial U}\right)_{i,j,k}\right] \delta U^{*}_{i,j,k} \tag{53}$$

This equation replaces equation 52. It yields the same results as equation 52 but requires fewer floating point operations and less memory. The diagonal wavefront algorithm is used in the solution of the Lower and Upper systems to allow vectorization or parallelization (see section 3).

When the source Jacobian, $\frac{\partial N}{\partial U}$, is zero the LU-SGS scheme is very efficient because the main block (i.e. the matrix multiplying $\delta U_{i,j,k}$) is diagonal and easy to invert. When there are chemical reactions the source Jacobian, given by equation 46, is not zero. Therefore, the resulting matrix is not diagonal and would be more expensive to invert. We therefore use the approximate source Jacobians discussed in the previous section to diagonalize the main block for chemically reacting flows.

5.2 Boundary Conditions

The various boundary condition options available in the code are described in section 4.4. These boundary conditions are in the form of algebraic or differential relationships for the flow variables on the boundary. The boundary conditions are satisfied numerically by setting the values of the flow variables in the boundary cells so that the boundary conditions are satisfied on the surface between the boundary cell and the first internal cell.

5.2.1 Free-Slip Walls

The zero gradient in pressure and temperature boundary conditions are easy to implement. The pressure and temperature within the boundary cell is simply set to the values in the first internal cell. The impermeable wall condition is set by reflecting the velocity vector about the surface. Mathematically,

$$\vec{V}_{boundary\ cell} = \left(\vec{V} - 2\vec{V}\cdot\vec{n}\right)_{internal\ cell}$$

where $\vec{V}$ is the velocity vector and $\vec{n}$ is the unit surface normal vector. The average velocity at the surface is therefore tangent to the surface.

5.2.2 No-Slip Walls

For no-slip walls the easy boundary conditions are for velocity and pressure. The velocity in the boundary cell is set equal to the negative of the velocity in the interior cell so that the velocity at the wall is zero. The pressure in the boundary cell is equal to the value in the first internal cell. The wall temperature depends on whether the wall is adiabatic, isothermal, or radiating/conducting.

If the wall is adiabatic (and nonradiating), the temperature gradient in the gas at the wall is zero. This is satisfied by setting the boundary cell temperature equal to the temperature in the first internal cell.

If the wall is isothermal, the temperature at the wall is specified. This is satisfied by extrapolating the temperature from the internal cell and wall into the boundary cell. Then when the boundary cell and internal cell temperatures are averaged the desired wall temperature is obtained. The boundary cell species densities are calculated from the temperature, pressure, and zero gradient in species mass fractions.

The most complicated case is when the radiating/conducting wall option is chosen. In this case the wall temperature is given by the complicated expression, equation 20. For numerical reasons it is often desirable to slow down the change in wall temperature during the initial transients, so we add a heat capacity (per unit volume), c_{hc}, to the wall. For simplicity, the heat capacity is lumped at the surface so that the expression for the heat conduction into the wall doesn't change. The resulting expression for the wall temperature is

$$c_{hc} t_{wall} \frac{\partial T_w}{\partial t} = q_g + q_s + q_r$$

or

$$c_{hc} t_{wall} \frac{\partial T_w}{\partial t} = -k \left. \frac{\partial T}{\partial n} \right|_w - k_s \frac{T_w - T_{iw}}{t_{wall}} - \sigma \left[\epsilon_w T_w^4 - \alpha_w T_{db}^4 \right] \tag{54}$$

The above expression is an ordinary differential equation that must be solved at each wall surface at each time step. The equation is highly nonlinear and very stiff because of the T_w^4 term in the radiative heat flux. The equation is solved using a linearized fully implicit procedure.

$$T_w^{n+1} = T_w^n + (q_g + q_s + q_r) \Big/ \left(\frac{c_{hc} t_{wall}}{\Delta t} - \frac{\partial q_g}{\partial T_w} - \frac{\partial q_s}{\partial T_w} - \frac{\partial q_r}{\partial T_w} \right)$$

When the wall capacity, c_{hc}, is set to zero this is equivalent to doing one Newton iteration on the nonlinear equation per flow solver time step.

5.2.3 Rarefied Gas Slip Walls

In section 4.4.3 expressions were given for the velocity at the edge of the Knudsen layer and temperature jump across the Knudsen layer. The actual boundary conditions for the Navier-Stokes solver are the values of the flow variables at the edge of the Knudsen layer, as indicated by the subscript s, not the values of the flow variables at the wall. In this section we describe the numerical treatment of these boundary conditions.

The first step is to find the temperature at the edge of the Knudsen layer, T_s. This temperature depends on the nature of the boundary. If the wall is adiabatic with no radiation, equation 24 holds and the temperature gradient is simply related to the slip surface velocities and velocity gradients. The velocities and gradients are evaluated from the velocities at the previous time step and the desired temperature gradient is obtained by setting the boundary cell temperature appropriately. For example, if the surface is the $j = 5/2$ boundary, then

$$T_{i,2,k} = T_{i,3,k} + \frac{\mu}{2k}\left[(u'_{i,3,k} + u'_{i,2,k})(u'_{i,3,k} - u'_{i,2,k}) + (w'_{i,3,k} + w'_{i,2,k})(w'_{i,3,k} - w'_{i,2,k})\right],$$

where u' and w' are two orthogonal components of the surface tangent velocity.

If the wall is either isothermal or radiating/conducting, T_s must be calculated using the temperature jump relation given in equation 25. The equation is rewritten as a function of the temperature ratio whose value should be zero.

$$f\left(\frac{T_s}{T_w}\right) =$$

$$\frac{T_s}{T_w}\left\{-\sqrt{\pi}\frac{2-\theta}{\theta}\left[\frac{kT_w}{p}\frac{A}{Vol}\left(\frac{T_3}{T_w} - \frac{T_s}{T_w}\right)\right] + \sum_{i=1}^{NS}\sqrt{\frac{2\tilde{k}T_w}{m_i}\frac{\bar{m}_s}{m_i}C_i^s}\sqrt{\frac{T_s}{T_w}}\right\}$$

$$-\sum_{i=1}^{NS}\sqrt{\frac{2\tilde{k}T_w}{m_i}\frac{\bar{m}_s}{m_i}C_i^s\frac{T_s}{T_w}} = 0$$

This nonlinear equation is solved using a Newton iteration.

$$\left(\frac{T_s}{T_w}\right)^{n+1} = \left(\frac{T_s}{T_w}\right)^n - \frac{f}{f'}$$

Currently, one iteration of the Newton iteration is performed per time step so that the equation is not solved completely on each time step, but is accurately solved when the solution is at steady state.

When the wall is isothermal, the wall temperature T_w doesn't change and the above procedure is used to calculate T_s, the temperature at the edge of the Knudsen layer. The temperature in the boundary cell is then determined by extrapolation. If the wall is radiating and/or there is heat conduction into the wall, the wall temperature is given by equation 26. In this case, equations 25 and 26 are a pair of coupled nonlinear algebraic equations for the T_s and T_w. The first equation is solved, as described above, using one step of a Newton iteration. For the solution of this equation the value of T_w from the previous time step is used. The second equation is then solved for T_w as described in the previous section while assuming that $\frac{T_s}{T_w}$ is constant. This procedure converges to the appropriate wall temperature and slip surface temperature at steady state.

The next step is to calculate the velocities at the edge of the Knudsen layer. These velocities are given by equations 21 through 23. The velocities are first transformed into the orthogonal surface normal coordinate system (x', y', z') where y' is the coordinate normal to the surface. The derivatives in equations 21 and 23 are then differenced. For example, for the $j = 5/2$ boundary this yields

$$\frac{\partial u'_s}{\partial y'} = \frac{2A}{Vol}\left[u'_{i,3,k} - u'_s\right].$$

The resulting algebraic expressions are solved for u'_s and w'_s to give

$$u'_s = \frac{C}{1+C} u'_{i,3,j}$$
$$w'_s = \frac{C}{1+C} w'_{i,3,j}$$

where

$$C = \left\{ \sqrt{\frac{\pi}{2}} \frac{2-\theta}{\theta} \left[\frac{\mu}{\sqrt{kT}} \frac{2A}{Vol} \right]_s \right\} / \sum_{i=1}^{NS} n_i^s \sqrt{m_i}.$$

The slip surface velocities are then transformed back to the global (x, y, z) coordinate system and the boundary cell values of velocity are determined by extrapolation.

The boundary cell values of pressure and species density are calculated as described in the previous section.

5.2.4 Supersonic Inflow

The supersonic inflow condition is satisfied by setting the flow variables in the boundary cells to the values specified by the user.

5.2.5 Supersonic Outflow

The supersonic outflow condition is satisfied by setting the flow variables in the boundary cells equal to the flow variables in the neighboring internal cell.

6 Parallel Navier-Stokes Implementation

The 3-D Navier-Stokes code "HANA" has been parallelized on the Intel Touchstone iPSC/860, Cray Y-MP, and Silicon Graphics Iris 4-D machines using domain decomposition. This development is under tasks 5 through 10 as described in section 2. The approach is essentially the same as that taken for the model code described previously in section 3.

The Navier-Stokes equations are solved using Lower-Upper Successive Gauss-Seidel relaxation (LU-SGS). The procedure uses a diagonal wavefront algorithm in which the inner most loop is over all points on the $i + j + k$ constant plane (see Figure 15). All points on this plane are independent of the other points on the plane so that they may all be updated simultaneously. This allows the code to vectorize over the inner loop.

The structure of the HANA code is shown in Figure 16. It is a multiple-zone (multi-block) code that uses multiple i, j, k ordered grids patched together at common boundaries. The hierarchical structure of the code reflects its multiple-zone nature. The main routine is a driver routine which calls subroutines to operate on zones. The main operations are "perform one iteration for a zone" and "transfer data between zones". This model was chosen because it fits nicely with the data partitioning and interprocessor communication approach required on parallel computers.

The following subsections present the specific implementation of the domain decomposition technique. Descriptions of the demonstration cases and timings on a single cpu Cray Y-MP computer are presented in the following section. Finally, comparisons are presented for the parallel efficiency, speed up and performance of pHANA on the different parallel machines.

6.1 Distributed Memory MIMD Computer

The iPSC/860 version of the pHANA code requires a host program and a node program to run. Due to the computationally demanding nature of the Navier-Stokes equations (especially for chemically reacting flows), both programs must be written in double precision format. The host program, which is usually run on the system resource manager (SRM), allocates a cube of nodes (processor-memory pairs). The host program also requires information on how each of the zones is divided in the i- and j-directions, the number of processors used, the name of the cube, the name and location of the node program, and a flag to tell the node program whether it is a new or restart calculation. The host program allows a cube to be allocated and released automatically when the program is running. It also allows the user to load a node program from the concurrent file system (cfs) by specifying the full path name of the node program. The Navier-Stokes solver is the node program which has to be loaded onto the nodes by the host program.

The node program (pHANA) was parallelized on the distributed memory computer using a permanent domain decomposition technique. The domain decomposition used in the Navier-Stokes implementation is the same as the domain decomposition for the model equation. The decomposition is limited to a "two-dimensional" decomposition, to reduce the search effort for interzone boundary conditions. The basic idea of domain decomposition is to divide the overall domain two-dimensionally (i.e., in the I-direction and J-direction only) into subdomains (subzones) which are assigned to separate processors. The K-direction is not affected by the decomposition. The number of subzones is equal to the number of processors used for the calculation. The domain decomposition is modified for the Navier-Stokes code implementation due to load balancing, faster i/o, and memory limitations. The modifications include: input/output and job management, initial conditions, and restart capabilities.

The input/output tasks, which were performed by the SRM in the model code, were moved to one of the nodes. In addition, this input/output node (processor) was dedicated to perform all job management activities. The reasons of this change were:

1. The amount of data that can be transferred from the SRM to the nodes is limited to 256 Kbytes. Node to node transfers do not have this limitation. Since the Intel Touchstone favors the passing of a few large blocks of data rather than many small blocks, several variables are combined and passed simultaneously. This reduced the overhead in establishing interprocessor communication links.

2. Dedicating a node to i/o avoids the load leveling problems seen with the model equation. I/o is generally performed by sequential code which cannot be parallelized. When one node shares i/o and solution duties, it is much slower than other nodes. Since the solutions on all nodes are synchronized, all other cpus have to wait until the node sharing i/o and solution duties is finished. Using a dedicated node for i/o and job management duties, the solver cpus have less idle time.

The second modification to the domain decomposition involved the initialization process. Previously, the mesh was input and the variables were initialized on the i/o node before the domain was split and distributed to the solver nodes The process was modified because of the 8 megabytes memory limitation of the nodes on the standard iPSC/860 machine, which leaves only 5 megabytes for data after the executable is loaded. This seriously limited the size of the problem that could be run on this computer. The modified initialization procedure is:

1. The i/o node reads in the original mesh from an external mesh file.

2. The i/o node splits the mesh into subdomains based on the specified number of divisions in the I and J-directions.

3. After the original mesh has been split into subdomains, the i/o node sends the subdomain meshes to the appropriate solver nodes.

4. The solver nodes now perform the flow field initialization based on the data received from the i/o node.

By distributing the mesh before initializing the flow variables, the memory required by the i/o node is minimize and larger cases can be calculated on the machine. However, the initial interprocessor communication increased due to the passing of the parameters defining the initial conditions.

The third modification to the domain decomposition procedure was the reading and writing of the restart file. Due to the memory limitation mentioned above. the restart file is now written in terms of the decomposed subdomain zones. When the code is restarted, the i/o node allocates just enough memory (or heap space) for one subdomain, reads the restart file for that subdomain, sends the data to the appropriate node and releases the memory. These operations are repeated until all of the subdomains in the restart file are input and sent to their nodes. This procedure allows HANA to use all of the available memory, where each cpu contributes about five megabytes of usable memory. Further, the restart files are now written in the concurrent file system (cfs) to speed up the read and write time. It appears that access time to a cfs restart file is an order magnitude faster than access time to a restart file on an SRM disk. However, the draw back of writing the restart file onto a cfs disk is that it cannot be accessed interactively.

A schematic diagram of the domain decomposition of a simple two-dimensional geometry is shown in Figure 17. The size of the subzones must be small enough to fit within the available local memory of the processor attached to that subzone. Calculations are driven using a global time step. Subzones are connected by means of interzone patches which provide communication between adjacent subzones, shown in Figure 18. This communication between the subzones on different nodes is done by copying the data from the sending subzone to an intermediate array on the same processor. This array is then sent to an intermediate array on the receiving processor and subsequently used as boundary conditions in the corresponding subzone on the receiving processor.

6.2 Shared Memory Cray Y-MP

The procedure for parallelizing on the Cray Y-MP is similar to the procedure used for the Navier-Stokes code implementation on the Intel Touchstone iPSC/860 (i.e.. macrotasked domain decomposition). A given zone is divided up automatically into subzones. The number of subzones is equal to the number of processors used in the calculation. Unlike the distributed memory MIMD version however, there is no processor dedicated for job management and i/o purposes. One of the processors is burdened with the job management and i/o responsibilities, in addition to number crunching. However, because the CRAY uses a small number of fast processors, the additional work load on this processor is in most cases negligible.

Parallelization is performed using macrotasking compiler directives. The conservative variable data for each subzone is isolated and can only be accessed the processor which is assigned to that subzone (private data). As with the Intel iPSC/860 the communication between the subzones is done using interzone patches. The boundary conditions on each subzone are copied into an intermediate array where data is shared with the adjacent processors. The macrotasking parallelization (large-grain parallelization) is done by isolating segments of the code that can be run in parallel into external subrou.ines. The job is divided into tasks to be run on different cpus by executing multiple A-level solver routines with each processing on a different subzone. The A-level solver routine is defined as external in the driver routine. To start a task, the compiler directive TSKSTART is used. Thus, TSKSTART is invoked several times with the same external routines as arguments, causing several processors to execute the solver routine in parallel. The memory which contains the data sets for each subzone is treated as private data. Upon the completion of the solver routine, the shared data (the interzone patch array) is modified by copying the updated boundary data. The L2-norm convergence level is also shared data where each processor contributes its subzone L2-norm. To avoid memory contention problems, a TSKWAIT *compiler directive is called on each time step prior to all boundary* condition updates. The boundary conditions are updated by unlocking values in the shared data area.

This domain decomposition technique is the same as the one used on the distributed memory MIMD computer. The transfer of data between subzones is different, as it is done using shared memory rather than message passing between processors.

A major problem with the Cray Y/MP is that it is not possible to obtain accurate timings of parallel runs when it is in the multiuser mode. The timings for task 3 were made with the NAS Cray YMP in single-user mode, which required the computer to be dedicated to our use. Clearly, such runs must be scheduled well in advance and must run very quickly. We were unable to get dedicated time on the NAS Cray YMP for timing of Navier-Stokes runs.

Another problem with the parallelized code on the Cray is that the domain decomposition reduces the vector lengths and, therefore, the vector efficiencies. For example, a zone which has an I-dimension of 80 may be divided into two zones with I-dimensions of 40. The zone with a dimension of 80 has a longer vector length than the zones with dimension 40 and will therefore take less than twice as long to update. The net result is in effect a reduction in parallel efficiency even without memory contention or load leveling problems. This problem may be avoided by increasing the problem size as the number of cpu increases.

6.3 Shared Memory Silicon Graphics Iris

The procedure for parallelizing on the Silicon Graphics Iris 4-D (SGI) is very similar to the procedure used for the Intel Touchstone iPSC/860 and the Cray Y-MP (i.e., macrotasked domain decomposition). Based on the number of user-specified divisions in the $i-$ and $j-$directions, a given zone is divided up automatically into subzones. No host program is necessary in this case, and one processor is responsible for the job management and i/o operations, in addition to number crunching. This does not cause a significant load leveling problem unless the number of grid points is very small.

The SGI is a shared memory MIMD type computer. The compiler parallelizes by generating threads (processes) from DO loop iterations. The DO loop is the basic work unit that is split into concurrent threads. Since parallelization on this machine is limited only to DO loop parallelization, the original thread is the master, and it controls the execution of all other threads. By splitting the A-level call loops in the driver routine, the code is executed in parallel. The C$DOACROSS compiler directive is used for the DO loop over the solver routine. The subzone number is the variable incremented in the DO loop. It is defined as a LASTLOCAL variable so that the appropriate values are sent to the slave threads. The time step number, time, and convergence level are defined to be shared. Data for each subzone is isolated and can only be accessed by one thread, thus preventing memory access collisions.

The transfer of data between subzones on different processors is through shared memory locations. The interzone communication is done by copying boundary condition data into an intermediate array. Data in this array is locked until all processors finish the time step. This is synchronized such that the boundary conditions are not updated and processors cannot start working on the next time step until all processors are finished working on the current time step. If the subdivision of the zones has a load balancing problem, or if other processes delay the execution of one of the threads, this synchronization will significantly reduce the parallel efficiency. However, synchronization is necessary to prevent a memory collision which could easily happen in a shared memory system. After the time step is finished in all subzones, data in the intermediate arrays is available to complete the transfer of data between subzones.

