MB_CNS Example Book.

Mechanical Engineering Report 2004/10
P. A. Jacobs
Centre for Hypersonics
The University of Queensland.

September 2004, Revised March 2005

Preface

MB_CNS is a collection of programs for the simulation of transient two-dimensional (or axisymmetric) flows. It is part of the larger collection of compressible flow simulation codes found at `http://www.mech.uq.edu.au/cfcfd/`. This manual is a collection of example simulations: scripts, results and commentary. It may be convenient for new users of the code to identify an example close to the situation that they wish to model and then adapt the scripts for that example.

This report will be updated occasionally with new examples and commentary. As the simulation codes are improved, we will try to maintain compatibility so that older examples are not "broken", however, this backward compatibility will not be guaranteed.

Contents

1 Introduction **3**

2 Mach 1.5 flow over a 20^o cone **5**
2.1 .sit file 8
2.2 Shell scripts 9
2.3 Notes 10

3 Flow of equilibrium-air over a sphere **12**
3.1 .sit file 12
3.2 Shell scripts 15
3.3 Notes 16

4 Hypersonic flow of ideal air over a blunt wedge **18**
4.1 .sit file 21
4.2 Shell scripts 23
4.3 Notes 24

5 Mach 3 flow over a sharp-nosed two-dimensional body **26**
5.1 .sit file 29
5.2 Shell scripts 30
5.3 Notes 31

6 Flow through a conical nozzle **32**
6.1 .sit file 35
6.2 Shell scripts 37
6.3 Notes 38

7 A section of an ideal compressible-flow vortex **40**
7.1 .sit file 41
7.2 Shell scripts 43
7.3 Notes 44

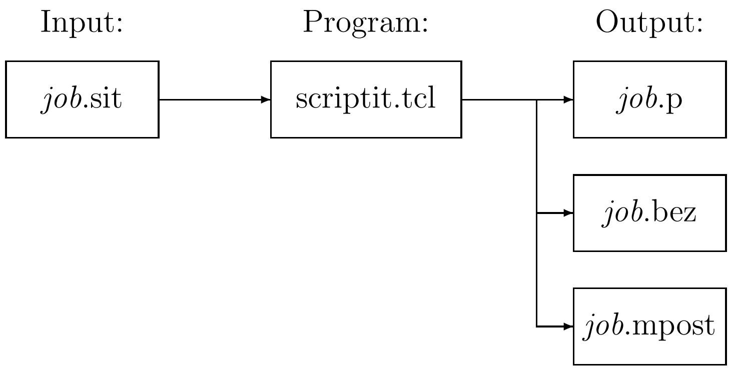
8 Pressure on a flat-faced cylinder **46**
8.1 .sit file 48
8.2 Shell scripts 49
8.3 Notes 51

1 Introduction

Setting up a simulation is mostly an exercise in writing a textual description of your flow and its bounding geometry. This is presented to the scriptit program as a text file, sometimes referred to as a ".sit" file or flow-specification file. Once you have prepared your flow specification file, the simulation data is generated in a number of stages:

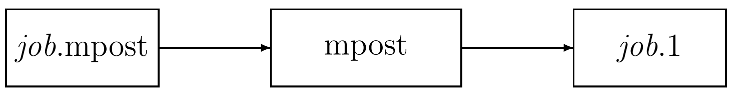
1a Create the geometry definition with the command.

$ scriptit.tcl -f *job*.sit -do-mpost



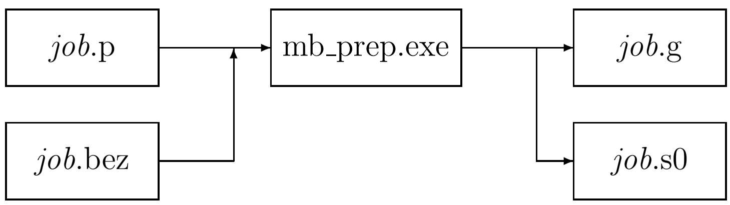
1b Check the geometry definition (visually) by using Metapost to make a viewable postscript file containing labelled nodes, block boundary curves and blocks. Metapost is distributed as part of the TEX document preparation system. It is most likely already installed on your UNIX/Linux system and there is a stand-alone binary for Win32 systems.

$ mpost *job*.mpost



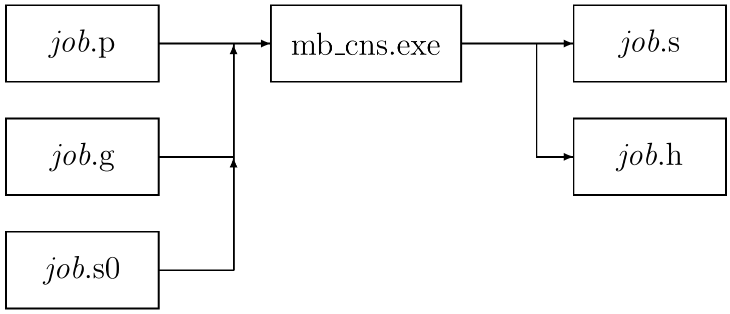
2 Generate a grid and initial flow solution (at $t = 0$).

$ mb_prep.exe -f *job*



3 Run the simulation code to produce flow data at subsequent times.

$ mb_cns.exe -f *job*



4 Extract subsets of the flow solution data for postprocessing. The specific command for this stage depends very much on what you want to do. The flow solution data is cell-averaged data associated with cell centres. You may extract the flow data for all cells at a particular time using `mb_post.exe` and reformat it for a particular plotting program or you may extract data along single grid lines (using `mb_prof.exe`) in a form ready for display with GNU-Plot or for further calculation. See the shell scripts in the examples for ideas on what can be done. Since the output of this stage is always a text file, you may look at the head of the file for hints as to what data is present.

Originally, `scriptit` was a C program which read your script, initialized some data structures and wrote the parameter (*job*.p) and geometry definition (*job*.bez) files. Currently, `scriptit` is implemented as a TCL program (see e.g. Reference [1]) that has a few extra procedures and your specification script is really a TCL script. It is worth the effort to learn just enough Tcl to be dangerous. The Web site `http://www.tcl.tk` has a good (short) starting guide.

After doing some initialization, `scriptit(.tcl)` sources your script file and assembles the geometry and flow specification data into a form that can be given to the main simulation codes (`mb_prep.exe` and `mb_cns.exe`). The advantage of this approach is that you have the full capability of the Tcl interpreter available to you from within your script. You can perform calculations so that you can parameterize your geometry, for example, or you can use Tcl control structures to make repetitive definitions much more concise. Additionally, you may use Tcl comments to add documentation to the script file.

The following examples should be studied in together with the HTML reference for mb_cns programs, and scriptit in particular. These hypertext manuals can be found in the documentation section at the URL: `http://www.mech.uq.edu.au/cfcfd/`.

2 Mach 1.5 flow over a 20^o cone

This is a small (in both memory and run time) example that is useful for checking that the simulation and plotting programs have been built or installed correctly. Assuming that you have the program executable files built and accessible on your system's search `PATH`, try the following commands:

```
$ cd ~/cfcfd/code/mb_cns/examples/cone20
$ ./cone20_run.sh
$ ./cone20_plot.sh
```

And, within a minite or so, you should end up with a number of files with various solution data plotted. The grid and initial solution are created and the time-evolution of the flow field is computed for 5 ms (with 1105 time steps being required). The commands invoke the shell scripts displayed in subsection 2.2.

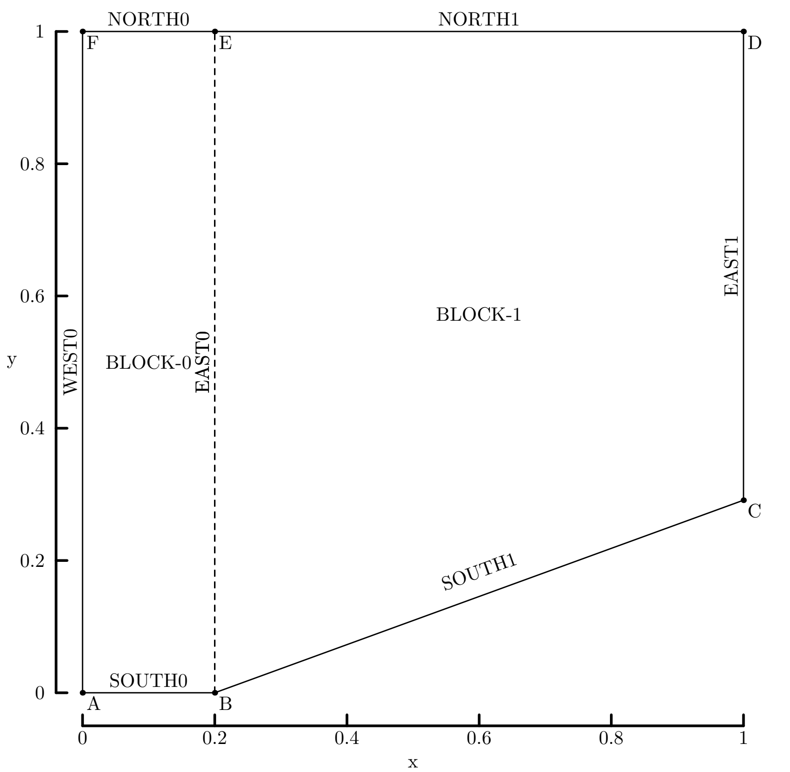


Figure 1: Schematic diagram of the geometry for a cone with 20 degree half-angle. This encapsulated postscript figure was generated from the Metapost file.

The free-stream conditions ($p_\infty = 95.84\,\text{kPa}$, $T_\infty = 1103\,\text{K}$ and $u_\infty = 1000\,\text{m/s}$) are related to the shock-over-ramp test problem in the original ICASE Report [2] and are set to give a Mach number of 1.5. From Chart 5 in Ref. [3], the expected steady-state shock wave angle is 49^o and, from Chart 6, the pressure coefficent is

$$\frac{p_{cone-surface} - p_\infty}{q_\infty} \approx 0.387$$

and the dynamic pressure for the specified free stream is $q_\infty = \frac{1}{2}\rho_\infty u_\infty^2 \approx 151.38\,\text{kPa}$. Figure 4 shows the pressure coefficient estimated as

$$C_p = \frac{f_x - p_\infty A}{q_\infty A}$$

from the simulated axial force, f_x, written into the simulation log file and frontal area of the cone, A.

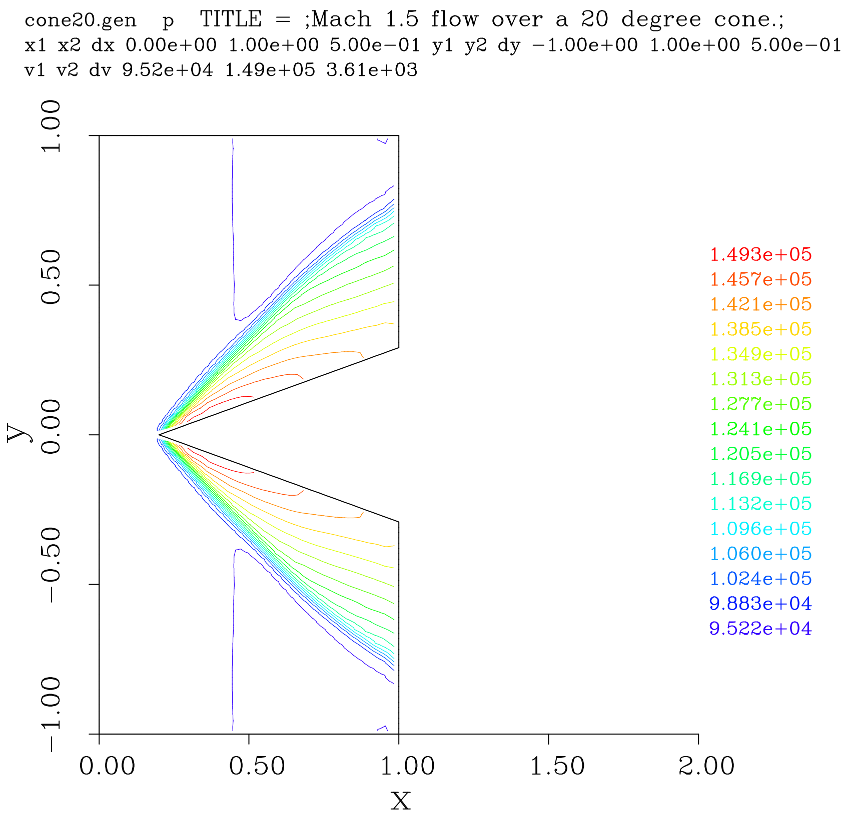


Figure 2: Pressure data for flow over a cone with 20 degree half-angle. The shock profile is not yet straight and the pressure field near the cone surface is not conically symmetric, although it would become more if we continued the simulation.