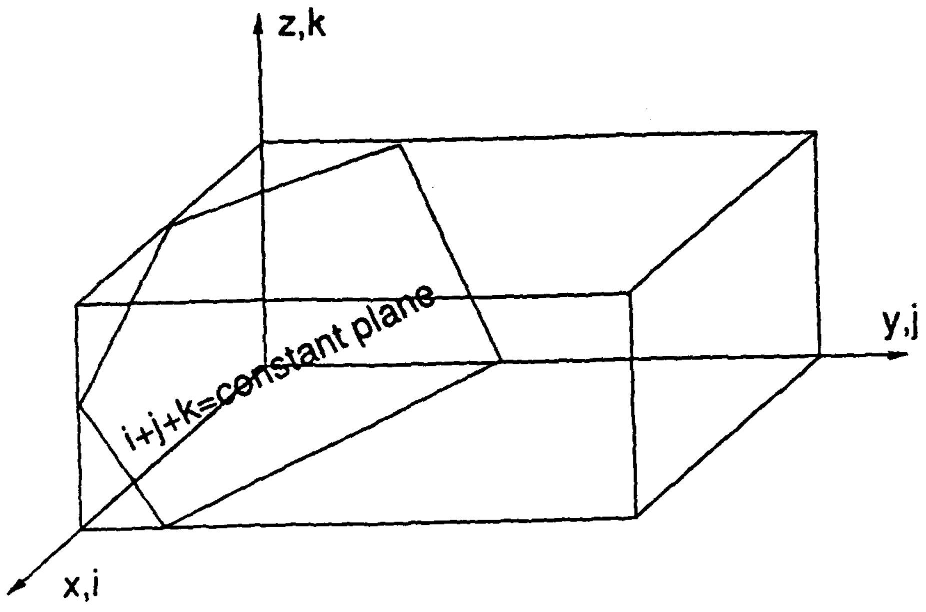
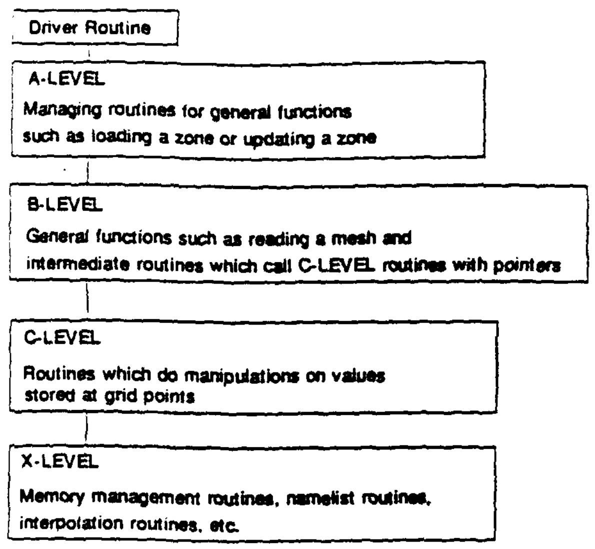


Figure 15: Diagonal Wave Plane (where $i + j + k$ is constant)



Driver Routine

A-LEVEL
Managing routines for general functions
such as loading a zone or updating a zone

B-LEVEL
General functions such as reading a mesh and
intermediate routines which call C-LEVEL routines with pointers

C-LEVEL
Routines which do manipulations on values
stored at grid points

X-LEVEL
Memory management routines, namelist routines,
interpolation routines, etc.

Figure 16: HANA code subroutine hierarchical levels

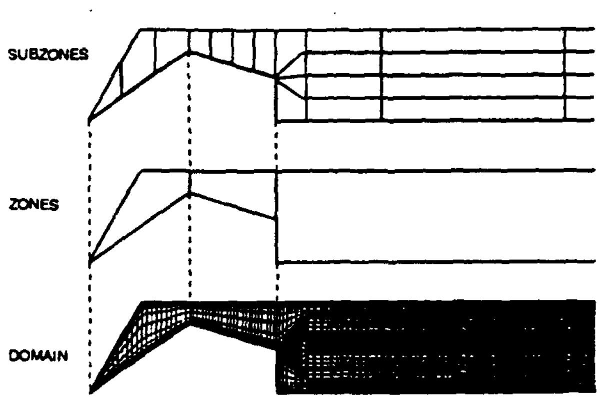


Figure 17: Domain Decomposition

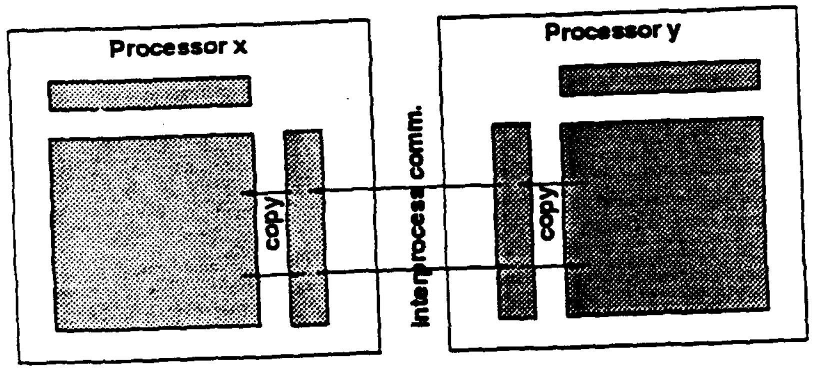


Figure 18: Interprocessor Communication

7 Test Cases

During tasks 6, 8, 10, and 11 we performed a suite of nine engineering calculations on three different computers using the parallelized Navier-Stokes code. The nine test cases are listed in Table 2 and are presented in this section. All cpu times given in this section are the single cpu Cray Y-MP time. Timings on other computers scale accordingly and the comparisons are presented in the next section.

Case No.	Description
1.	RAE-2822 airfoil
2.	Unsteady flow past a cylinder
3.	Mars penetrator with CO_2 chemistry model
4.	Hydrogen-air shock induced combustion
5.	Premixed hydrogen-air past a 10^o ramp
6.	Hydrogen-air oblique detonation ram accelerator
7.	Methane-air shock induced combustion
8.	Methane-air thermally choked and transdetonative ram accelerator
9.	Ideal gas 3-D ram accelerator

Table 2: Suite of Test Cases

7.1 RAE-2822 Airfoil

The first test case is a 2D flow field that is representative of a class of viscous-shock flow interactions commonly found on transonic vehicles. A two-zone body fitted C-mesh consisting of 104×44 and 104×44 points was used in the calculation. The 24 inch chord RAE-2822 airfoil has a sharp trailing edge, a moderate amount of aft chamber, and a 12.1% thick supercritical section. This airfoil has been widely used for code validations. The case computed is a Mach 0.73 flow at a 3.19^o angle of attack. A converged solution was achieved within 2000 iterations requiring approximately 30 minutes on a Cray Y-MP. The mesh, pressure contours and the pressure coefficient from the computed solution are shown in Figures 19, 20, and 21.

The computed solution shows a very good comparison with the experimental results. A discrepancy in the pressure coefficient is due to neglect of turbulence. Experimentally, a boundary-layer trip strip was placed at 3% chord.

7.2 Unsteady flow past a cylinder

The second test case is an unsteady Mach 0.45 flow past a cylinder at a Reynolds number of 110.000. The case is selected to demonstrate pHANA's unsteady flow

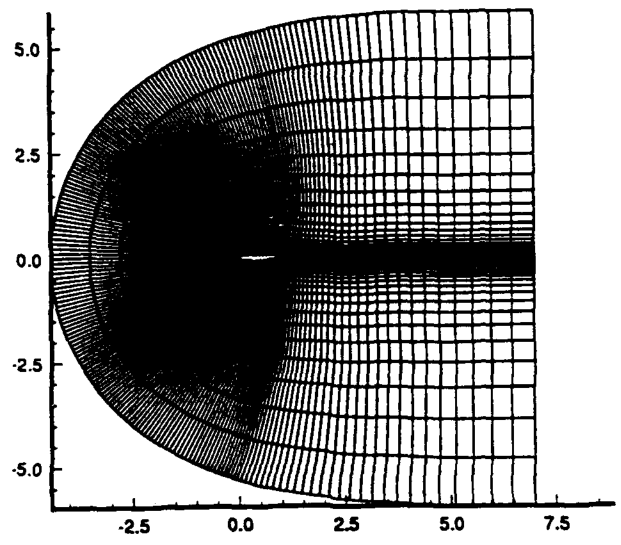


Figure 19: RAE-2822: Two-zone Mesh

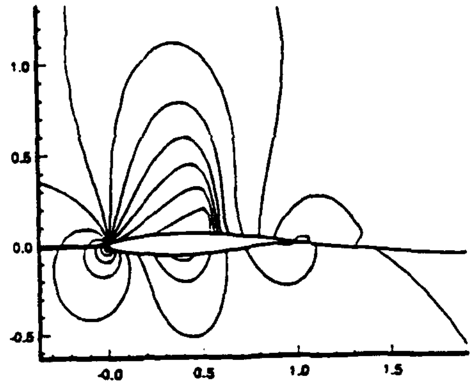


Figure 20: RAE-2822: Pressure Contours

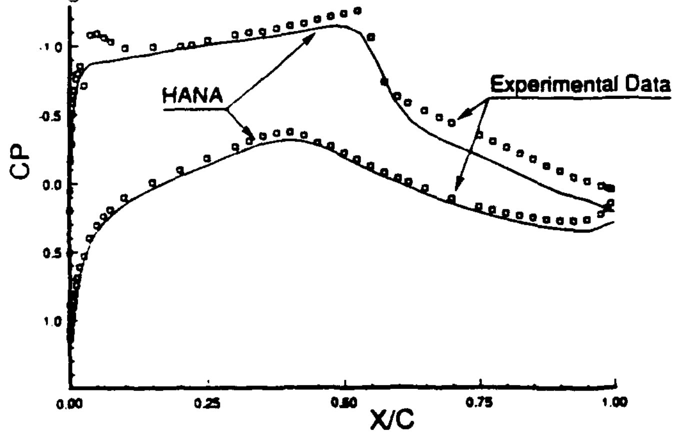


Figure 21: RAE-2822: Computed Pressure Coefficient vs. Experimental Data

analysis capability. A three-zone mesh consisting of 40×40, 40×40, and 40×96 points was used in the calculation. The calculation was performed using an explicit method and was carried over to 10,000 time steps taking approximately 20 cpu minutes. Streamlines of the results at different times are shown in Figure 22.

7.3 Mars penetrator with CO_2 chemistry model

The third case is a calculation of hypersonic flow of Martian atmospheric gas over an axisymmetric aerobrake. The geometry for this vehicle is shown in Figure 23. The calculation was done using a 50×60 single zone mesh which ignores the afterbody of the vehicle. The freestream conditions are $P = 7.36N/m^2$, $T = 149^oK$, $U = 4783m/s$, and $M = 24$. The freestream chemical mass fractions for CO_2 and N_2 are 0.9685 and 0.0315 respectively. A 1000^oK isothermal no-slip wall boundary condition is used on the surface.

The chemistry model used includes 8 species and 12 reactions [64], as shown in Table 5. The temperature used in the Arrhenius rate expression is the geometric mean of the translational and vibrational temperatures, $\overline{T} = \sqrt{TT_v}$. The Landau-Teller coefficients for the vibration/electronic energy source term are also taken from [64]. At these conditions, the effect of ionization is negligible so it was neglected in the chemistry model. A converged solution was obtained in 6000 iterations requiring about 1.0 Cray Y-MP cpu hour.

The computed temperature contours for this calculation is shown in Figure 24. The flow is energetic enough to cause significant decomposition of CO_2 but much less dissociation of N_2. As expected, the temperature peak behind the shock is visible all along the aerobrake surface.

7.4 Hydrogen-air shock induced combustion

The fourth test case is the shock-induced combustion of a Hydrogen-air mixture. Shock-induced combustion phenomena, ranging from decoupled shock-deflagration systems to overdriven oblique detonation waves, were experimentally investigated in the mid 1960's and early 1970's. Experiments mostly consisted of firing ballistic and other blunt body projectiles through explosive mixtures such as H_2/O_2 and $H_2/$Air. Calculations of this type of reaction are very demanding in terms of numerical robustness and accuracy, since the reactions usually occur very fast with significant energy release which takes place in a very short distance. Two results from Lehr's [48] experiment were modeled numerically. The two cases involve a 15mm sphere-cylinder shaped projectile moving through a stoichiometric mixture of hydrogen-air ($P_\infty = 42662Pa, T_\infty = 250^oK$) at velocities of 2605 m/s and 1685 m/s, respectively. Inviscid calculations were performed on a 72×65 grid Converged solutions were obtained within 4000 iterations or about 40 cpu minutes. The combustion model used is listed in Table 6. It includes 6 reacting species (H_2, O_2, H_2O, H, O, and OH)

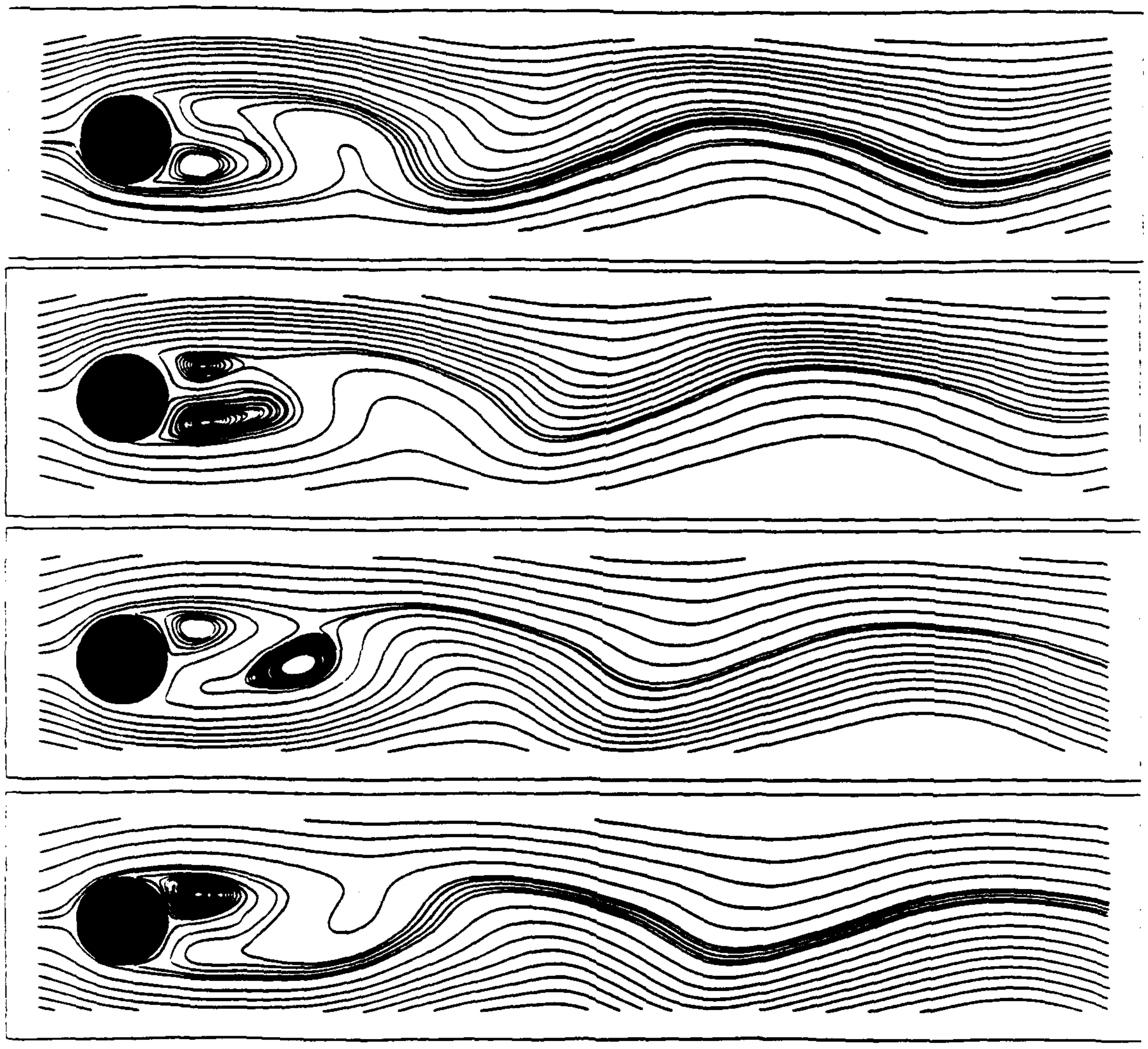


Figure 22: Streamlines - M=0.45 Past a Cylinder at Re=110,000

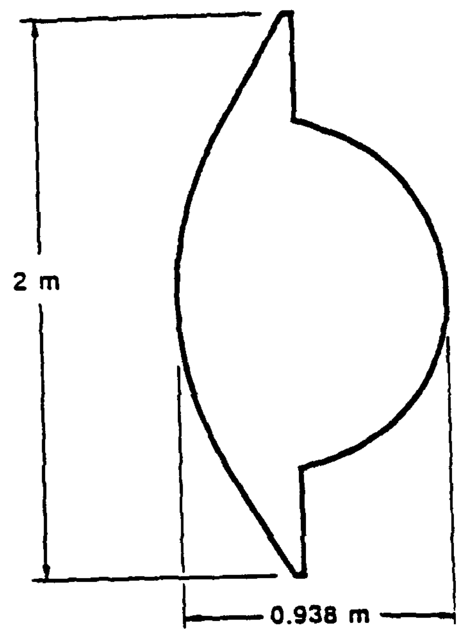


Figure 23: Geometry of the Mars Penetrator

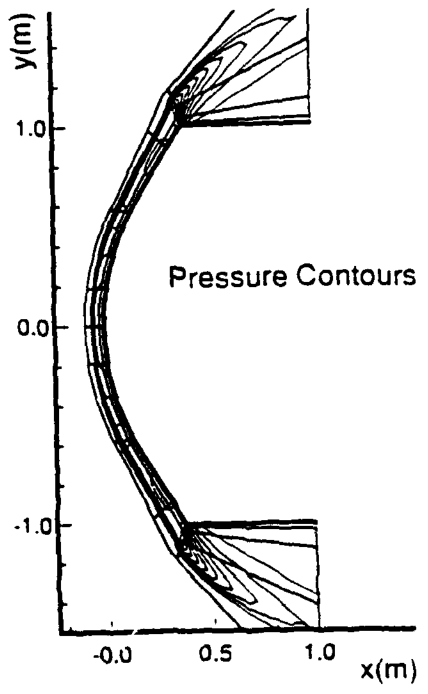
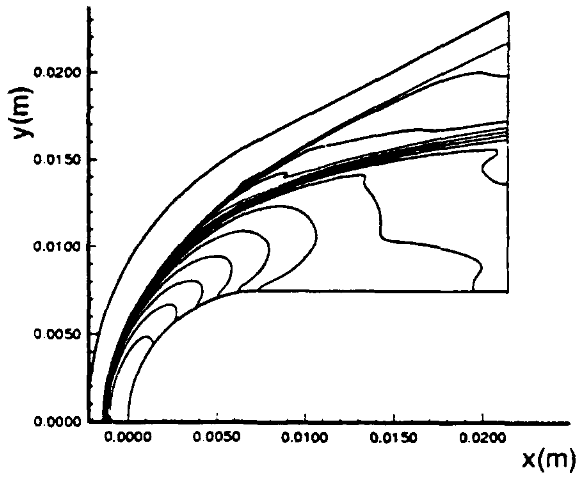


Figure 24: Temperature contours for the Mars penetrator



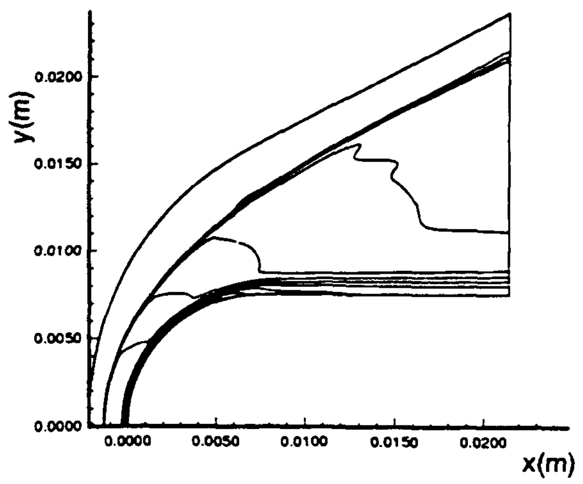


Figure 25: Hydrogen-Air Shock Induced Combustion

and 1 inert species (N_2) and is similar to the model we used in previous calculations [71, 72, 73], and the model used by Yungster [66, 67, 68].

Experimentally, Lehr found that the superdetonative case resulted in a coupled shock-deflagration system, whereas the subdetonative case yielded a decoupled shock-deflagration system. These two cases were also modeled numerically by [66] and [72]. Figures 25a and 25b show the computed temperature contours for the two cases along with experimental shadowgraph from [48]. The results correctly predict the location and the shape of the wave. In the superdetonative case, it appears that the combustion front decouples from the shock wave around the shoulder of the projectile. This phenomenon was also seen in Lehr's experimental shadowgraph. Unlike the superdetonative case, the combustion front in the subdetonative case is decoupled from the bow shock wave and is separated by an induction region. The temperature appears to be constant in the induction region and is followed by a sharp rise at the combustion front.