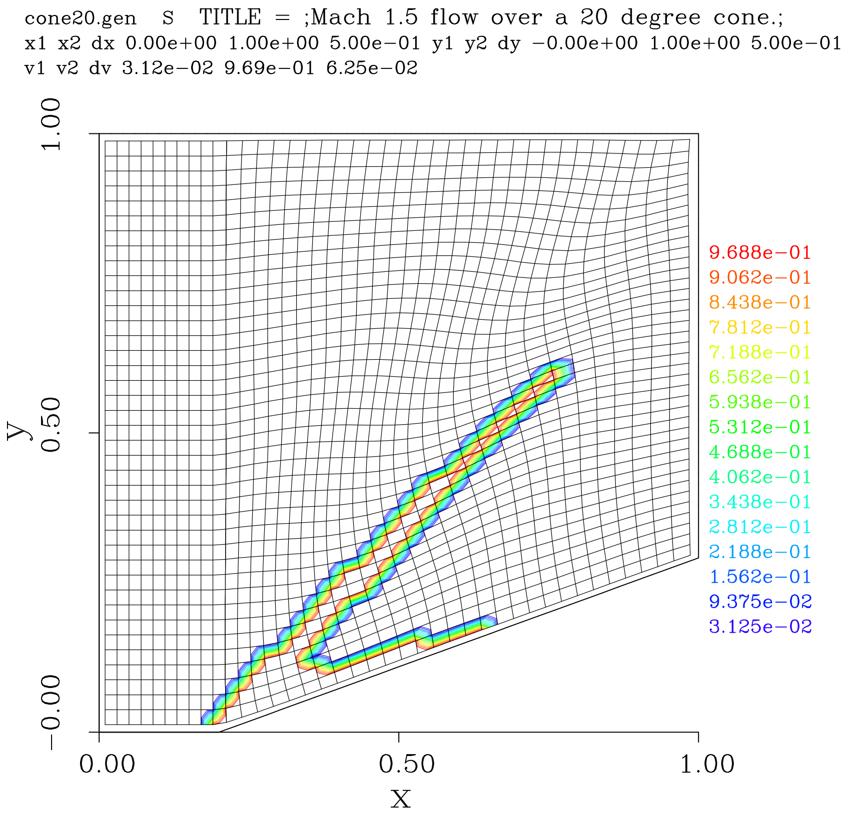


Figure 3: Shock-sensor data for flow over a cone with 20 degree half-angle. For the `adaptive` flux calculator, this sensor indicates the regions of the flow where the more dissipative scheme should be used.

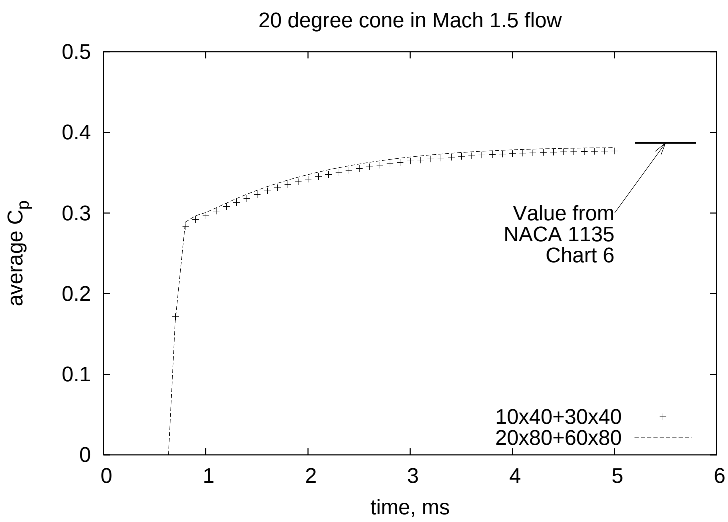


Figure 4: Evolution of the axial (drag) force for flow over a cone with 20 degree half-angle for two mesh resolutions.

2.1 .sit file

```
# cone20.sit
# Mach 1.5 flow over a 20 degree cone.

# Set up two quadrilaterals in the (x,y)-plane be first defining
# the corner nodes, then the lines between thos corners and then
# the boundary elements for the blocks.
BEGIN_GEOMETRY
    NODE a 0.0 0.0
    NODE b 0.2 0.0
    NODE c 1.0 0.29118
    NODE d 1.0 1.0
    NODE e 0.2 1.0
    NODE f 0.0 1.0

    LINE ab a b
    LINE bc b c
    LINE af a f
    LINE be b e
    LINE cd c d
    LINE fe f e
    LINE ed e d

    # Define the boundaries.
    POLYLINE north0 1 + fe
    POLYLINE east0  1 + be
    POLYLINE south0 1 + ab
    POLYLINE west0  1 + af
    POLYLINE south1 1 + bc
    POLYLINE east1  1 + cd
    POLYLINE north1 1 + ed
END_GEOMETRY

BEGIN_FLOW
    # Gas and flow properties
    GAS_TYPE perf_air_14
    GAS_STATE initial 5955.0      0.0 0.0  304.0  1.0
    GAS_STATE inflow  95.84e3 1000.0 0.0 1103.0  1.0

    # Set the boundary discretisation before building the blocks.
    DISCRETISE north0 10 0 0 0.0
    DISCRETISE east0  40 0 0 0.0
    DISCRETISE south0 10 0 0 0.0
    DISCRETISE west0  40 0 0 0.0
    DISCRETISE north1 30 0 0 0.0
    DISCRETISE south1 30 0 0 0.0
    DISCRETISE east1  40 0 0 0.0

    # Inflow and outflow boundaries.
    BOUNDARY_SPEC west0 SUP_IN  inflow
    BOUNDARY_SPEC east1 EXTRAPOLATE_OUT

    # Define two blocks with a common boundary.
    # Although we have used integers in the block names,
    # any unique names would be fine.
    # Same goes for the other entities such as nodes, and boundaries.
    BLOCK block-0  + north0 + east0 + south0 + west0
    BLOCK block-1 + north1 + east1 + south1 + east0
    CONNECT_BLOCKS block-0 east block-1 west
    # Have specified an area-orthogonality grid for block-1,
    # just for fun.
    GRID_TYPE block-1 AO

    # Assign the initial gas states
    FILL_BLOCK block-0 initial
    FILL_BLOCK block-1 initial
END_FLOW
```

```
BEGIN_CONTROL
    TITLE Mach 1.5 flow over a 20 degree cone.
    CASE_ID 5
    AXISYMMETRIC
    VISCOUS
    FLUX_CALC adaptive
    MAX_TIME  5.0e-3
    MAX_STEP  3000
    TIME_STEP 1.0e-6
    DT_PLOT   1.5e-3
    DT_HISTORY 10.0e-5
    HISTORY_CELL block -1 10 1
END_CONTROL

# Name the output files and build them.
MPOST FILE cone20.mpost
MPOST SCALES 0.12 0.12
MPOST XAXIS 0.0 1.0 0.2 -0.05
MPOST YAXIS 0.0 1.0 0.2 -0.04
BEZIER_FILE cone20.bez
PARAM_FILE cone20.p
BUILD

QUIT
```

2.2 Shell scripts

```
#! /bin/sh
# cone20_run.sh
# exercise the Navier-Stokes solver for the Cone20 test case.
# It is assumed that the path is set correctly.

# Generate the Bezier, Input parameter and MetaPost files from the Script File.
# The MetaPost file provides us with a graphical check on the geometry
# specification and it is worth creating if metapost (mpost) is available.
if [[ -f /usr/bin/tclsh ]]
then
    echo "Use the new scriptit."
    scriptit.tcl -f cone20.sit -do-mpost > cone20.scriptit-log
    if [[ -f cone20.mpost ]]
    then
        mpost cone20.mpost
    fi
else
    echo "Use the old scriptit."
    scriptit.exe < cone20.sit > cone20.log
fi

# Generate the Grid and Initial Solution Files.
mb_prep.exe -f cone20

# Integrate the solution in time,
# recording the axial force on the cone surface.
#
# The following environment variables allow the shared-memory version
# of the code to use one thread for each of the two blocks.
# The stacksize requirements may increase as the code develops and
# more elements are added to the internal data structures.
export OMP_NUM_THREADS=2
export KMP_STACKSIZE=8m
time mb_cns.exe -f cone20 -force 1 2

# Extract the solution data and reformat.
# If no time is specified, the first solution found is output.
mb_post.exe -fp cone20.p -fg cone20.g -fs cone20.s -fo cone20 -generic
```

```
# Extract the average coefficient of pressure from the axial force
# records that were written to the simulation log file.
awk -f cp.awk mb_cns.log > cone20_cp.dat

echo "At this point, we should have a new solution"
echo "Run cone20_plot.sh next"
```

```
# cone20_plot.sh
# Pick up the reformatted data and make plots of:

# 1. Shock indicator
mb_cont.exe -fi cone20.gen -fo cone20_S.ps -var 9 -ps -colour \
            -xrange 0.0 1.0 0.5 -yrange -0.0 1.0 0.5 -mesh

# 2. Pressure.
mb_cont.exe -fi cone20.gen -fo cone20_p.ps -var 6 -ps -colour \
            -mirror -xrange 0.0 1.0 0.5 -yrange -1.0 1.0 0.5
mb_cont.exe -fi cone20.gen -fo cone20.gif -var 6 -gif -colour \
            -mirror -xrange 0.0 1.0 0.5 -yrange -1.0 1.0 0.5

# 3. The mesh alone.
mb_cont.exe -fi cone20.gen -fo cone20_mesh.ps -var 6 -ps -colour \
            -xrange 0.0 1.0 0.5 -yrange -0.0 1.0 0.5 -mesh -nocontours

# 4. The average coefficient of pressure on the cone surface.
#    We assume that the high-resolution data file is also available.
gnuplot <<EOF
set term postscript eps enhanced 20
set output "cone20_cp.ps"
set style line 1 linetype 1 linewidth 3.0
set title "20 degree cone in Mach 1.5 flow"
set xlabel "time, ms"
set ylabel "average C_p"
set xtic 1.0
set ytic 0.1
set yrange [0:0.5]
set key bottom right
set arrow from 5.2,0.387 to 5.8,0.387 nohead linestyle 1
set label "Value from\nNACA 1135\nChart 6" at 5.0,0.3 right
set arrow from 5.0,0.3 to 5.5,0.387 head
plot "cone20_cp.dat" using 1:2 title "10x40+30x40", \
     "cone20_cp_hi-res.dat" using 1:2 title "20x80+60x80" with lines
EOF
```

2.3 Notes

- Run time is approximately 40 seconds on a personal computer with a Celeron 2.4Ghz processor.
- This cone20.sit file should work in both the old C-scriptit and the newer Tcl-scriptit.
- The scriptit program converts all string labels to uppercase. You may specify upper or lower case command names but be careful not to mix cases in a single name; the command will not be found.

- There is a shell script (`cone20_mpi.sh`) to run the MPI version of the simulation code for this example.
- There is also a batch file for running the example on a MS-Windows system.
- Awk script for extracting x-force data from the simulation log file.

```
# cp.awk
# Extract the simulation times and axial force values from the log file.
# The relevant lines in mb_cns.log start with the string "XFORCE"
# and are of the form:
#    XFORCE: t n jb ibndy fx_p fx_v [jb ibndy fx_p fx_v [jb ...]]
# Present the axial force as an average coefficient of pressure to
# compare with that obtained from NACA 1135.

BEGIN {
    p_inf = 95.84e3;  # Pa
    T_inf = 1103.0;   # K
    u_inf = 1000.0;   # m/s
    R = 287;          # J/kg.K
    r_base = 0.29118; # m
    rho_inf = p_inf / (R * T_inf);  # kg/m**3
    q_inf = 0.5 * rho_inf * u_inf * u_inf;  # Pa
    A = 3.14159 * r_base * r_base;  # m**2
    print "# time (ms) Cp";
    print "# rho_inf= ", rho_inf, " q_inf= ", q_inf, " A= ", A
}

/XFORCE/ {
    # Select just the simulation time and the force on the cone surface.
    t = $2;  # in seconds
    f = $6;  # pressure force in Newtons
    # The coefficient of pressure is based on the difference
    # between the cone surface pressure and the free-stream pressure.
    Cp = (f / A - p_inf) / q_inf;
    print $2*1000.0, Cp;
}
```

- The command: `$ mb_compare.sh cone20`
  will compare the newly-computed solution with a reference solution stored in compressed files in the `./reference` subdirectory. If all is well, you should get a report with zero differences for each of the files except the log-file. The log-file will almost certainly contain differences with respect to run times (or wall-clock times).

3 Flow of equilibrium-air over a sphere

This example is a good starting-point for the modelling of hypersonic flow over blunt bodies. It shows the use of arcs and the use of a look-up table as the equation of state for a gas in chemical equilibrium but it remains geometrically simple by using a single-block grid. Also, the .sit file makes use of the Tcl language to parameterize the simulation's specification.

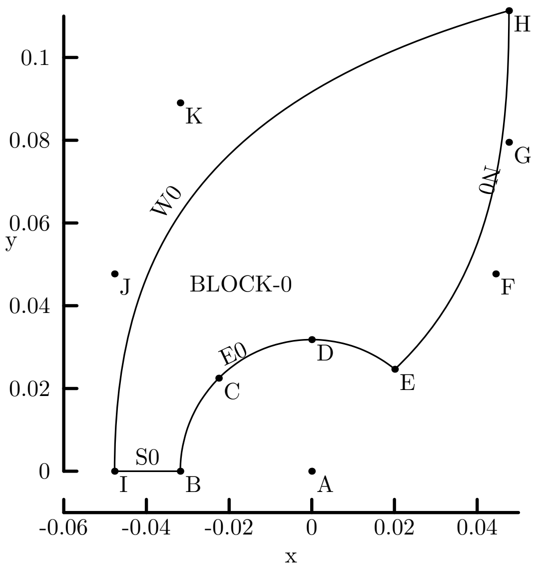


Figure 5: Schematic diagram of the geometry for a sphere wrapped by a single-block grid.

The free-stream condition ($p_\infty = 20\,\text{kPa}$, $T_\infty = 296\,\text{K}$, $u_\infty = 4.68\,\text{km/s}$) corresponds to Case 3 in Ref. [4] with $M_\infty = 13.6$. According to Sawada & Dendou [4], the air is close to being in chemical equilibrium and there is a very thin boundary layer. The results show that the inviscid simulation does indeed capture some of the high-temperature chemistry influence. Ideal stagnation temperature would be 11204 K whereas the simulated temperature along the stagnation line rises to only 6081 K. Secondly, the stand-off distance for an ideal gas is expected to be approximately 4.63 mm. In Fig. 7 the simulated shock stand-off distance is 2.66 mm near the stagnation point. This is within 3% of the experimental value obtained by D. Reda in Sandia's Ballistics Range (see [4]).

3.1 .sit file

```
# file: ss3.sit
#
# Sphere in equilibrium air modelling Case 3 from
#     K. Sawada & E. Dendou (2001)
#      Validation of hypersonic chemical equilibrium flow calculations
#      using ballistic-range data.
```

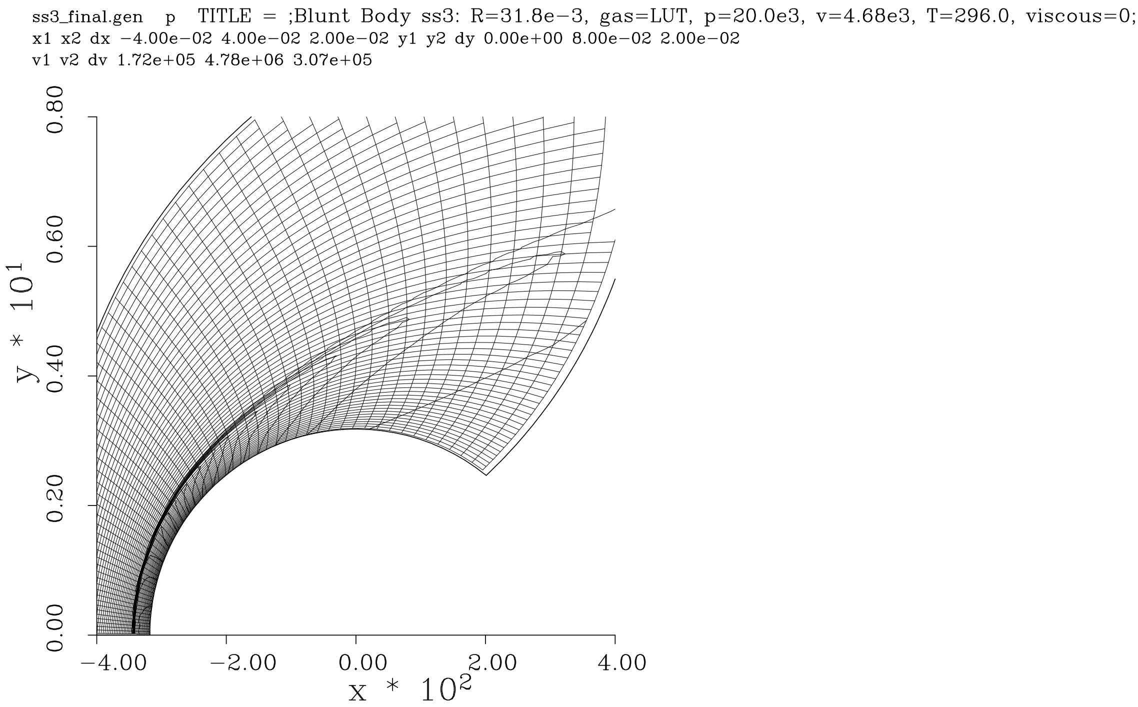


Figure 6: Mesh for flow over a sphere.

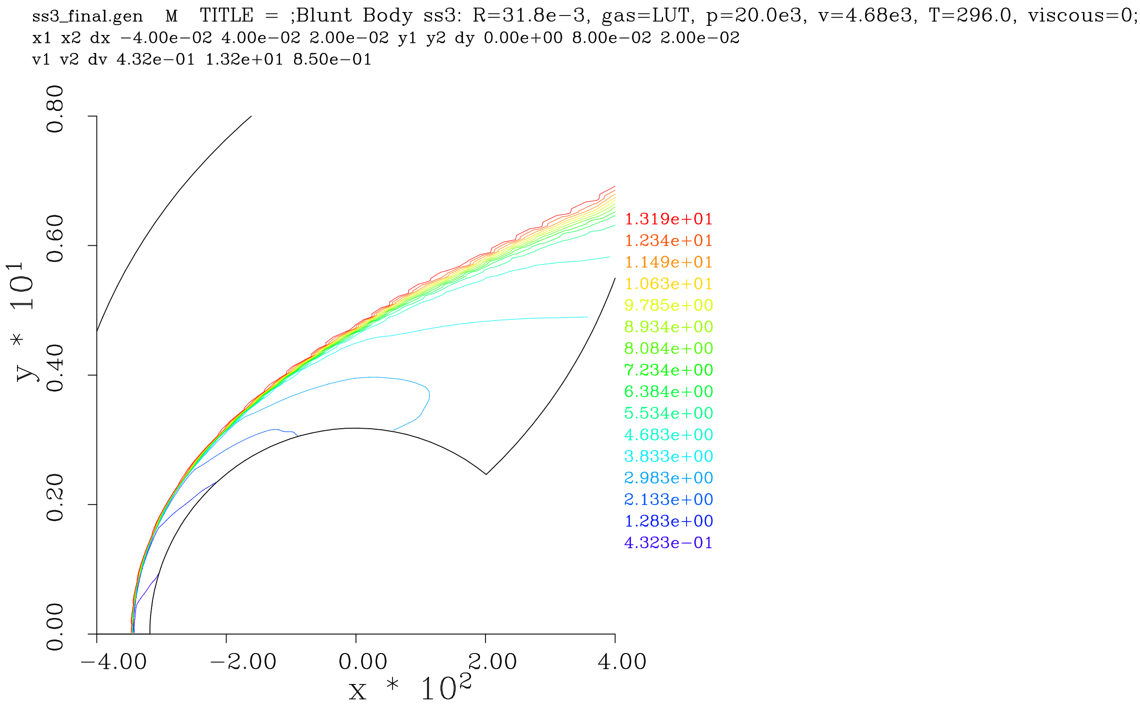


Figure 7: Mach number data for equilibrium-air flow over a sphere.

```
#     Shock Waves (2001) Vol. 11, pp 43--51
#
# Experimental shock stand-off distance is 2.59mm
# Sawada & Dendou CFD value:              2.56mm
#
# This script derived from rbody, 22-Jan-2004.
# PJ
#
# The grid is a bit wasteful because the shock lies close to
# the body for equilibrium air, however, this grid layout
# (as used in rbody) allows us to play with perfect-gas models
# without hitting the inflow boundary with the shock.

# The following JOB name is used to build file names at the end.
set JOB "ss3"

# Radius of body
set R 31.8e-3;                    # m
set T_body 296.0;                 # surface T, not relevant for inviscid flow
set body_type "sphere";           # choose between "cylinder" and "sphere"

# Free-stream flow definition
set p_inf 20.0e3;                 # Pa
set T_inf 296.0;                  # degrees K
set u_inf 4.68e3;                 # flow speed, m/s

# For equilibrium chemistry, use the look-up-table.
set gas_name LUT;

set do_viscous 0;                 # flag for viscous/inviscid calc
set nn  60;                       # grid resolution, both ix and iy
set t_final [expr 10.0 * $R / $u_inf]; # allow time to settle at nose
set t_plot  [expr $t_final / 1.0];     # plot only once

BEGIN_GEOMETRY
    set deg2rad [expr atan2(0.0,-1.0) / 180.0]
    set alpha1 [expr 135.0 * $deg2rad]
    set alpha2 [expr 50.8 * $deg2rad]

    node a 0.0 0.0
    node b [expr -1.0 * $R] 0.0
    node c [expr cos($alpha1) * $R] [expr sin($alpha1) * $R]
    node d 0.0 $R
    node e [expr cos($alpha2) * $R] [expr sin($alpha2) * $R]
    node f [expr 1.4 * $R] [expr 1.5 *$R]
    node g [expr 1.5 * $R] [expr 2.5 * $R]
    node h [expr 1.5 * $R] [expr 3.5 * $R]
    node i [expr -1.5 * $R] 0.0
    node j [expr -1.5 * $R] [expr 1.5 * $R]
    node k [expr -1.0 * $R] [expr 2.8 * $R]

    # east boundary
    arc  bc b c a
    arc  cd c d a
    arc  de d e a

    # north boundary (reversed)
    bezier eh e f g h

    # south boundary
    line ib i b

    # west boundary
    bezier ih i j k h

    polyline n0   1 - eh
    polyline s0   1 + ib
    polyline e0   3 + bc + cd + de
    polyline w0   1 + ih
END_GEOMETRY
```

```
BEGIN_FLOW
    gas_type $gas_name
    gas_state inflow $p_inf $u_inf 0.0 $T_inf 1.0
    gas_state initial [expr 0.3 * $p_inf] 0.0 0.0 $T_inf 1.0

    discretise n0  $nn 0 1 1.2
    discretise s0  $nn 0 1 1.2
    discretise e0  $nn 1 0 1.1
    discretise w0  $nn 1 0 1.1

    boundary_spec w0 sup_in inflow
    boundary_spec n0 sup_out
    boundary_spec e0 fixed_T $T_body

    block block-0 + n0 + e0 + s0 + w0

    fill_block block-0 initial
END_FLOW

BEGIN_CONTROL
    set TitleText "Blunt Body $JOB: R=$R, gas=$gas_name, p=$p_inf, v=$u_inf,"
    append TitleText " T=$T_inf, viscous=$do_viscous"
    title $TitleText
    case_id 0
    if { $do_viscous } {
        viscous
        viscous_delay $t_plot
    }
    if { [string equal $body_type "sphere"] } {
        axisymmetric
    }
    flux_calc adaptive
    max_time  $t_final
    max_step  400000
    time_step 1.0e-8
    cfl 0.40
    dt_plot   $t_plot
    history_cell block-0 $nn 1
    dt_history 1.0e-6
END_CONTROL

bezier_file $JOB.bez
param_file  $JOB.p
mpost_file  $JOB.mpost
mpost_scales [expr 0.02 / $R] [expr 0.02 / $R]
# The following specs for the axes required a bit of fiddling
# to get the desired effect.
# If you change the radius, you'll probably have to adjust them again.
mpost xaxis -0.060 0.050 0.020 -0.010
mpost yaxis  0.0   0.110 0.020 -0.060
build

quit
```