7.5 Premixed hydrogen-air past a 10^o ramp

In the fifth test case the HANA calculations are compared with calculations presented by Chitsomboon et. al [77] who used the Roger and Chinitz two-step combustion model. The test case involves a supersonic premixed stoichiometric hydrogen-air mixture which is ignited by the boundary layer and a shock wave induced by a 10^o compression ramp. The $M = 4$ flow was initially at 900^oK and 1 atmosphere. The same combustion model as in the previous case was used in this calculation. The shock and the boundary layer raise the temperature and start the combustion process. The calculation was performed using a 80×50 grid requiring 5000 iterations and 45 cpu minutes. Results are compared in Figures 26a through Figures 26c. The HANA results compare favorably with the results of Chitsomboon. However, our results

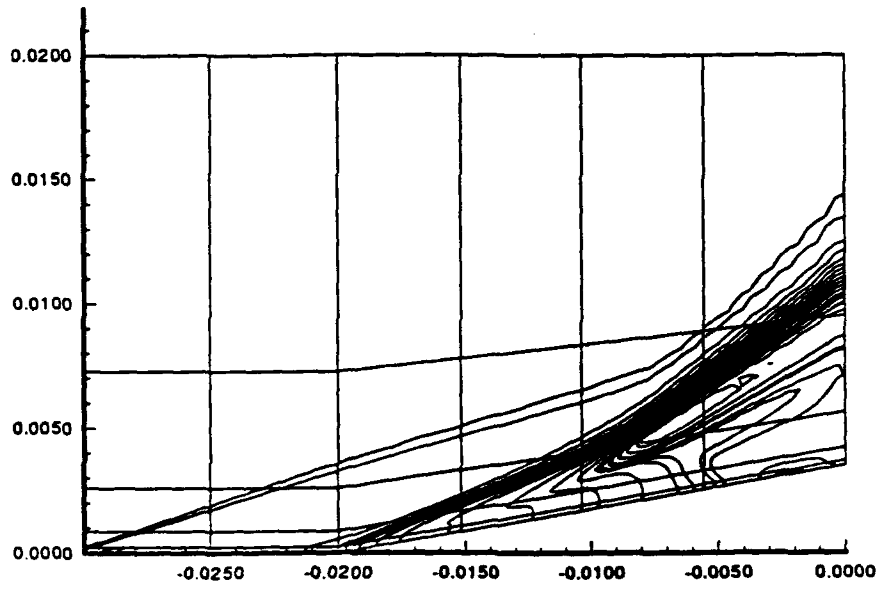


Figure 26: Hydrogen-Air Past a 10°*Ramp*

show significantly larger H_2O and smaller OH mass fractions than their results. The differences are due to the different chemistry models used. Our model yields somewhat faster and more complete reactions than the global two-step-reaction of Roger and Chinitz. The same behavior was also observed by Shuen and Yoon [70].

7.6 Hydrogen-air oblique detonation ram accelerator

The sixth test case involves the superdetonative operation mode of a ram accelerator in a stoichiometric hydrogen-air mixture. Results from the calculation are compared to Yungster's calculation [67]. The geometry is a 15 cm long projectile with a 14° cone half angle and a 1.95 cm maximum diameter inside a 3.0 cm diameter tube. The projectile is moving through a 300°K, 1 atmosphere, stoichiometric hydrogen-air mixture, and its wall is assumed to be isothermal at 600°K. The geometry and the flow parameters are similar to those used by Yungster. However, information was not available for the radius of the round corner at the transition region between the nose and the body of the projectile. A 135 x 51 mesh and a 7 species, 8 step chemistry model was used in the calculation. A converged laminar calculation is achieved within 8000 iterations requiring about 2 cpu hours. The three different Mach numbers considered are: $M = 6.7, 7.5$, and 8.0. Temperature contours for these cases are shown in Figure 27, Figure 28, and Figure 29 for $M = 6.7, 7.5, 8.0$, respectively. Water vapor mass fraction contours for these cases are shown in Figure 30, Figure 31, and Figure 32. The pressure profiles at the projectile and the tube walls for the

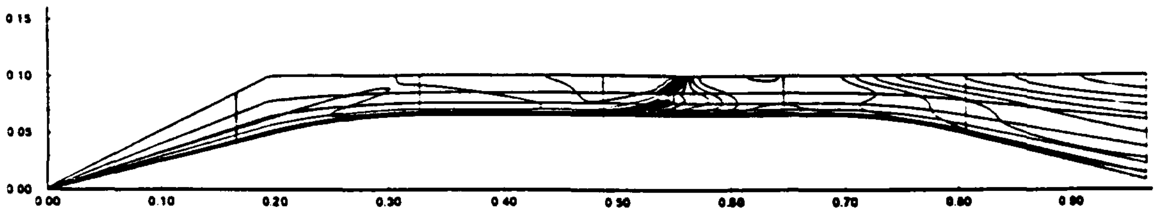


Figure 27: $M = 6.7$: Temperature Contours

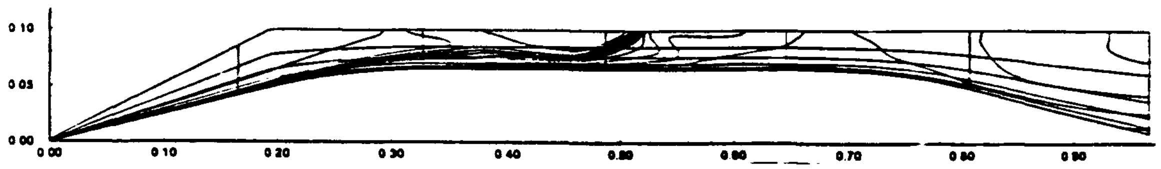


Figure 28: $M = 7.5$: Temperature Contours

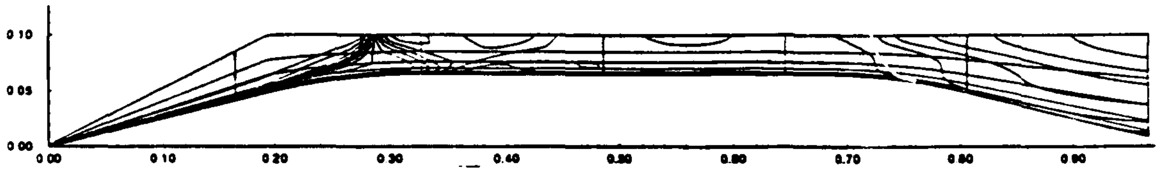


Figure 29: $M = 8.0$: Temperature Contours

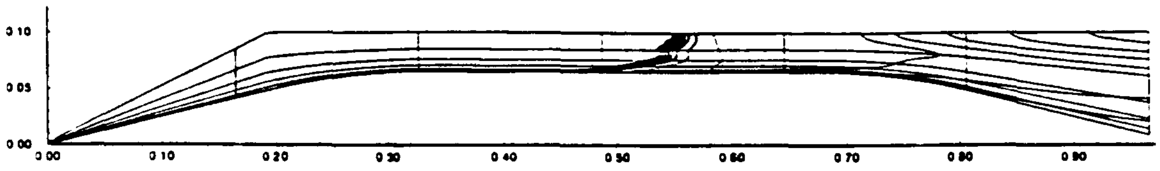


Figure 30: $M = 6.7$: Water Vapor Mass Fraction Contours

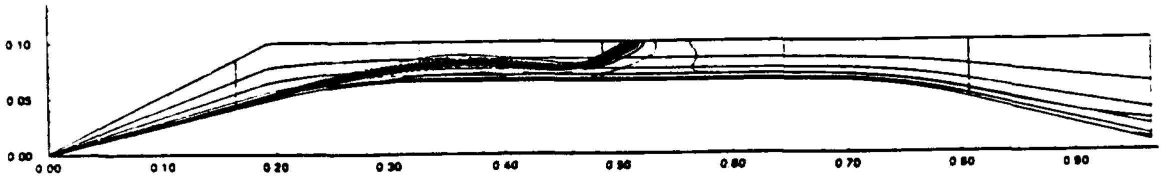


Figure 31: $M = 7.5$: Water Vapor Mass Fraction Contours

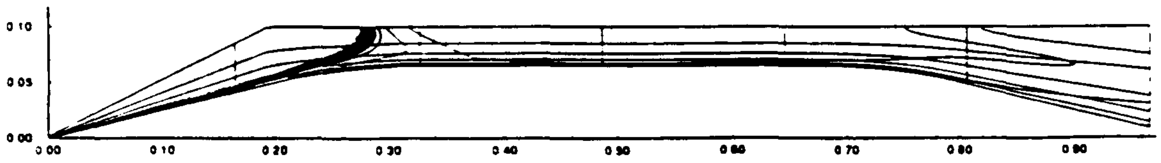


Figure 32: $M = 8.0$: Water Vapor Mass Fraction Contours

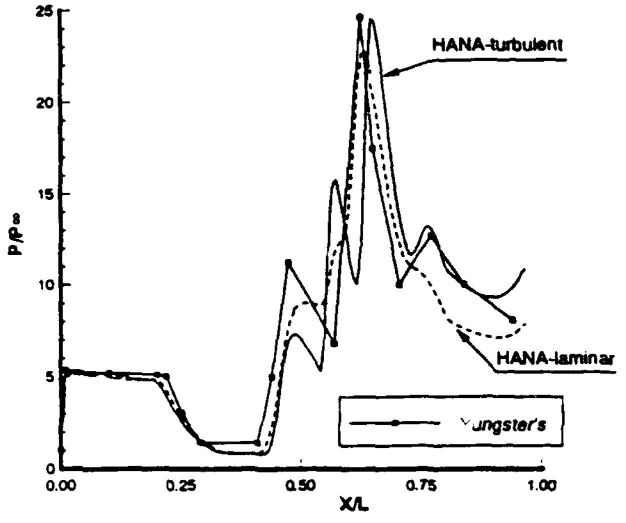


Figure 33: $M = 6.7$: Pressure Profile along Projectile Wall

$M = 6.7$ case are compared to results taken from Yungster's calculations in Figure 33 and Figure 34. As previously computed, the combustion begins in the boundary layer region, propagates outwards to fill the whole gap between the projectile wall and the tube wall, and finally establishes a shock-deflagration front. Results show good agreement with Yungster's calculation. Small discrepancies are due to the difference in the expansion region at the nose/body junction of the projectile.

7.7 Methane-air shock induced combustion

The seventh set of test cases was selected from the French-German Research Institute of Saint-Louis (ISL) shock tube experiments [80]. At ISL, the two experiments described in Table 3 were performed using a stoichiometric methane-air mixture at $P_\infty = 51000Pa$, $T_\infty = 295°K$, and $U_\infty = 2333$m/s.

Case No.	Description
1.	Blunt cylinder with different chemistry models
2.	Cylinders of various diameters

Table 3: Methane-Air Combustion Cases

Four different chemistry models were used for the calculations of the blunt cylinder, they are: a "full" model, a quasi-global, a three-step, and a single-step global combustion model. The full model was derived from several published methane-air

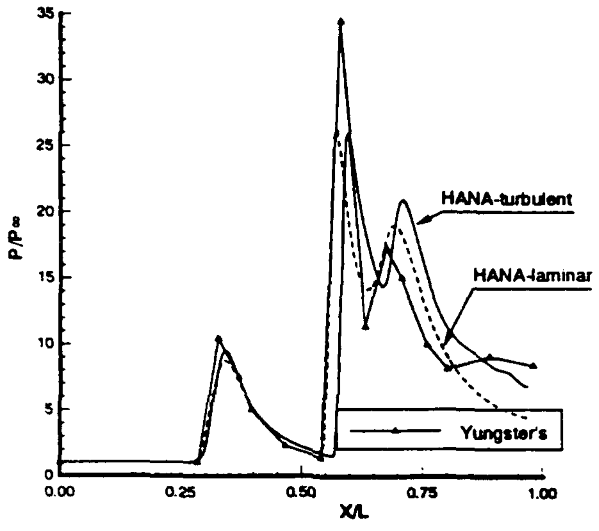


Figure 34: $M = 6.7$: Pressure Profile along Tube Wall

combustion models. There is significant variation in some of the reaction rates calculated from the published rate coefficients for the different models. The rate which gave the best solution for shock induced combustion on a blunt body was chosen. The final model consists of 13 species and 19 elementary reaction steps, see Table 7. This model is called full because it contains no global reactions. It is, however, truncated and a detailed sensitivity study was not performed to determine that all ignored reactions are insignificant. Note that for simplicity, nitrogen is assumed to be inert in this model.

The second model tested was due to Wesbrook and Dryer and is called the quasi-global reaction model. It includes 10 species and 12 reaction steps, see Table 8. The model was first used by Edelman and Fortune [82], who combined a single reaction of fuel and oxidizer with a detailed reaction mechanism of a $CO - H_2 - O_2$ system. It provides a better equilibrium temperature and a more accurate representation of concentrations of the products and reactants than the global reaction alone. The first reaction in this model is a phenomenological model and the reaction rate is given in the form of

$$k_{ov} = CT^n exp(-E/kT)\,[Fuel]^a\,[Oxidizer]^b\,.$$

A small modification was made to INCA to accommodate this form such that

$$k_f = k_{arr}\,[CH_4]^{-1.3}\,[O_2]^{-0.2}\,,$$

where k_{arr} is the standard Arrhenius form of the reaction rate. It is important to note that after some careful comparisons to the original model listed in Westbrook and Dryer [75], reactions involving H_2O_2 and HO_2 were omitted without any noticeable differences. Thus, the model was reduced from 12 species and 21 reactions to 10 species and 12 reactions.

The last two chemistry models are global models that were developed by Westbrook and Dryer [75, 76] and used in the ram accelerator calculations by Nusca [74, 81]. They are a three-step model having 7 species and a one-step model having 5 species.

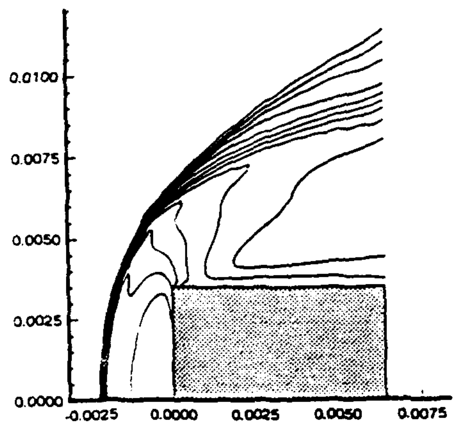
In the first case, the geometry is a 7 mm diameter blunt cylinder. Calculations were performed on a 72 × 65 mesh. Temperature contours from the laminar calculations for all four models are shown in Figures 35a through Figures 35d. Both the full model and the quasi-global model predict a coupled shock-deflagration system near the stagnation region which then decouples at the shoulder of the cylinder. The two contour plots show a remarkable resemblance to the interferometry pictures from ISL [80]. Unfortunately, a more quantitative comparison to the experimental results is not possible at present. On the other hand, the three-step and global models grossly underpredict the amount of combustion.

Figures 35e and 35f show the stagnation line pressure and temperature comparison between the full model and the quasi-global model. It appears that the two models compare favorably. However, the quasi-global model slightly overpredicts the equilibrium temperature. It is important to note that the full model used here is a significant simplification of the complete model listed in [76], and that there is much uncertainty in some of the rate coefficients. Thus, the quasi-global model appears to yield a reasonable result with about 20% saving in cpu-time compared to the full model.

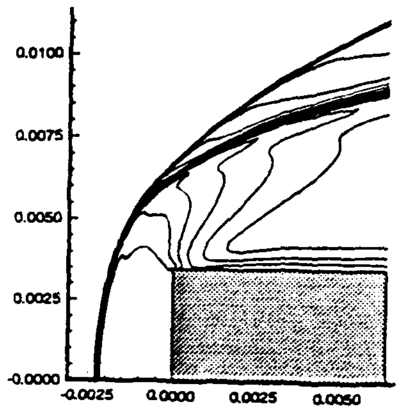
To determine the effects of turbulence on the shock and combustion front locations, the Baldwin-Lomax turbulence model was used for a calculation with the quasi-global chemistry model. The computed temperature contours are shown in Figures 35g. The shock/deflagration location is not affected by turbulence. However, the turbulence appears to enhance the combustion process behind the shoulder of the cylinder. The turbulent result is not significantly different from the laminar solution.

The second case involves two-dimensional rods of varying diameter. The flow field was modeled on a 72 × 65 two-dimensional blunt body mesh. Results from the calculations for 7mm, 3mm, 1mm, and 0.5mm diameters rods are shown in Figures 36a through 36d, respectively. Notice how a coupled shock deflagration system becomes a decoupled system as the rod diameter is reduced. However, ISL reported that there is a sharp ignition onset between the 1mm and 3mm diameter rods. This discrepancy could be due to the inaccuracy of the quasi-global model in the calculation of induction time. The position of the pressure transducers in the experiments is crucial. They may have been placed too far away from the rod itself to be able to identify the deflagration location for all rod diameters.

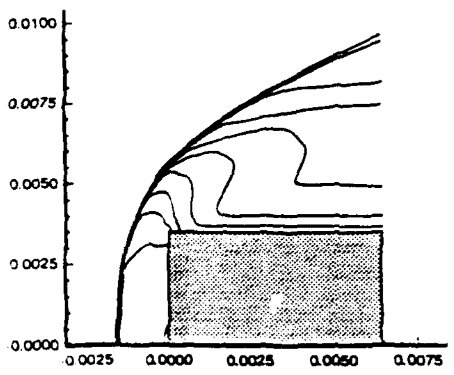
Even though the two "calibration" cases above do not show an exact agreement with the experimental findings, they show a promising trend that such a complicated combustion can be modeled using the quasi-global chemistry model.



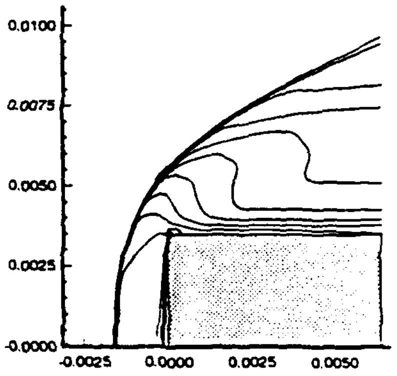
Full Model



Quasi-Global Model

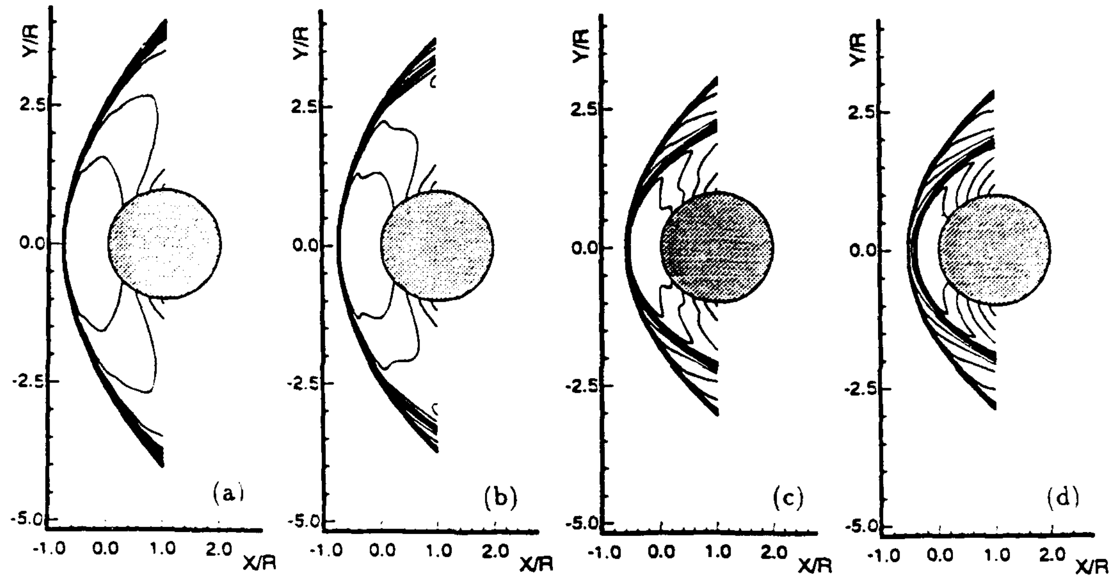


Three-Step Model



Single-Step Model

Figure 35: Blunt Cylinder with different Chemistry Models



(a) 7mm, (b)3mm, (c)1mm, (d)0.5mm

Figure 36: Rods different Diameters

7.8 Methane-air ram accelerator

The eighth set of test cases is Methane-air combusting flow about a ram accelerator operating in the thermally chocked, transdetonative, and superdetonative modes. Results are compared with experimental data from the University of Washington.

Depending on the speed of the projectiles and the Chapman-Jouget (C-J) detonation speed of the mixture, the ram accelerator operation modes can be divided into three regimes [69, 78]: the thermally choked subsonic combustion mode, the transdetonative mode, and the superdetonative mode. The thermally choked mode operates at speeds up to about 85% of the detonation speed of the mixture. This mode is usually characterized experimentally by a very sharp rise in the tube pressure profiles somewhere along the second half of the projectile body. The combustion usually occurs behind the projectile, spreads to the full tube diameter, and forces the flow to thermally choke at some distance downstream from the projectile [69]. The transdetonative operation mode is usually observed when the velocity of the projectile is 85% to 115% of the local C-J speed of the mixture [69]. In this regime, it appears that projectiles can make a smooth transition from the thermally choked mode (subdetonative speed) to the superdetonative operation mode in a single mixture [69]. The superdetonative operation mode requires that the projectile travels at a minimum speed of 115% of the mixture C-J speed [65]. The basic principles of this operation mode are similar to the oblique detonation wave engine and the scram-accelerator [79].

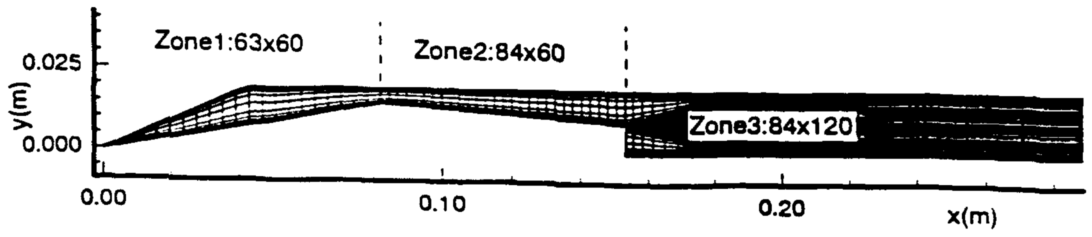


Figure 37: Ram Accelerator Mesh and Geometry

In a previous report [73], we concluded that by increasing the projectile's speed, the normal shock (which is the characteristic of the thermally choked subsonic combustion mode) moves downstream on the projectile and detaches from the projectile's body as it enters into the transdetonative operation mode. As the transdetonative mode is entered, an oblique shock appears which is attached at the base of the projectile. Since the shock strength is not sufficient to render the flow behind it subsonic over the entire domain, the flow behind this oblique shock is a mixed. Part of the combustion occurs supersonically and the rest occurs subsonically. These previous calculations, however, only considered the bulk combustion processes and neglected the effects of the boundary-layer interactions. An objective of the present calculation was, therefore, to analyze the importance of boundary layers in the ram accelerator flow fields. It is shown that the "free-slip wall" assumption can only be used in the thermally choked calculations, and boundary-layer interactions are extremely important in higher velocity operation modes.

The geometry considered is shown in Figure 37. This geometry is the actual ram accelerator geometry given in [78]. The tube diameter is 38 mm. The projectile has a 10° nose cone half-angle, a forebody length of 82.06 mm, a body length of 71.1 mm, and a base diameter of 17.8 mm. The three-zone computational domain consisted of 63×60, 84×60, and 84×120, mesh points clustered near the projectile wall and the tube wall. The free-stream mixture is $2.5CH_4 + 2O_2 + 5.5N_2$ at 31 atmospheres and $300^\circ K$. The Chapman-Jouget (CJ) detonation velocity for this mixture is 1770 m/s [78]. Calculations were performed under a laminar flow assumption. A converged solution was usually achieved within 9,000 iterations requiring approximately 10 Cray Y/MP cpu hours/case. The domain was initialized with free stream conditions with a normal shock at the base of the projectile. Behind the shock, the pressure was set to 10 times the free stream value, the temperature was set to $2500^\circ K$, and the rest of the field variables were set to free-stream conditions. The boundary conditions were: no-slip on the projectile surface, free-slip on the tube wall, a symmetry plane on the center line behind the projectile, and supersonic outflow. The combustion model for the ram accelerator calculations is the quasi-global methane-air model described in the previous section.

A total of ten runs were made to demonstrate the thermally choked and the

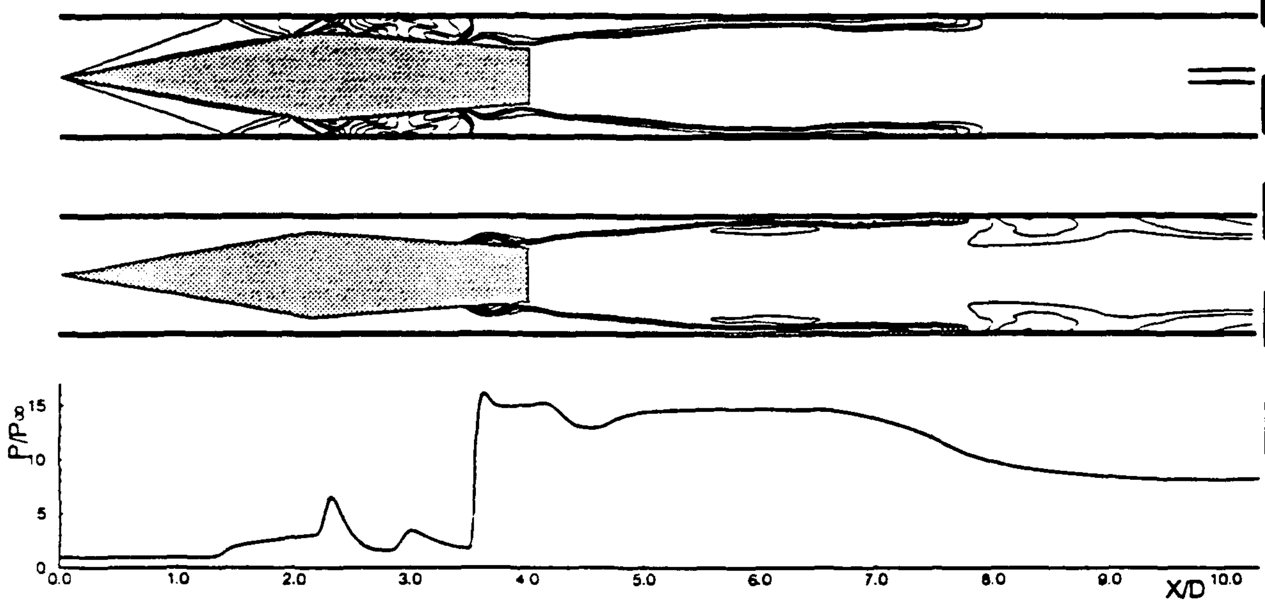


Figure 38: $M_{CJ} = 0.71$ Ram Accelerator

transdetonative operation modes of the ram accelerator. The ten runs are listed in Table 4.

Case No.	*Description*	U_∞(m/s)	U/U_{CJ}
1.	Thermally Choked	1250	0.71
2.	Thermally Choked	1350	0.76
3.	Transdetonative	1470	0.83
4.	Transdetonative	1540	0.87
5.	Transdetonative	1625	0.92
6.	Transdetonative	1685	0.95
7.	Transdetonative	1760	0.99
8.	Transdetonative	1840	1.04
9.	Transdetonative	1965	1.11
10.	Transdetonative	2015	1.14

Table 4: Ram Accelerator Cases

The computed temperature contours, and pressure profiles along the tube wall for the ten cases are shown in Figures 38 through 47, respectively. In the thermally choked mode, one can see clearly the series of oblique waves which culminate in a "normal" shock. This normal shock is stabilized on the projectile's body by the massive heat addition behind the projectile and the thermal choking point which occurs about 1 projectile length behind the base. It is important to note that the location of this normal shock is strongly affected by the amount of heat addition behind the