3.2 Shell scripts

```
#!/bin/sh
# file: ss3_setup_lut.sh

cp ~/cfcfd/code/cea_tables/cea_table_air.txt ./cea_table.txt
cea_to_binary.exe -extrapolate
mv cea_lut.dat lut.dat

echo "We should now have a Look-Up-Table for air"
```

```
# ss3_run.sh
# Shell script to set up and run Sawada & Dendou's sphere case 3.
#

# For a clean start
scriptit.tcl -f ss3.sit -do-mpost > ss3.scriptit.log
mpost ss3.mpost
mb_prep.exe -f ss3

# The main event
time mb_cns.exe -f ss3
```

```
# ss3_post.sh
#
TFINAL=67.0e-6

mb_post.exe -fp ss3.p -fg ss3.g -fs ss3.s -fo ss3_final -t $TFINAL \
    -logrho -generic

mb_cont.exe -fi ss3_final.gen -fo ss3_final_mesh.ps -ps -var 6 -mesh \
   -xrange -40.0e-3 40.0e-3 20.0e-3 -yrange 0.0 80.0e-3 20.0e-3
mb_cont.exe -fi ss3_final.gen -fo ss3_final_p.ps -ps -var 6 -colour \
   -xrange -40.0e-3 40.0e-3 20.0e-3 -yrange 0.0 80.0e-3 20.0e-3
mb_cont.exe -fi ss3_final.gen -fo ss3_final_T.ps -ps -var 5 -colour \
   -xrange -40.0e-3 40.0e-3 20.0e-3 -yrange 0.0 80.0e-3 20.0e-3
mb_cont.exe -fi ss3_final.gen -fo ss3_final_M.ps -ps -var 7 -colour \
   -xrange -40.0e-3 40.0e-3 20.0e-3 -yrange 0.0 80.0e-3 20.0e-3
mb_cont.exe -fi ss3_final.gen -fo ss3_final_logrho.ps -ps -var 2 -colour \
   -xrange -40.0e-3 40.0e-3 20.0e-3 -yrange 0.0 80.0e-3 20.0e-3

mb_prof.exe -fp ss3.p -fg ss3.g -fs ss3.s -fo ss3_stag_line.data \
    -t $TFINAL -yline 0 1
awk -f locate_shock.awk ss3_stag_line.data > ss3.result
```

3.3 Notes

- This simulation reaches a final time of 67.95 μs in 5192 steps and, on a Celeron 2.4 Ghz system, this takes 7 min, 53 s of CPU time. This timing will be a bit sensitive to the state of the code because the large data structures appear to be causing a lot of cache misses. If we cut down on the amount of storage for each cell and reduce the size of the temporary arrays, we can achieve significant reductions in the CPU time.

- Awk script for extracting the shock location from the stagnation-line flow data.

```
# locate_shock.awk

BEGIN {
     p_old = 0.0;
     x_old = -2.0;    # dummy position
     y_old = -2.0;
     p_trigger = 2.0e6; # something midway between free stream and stagnation
     shock_found = 0;
}
```

```
$1 != "#" { # for any non-comment line, do something
    p_new = $7;
    x_new = $1;
    y_new = $2;
    # print "p_new=", p_new, "x_new", x_new, "y_new", y_new
    if ( p_new > p_trigger && shock_found == 0 ) {
        shock_found = 1;
        frac = (p_new - p_trigger) / (p_new - p_old);
        x = x_old + frac * (x_new - x_old);
        y = y_old + frac * (y_new - y_old);
        print "shock-location= ", x, y
    }
    p_old = p_new;
    x_old = x_new;
    y_old = y_new;
}

END {
    if ( shock_found == 0 ) {
        print "shock not located";
    }
    print "done."
}
```

4 Hypersonic flow of ideal air over a blunt wedge

This example is a partial solution to the CFD exercise for the MECH4470 class in 2004. Because the original specification was given in nondimensional form, an arbitrary 10 mm nose radius has been selected for the inviscid simulation. This is also a reasonable size for a possible wind tunnel experiment. The free-stream condition was specified as having a Mach number of 5 and the gas was specified as ideal air. Choosing particular values of $p_\infty = 100\,\text{kPa}$, $T_\infty = 100\,\text{K}$, lead to a free-stream velocity of $u_\infty = 1002\,\text{m/s}$ and a dynamic pressure of $q_\infty = 1.75\,\text{MPa}$.

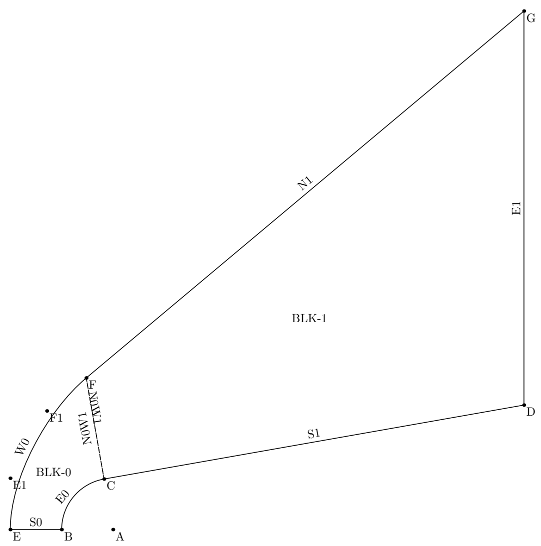


Figure 8: Schematic diagram of the geometry for the blunted 10 degree wedge.

The simulation is started with low-pressure conditions throughout the flow domain and free-stream conditions applied to the inflow boundary (the west boundary of blk-1 and the north boundary of blk-1). The flow data is allowed to evolve until $t_{final} = 399\,\mu\text{s}$, which corresponds to a particle of the free-stream travelling 40 nose radii. The axial force (shown in Fig.10) is seen to settle to a value of 28260 N in that time. This corresponds to a drag coefficient of 0.666.

The surface pressure (shown normalised in Fig. 11) has been extracted from the solution file by `mb_prof` by selecting the east-most line of cells of the first block and the south-most line of cells of the second block. The selected data is filtered by an Awk script to produce the normalised data (and the Newtonian reference data) as plotted.

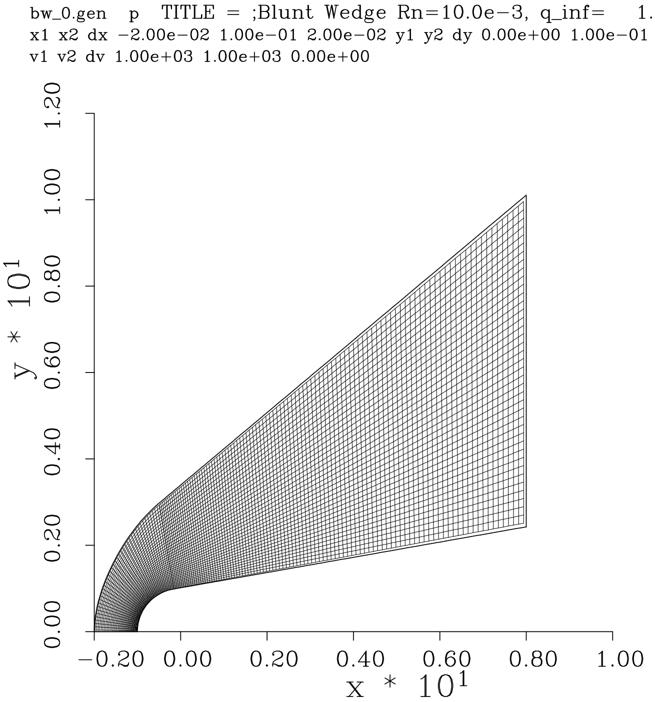


Figure 9: Mesh for the blunt wedge exercise.

Figure 10: History of the axial forces for the blunt-wedge exercise.

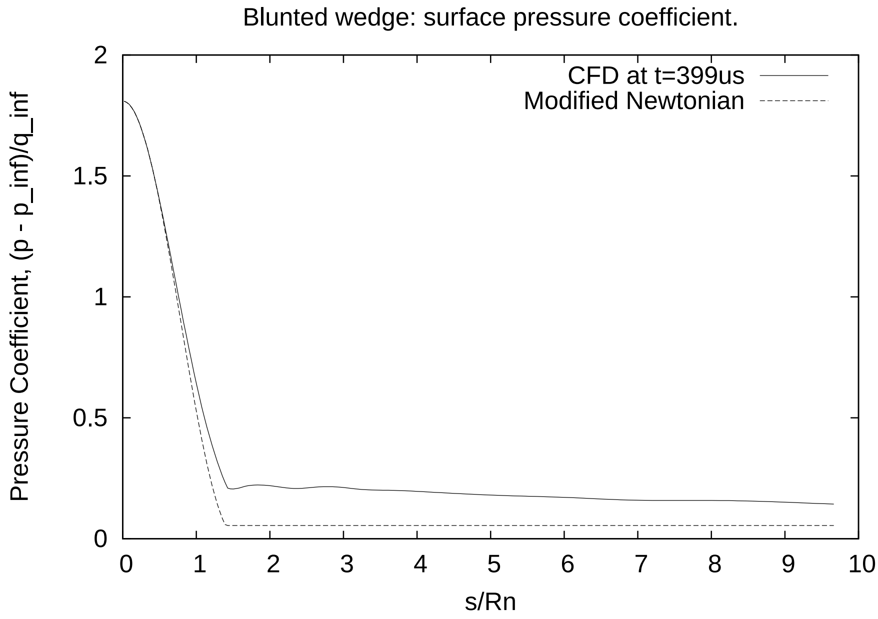


Figure 11: Surface pressure coefficient data for the blunt-wedge exercise.

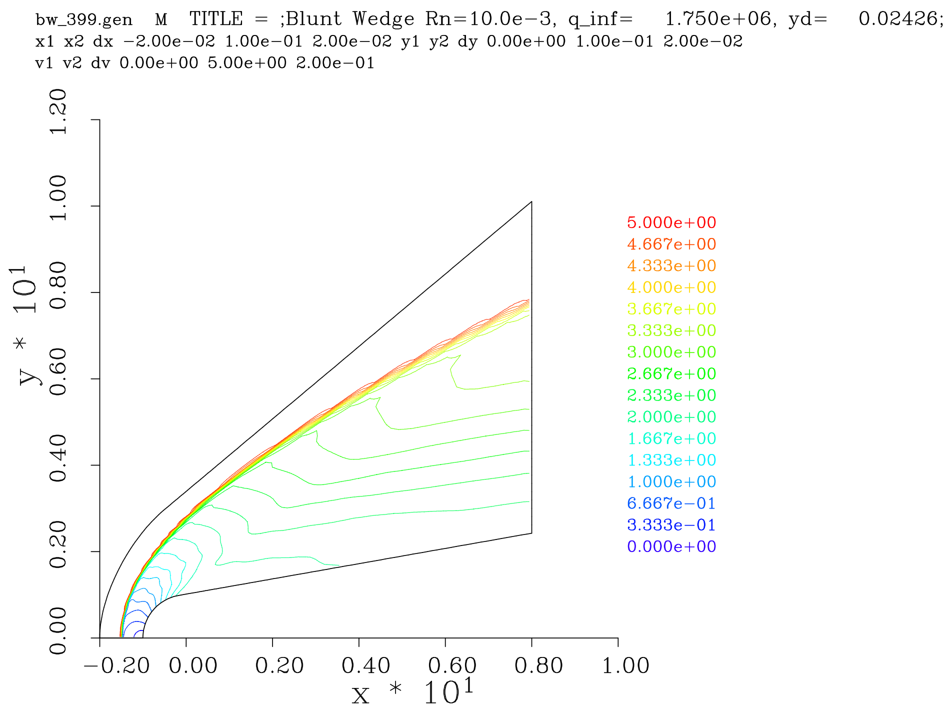


Figure 12: Mach number data for the blunt-wedge exercise.