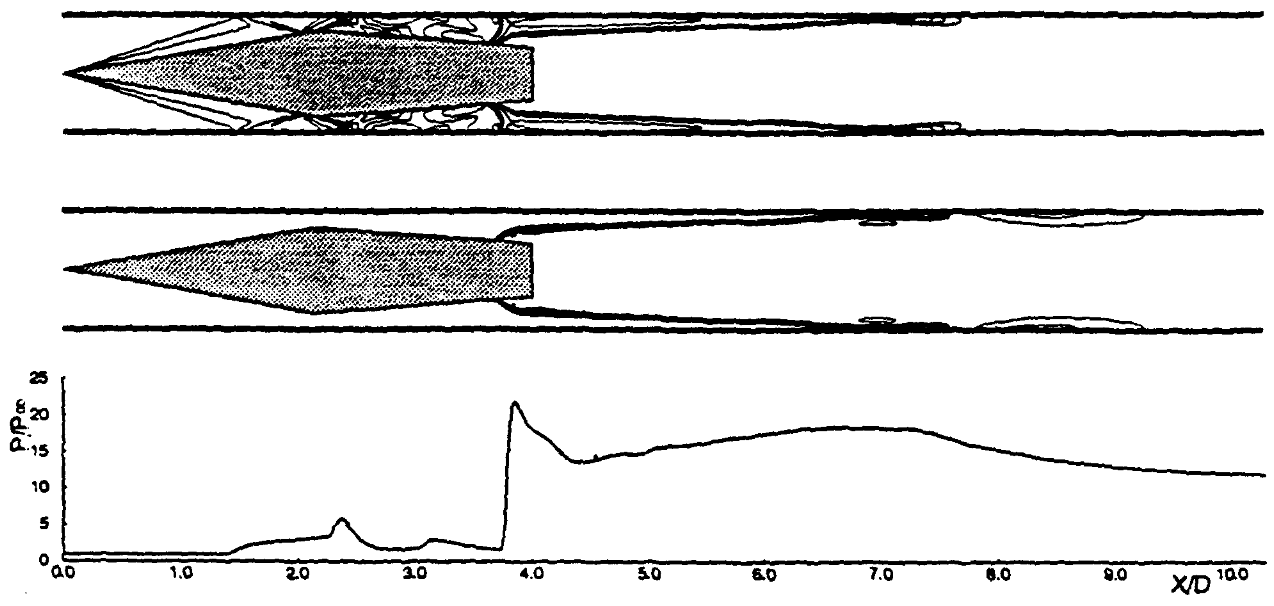


Figure 39: $M_{CJ} = 0.76$ Ram Accelerator

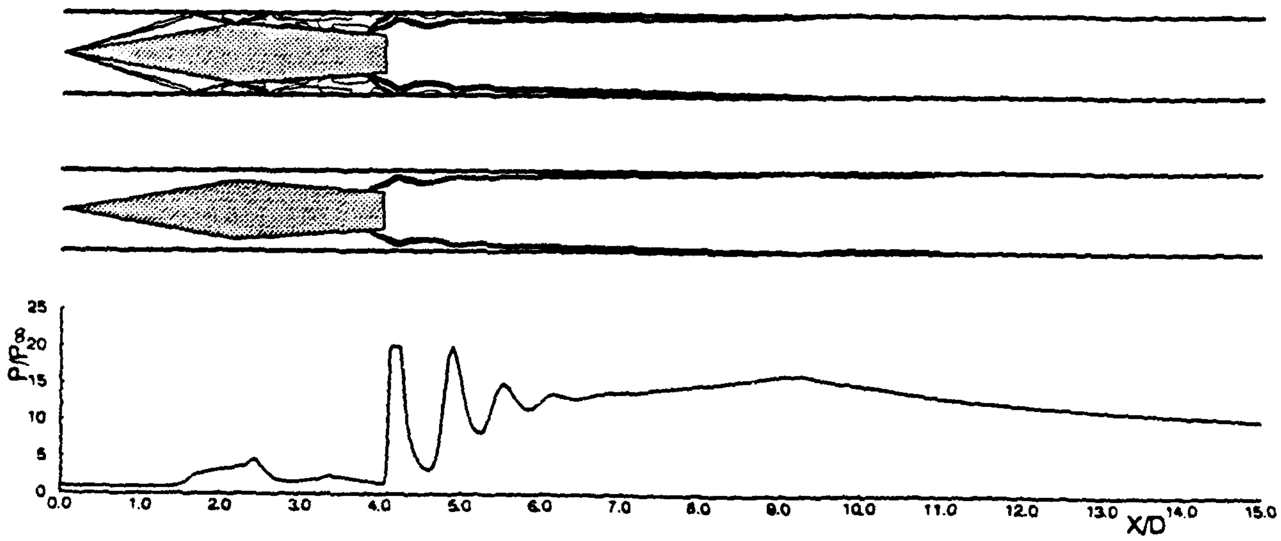


Figure 40: $M_{CJ} = 0.83$ Ram Accelerator

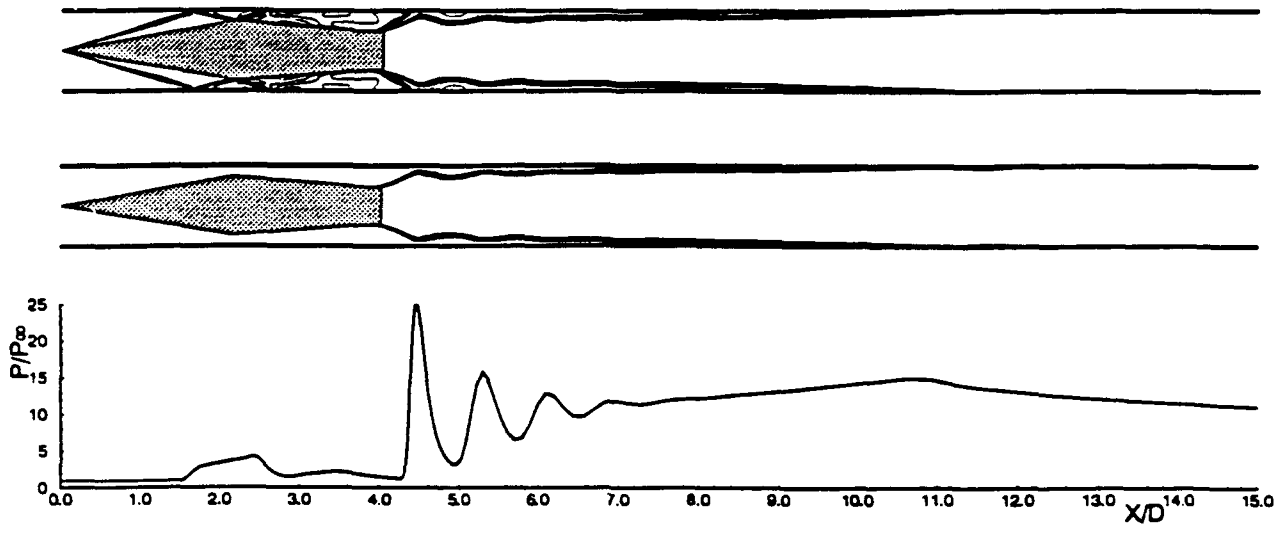


Figure 41: $M_{CJ} = 0.87$ Ram Accelerator

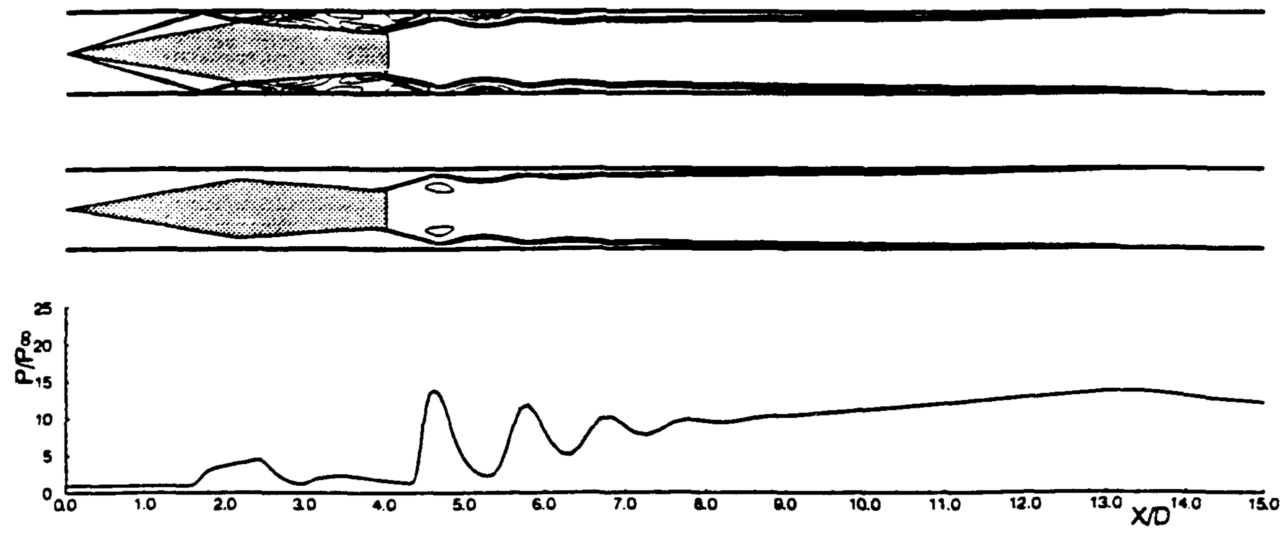


Figure 42: $M_{CJ} = 0.92$ Ram Accelerator

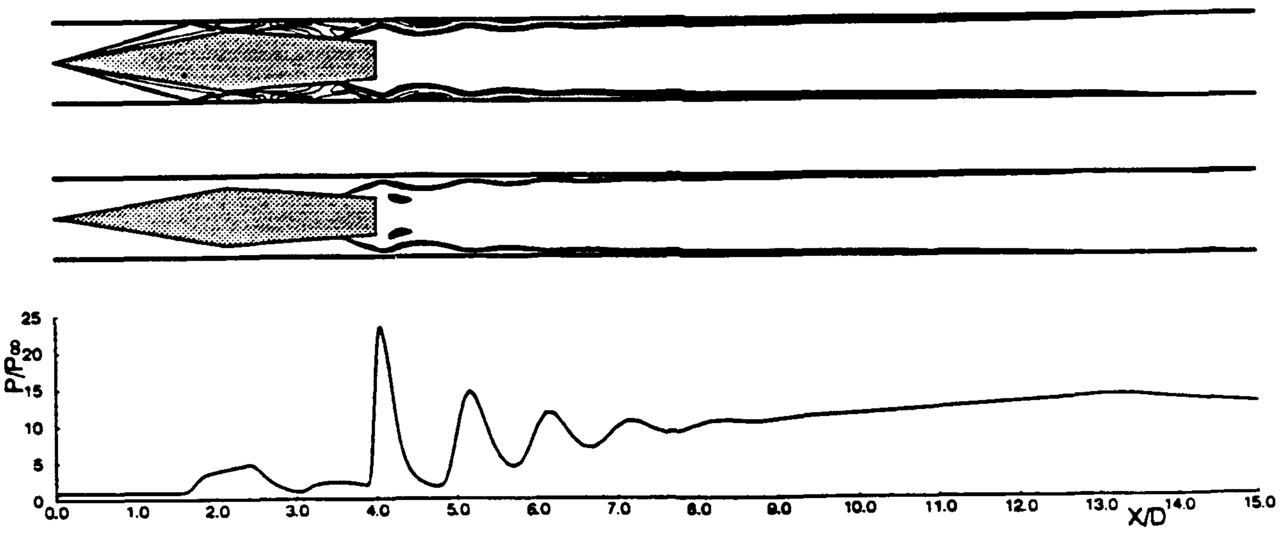


Figure 43: $M_{CJ} = 0.95$ Ram Accelerator

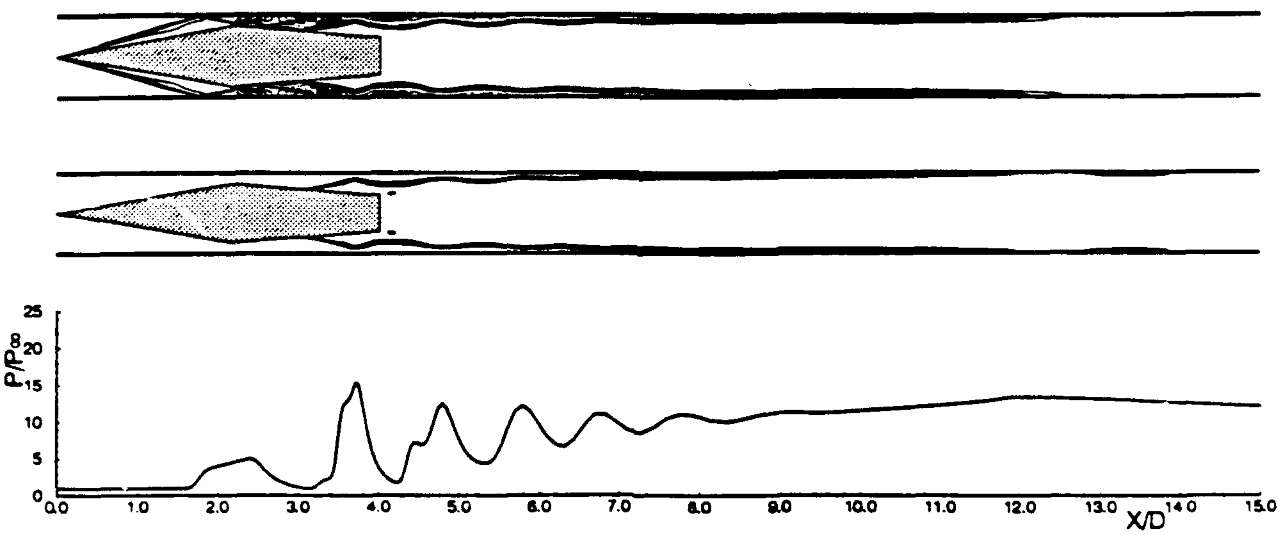


Figure 44: $M_{CJ} = 0.99$ Ram Accelerator

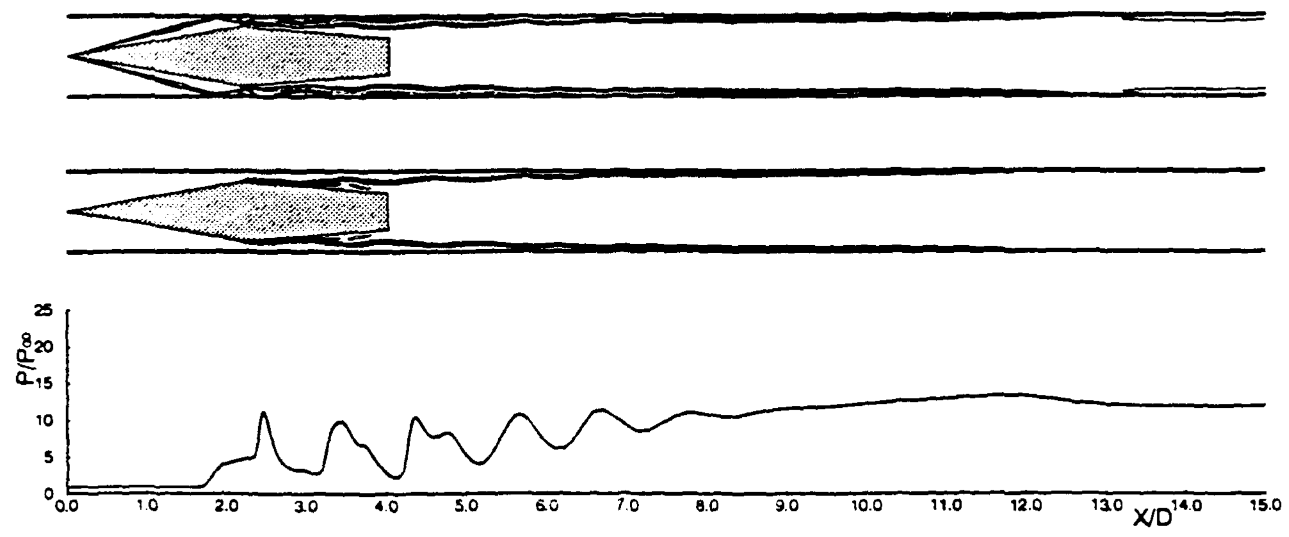


Figure 45: $M_{CJ} = 1.04$ Ram Accelerator

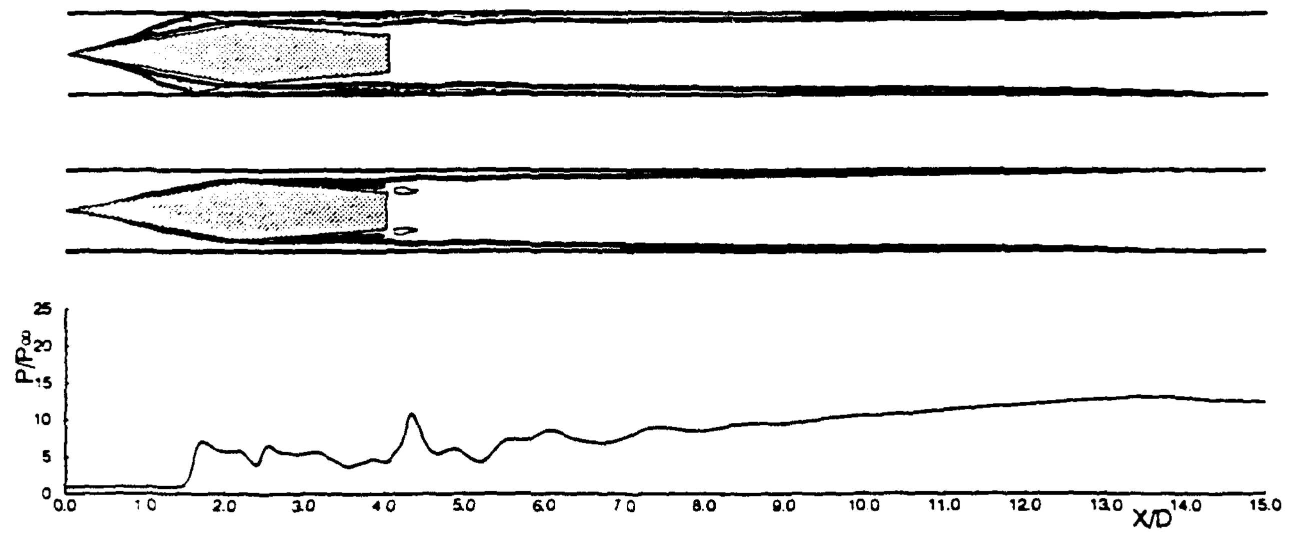


Figure 46: $M_{CJ} = 1.11$ Ram Accelerator

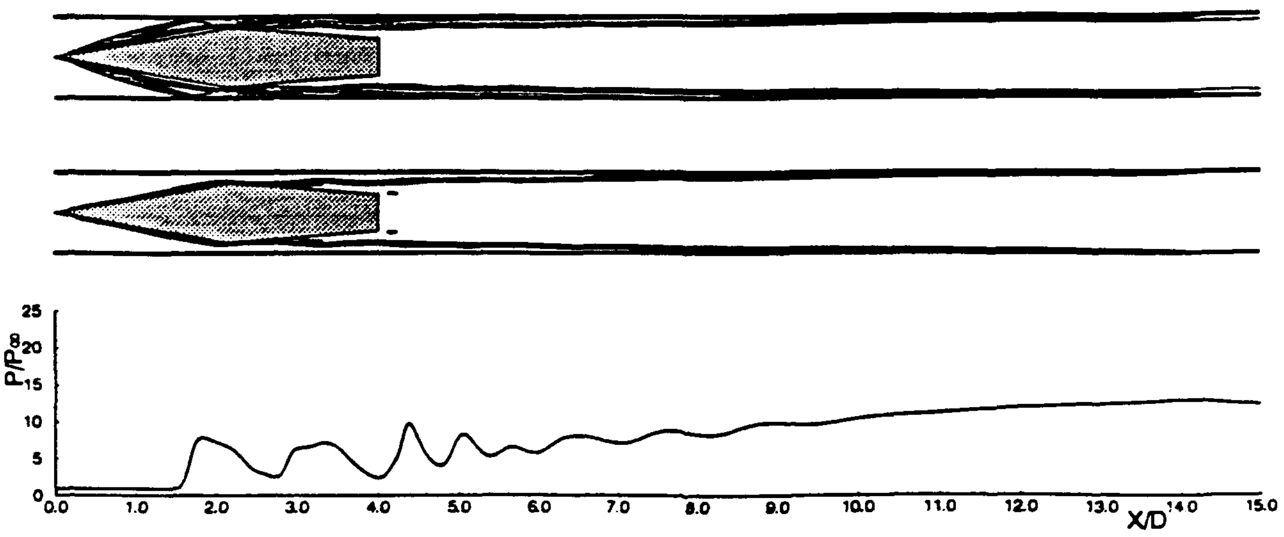


Figure 47: $M_{CJ} = 1.14$ Ram Accelerator

base. Thus, it depends on the reaction rates used in the combustion model. By using different combustion models, the unstart phenomenon has been experienced in which the normal shock and combustion front move upstream of the throat, and the projectile experiences negative thrust. A small boundary layer combustion region also appears following the normal shock. This boundary layer combustion is suspected to be due to the adverse pressure gradient behind the shock, which causes the laminar boundary layer to separate. In the separation region, the temperature and residence times are high enough to initiate combustion. A similar behavior has also been observed in the oblique detonation mode calculations of Yungster [68]. The first dip in the pressure profiles corresponds to the expansion region at the base of the projectile. Behind this expansion point, the pressure increases until it reaches a local maximum where the combustion front reaches the tube wall. Downstream of this expansion point, the flow chokes and becomes transonic. The two thermally choked calculations appear to agree with the experimental results observed at the University of Washington [78]. The base region acts as a flame holder at these velocities and there is no combustion in the boundary layer upstream of the normal shock.

In the transdetonative operation mode, the culminating shock is no longer a normal shock. It changes its characteristics and becomes an oblique shock wave. As the velocity is increased, the shock recedes along the body of the projectile. Since the shock is no longer normal, it is not sufficient to render the flow behind it subsonic over the entire tube area. The flow behind this oblique shock is mixed flow, where a full combustion is reached in a very short distance in the subsonic region but much slower in the supersonic part. This oblique shock is followed immediately by a combustion front which starts at the boundary layer. This partial combustion

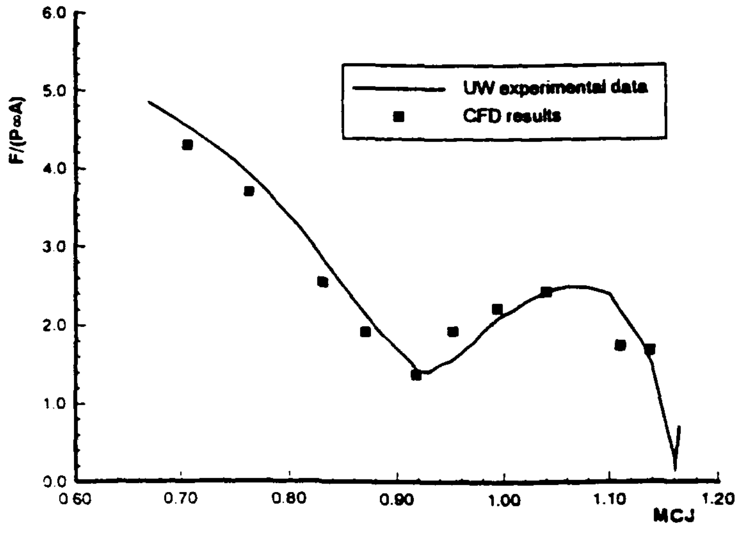


Figure 48: Thrust Coefficient of the Ram Accelerator

on the body of the projectile has also been observed experimentally. As the velocity is increased, the location of the shock/deflagration system continues to recede along the body until it reaches the projectile's base. This point coincides with the lowest thrust coefficient which occurs at $U/U_{CJ} = 0.92$. In contrast to the inviscid results [73], beyond this velocity the shock/deflagration front now moves forward on the projectile's body. This phenomenon is related to the upward sweep in the thrust coefficient. At velocities higher than 1840 m/s, combustion occurs prematurely in the boundary layer at the nose region of the projectile. This behavior has also been observed in the oblique detonation mode calculations of Yungster [68] and experimental luminosity data [69, 78, 83]. In contrast to Yungster's results however, the full tube combustion is not reached until about 2 to 4 projectile's length downstream. The location where a full tube combustion begins is also observed as the peak pressure in the tube wall pressure profiles. These locations (where a full tube combustion begins) appear to be further back than the experimental data. This discrepancy is probably due to neglected turbulence and three-dimensional effects. This downstream/upstream movement of the shock/deflagration front has also been observed by Nusca [81]. However, since his results were obtained by coupling two different codes in an iterative manner (where the flow field is partitioned by a normal shock), the culminating shock was always a normal shock.

The computed thrust coefficients for different velocities are compared to the thrust coefficient derived from the experimental observations in Figure 48. The computed thrust coefficient shows very good agreement with the experimental data. It is important to note that, after some careful comparisons to the experimental data, there are two discrepancies. First, the pressure profiles along the tube wall indicate that this shock/deflagration system has a very high peak pressure which is not observed in the experimental results. This phenomenon may be due to simplified free-slip boundary conditions applied at the tube wall. Second, the first pressure peak behind the throat

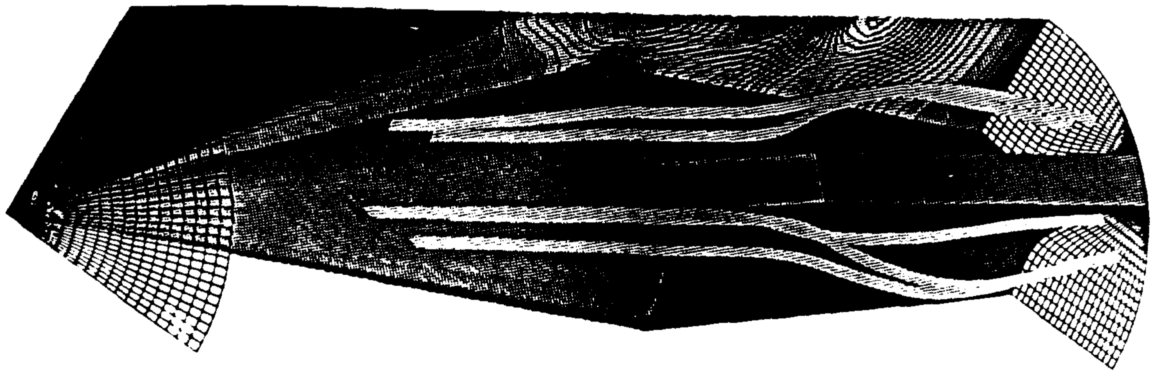


Figure 49: Mesh and stream-tubes for Ideal Gas 3-D Ram Accelerator

is lower in the computed results than that observed experimentally. This discrepancy is shown in the next section to be due to fin interactions. Fin effects might also account for indications in the luminosity data that there is combustion at the throat.

Results from this set of calculations indicate the importance of the boundary layer combustion in the transdetonative operation mode of the ram accelerator. It also gives a possible explanation of the second upward sweep in the thrust data. Finally, the overall results show a very good agreement with the experimental results and trends from the University of Washington. However, there are several aspects neglected in this calculation which are suspected to play a significant role in numerical simulations of the transdetonative operation mode. One is the effect of the highly swept fins which will generate three-dimensional shock waves and vortices along the projectile body. The three-dimensional shock waves, vortices, and boundary layer may interact and initiate throat combustion at lower velocities than predicted by the axisymmetric calculations. The three-dimensional shock and the throat combustion could also explain the underprediction of the first pressure peak downstream of the throat.

7.9 Ideal Gas 3-D Ram Accelerator

The final test case, a perfect gas calculation for a three-dimensional ram accelerator projectile, was selected to elucidate the effects of fin interactions. The projectile geometry is similar to the one used at the University of Washington [69]. However, to simplify the computation, the height of the fin is truncated slightly to allow a 3 mm gap between the fin and the tube wall. The gap in the experiment is essentially zero (approximately 0.05 mm). Further, a laminar flow and free slip walls were assumed. The free stream conditions were $P_\infty = 31$ atmospheres, $T_\infty = 300^\circ K$, and $U_\infty = 1540$ m/s. The calculation was performed with the $40 \times 20 \times 20$ single-zone mesh shown in Figure 49. The computed stream-tubes (see Figure 49) show a vortex rollup along the side of the fin. Figure 50 shows the tube wall pressure profiles at three different orientations with respect to the fin (0°, 15°, and 45°). At 0° the jump in the first pressure peak was about 30 times the free-stream pressure. This peak has never been seen in the axisymmetric calculations and is solely due to the fin effect. The jump

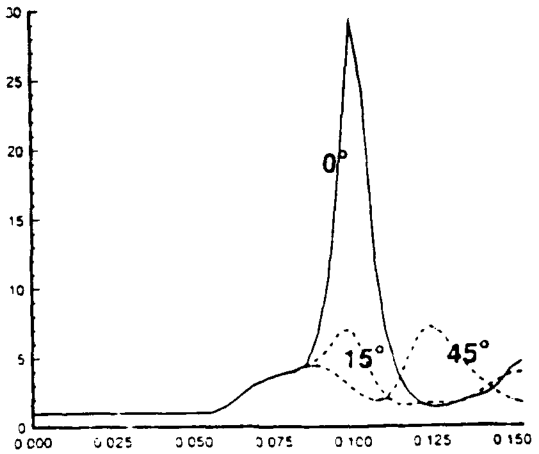


Figure 50: Tube Wall Pressure Profiles at 0°. 15°. 45° angles to the fin

is also associated with a three-dimensional oblique shock reflection which is shown in the 45° pressure trace profile as a second pressure peak. In the corresponding axisymmetric calculation, the tube wall pressure profile was similar to the one at 45° without the second pressure peak.

The oblique shock wave due to the fin interactions changes the flow features in the ram accelerator projectile flow fields. It also sheds light on the discrepancies in the comparison between the experimental pressure profiles and results from axisymmetric calculations. Unfortunately, this calculation does not give information on the effects of the shock wave on the boundary layer and combustion initiation. Obviously, the wave will raise the temperature behind it significantly which may be enough to start the combustion for some cases. More three-dimensional calculations should be performed including boundary layer effects and finite-rate combustion.

Reaction				C	n	E/k
1	$CO + M$	$=$	$C + O + M$	$4.5\text{x}10^{19}$	-1.0	128.900
2	$CO_2 + M$	$=$	$CO + O + M$	$3.7\text{x}10^{14}$	0	52.500
3	$O_2 + M$	$=$	$2O + M$	$2.0\text{x}10^{21}$	-1.5	59.500
4	$CO + CO$	$=$	$CO_2 + C$	$2.3\text{x}10^{09}$	0.5	65.710
5	$CO + O$	$=$	$O_2 + C$	$3.9\text{x}10^{13}$	-0.18	69.200
6	$CO_2 + O$	$=$	$O_2 + C$	$1.7\text{x}10^{13}$	0	26.500
7	$N_2 + M$	$=$	$2N + M$	$7.0\text{x}10^{21}$	-1.6	113.200
8	$NO + M$	$=$	$N + O + M$	$1.1\text{x}10^{17}$	0	75.500
9	$CO + N$	$=$	$NO + C$	$2.9\text{x}10^{11}$	0.5	53.630
10	$N_2 + O$	$=$	$NO + N$	$6.4\text{x}10^{17}$	-1.0	38.370
11	$NO + CO$	$=$	$CO_2 + N$	$4.6\text{x}10^{08}$	0.5	12.070
12	$NO + O$	$=$	$O_2 + N$	$8.4\text{x}10^{12}$	0	19.450

Table 5: Chemistry model for Mars atmospheric gases

No.	Reaction	C	n	E/k
1.	$H + O_2 \Longleftrightarrow OH + O$	2.2E14	0.0	8455.0
2.	$O + H_2 \Longleftrightarrow OH + H$	7.5E13	0.0	5586.0
3.	$H_2 + OH \Longleftrightarrow H + H_2O$	2.0E13	0.0	2600.0
4.	$OH + OH \Longleftrightarrow O + H_2O$	5.3E12	0.0	503.0
5.	$H_2 + M \Longleftrightarrow 2H + M$	5.5E18	-1.0	51987.0
6.	$H_2O + M \Longleftrightarrow OH + H + M$	5.2E21	-1.5	59386.0
7.	$OH + M \Longleftrightarrow O + H + M$	8.5E18	-1.0	50830.0
8.	$O_2 + M \Longleftrightarrow O + O + M$	7.2E18	-1.0	59340.0

Table 6: Hydrogen-Air Combustion Model

No.	Reaction	C	n	E/k
1	$CH_4 + M \Longleftrightarrow CH_3 + H + M$	1.50E+17	0.0	44600.0
2	$CH_4 + H \Longleftrightarrow CH_3 + H_2$	2.00E+14	0.0	5982.7
3	$CH_4 + OH \Longleftrightarrow CH_3 + H_2O$	3.00E+13	0.0	2513.7
4	$CH_4 + O \Longleftrightarrow CH_3 + OH$	2.00E+13	0.0	3469.0
5	$CH_3 + O \Longleftrightarrow CH_2O + H$	8.00E+13	0.0	502.7
6	$CH_3 + O_2 \Longleftrightarrow CH_2O + OH$	4.00E+13	0.0	8798.1
7	$CH_2O + M \Longleftrightarrow CO + H_2 + M$	2.00E+16	0.0	17596.2
8	$CH_2O + OH \Longleftrightarrow CHO + H_2O$	2.50E+13	0.0	502.7
9	$CH_2O + O \Longleftrightarrow CHO + OH$	3.00E+13	0.0	0.0
10	$CH_2O + H \Longleftrightarrow CHO + H_2$	1.70E+13	0.0	1508.2
11	$CHO + OH \Longleftrightarrow CO_2 + H$	1.00E+14	0.0	0.0
12	$CHO + M \Longleftrightarrow CO + H + M$	2.00E+12	0.5	14479.2
13	$CO + OH \Longleftrightarrow CO_2 + H$	5.50E+11	0.0	543.0
14	$H + O_2 \Longleftrightarrow OH + O$	2.20E+14	0.0	8455.0
15	$O + H_2 \Longleftrightarrow OH + H$	7.50E+13	0.0	5586.0
16	$OH + H_2 \Longleftrightarrow H_2O + H$	2.00E+13	0.0	2600.0
17	$OH + OH \Longleftrightarrow H_2O + O$	5.30E+12	0.0	503.0
18	$H_2 + M \Longleftrightarrow H + H + M$	5.50E+18	-1.0	51987.0
19	$H_2O + M \Longleftrightarrow OH + H + M$	5.20E+21	-1.5	59386.0

Table 7: Methane-Air "Full" Combustion Model

No.	Reaction	C	n	E/k
1	$CH_4 + 1.5O_2 \Longrightarrow CO + 2H_2O$	4.00E+10	0.0	24333.0
2	$CO + OH \Longleftrightarrow CO_2 + H$	1.50E+07	1.3	-402.2
3	$CO + O_2 \Longleftrightarrow CO_2 + O$	3.10E+11	0.0	18903.3
4	$CO + O + M \Longleftrightarrow CO_2 + M$	5.90E+15	0.0	2061.3
5	$H + O_2 \Longleftrightarrow OH + O$	2.20E+14	0.0	8446.2
6	$O + H_2 \Longleftrightarrow OH + H$	1.80E+10	1.0	4474.5
7	$OH + H_2 \Longleftrightarrow H_2O + H$	2.20E+13	0.0	2564.0
8	$H_2O + O \Longleftrightarrow OH + OH$	6.80E+13	0.0	9250.6
9	$OH + M \Longleftrightarrow O + H + M$	8.00E+19	-1.0	52135.0
10	$O_2 + M \Longleftrightarrow O + O + M$	5.10E+15	0.0	57816.1
11	$H_2 + M \Longleftrightarrow H + H + M$	2.20E+14	0.0	48263.8
12	$H_2O + M \Longleftrightarrow OH + H + M$	2.20E+16	0.0	52788.6

Table 8: Methane-Air Quasi-Global Combustion Model

8 Evaluation and Comparisons of Parallel Performance

In this section, timings for six of the test cases described in the previous section are presented. From these timings, the parallel performance of pHANA can be estimated.

Two parameters commonly used to measure the parallel performance of an application program are speedup and efficiency. Speedup is defined as the ratio of the time required to run a particular application on one processor to that required to run on N processors. Efficiency is defined as

$$\text{Efficiency} = \text{Speedup}/N,$$

or the fraction of the maximum possible speedup obtainable with N processors. Amdahl's law relation is often used to determine the maximum possible performance of an algorithm. Using Amdahl's relation, the speedup and efficiency are given by

$$\text{Speedup} = \frac{N}{p + (1-p)N}$$

$$\text{Efficiency} = \frac{1}{p + (1-p)N}$$

where p is the fraction of the calculation that is performed in parallel. Obviously, algorithms that can give $p = 1$ are of particular interest. With such algorithms there is no inherent limit placed on the speedup by the algorithm itself. The current algorithm is designed to achieve nearly 100% or $p = 1$ parallelism. However, in practice, there are several factors that limit the performance of the current implicit algorithm:

1. load balancing (t_{lb})
2. interprocess communication/data swapping and synchronization (t_c)
3. additional work due to domain decomposition (t_{addl})
4. reduction in vector length (t_{vect})
5. competition for cpu time among users (for multi-user systems) (t_{extr})

Therefore, the time required to complete a calculation on N processors (t_N) is

$$t_N = (t_1/N) + t_{lb} + t_c + t_{addl} + t_{vect} + t_{extr}$$

where t_1 is the time required to solve the same problem on one processor.

Load balancing refers to the even distribution of the computational time among the processors. For our implicit algorithm, it can be eliminated by satisfying a simple formula:

$$MOD([ID + (n_i - 1) * 4], n_i) = MOD([JD + (n_j - 1) * 4], n_j) = 0.$$

where ID, JD are the number of mesh points in the i- and j-directions, and n_i and n_j are the number of divisions used in each direction to decompose the domain. In this section, the load balancing problem has been eliminated in all cases by selecting combinations for the domain decomposition such that perfect load balancing is achieved.

Interprocess communication/data swapping and synchronization are an important consideration in the design of parallel algorithms. Even for a perfectly parallel algorithm, the total computational time is always augmented by the time to perform the necessary communication between processors. The communication time is dependent on the hardware bandwidth, the message passing software, and the algorithm. The ratio of the time required for message passing to the total run time depends on the amount of work (number crunching) on each processor. It is approximately inversely proportional to the square root of the number of mesh points assigned to each processor and to the number of species included in the calculation. For a distributed memory system, it is generally advantageous to minimize the number of messages passed and to maximize the size of each message. This reduces the effect of latency time (or the time required to send a 0-byte message) on the parallel efficiency.

The domain decomposition technique adopted in the current implicit algorithm formulation is not perfect. One of its drawbacks is that additional work is required as more processors are used to solve a problem in a parallel manner. The additional work occurs at the mesh cell faces that define the boundaries between subzones. For a single processor calculation the flux through each face must only be calculated once per time step. With multiple processors it must be calculated twice; once for each of the subzones on either side of the face. Thus, for parallel calculations, the added work to compute these fluxes is approximately

$$t_{addl} \propto [(JD * (n_i - 1)) + (ID * (n_j - 1))].$$

Of course this work is only involved in the flux and implicit flux Jacobian calculations, which together are approximately 40% of the total work. Therefore the final additional work due to subdomain decomposition is:

$$t_{addl}/t_1 \simeq 0.40 * \left[\frac{n_i - 1}{ID} + \frac{n_j - 1}{JD}\right].$$

A 2-cell interprocess communication patch was chosen over a 1-cell patch which would require half as many additional subzone boundary cells due to domain decomposition. However, a 1-cell patch would require additional calculations and information to be sent for the explicit and implicit fluxes to the neighboring processors, which is less cost effective. It is obvious that for a constant problem size the additional work due to parallelization increases with increasing number of processors. The best parallel efficiency is obtained by solving the largest possible problem that will fit in the available memory for a given number of processors. While this approach will optimize parallel efficiency, it may not satisfy the most common goal, which is to minimize the run time for a constant size problem.

The reduction in vector length is a very important consideration if a parallel algorithm is to be used on a parallel vector machine. It also affects performance on machines which utilize pipelining architectures. The domain decomposition divides the overall domain into smaller subdomains which results in shorter vector lengths. Even without any other factors affecting its performance, the time to complete a calculation increases with the reduction in vector length. In this report, vector length reduction is especially noticeable in the Cray Y-MP performance, and will be addressed later in this section.

The two shared memory systems tested in this work are the Cray Y-MP and the SGI 4D. Both systems are multi-user/multi-tasking systems where several users can run their specific application programs simultaneously. On this type of system, unfortunately, the presence of other users affects the performance of parallel algorithms. Other users programs will unevenly delay the execution of the pHANA calculations on some of the processors. This leads to an uncontrollable load leveling problem as the other processors sit idle while the delayed processor catches up.

In the following subsections, timing results for three parallel computers are presented.

8.1 Distributed Memory MIMD

The parallel HANA code (pHANA) was tested on an Intel Touchstone iPSC/860 for six of the cases described in section 7. The parallel speedup and efficiency for these cases are shown in Figures 51 and 52. The efficiency of the code is different for the different cases. As expected, better efficiencies are achieved for the cases with larger meshes.

Figure 53 shows how the cpu time is spent for the axisymmetric combusting ram accelerator case. By holding the number of mesh cells-per-processor constant, the domain decomposition overhead (t_{addl}) is constant and the overhead due to reduction of vector length for pipelining (t_{vect}) is eliminated. Thus, the reduction in efficiency as the number of processors is increased is attributable only to the interprocess communication overhead (t_c). Note that for this case, the calculation solves 14 sets of partial differential conservation equations (10 species, 3 momentum, and 1 energy). This maximizes the amount of work on each node. Therefore, the ratio of communication overhead to the total amount of work is minimized, resulting in a very high parallel performance (see Figures 51 and 52). It appears that when a maximum amount of work is assigned equally to all cpus, only 25% or less of the total time is spent doing "noncomputational" activities when the number of processors is less than 64.

As depicted in Figure 51, the speedup increases as the amount of work relative to the communication time overhead increases. Figure 54 illustrates how the cpu time is spent for a calculation where the number of cpus is increased and the size of the problem is held constant. It shows the distribution of cpu time for the hydrogen-air shock

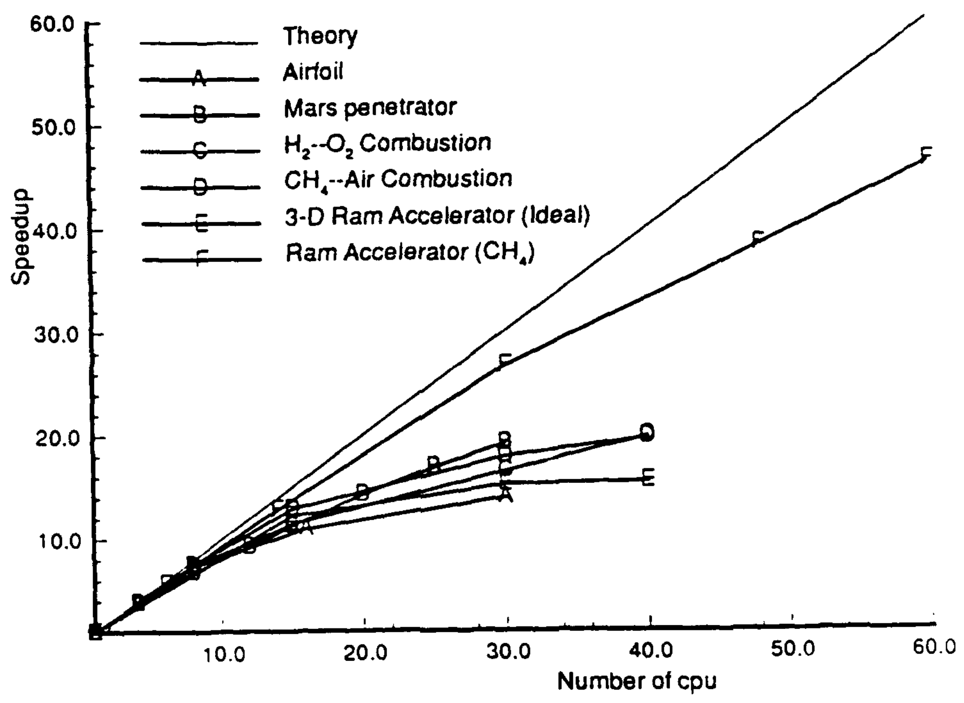


Figure 51: Speed on Intel Touchstone iPSC/860

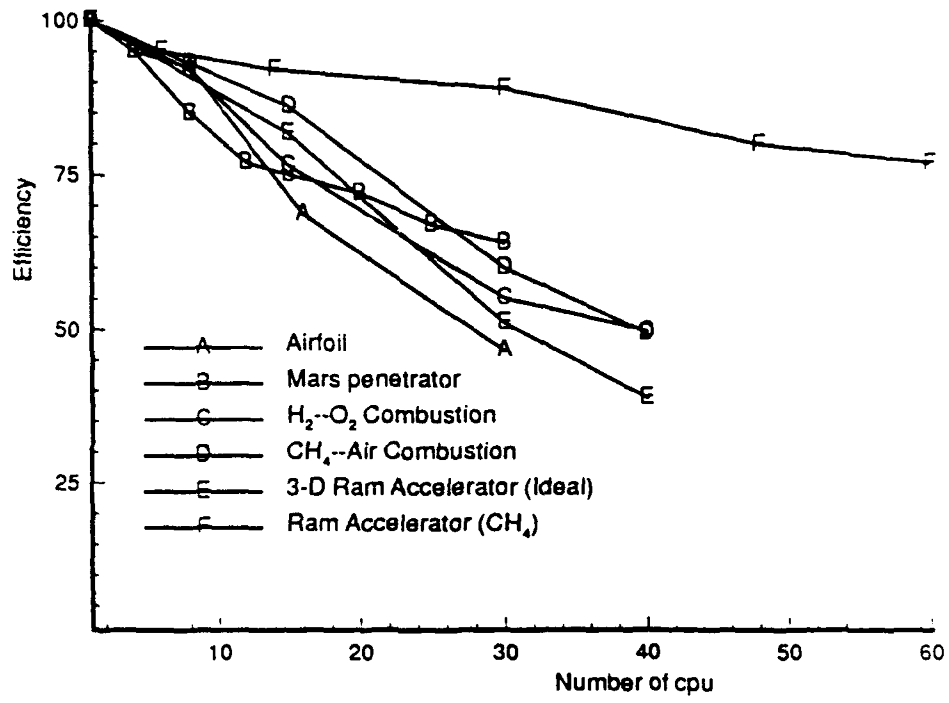


Figure 52: Efficiency on Intel Touchstone iPSC/860

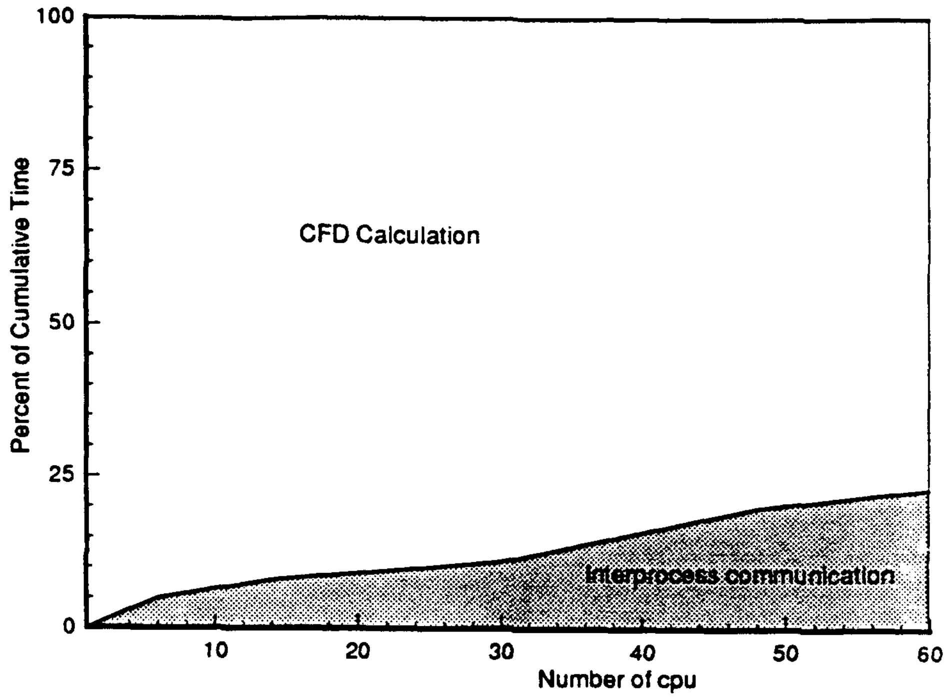


Figure 53: iPSC/860: Distribution of CPU time when the amount of work/node is constant

induced combustion case. For this case, increasing the number of processors utilized in the calculation increases the percentage of work due to domain decomposition and decreases the working vector length. To isolate the interprocess communication overhead time (t_c), a set of calculations was performed varying the number of cpus but holding the work per node constant. Also the effect of the reduction in vector length overhead time (t_{vect}) was approximated by running a set of 1-cpu calculations where the dimension of the problem was changed. Unlike the axisymmetric combusting ram accelerator case, the hydrogen-air shock induced combustion only requires 11 partial differential conservation equations (7 species, 3 momentum, and 1 energy). Therefore, each subdomain is smaller and requires less work than that for the ram accelerator case. This results in lower parallel efficiency and a higher ratio of interprocess communication overhead to the total computational time. This ratio increases linearly after 15 processors. More than 25% of the total cumulative time is spent performing "noncomputational" activities when the number of processors exceeds 40.

Finally, the floating point operations per-second (flops) for the combusting ram accelerator case are given in Table 9. The flops are calculated based on the floating point operation count provided by the hardware performance monitor on the Cray Y-MP. It appears that at least 48 processors are required to obtain a speed comparable to a single cpu Cray Y-MP for this case.

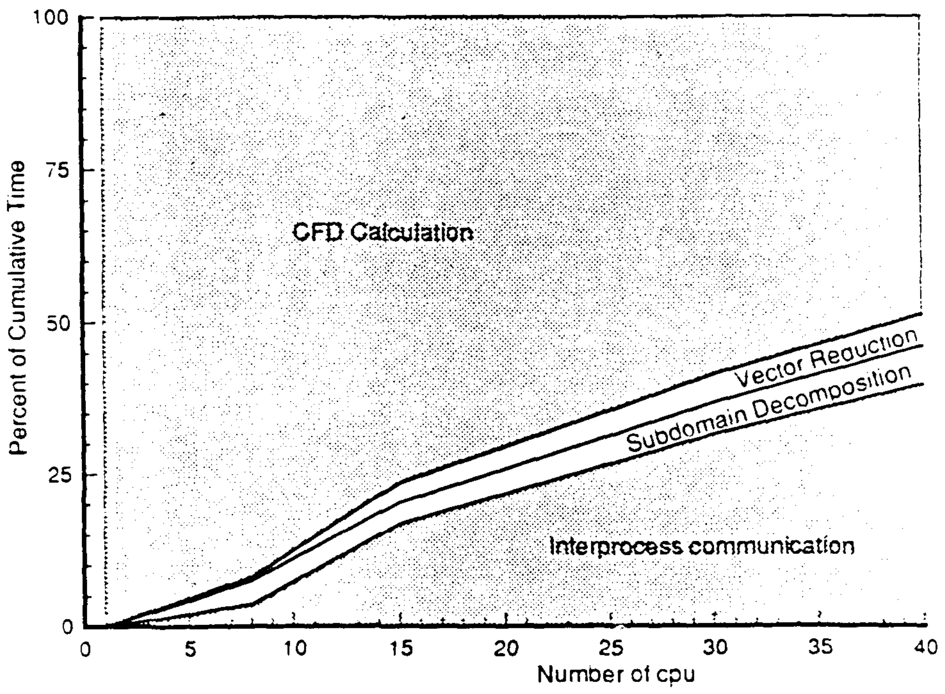


Figure 54: iPSC/860: Distribution of CPU time for constant problem size

Number of Processors	Speedup	Efficiency	MFlops
1	1.0	100.0	3.00
6	5.70	95.0	17.1
14	12.88	92.0	38.7
30	26.71	89.0	80.17
48	38.40	80.0	115.3
60	46.20	76.9	138.7

Table 9: iPSC/860: Floating point operations per-second for the ram accelerator case.