4.1 .sit file

```
# bw.sit
# MECH4470/CFD Exercise: Hypersonic flow over a blunt wedge.
# PJ, 06-Oct-04

# Geometry
set Rn    10.0e-3;                 # radius of cylindrical nose
set xEnd  [expr 8.0 * $Rn];        # downstream extent of wedge
set alpha [to_radians 10.0];       # angle of wedge wrt free stream
set delta 10.0e-3;                 # offset for inflow boundary

# Free stream
set g_gas 1.4;                     # Ideal Air
set R_gas 287.0;
set M_inf 5.0;                     # Specified Mach number
set p_inf 100.0e3;                 # Select a static pressure
set T_inf 100.0;                   # and a temperature
# We need to determine velocity.
set a_inf [expr sqrt($T_inf * $R_gas * $g_gas)]; # sound speed
set u_inf [expr $M_inf * $a_inf];                # velocity
# Also, handy to know dynamic pressure for nondimensionalization
# of the pressures and drag forces.
set q_inf [expr 0.5 * $g_gas * $p_inf * $M_inf * $M_inf]
puts "Free-stream velocity, u_inf= $u_inf"
puts "      static  pressure, p_inf= $p_inf"
puts "     dynamic  pressure, q_inf= $q_inf"

# For transient simulation, we start with a low pressure.
set p_init 1000.0;
set T_init  100.0;

BEGIN_GEOMETRY
    # First, specify surface of cylinder and wedge
    node a  0.0 0.0; # Centre of curvature for nose
    set  xb [expr -1.0 * $Rn]
    node b  $xb 0.0
    set  xc [expr -1.0 * $Rn * sin($alpha)]
    set  yc [expr $Rn * cos($alpha)]
    node c  $xc $yc
    arc  bc b c a
    # Down-stream end of wedge
    set  yd [expr $yc + ($xEnd - $xc) * tan($alpha)]
    node d  $xEnd $yd
    puts "height at end of plate yd= $yd"
    line cd c d

    # Outer-edge of flow domain has to contain the shock layer
    # Allow sufficient for shock stand-off at the stagnation line.
    set R2 [expr $Rn + $delta]
    set  xe [expr -1.0 * $R2]
    node e  $xe 0.0
    # The shock angle for a 10 degree ramp with sharp leading edge
    # is 20 degrees (read from NACA 1135, chart 2),
    # however, the blunt nose displaces the shock a long way out
    # so we allow some more space.
    # We need to set the boundary high enough to avoid the shock
    set  R3 [expr $Rn + 2.0 * $delta]
    set  xf [expr -1.0 * $R3 * sin($alpha)]
    set  yf [expr $R3 * cos($alpha)]
    node f  $xf $yf
    # Now, put in intermediate control points so that we can use
    # cubic Bezier curve for the inflow boundary around the nose
    # and a straight line downstream of point f.
    node e1 $xe $delta
    set  alpha2 [to_radians 40.0]
    set  xf1 [expr $xf - $delta * cos($alpha2)]
    set  yf1 [expr $yf - $delta * sin($alpha2)]
    node f1 $xf1 $yf1
```

```
    bezier  ef e e1 f1 f
    set  yg [expr $yf + ($xEnd - $xf) * tan($alpha2)]
    node g  $xEnd $yg
    line fg f g

    # Define straight-line segments between surface
    # and outer boundary.
    line eb e b
    line fc f c
    line dg d g

    # Assemble the curve segments into polylines
    # that will become the block boundaries.
    polyline n0w1 1 + fc
    polyline s0   1 + eb
    polyline e0   1 + bc
    polyline w0   1 + ef
    polyline n1   1 + fg
    polyline e1   1 + dg
    polyline s1   1 + cd
END_GEOMETRY

BEGIN_FLOW
    gas_type PERF_AIR_14
    gas_state inflow  $p_inf   $u_inf 0.0 $T_inf   1.0
    gas_state initial $p_init  0.0    0.0 $T_init  1.0

    set nnx0 40
    set beta0 1.2
    set nny0 40
    discretise n0w1 $nnx0 0 1 $beta0
    discretise s0   $nnx0 0 1 $beta0
    discretise e0   $nny0 0 0 0.0
    discretise w0   $nny0 0 0 0.0

    set nnx1 100
    set nny1 $nnx0;  # connecting a north edge to a west edge
    set beta1 1.2
    discretise n1   $nnx1 1 0 $beta1
    discretise s1   $nnx1 1 0 $beta1
    discretise e1   $nny1 0 0 0.0

    boundary_spec w0 sup_in inflow
    boundary_spec n1 sup_in inflow
    boundary_spec e1 extrapolate_out

    block blk-0 + n0w1 + e0   + s0   + w0
    block blk-1 + n1   + e1   + s1   - n0w1

    connect_blocks blk-0 north blk-1 west

    fill_block blk-0 initial
    fill_block blk-1 initial
END_FLOW

BEGIN_CONTROL
    set title_string "Blunt Wedge Rn=$Rn, "
    append title_string "q_inf=[format "%12.3e" $q_inf], "
    append title_string "yd=[format "%10.5f" $yd]"
    title $title_string
    case_id 0
    flux_calc adaptive
    set t_final [expr 40.0 * $Rn / $u_inf]
    puts "Final time= $t_final"
    max_time  $t_final
    max_step  500000
    time_step 1.0e-8
    dt_plot   $t_final
    history_cell blk-0 $nnx0 1
    dt_history [expr $t_final / 100.0]
END_CONTROL
```

```
bezier_file bw.bez
param_file  bw.p
mpost_file  bw.mpost
mpost_scales 1.5 1.5
build

quit
```

4.2 Shell scripts

```
# bw_prep.sh
#
scriptit.tcl -f bw.sit -do-mpost > bw.scriptit.log
mpost bw.mpost
mb_prep.exe -f bw
mb_post.exe -fp bw.p -fg bw.g -fs bw.s0 -fo bw_0 -generic

XMIN=-20.0e-3
XMAX=100.0e-3
YMIN=0.0
YMAX=100.0e-3
TIC=20.0e-3
mb_cont.exe -fi bw_0.gen -fo bw_0_mesh.ps -ps -var 6 -mesh \
        -xrange $XMIN $XMAX $TIC -yrange $YMIN $YMAX $TIC
```

```
# bw_run.sh
#
time mb_cns.exe -f bw -force 0 1 -force 1 2
mv mb_cns.log bw.mb_cns.log
echo "Done"
```

```
# bw_post.sh

TIMES="399"
XMIN=-20.0e-3
XMAX=100.0e-3
YMIN=0.0
YMAX=100.0e-3
TIC=20.0e-3

for TME in $TIMES
do
    mb_post.exe -fp bw.p -fg bw.g -fs bw.s -fo bw_$TME -t $TME.0e-6 -generic

    mb_cont.exe -fi bw_$TME.gen -fo bw_"$TME"_p.ps -ps -var 6 -colour \
        -xrange $XMIN $XMAX $TIC -yrange $YMIN $YMAX $TIC
    mb_cont.exe -fi bw_$TME.gen -fo bw_"$TME"_M.ps -ps -var 7 -colour \
        -levels 0.0 5.0.0 0.2 \
        -xrange $XMIN $XMAX $TIC -yrange $YMIN $YMAX $TIC
    mb_cont.exe -fi bw_$TME.gen -fo bw_"$TME"_sonic.ps -ps -var 7 -colour \
        -levels 0.0 1.0 0.2 \
        -xrange $XMIN $XMAX $TIC -yrange $YMIN $YMAX $TIC
done
```

```
# bw_force.sh

# Plot the surface pressure on the wedge
TME=399
NX=40
mb_prof.exe -fp bw.p -fg bw.g -fs bw.s -fo bw_surface.dat \
    -t $TME.0e-6 -xline 0 $NX -yline 1 1
awk -f surface_pressure.awk bw_surface.dat > bw_surface_p_coeff.dat

gnuplot <<EOF
set term postscript eps 20
set output "bw_surface_pressure.eps"
set title "Blunted wedge: surface pressure coefficient."
set xlabel "s/Rn"
set ylabel "Pressure Coefficient, (p - p_inf)/q_inf"
set yrange [0.0:2.0]
plot "bw_surface_p_coeff.dat" using 1:2 title "CFD at t=399us" with lines, \
     "bw_surface_p_coeff.dat" using 1:3 title "Modified Newtonian" with lines
EOF

# Plot the axial force coefficient.
awk -f xforce.awk bw.mb_cns.log > bw_xforce.dat

gnuplot <<EOF
set term postscript eps 20
set output "bw_xforce.eps"
set title "Blunted wedge: x-force history"
set xlabel "t, microseconds"
set ylabel "x-force, N"
set yrange [0:35000]
set key top left
plot "bw_xforce.dat" using 1:2 title "total" with lines, \
     "bw_xforce.dat" using 1:3 title "cylinder" with lines, \
     "bw_xforce.dat" using 1:4 title "wedge" with lines
EOF
```

4.3 Notes

- This simulation reaches a final time of 399 μs in 5223 steps and, on a Celeron 2.4 Ghz system, this takes 10 min, 11 s of CPU time.
- Selection of the `mb_cns.log` file showing some x-force data as written during the simulation. See the function `print_forces()` in `cns_xforce.c` for details of the format.

```
Step=    420 t= 2.378e-05 dt= 5.958e-08 WC=50.0 WCtFT=749.9 WCtMS=59473.8
CFL_min = 1.645476e-03, CFL_max = 4.949394e-01, dt_allow = 5.957893e-08
Smallest CFL_max so far = 3.381198e-02 at t = 1.000000e-07
 dt[0]=5.957893e-08 dt[1]=7.096680e-08
There are 2 active blocks.
Global Residual (for density) = 1.252045e-01
XFORCE: 2.396144e-05 2 0 1 2.372006e+04 0.000000e+00 1 2 7.255583e+02 0.000000e+00
```

- Awk filter for extracting the x-force data from the simulation log file.

```
# xforce.awk
# Extract the simulation times and axial force values from the log file.

BEGIN {
    print "# time (microseconds) x-force-total  only-cylinder only-wedge";
```

```
}

/XFORCE/ {
    # Select just the simulation time and the pressure forces.
    t = $2;  # in seconds
    fx_p_0 = $6;  # force on cylinder in Newtons
    fx_p_1 = $10; # wedge surface
    print $2*1.0e6, fx_p_0 + fx_p_1, fx_p_0, fx_p_1;
}
```

- Awk filter for normalising the surface pressure data.

```
# surface_pressure.awk
# Normalise the surface pressure with free-stream dynamic pressure and
# compute the distance around from the stagnation point.

BEGIN {
    q_inf = 1.750e6;    # free-stream dynamic pressure
    p_inf = 100.0e3;    # free-stream static pressure
    Rn = 10.0e-3;       # nose radius
    xold = -Rn;         # location of the stagnation point
    yold = 0.0;
    s = 0.0;            # distance around from stagnation point
    count = 0;
    pi = 3.1415927;
    cone_angle = 10.0/180.0 * pi;
    print "# s/Rn  Cp(CFD)  Cp(Newton)  x(m)  y(m)";
}

$1 != "#" {
    count += 1;
    x = $1;             # cell-centre position
    y = $2;
    p = $7;             # cell-centre pressure
    if ( count == 1 ) p_pitot = p;  # Close enough to the stagnation point.
    dx = x - xold;
    dy = y - yold;
    s += sqrt(dx * dx + dy * dy);
    # Estimate Cp using Modified Newtonian Model.
    theta = 0.5 * pi - (s/Rn);  # local angle of surface
    if (theta < cone_angle) theta = cone_angle;
    Cp_MN = (p_pitot - p_inf) / q_inf * sin(theta) * sin(theta);
    print s/Rn, (p - p_inf)/q_inf, Cp_MN, x, y;
    xold = x;
    yold = y;
}
```

5 Mach 3 flow over a sharp-nosed two-dimensional body

The specifications for this example come from section 5.2 in JD Anderson's Hypersonics book [5]. It shows the use of a `spline` curve as well as being a source of test data for the Method-of-Characteristics for rotational flow. Data for the spline points was computed from

$$\frac{y}{y_e} = -0.008333 + 0.609425\left(\frac{x}{y_e}\right) - 0.092593\left(\frac{x}{y_e}\right)^2$$

where $y_e = 1.0$.

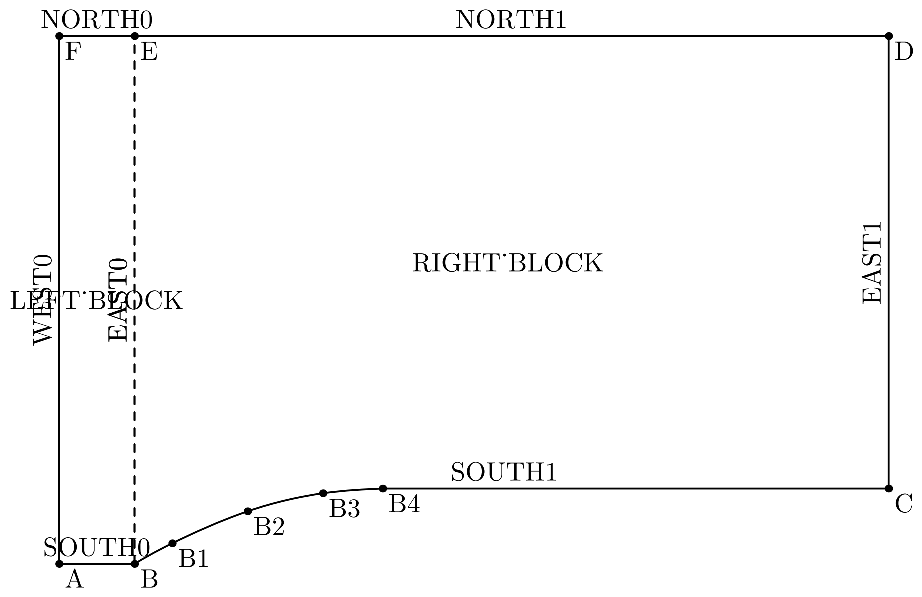


Figure 13: Schematic diagram of the geometry for the sharp body.

The surface pressure (shown in Fig. 15) has been extracted from the solution file by `mb_prof` by selecting the south-most line of cells of both blocks. The pressure field (Fig. 16) shows the curved shock clearly.

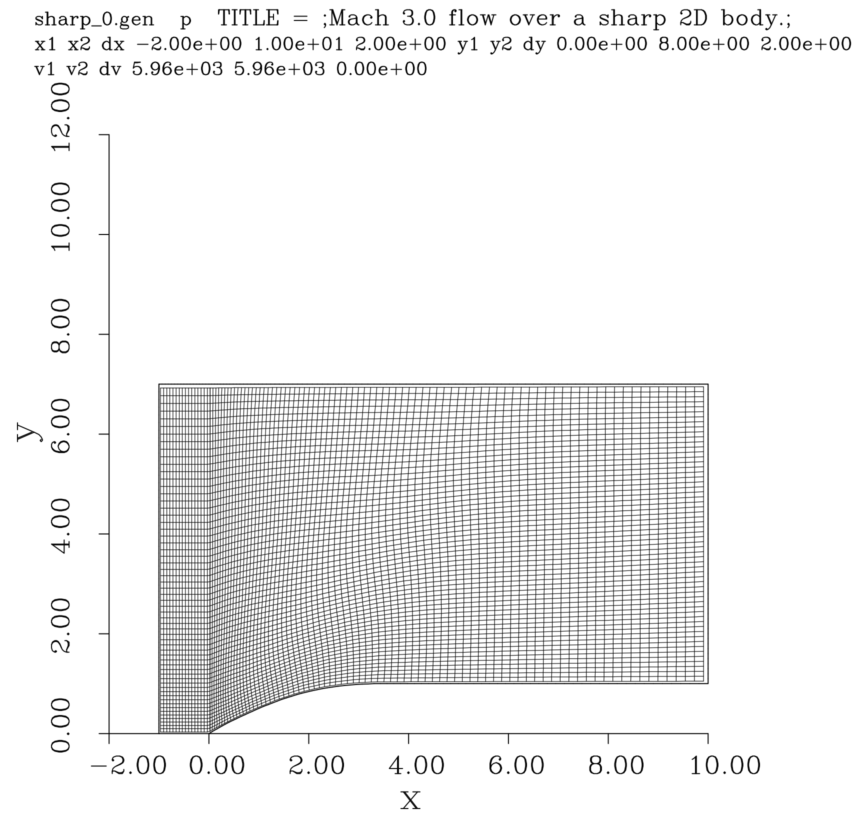


Figure 14: Mesh for the sharp body exercise.

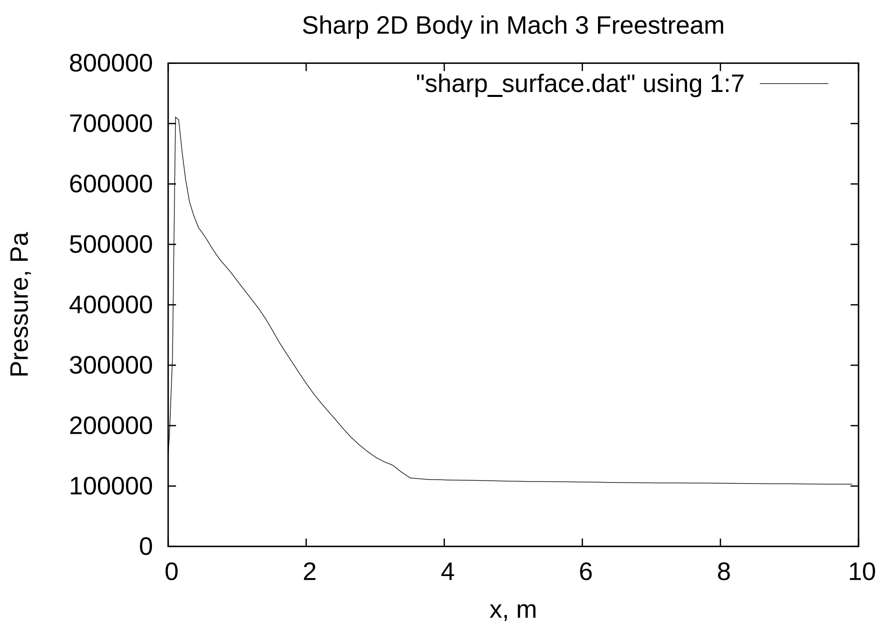


Figure 15: Pressure data along the body surface.

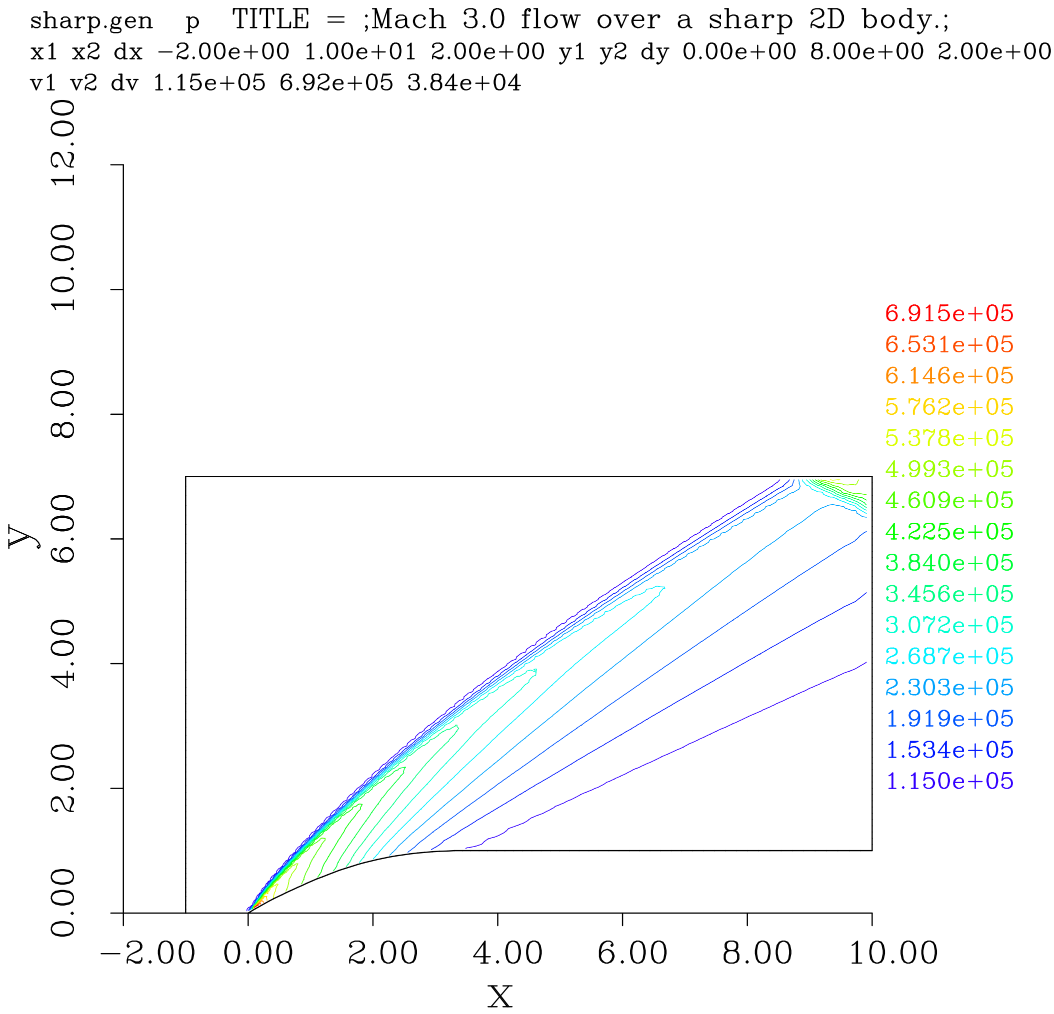


Figure 16: The pressure field for flow over a sharp body. Note that the shock reflects from the upper boundary, which has a SLIP_WALL boundary condition by default.

5.1 .sit file

```
# sharp.sit
# Mach 3.0 flow over a curved 2D-planar body.

# Set up two blocks, one upstream of the body.
BEGIN_GEOMETRY
   NODE a  -1.0    0.0
   NODE b   0.0    0.0
   NODE b1  0.5    0.2732
   NODE b2  1.5    0.6975
   NODE b3  2.5    0.9365
   NODE b4  3.291  1.0
   NODE c   10.0   1.0
   NODE d   10.0   7.0
   NODE e   0.0    7.0
   NODE f  -1.0    7.0

   LINE    ab   a b
   SPLINE  bb4  4 b b1 b2 b3 b4
   LINE    b4c  b4 c
   LINE    af   a f
   LINE    be   b e
   LINE    cd   c d
   LINE    fe   f e
   LINE    ed   e d

   # Define the boundaries
   POLYLINE north0 1 + fe
   POLYLINE east0  1 + be
   POLYLINE south0 1 + ab
   POLYLINE west0  1 + af
   POLYLINE south1 2 + bb4 + b4c
   POLYLINE east1  1 + cd
   POLYLINE north1 1 + ed
END_GEOMETRY

BEGIN_FLOW
   # Gas and flow properties
   GAS_TYPE perf_air_14
   GAS_STATE initial 5955.0     0.0 0.0  304.0 1.0
   GAS_STATE inflow  95.8e3 2000.0 0.0 1103.0 1.0

   # Set the boundary discretisation before building the blocks
   DISCRETISE north0 16 0 0 0.0
   DISCRETISE east0  60 1 0 1.3
   DISCRETISE south0 16 0 0 0.0
   DISCRETISE west0  60 1 0 1.3
   DISCRETISE north1 80 1 0 1.2
   DISCRETISE south1 80 1 0 1.2
   DISCRETISE east1  60 0 0 0.0

   # Inflow and outflow boundaries
   BOUNDARY_SPEC west0 SUP_IN  inflow
   BOUNDARY_SPEC east1 SUP_OUT

   # Define two blocks with a common boundary
   BLOCK left_block  + north0 + east0 + south0 + west0
   BLOCK right_block + north1 + east1 + south1 + east0
   CONNECT_BLOCKS left_block east right_block west
   GRID_TYPE right_block AO

   # Assign the initial gas states
   FILL_BLOCK left_block  initial
   FILL_BLOCK right_block initial
END_FLOW

BEGIN_CONTROL
   TITLE Mach 3.0 flow over a sharp 2D body.
```

```
    CASE_ID 0

    FLUX_CALC ausmdv
    MAX_TIME  15.0e-3
    MAX_STEP  2500
    TIME_STEP 1.0e-6
END_CONTROL

# Name the output files and build them.
BEZIER_FILE sharp.bez
PARAM_FILE sharp.p
MPOST FILE sharp.mpost
MPOST SCALES 0.01 0.01
BUILD

QUIT
```