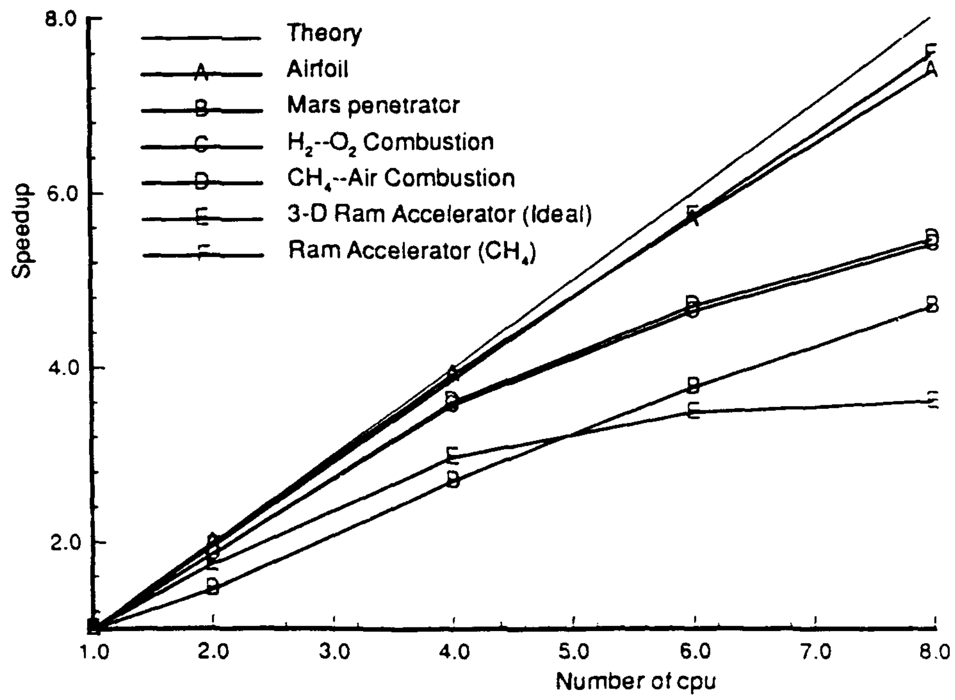


Figure 55: Speedup on Cray Y-MP

8.2 Shared Memory Cray Y-MP

The parallel speedup and efficiency on the Cray Y-MP for the six test cases are shown in Figures 55 and 56. The performance of the code appears to change for the different cases. The best performance is achieved for the cases with the longest vector lengths.

Figure 57 shows how the cpu time is spent on the axisymmetric combusting ram accelerator case. In this case, the amount of work and the vector length on each cpu is held constant. The reduction of efficiency is due only to the interprocess communication overhead (t_c) and synchronization. This overhead is shown to be less than 10% for 8 cpus. Unfortunately the same result does not apply for smaller cases where the number of mesh cells is held fixed while the number of processors is increased.

To investigate the effect of varying vector length on the performance of the code, we have selected two sets of 1-cpu runs where each dimension is changed while holding others constant. The first set fixes $JD = 11$ while varying the number of mesh points in the I-direction, and the second fixes $ID = 53$ while varying the number of mesh points in the J-direction. Results from this test for are shown in Figure 58 for the hydrogen-air shock induced combustion case. The vector length is a very important factor in the achieved speed on the Cray Y-MP. This result is crucial for understanding the reduction of efficiency of a parallel algorithm as the number of processors is increased. Using the domain decomposition technique, a computational domain is broken up into smaller subdomains which are assigned to different processors. A good

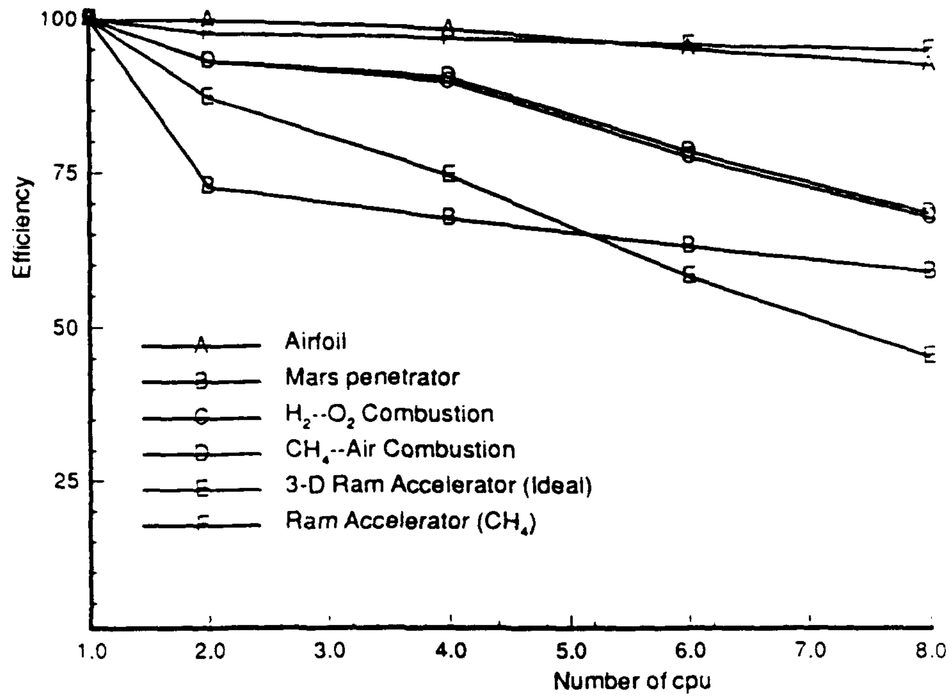


Figure 56: Efficiency on Cray Y-MP

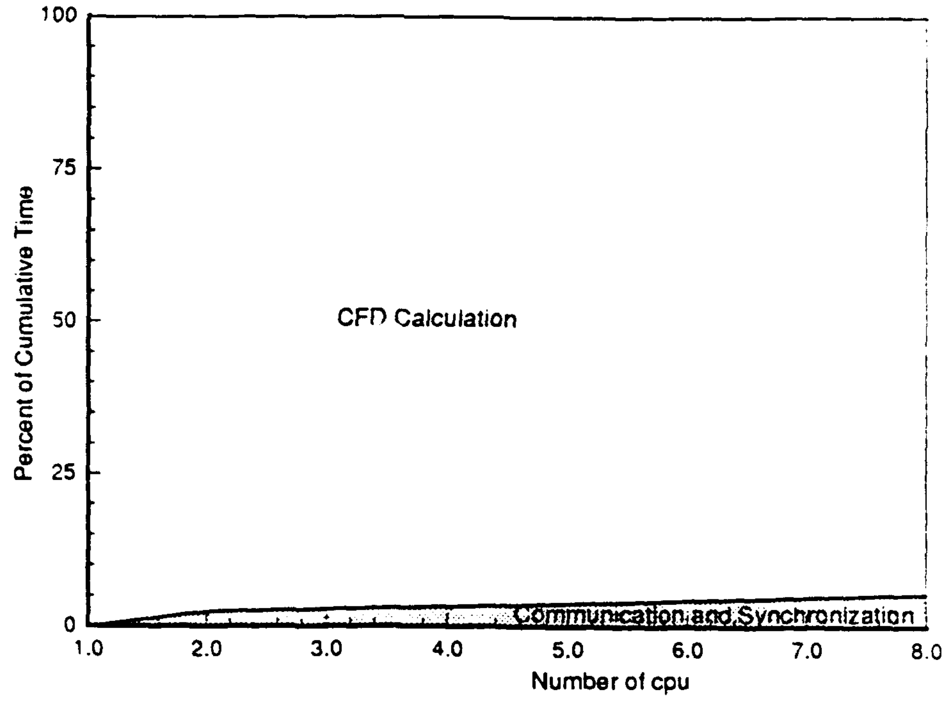


Figure 57: Cray Y-MP: Distribution of CPU time when the amount of work/node is constant

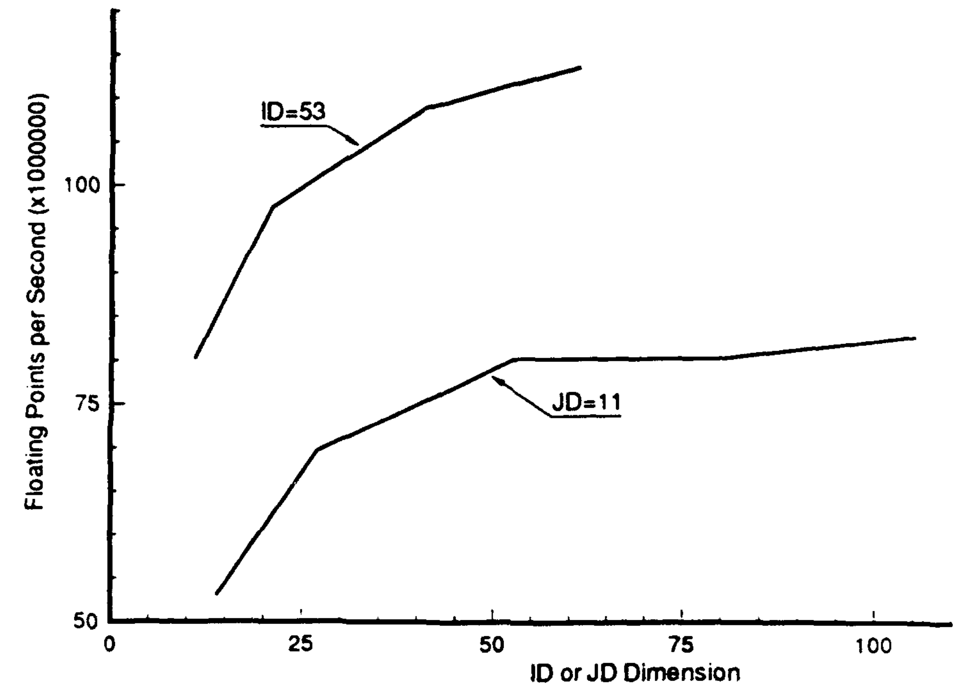


Figure 58: Cray Y-MP: Effects of Vector Length Reductions

parallel performance can only be obtained if the dimensions of these subdomains are large enough such that the effect of vectorization can be neglected. For our test cases, a vector length of at least 50 is required for this to be true.

Figure 59 illustrates further how the cpu time is spent for a calculation in which the number of cpus is increased and the size of the problem is held constant. As the number of processors is increased, the additional work due to domain decomposition increases and the vector length of each subdomain decreases. The overhead time associated with reduction in vector length (t_{vect}) is approximated by running a set of 1-cpu calculations where the dimension of the problem was changed (see, Figure 58). As mentioned previously, the vector length in this case is much shorter than is optimum and the run, therefore, results in lower parallel efficiency. The ratio of overhead time due to vectorization loss to the total computational time is increased. This represents more than 20% of the total cpu time for a constant-size parallel run with eight cpus.

Finally the floating point operations per second (flops) measured for the combusting ram accelerator case are given in Table 10. This performance can be improved by increasing the number of mesh cells in the calculation.

8.3 Shared Memory Silicon Graphics Iris

The parallel speedup and efficiency on the SGI 4D/380 for the six test cases are shown in Figures 60 and 61. As for the other two computers, the performance of the code is different for different problems. The performance appears to be excellent for four or

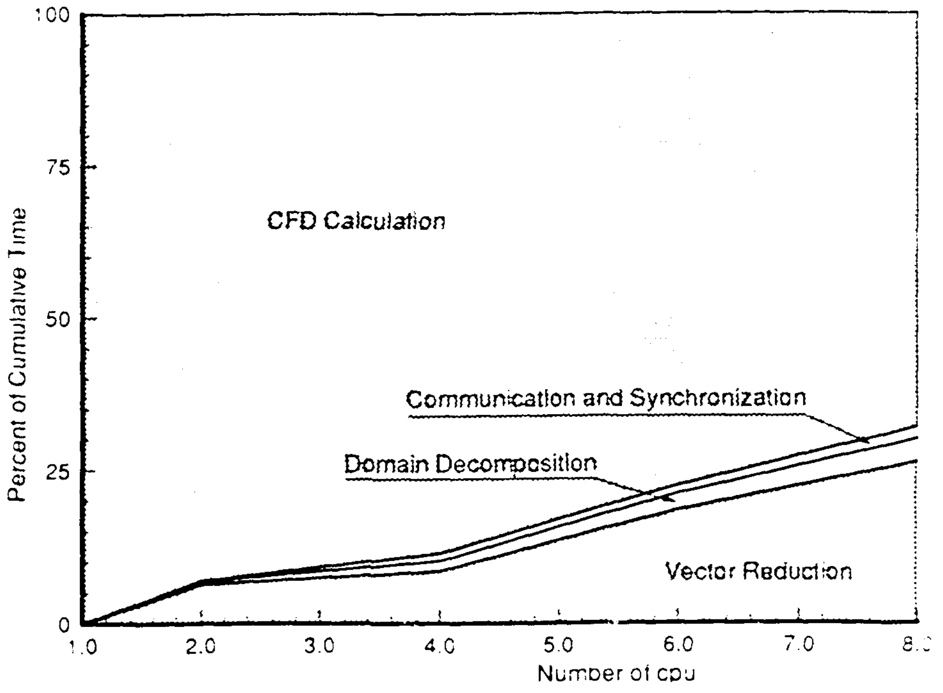


Figure 59: Cray Y-MP: Distribution of CPU time for constant problem size

Number of Processors	Speedup	Efficiency	MFlops
1	1.0	100.0	111.62
2	1.95	97.6	217.9
4	3.87	96.8	432.3
6	5.74	95.7	640.7
8	7.58	94.7	845.9

Table 10: Cray Y-MP: Floating point operations per-second for the ram accelerator case.

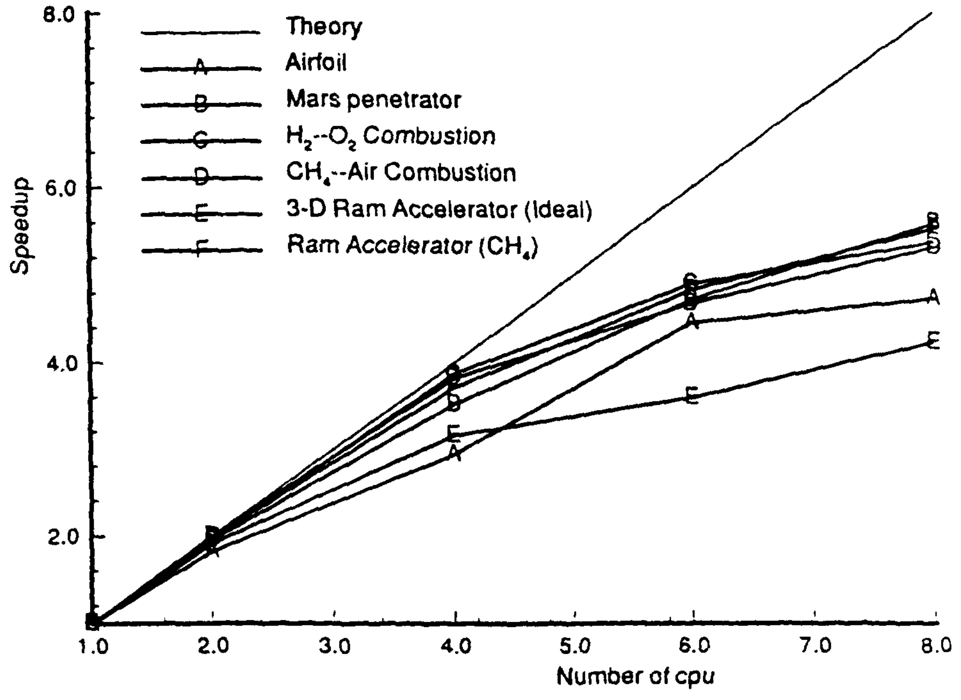


Figure 60: Speedup on SGI 4-D/380

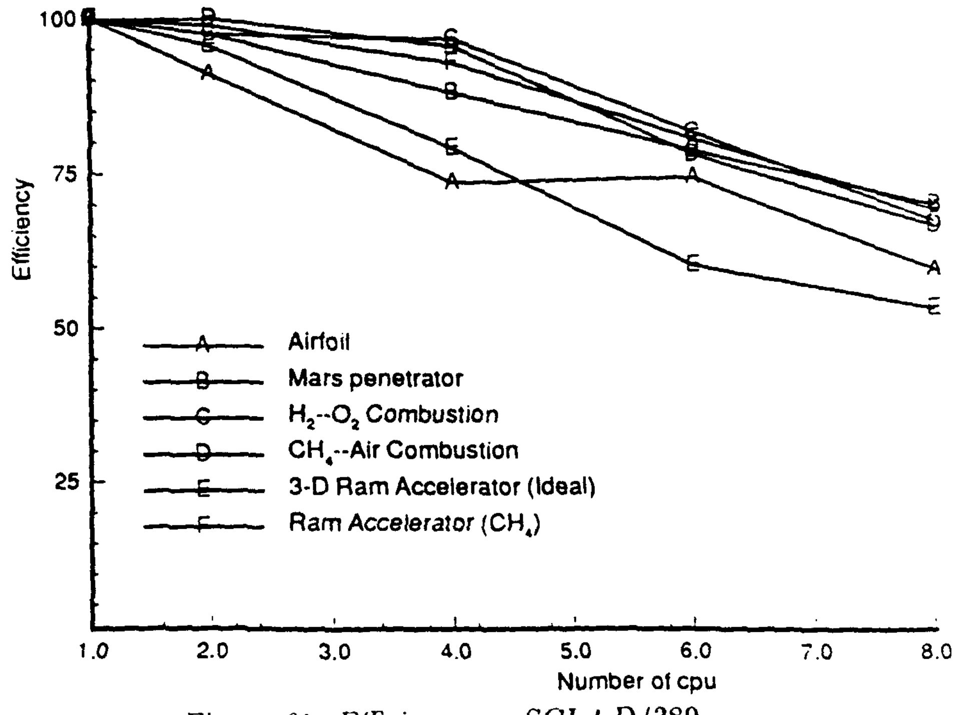


Figure 61: Efficiency on SGI 4-D/380

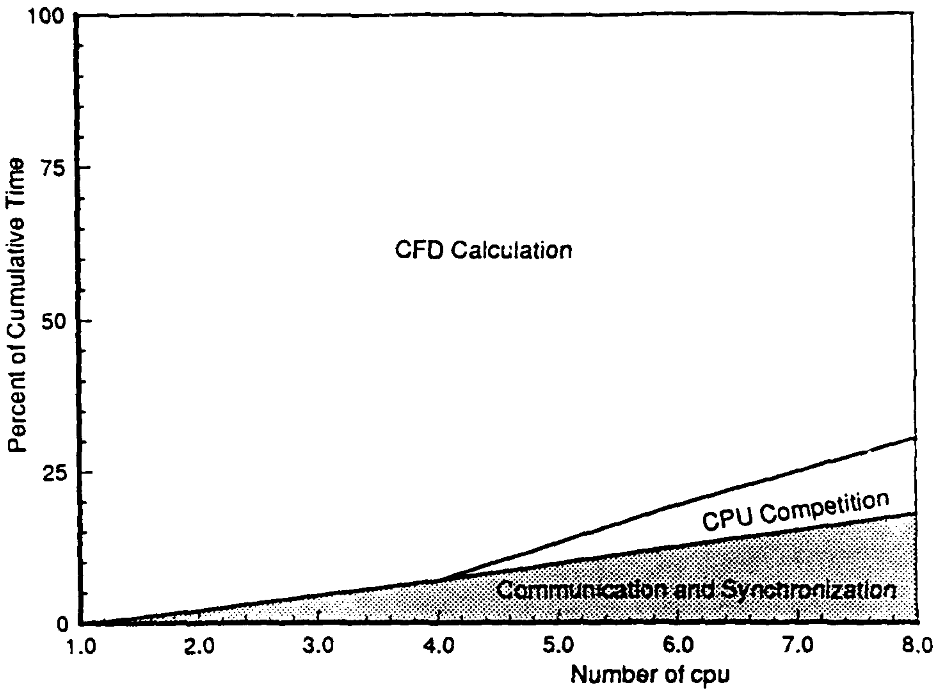


Figure 62: SGI 4-D/380: Distribution of CPU time when the amount of work/node is constant

less processors. However, when a larger number of cpu's are used, the performance drops significantly. As mentioned previously, the load balancing problem has been eliminated for our test cases. However, in a multi-user system one cpu may be working on two or more applications at a time. Multiple applications running at the same time resulting in cpu competition which causes some cpus to lag behind the others. The faster cpus must then wait for the slower ones to catch up. A second factor is the relatively low bandwidth on the SGI bus. The bandwidth is too small and cpus often have to wait to access memory when more than four cpus are working on a problem.

Unlike the iPSC/860 and the Cray Y-MP discussed previously, the best performance on the SGI machine was not achieved for the largest test case. This is due to the smaller cache memory available on the machine. The cache was not large enough for the combusting ram accelerator case to run without significant cache misses.