5.2 Shell scripts

```
#!/bin/sh
# sharp_prep.sh
# A sharp axisymmetric body as described in Andersons Hypersonics text.

# Generate the Bezier and Input parameter files from the Script File.
scriptit.tcl -f sharp.sit -do-mpost > sharp.scriptit.log
mpost sharp.mpost

# Generate the Grid and Initial Solution Files.
mb_prep.exe -f sharp

# Extract the initial solution data and reformat.
mb_post.exe -fp sharp.p -fg sharp.g -fs sharp.s0 -fo sharp_0 -generic

# Pick up the reformatted data and make a mesh plot.
mb_cont.exe -fi sharp_0.gen -fo sharp_0_mesh.ps -var 6 -ps -mesh \
    -mirror -xrange -2.0 10.0 2.0 -yrange 0.0 8.0 2.0

echo At this point, we should be ready to start the simulation.
```

```
#!/bin/sh
# sharp_run.sh
# Exercise the Navier-Stokes solver for a sharp axisymmetric body.

# Integrate the solution in time.
time mb_cns.exe -f sharp

echo At this point, we should have a final solution in sharp.s
```

```
#!/bin/sh
# sharp_post.sh
# Sharp axisymmetric body, extract data and plot it.

# Extract the solution data over whole flow domain and reformat.
mb_post.exe -fp sharp.p -fg sharp.g -fs sharp.s -fo sharp -generic

# Pick up the reformatted data and make a contour plot.
mb_cont.exe -fi sharp.gen -fo sharp.gif -var 6 -gif -colour -mirror \
```

```
    -xrange -2.0 10.0 2.0 -yrange 0.0 8.0 2.0

mb_cont.exe -fi sharp.gen -fo sharp_p.ps -var 6 -ps -colour -mirror \
    -xrange -2.0 10.0 2.0 -yrange 0.0 8.0 2.0

# Extract surface pressure and plot.
mb_prof.exe -fp sharp.p -fg sharp.g -fs sharp.s -fo sharp_surface.dat \
    -yline 0 1 -yline 1 1

gnuplot <<EOF
set term postscript eps 20
set output "sharp_surface_p.eps"
set title "Sharp 2D Body in Mach 3 Freestream"
set xlabel "x, m"
set ylabel "Pressure, Pa"
set xrange [0.0:10.0]
set yrange [0.0:800.0e3]
plot "sharp_surface.dat" using 1:7 with lines
EOF

echo At this point, we should have a plotted data.
```

5.3 Notes

- This simulation reaches a final time of 15 ms in 2021 steps and, on a Celeron 2.4 Ghz system, this takes 3 min, 40 s of CPU time.

6 Flow through a conical nozzle

Good quality experimental data for wall pressure distribution in a conical nozzle with a circular-arc throat profile and a is available in Ref. [6]. In the original experiment the flow of air through the facility was allowed to reach steady state and static pressures were measured at a large number of points along the nozzle wall. In contrast, the present simulation is transient with just the transonic plus supersonic parts of the flow field reaching steady state.

Figure 17 shows the outline of the simulated flow domain which is set up to approximate the largest subsonic area ratio used in the experiment. The relatively long upstream part of the simulated tube provides the gas through an unsteady expansion from zero speed and pressure of 500 kPa (state 4) up to a small Mach number (state 3). These state labels refer to the those for the hypothetical shock tube problem in which state 1 is the initial low-pressure condition, state 4 is the initial high-pressure condition, state 2 is the post-shock condition of the low-pressure gas, and state 3 is the expanded high-pressure gas condition. Assuming that flow in the subsonic and transonic regions of the nozzle is steady, the expected Mach number is $M_3 = 0.13812$ for an area ratio of $A_3/A_* = 4.2381$.

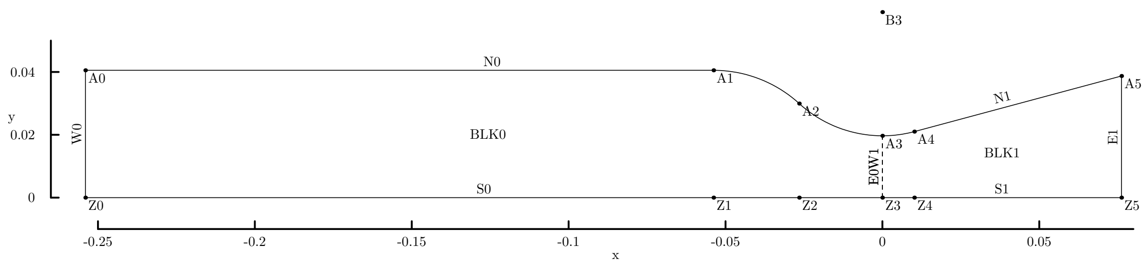


Figure 17: Schematic diagram of the full flow domain for the duct and conical nozzle.

Along the C_+ characteristic connecting states 4 and 3, the Riemann invariant can be written as

$$J_+ = u + \frac{2a}{\gamma - 1} \quad ,$$

and so the relation between states 4 and 3 can be written as

$$\frac{a_4}{a_3} = 1 + \frac{\gamma - 1}{2} M_3 \quad .$$

Because the unsteady expansion is isentropic and $a = \sqrt{\gamma R T}$, the pressure ratio can be written as

$$\frac{p_4}{p_3} = \left[1 + \frac{\gamma - 1}{2} M_3 \right]^{2\gamma/(\gamma-1)}$$

which gives a specific pressure ratio of $p_4/p_3 = 1.2102$. Since the experiment used a steady expansion from a large reservoir at stagnation conditions to the equivalent of our state 3, the corresponding stagnation pressure for state 3 is computed from

$$\frac{p_{03}}{p_3} = \left[1 + \frac{\gamma - 1}{2} M_3^2\right]^{\gamma/(\gamma-1)}$$

which gives $p_{03} = 418.7\,\text{kPa}$ in the current simulation.

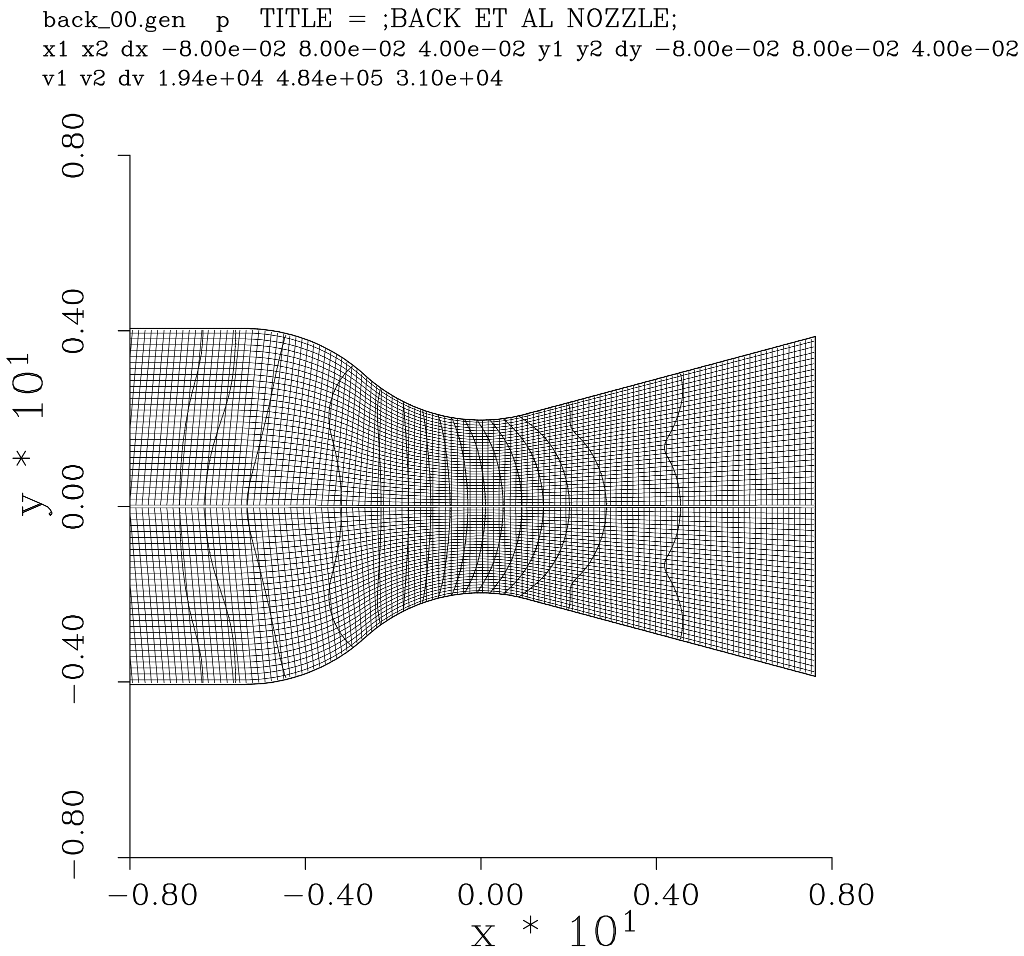


Figure 18: Mesh of lines joining the centres of every-second finite-volume cell with pressure contours superimposed.

Figure 19 shows the pressure distribution throughout the flow domain at $t = 1.0\,\text{ms}$. A shock, starting at the transition from circular arc to straight wall in the early supersonic part of the nozzle, can be seen propagating toward the centreline as the flow proceeds to the exit plane.

The flow in the nozzle is reasonably steady, as indicated by the histories shown in Fig. 20 but the unsteady expansion can be seen reflecting from the inflow boundary at $x = -0.245\,\text{m}$ in Fig.19.

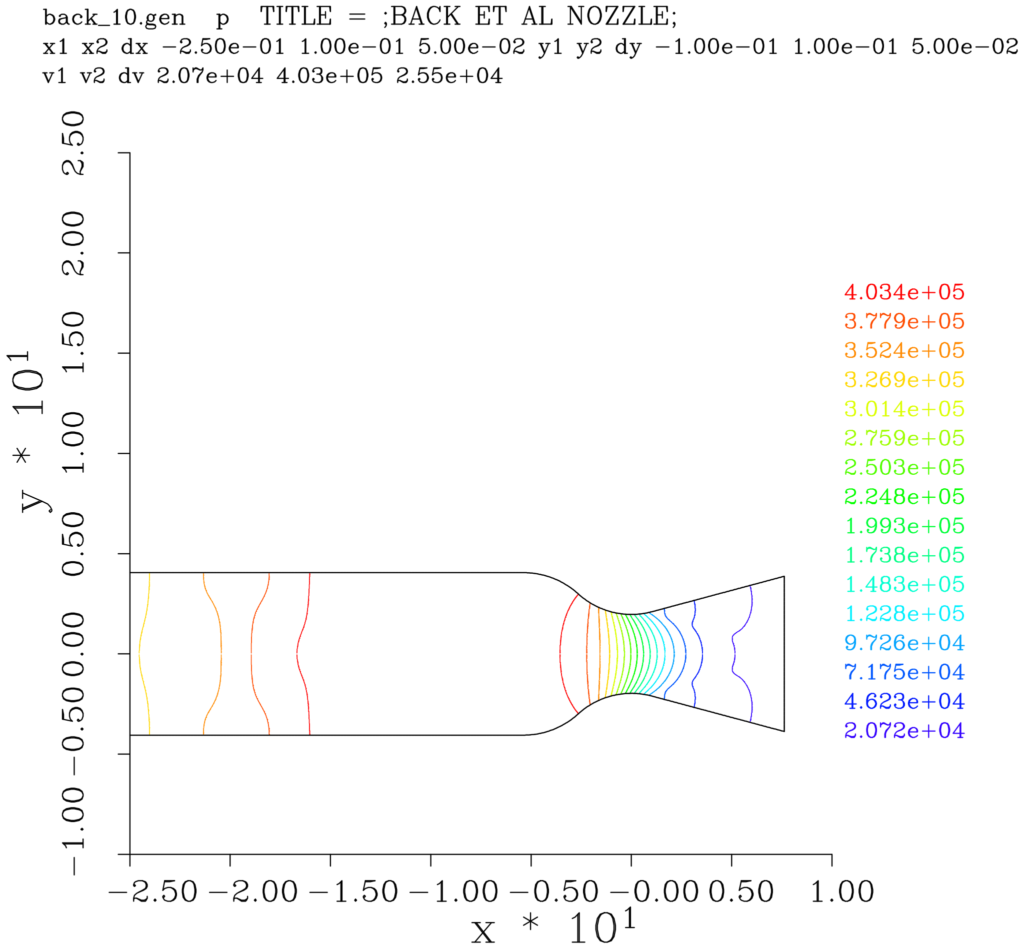


Figure 19: Pressure contours within the flow domain at 1.0 ms.