Figure 62 shows how the cpu time was spent on the axisymmetric combusting ram accelerator case. In this case, the amount of work on each cpu is constant. The reduction of efficiency is due only to the interprocess communication overhead (t_c), synchronization, competition with other applications for the cpus, and the low bus bandwidth. This overhead is shown to be more than 25% for 8 cpus.

Figure 63 illustrates further how the cpu time is spent for the constant-size problem. As the number of processors is increased, the additional work due to domain decomposition increases. The overhead time associated with reduction in vector length (t_{vect}) is seen to be minimal since the SGI is not a vector machine. The amount of

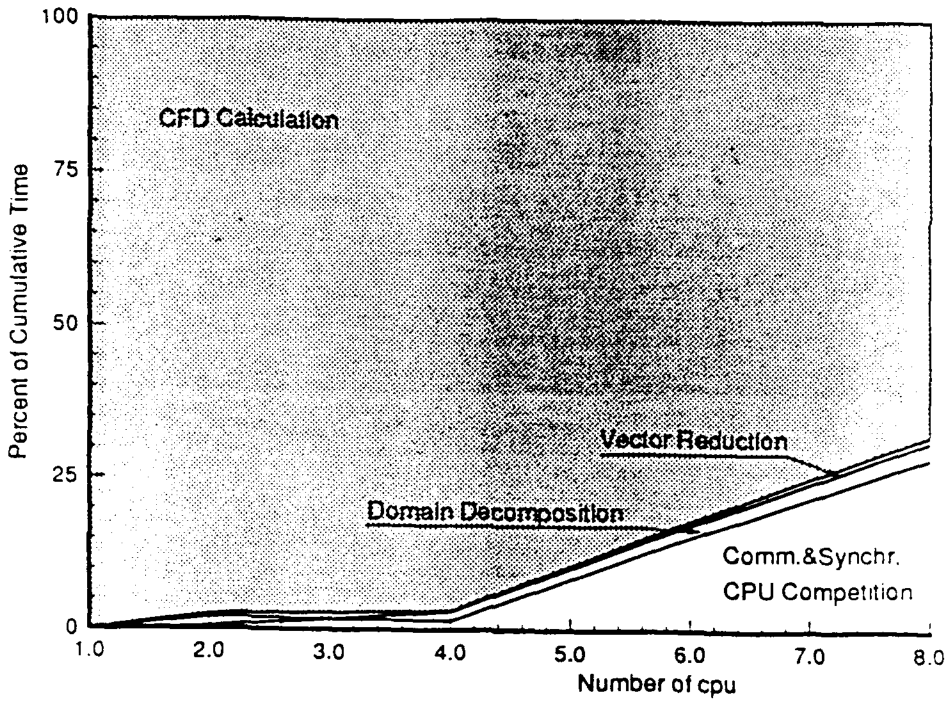


Figure 63: SGI 4-D/380: Distribution of CPU time for constant problem size

time required for communication and synchronization appears to be less than that for the combusting ram accelerator case. The cpu time spent on noncomputational activities is fairly low when less than 4 processors are utilized. However, this overhead time jumps significantly for higher number of cpus due to cpu competition and the inability of the bus to handle the traffic.

The floating points operations per second (flops) measured for the combusting ram accelerator case are given in Table 11. The flops are calculated based on the number of floating points given by hardware performance monitor on Cray Y-MP.

Number of Processors	Speedup	Efficiency	MFlops
1	1.0	100.0	2.69
2	1.98	98.9	5.33
4	3.72	93.0	10.01
6	4.84	80.7	13.02
8	5.56	69.5	14.96

Table 11: SGI 4-D/380: Floating point operations per second for the ram accelerator case.

9 Conclusions

As a result of this investigation we can make the following conclusions.

- A model code has been written which solves a set of Laplace equations on a multiple zone mesh using point Gauss-Seidel relaxation and a diagonal wavefront algorithm. The code was written to simulate the operation of a Navier-Stokes code as closely as possible. The model code was parallelized on the Cray Y-MP, Intel Touchstone iPSC/860, and Silicon Graphics Iris 4-D/380 (SGI) machines.

- The 3D Navier-Stokes code was parallelized using domain decomposition on the Cray Y-MP, Intel Touchstone iPSC/860, and SGI 4-D/380 computers. The governing equations and solution algorithm for this code, pHANA, are detailed in sections 4 and 5.

- Flow field and parallel performance results are given for the calculations on the three computers. The test cases include two and three-dimensional flows at subsonic, transonic, and supersonic speeds. Thermochemical models vary from nonreacting ideal gas to combusting gases.

- The main deterrent to parallel performance on the Intel iPSC/860 was interprocess communication. Improvements to the hardware which reduce the interprocessor communication latency should dramatically improve the parallel performance of the Navier-Stokes code particularly for cases which can not be sized to fully utilize the memory available with each cpu.

- The main deterrent to parallel performance on the CRAY Y-MP was the reduction in vector length due to domain decomposition. Vector length was not as critical to performance on the other two computers.

- The main deterrent to parallel performance on the SGI 4-D/380 was memory contention, low bus bandwidth, and competition for CPU time with other processes.

- For all computers, increasing the size of the problem as the number of processors increased yielded a better performance than holding the problem size constant as the number of processors increased.

References

[1] Peterson, V.L.,"The Impact of Supercomputers on the Aerospace Sciences," AIAA 24th Aerospace Sciences Meeting, Reno, NV, January 1986.

[2] Denning,P.J.,"Parallel Computation," *American Scientist*, Vol. 73, No. 4, July 1985, pp322-323.

[3] Moore,R., Nassi, I., O'Neil, J., and Siewiorek, D.P.,"The Encore Multimax: A Multiprocessor Computing Environment.", Encore Technical Report No. ETR 86-004, Encore Computer Corporation, Marlboro, MA, 1986.

[4] Anderson, D. A. , Tannehill, J. C. , and Pletcher, R. H. *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, 1984.

[5] Baldwin, B. and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper No. 78-257, Jan. 1978.

[6] Launder, B.E., and Spalding, D.B., "The Numerical Computation of Turbulent Flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 3, 1974.

[7] Vinokur, M., and Lui, Y., "Equilibrium Gas Flow Computations II: An Analysis of Numerical Formulations of Conservation Laws," AIAA Paper No. 88-0127, 1988.

[8] Srinivasan, S., Tannehill, J.C., and Weilmuenster, K.J., "Simplified Curve Fits for the Thermodynamic Properties of Equilibrium Air," ISU-ERI-Ames-86041, Iowa State University, 1986.

[9] Vincenti, W.G., and Kruger, C.H., *Introduction to Physical Gas Dynamics*, Robert E. Krieger Publishing Company, Malabar, FL 1965.

[10] Lee, J.H., "Basic Governing Equations for the Flight Regimes of Aeroassisted Orbital Transfer Vehicles," AIAA Paper No. 84-1729, 1984.

[11] Park, C., "Assessment of Two-Temperature Kinetic Model for Ionizing Air," *Journal of Thermophysics and Heat Transfer*, Vol. 3, No. 3, July 1989.

[12] Candler, G.V., and MacCormack, R.W., "The Computation of Hypersonic Ionized Flows in Chemical and Thermal Nonequilibrium," AIAA Paper No. 88-0511, 1988.

[13] Srinivasan, S., and Tannehill, J.C., "Simplified Curve Fits for the Transport Properties of Equilibrium Air," NASA CR-178411, 1987.

[14] Gokcen, T., MacCormack, R.W., and Chapman, D.R., "Computational Fluid Dynamics Near the Continuum Limit," AIAA Paper 87-0115.

[15] Kogan, M. N. , *Rarefied Gas Dynamics*, Plenum Press, NY 1969.

[16] Gupta, R.N., Scott, C.D., and Moss, J.N., "Slip-Boundary Equations for Low-Reynolds-Number Multicomponent Gas Flows," AIAA Paper 84-1732.

[17] Gupta, R.N., Scott, C.D., and Moss, J.N., "Slip-Boundary Equations for Multicomponent Nonequilibrium Airflow," NASA TP 2452, Nov. 1985.

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

[19] Yoon, S., and Kwak, D., "Artificial Dissipation Models for Hypersonic External Flow," AIAA Paper No. 88-3708, 1988.

[20] Peery, K.M. and Imlay, S.T., "An Efficient Implicit Method for Solving Viscous Multi-Stream Nozzle/Afterbody Flow Fields," AIAA Paper No. 86-1380, June 1986.

[21] Anderson, W.K., Thomas, J.L., and VanLeer, B., "A Comparison of Finite Volume Flux Vector Splitting for the Euler Equations," AIAA Paper No. 85-0122, 1985.

[22] Steger, J.L. and Warming, R.F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite- Difference Methods," *Journal of Comp. Phys.*, Vol. 40, pp 263-293, 1981.

[23] Yee, H.C., Warming, R.F., Harten, A., "Implicit Total Variation Diminishing (TVD) Schemes for Steady State Calculations," J. of Comp. Phys., Vol 57, pp. 327-360, 1985.

[24] Pulliam,T.H. and Steger,J.L.,"On Implicit Finite Difference Simulations of Three Dimensional Flows," AIAA Paper No. 78-10, Jan. 1978.

[25] van Leer, B., "Flux-Vector Splitting for the Euler Equations," ICASE Report 82-30, Sept., 1982.

[26] Harten, A., Lax, P.D., and Van Leer, B., "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws," SIAM Review, Vol. 25, No. 1, Jan. 1983, pp. 35-61.

[27] Grossman, B., and Walters, R.W., "An Analysis of Flux-Split Algorithms for Euler's Equations with Real Gases," AIAA paper 87-1117CP, 1987.

[28] Engquist, B., and Osher, S., "One-Sided Difference Approximations for Nonlinear Conservation Laws," Mathematics of Computation, Vol. 36, No. 154, April 1981, pp. 321-351.

[29] Osher, S., and Solomon, F., "Upwind Difference Schemes for Hyperbolic Systems of Conservation Laws," Mathematics of Computation, Vol. 38, No. 158, April 1982, pp. 339-374.

[30] Harten, A., "On a Class of High Resolution Total-Variation-Stable Schemes," NYU Report, Oct., 1982; SIAM J. Num. Anal, Vol. 21, 1984, pp 1-23.

[31] Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[32] Peery, K.M., Imlay, S.T., and Katsandres, J.T., "Real Gas Blunt-Body Flow Simulations," AIAA Paper No. 87-2179, June 1987.

[33] Peery, K.M. and Imlay, S.T., "Blunt-Body Flow Simulations," AIAA Paper No. 88-2904, July 1988.

[34] Park, C., and Yoon, S., "A Fully-Coupled Implicit Method for Thermo-Chemical Nonequilibrium Air at Sub-Orbital Speeds," AIAA Paper No. 89-1974, June 1989.

[35] Jachimowski, C.J., "An Analytical Study of the Hydrogen-Air Reaction Mechanism with Application to Scramjet Combustion," NASA-TP-2791.

[36] Lijewski, L.E., et al., "Program EAGLE User's Manual: *Vol I - Introduction and* Grid Applications," Air Force Armament Laboratory, AFATL-TR-88-117, VOL I, September 1988.

[37] Thompson, J.F., Gatlin, B., "Program EAGLE User's Manual: Vol II - Surface Generation Code," Air Force Armament Laboratory, AFATL-TR-88-117, VOL II, September 1988.

[38] Thompson, J.F., Gatlin, B., "Program EAGLE User's Manual: Vol III - Grid Generation Code," Air Force Armament Laboratory, AFATL-TR-88-117, VOL III, September 1988.

[39] Cheatwood, F.M., DeJarnette, F.R., and Hamilton, H.H., "Geometrical Description for a Proposed Aeroassist Flight Experiment Vehicle," NASA Technical Memorandum 87714, July 1986.

[40] Hair, L.M., Engel, C.D., Shih, P., Bithell, R., and Bowman, M., "Aerothermal Test Data of Low L/D Aerobraking Orbital Transfer Vehicle at Mach 10," NASA LaRC Test 117, Remtech Report RTR 069-1, April 1983.

[41] Hornung, H.G., "Non-equilibrium Dissociating Nitrogen Flow over Spheres and Circular Cylinders, " J. Fluid Mech. Vol.53, part 1, pp.149-176, 1972.

[42] Lobb. R.K., "Experimental Measurement of Shock Detachment Distance on Spheres Fired in Air at Hypervelocities, " The High Temperature Aspects of Hypersonic Flow, edited by W.C. Nelson, Pergammon, NY., 1964.

[43] Gnoffo. P.A., "Code Calibration Program in Support of the Aeroassist Flight Experiment, " J. Spacecraft, Vol.27, No.2, pp.131-142, March-April, 1990.

[44] Gnoffo, P.A., "Application of Program LAURA to Three-Dimensional AOTV Flowfields, " AIAA-86-0565, 1986.

[45] Yungster. S., Eberhardt, S., and Bruckner, A.P., "Numerical Simulation of Shock-Induced Combustion Generated by High-Speed Projectiles in Detonable Gas Mixtures," AIAA-89-0673. 1989.

[46] Yungster, S. and Bruckner, A.P., "A Numerical Study of the Ram Accelerator Concept in the Superdetonative Velocity Range. " AIAA-89-2677, 1989.

[47] Evans. J.S., and Schexnayder, C.J.Jr., "Influence of Chemical Kinetics and Unmixedness on Burning in Supersonic Hydrogen Flames, " AIAA. J. vol.18, No.2, pp.188-193, 1979.

[48] Lehr. H.F., "Experiments on Shock-Induced Combustion," Aeronatica Acta. Vol.17. pp.589-597. 1972.

[49] Imlay. S.T., "A Solution Adaptive Grid/Navier-Stokes Solution Procedure," AIAA Paper No. 87-2180, June 1987.

[50] Thomas. J.L., and Walters, R.W., "Upwind Relaxation Algorithms for the Navier-Stokes Equations," AIAA Paper No. 85-150, July 1985.

[51] MacCormack. R. W.. "Current Status of Numerical Solutions of the Navier-Stokes Equations," AIAA Paper No. 85-0032, Jan. 1985.

[52] Chakravarthy, S.R., "Implicit Upwind Schemes Without Approximate Factorization," AIAA Paper No. 84-0165, Jan. 1984.

[53] Imlay. S.T., "Numerical Solution of 2-D Thrust Reversing and Thrust Vectoring Nozzle Flowfields," AIAA Paper No. 86-0203, Jan. 1986.

[54] Imlay. S.T. "Implicit Time-Marching Solution of the Navier- Stokes Equations for Thrust Reversing and Thrust Vectoring Nozzle Flows," NASA CR 4026, Nov. 1986.

[55] Eberhardt. S., and Imlay, S.. "A Diagonal Implicit Scheme for Computing Flows with Finite-Rate Chemistry," AIAA Paper No. 90-1577, June 1990.

[56] Degrez, G. and Ginoux, J.J., "Surface Phenomena in a Three-Dimensional Skewed Shock Wave/ Laminar Boundary Layer Interaction," *AIAA Journal*, Vol. 22, No. 12, December 1984, pp. 1764-1769.

[57] Imlay, S.T., Roberts, D.W., and Soetrisno, M., "An Efficient 3D Navier-Stokes Analysis for Evaluation of Hypersonic Vehicles," Final report for NASA Contract No. NAS8-37406, October 1990.

[58] Cook, D.H., et al., "Aerofoil RAE 2822 - Pressure Measurements, and and Boundary Layer and Wake Measurements," AGARD-AR-138, J. Barche, AGARD, 1979, pp. A6-1 to A6-77.

[59] Spaid, W., et al., "An Experimental Study of Transonic Flow About a Supercritical Airfoil," NASA TM-81336, 1983.

[60] Chan, J.S., "Multizone Navier-Stokes Computations of Viscous Transonic Flows Around Airfoils," AIAA Paper No. 99-0103, Jan. 1988.

[61] Visbal, M.R., and Shang, J.S., "Comparative Study Between the Navier-Stokes Algorithms for Transonic Airfoils," *AIAA Journal* , Vol. 24, April 1986, pp. 599-606.

[62] Mason, M.L., Putnam, L.E., and Re, R.J., "The Effect of Throat Contouring on Two-Dimensional Converging-Diverging Nozzles at Static Conditions," NASA TP-1704, August 1980.

[63] Imlay, S.T., Roberts, D.W., Soetrisno, M., and Eberhardt, S., "Nonequilibrium Thermo-Chemical Calculations using a Diagonal Implicit Scheme," AIAA Paper No. 91-0468, Jan. 1991.

[64] Candler, G.V., "Computation of Thermo-Chemical Nonequilibrium Martian Atmospheric Entry Flows," AIAA Paper No. 90-1695, 1990.

[65] Kull, A.E., Burnham, E.A., Knowlen, C., Bruckner, A.P., Hertzberg, A., "Experimental Studies of Superdetonative Ram Accelerator Modes, " AIAA-89-2632.

[66] Yungster, S., Eberhardt, S., Bruckner, A.P, "Numerical Simulation of Shock-Induced Combustion Generated by High-Speed Projectiles in Detonable Gas Mixtures, " AIAA-89-0673.

[67] Yungster S., Bruckner, A.P, "A Numerical Study of the Ram Accelerator Concept in the Superdetonative Velocity Range, " AIAA-89-2677.

[68] Yungster S., "Numerical Study of Shock-Wave/Boundary Layer Interactions in Premixed Hydrogen-Air Hypersonic Flows, " AIAA-91-0413.

[69] Burnham, E.A., Kull, A.E., Knowlen, C., Bruckner, A.P., Hertzberg, A., "Operation of the Ram Accelerator in the Transdetonative Velocity Regime, " AIAA-90-1985.

[70] Shuen, J.S. and Yoon, S., "Numerical Study of Chemically Reacting Flows Using an LU Scheme, " AIAA-88-0436.

[71] Imlay, S.T, Roberts, D.W., Soetrisno, M., "Nonequilibrium Thermo Chemical Calculations using a Diagonal Implicit Scheme, " AIAA-91-0468.

[72] Imlay, S.T, Roberts, D.W., Soetrisno, M., "HANA: A Three Dimensional Navier-Stokes Code for Chemically Reacting Flows, " AIAA-91-2153.

[73] Soetrisno, M., Imlay, S.T. "Simulation of The Flow Field of a Ram Accelerator, " AIAA-91-1915.

[74] Nusca, M "Numerical Simulation of Reacting Flow in a Thermally Choked Ram Accelerator Projectile Launch System, " AIAA-91-2490.

[75] Westbrook, C. K., Dryer, F. L., "Simplified Reaction Mechanisms for the Oxidation of Hydrocarbon Fuels in Flames," Combustion Science and Technology, Vol. 27, pp. 31-43, 1981.

[76] Westbrook, C. K., Dryer, F. L., "Chemical Kinetic Modeling of Hydrocarbon Combustion, " Progress in Energy and Combustion Science, Vol. 10, No. 1, pp. 1-57, 1984.

[77] Chitsomboon, T., Kumar, A., Tiwari, S. N., "Numerical Study of Finite Rate Supersonic Combustion Using Parabolized Equations, " AIAA-87-0088, 1987.

[78] Chew, G., Knowlen, C., Burnham, E. A., Hertzberg, A., Bruckner, A. P., "Experiments on Hypersonic Ramjet Propulsion Cycles Using a Ram Accelerator, " AIAA-91-2489, 1991.

[79] Humphrey, J. W., "Parametric Study of an ODW Scramaccelerator for Hypersonic Test Facilities, " AIAA-90-2470, 1990.

[80] Srulijes, J., Smeets, G., Seiler, F., George, A., Mathieu, G., and Resweber, R., "Shock Tube Validation Experiments for the Simulation of Ram Accelerator Related Combustion and Gasdynamic Problems, " Presented at the 18th International Symposium on Shock Waves, in Sendai, Japan, July 21-26, 1991.

[81] Nusca, M., "Navier-Stokes Simulation of Fluid Dynamic and Combustion Phenomena in the Ram Accelerator, " Presented at the 28th JANNAF Combustion Subcommittee Meeting, in San Antonio, Texas, October 28, 1991.

[82] Edelman, R. B., and Fortune, O. F., "A Quasi-Global Chemical Kinetic Model for the Finite Rate Combustion of Hydrocarbon Fuels with Application to Turbulent Burning and Mixing in Hypersonic Engines and Nozzles, " AIAA-69-0086.

[83] Knowlen, C., Burnham, E., Bruckner, A.P., Hertzberg, A., "Ram Accelerator Combustion Phenomena," Presented at the 28th JANNAF Combustion Subcommittee Meeting, in San Antonio, Texas, October 28, 1991.