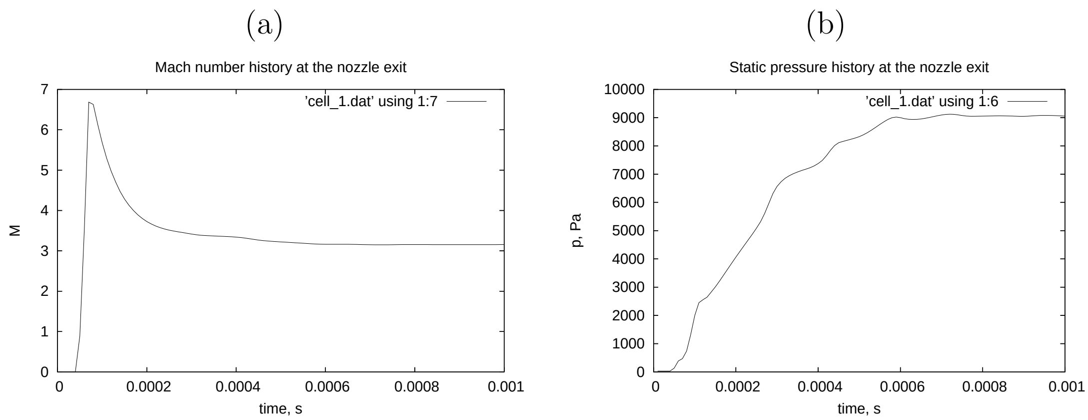


Figure 20: Development of the flow at a "history point" near the centre of the exit plane: (a) Mach number; (b) static pressure.

Figure 21 shows that the simulation matches the experimental data closely. The reflected expansion is shown clearly in the left figure and indicates that, at the time of writing this report, the subsonic boundary condition is not working as well as it should.

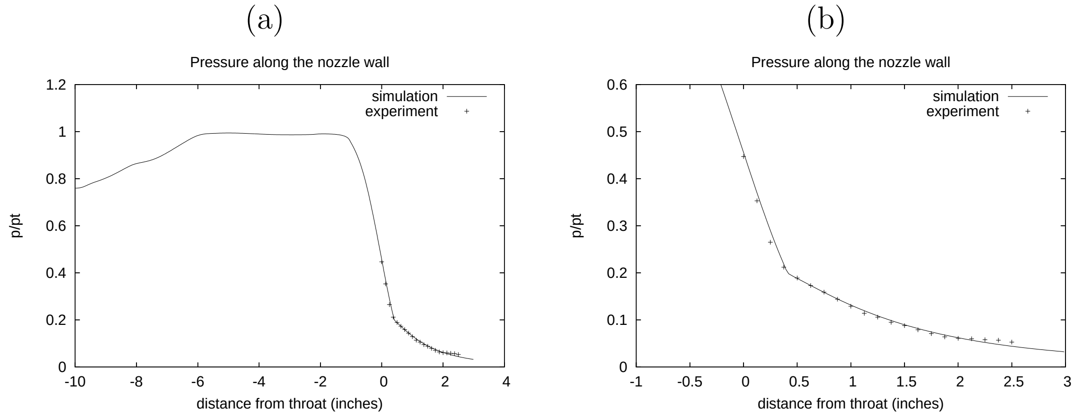


Figure 21: Normalised pressure distribution along the nozzle wall: (a) full length of flow domain; (b) just the supersonic part of the nozzle.

6.1 .sit file

```
# back.sit
# Conical nozzle from Back, Massier and Gier (1965)

BEGIN_GEOMETRY
   NODE a0  -0.254     0.040525
   NODE z0  -0.254     0.0
   NODE a1  -0.053812  0.040525
   NODE z1  -0.053812  0.0
   NODE a2  -0.026518  0.029954
   NODE z2  -0.026518  0.0
   NODE b3   0.0       0.059055
   NODE a3   0.0       0.019685
   NODE z3   0.0       0.0
   NODE a4   0.010190  0.021026
   NODE z4   0.010190  0.0
   NODE a5   0.076200  0.038712
   NODE z5   0.076200  0.0

   # Lines that run vertically.
   LINE z0a0 z0 a0
   LINE z3a3 z3 a3
   LINE z5a5 z5 a5

   # Lines that run along the x-axis
   LINE z0z3 z0 z3
   LINE z3z5 z3 z5

   # Lines and arcs for the tube and nozzle wall
   LINE a0a1 a0 a1
   ARC  a1a2 a1 a2 z1
   ARC  a2a3 a2 a3 b3
   ARC  a3a4 a3 a4 b3
   LINE a4a5 a4 a5

   # Define the boundaries that will be used to
```

```
    # build the blocks.
    POLYLINE n0    3 + a0a1 + a1a2 + a2a3
    POLYLINE s0    1 + z0z3
    POLYLINE w0    1 + z0a0
    POLYLINE e0w1 1 + z3a3

    POLYLINE n1    2 + a3a4 + a4a5
    POLYLINE s1    1 + z3z5
    POLYLINE e1    1 + z5a5
END_GEOMETRY

BEGIN_FLOW
    # Gas and flow properties
    GAS_TYPE PERF_AIR_14
    GAS_STATE stagnation   500.0e3   0.0 0.0   300.0 1.0
    GAS_STATE low_pressure 30.0      0.0 0.0   300.0 1.0

    # Set the boundary discretisation before building the blocks
    DISCRETISE n0    360 0 0 0.0
    DISCRETISE s0    360 0 0 0.0
    DISCRETISE w0     60 0 0 0.0
    DISCRETISE e0w1   60 0 0 0.0

    DISCRETISE n1    120 0 0 0.0
    DISCRETISE s1    120 0 0 0.0
    DISCRETISE e1     60 0 0 0.0

    # BOUNDARY_SPEC w0 SUBSONIC_IN stagnation
    BOUNDARY_SPEC e1 SUP_OUT

    # Define blocks
    BLOCK blk0 + n0    + e0w1 + s0    + w0
    BLOCK blk1 + n1    + e1    + s1    + e0w1

    CONNECT_BLOCKS blk0 east blk1 west

    # Assign the initial gas states
    FILL_BLOCK blk0 stagnation
    FILL_BLOCK blk1 low_pressure
END_FLOW

BEGIN_CONTROL
    TITLE Back et al nozzle
    CASE_ID 0

    AXISYMMETRIC
    FLUX_CALC Adaptive

    MAX_TIME   1.00e-3
    MAX_STEP   5000
    TIME_STEP   1.0e-7
    DT_PLOT     0.2e-3
    DT_HISTORY 10.0e-6

    HISTORY_CELL blk1    1   5    # At throat
    HISTORY_CELL blk1 120   5    # At exit plane
END_CONTROL

# Name the output files and build them.
BEZIER_FILE back.bez
PARAM_FILE   back.p
MPOST FILE   back.mpost
MPOST SCALES 0.8 0.8
MPOST XAXIS   -0.250 0.080 0.05 -0.01
MPOST YAXIS   0.0 0.050 0.020 -0.265
BUILD

EXIT
```

6.2 Shell scripts

```
#!/bin/sh
#  back.bat
#  Exercise the Navier-Stokes solver for the conical nozzle
#  as used by Back, Massier and Gier (1965) AIAA J. 3(9):1606-1614.

#  It is assumed that the path is set and that
#  the script file "back.sit" has been correctly written.

#  Stage 1:
#  Generate the input parameter and Bezier files
#  (back.p and back.bez respectively)
#  from the script file.
#  A record of the transactions is recorded in the log file
#  "back.log" so that, if anything goes wrong,
#  you can browse the log file and diagnose the problem.

# scriptit.exe < back.sit > back.log
scriptit.tcl -f back.sit -do-mpost > back.log
mpost back.mpost

#  Stage 2:
#  Pick up the input parameter and Bezier files
#  and generate the grid and initial solution files
#  (back.g and back.s0).
#  Write these files as binary data.

mb_prep.exe -wb -f back

#  Stage 3:
#  Pick up the grid and initial solution files as
#  binary data and integrate the solution in time.
#  The solution data at later times will be written
#  to the solution output file "back.s"

time mb_cns.exe -rb -wb -f back

#  Finished:
```

```
#!/bin/sh
#  back_plot.sh

# Stage 4: Extract particular times from the solution set.
# The following line extracts the first solution in the "back.s" file.
#
mb_post.exe -rb -fp back.p -fg back.g -fs back.s \
            -t 0.0 -fo back_00 -generic

# Pick up the reformatted data and make a plot of the mesh.
# Note that this plotted mesh is created by joining cell-centres.
# It is not the *true* mesh used by the flow solver.
# The ixskip, iyskip also allow the plotted mesh to be coarser
# than the true mesh.
#
mb_cont.exe -fi back_00.gen -fo back_00.ps -ps -mesh -var 6 \
            -ixskip 2 -iyskip 2 -mirror \
            -xrange -0.080 0.080 0.040 -yrange -0.080 0.080 0.040

# Static pressure contours after nozzle-flow settles.
#
mb_post.exe -rb -fp back.p -fg back.g -fs back.s \
            -t 1.0e-3 -fo back_10 -generic

mb_cont.exe -fi back_10.gen -fo back_10.ps -ps -colour -var 6 \
```

```
        -mirror \
        -xrange -0.250 0.100 0.050 -yrange -0.100 0.100 0.050

# Finished:
```

```
# back_history.sh
# Extract the flow history data at the nozzle exit plane.
# This is then plotted using gnuplot and an assessment
# can be made as to whether the flow has reached steady state.

mb_hist.exe -fi back.h -fo cell_1.dat -ncell 2 -cell 1

gnuplot <<EOF
set term postscript eps 20
set output 'back_history_M.eps'

set title 'Mach number history at the nozzle exit'
set xrange [0.0:1.0e-3]
set xlabel 'time, s'
set ylabel 'M'

plot 'cell_1.dat' using 1:7 with lines
EOF

gnuplot <<EOF
set term postscript eps 20
set output 'back_history_p.eps'

set title 'Static pressure history at the nozzle exit'
set xrange [0.0:1.0e-3]
set xlabel 'time, s'
set ylabel 'p, Pa'

plot 'cell_1.dat' using 1:6 with lines
EOF
```

6.3 Notes

- The simulation reaches a final time of 1 ms in 4535 steps and, on a Celeron 2.4 Ghz system, this takes 40 min, 36 s of CPU time. This is equivalent to 19.2 μs per cell per predictor-corrector time step.

- If the code is compiled with the GNU C compiler for debugging, the run time is approximately 1.6 times longer than for the standard optimisation level. This debugging mode eliminates optimisation, includes debugging symbols in the code and is linked to the Electric-Fence debugging library for malloc(). Peace of mind comes at a price.

- The pressure is normalised with respect to the stagnation pressure using the following AWK script.

```
# normalize.awk
# Normalize the surface pressure over the length of the nozzle.
```

```
BEGIN {
    p0 = 418.7e3
    print "# Normalized surface pressure for the Back nozzle (simulation)"
    print "# x(inches) p/pt"
}

$1 != "#" {  # For non-comment lines in the data file do...
   p = $7
   r = $2
   x = $1
   print x/0.0254, p/p0
}
```

7 A section of an ideal compressible-flow vortex

This flow example was used by Ian Johnston in his thesis and it comes with an analytic solution [7]. With respect to MB_CNS, it illustrates the use of a specified flow profile as an input and it shows the use of profile extraction, again.

The flow domain (Fig. 22) includes only part of the first quadrant of an ideal vortex flow in inviscid air with $R = 287$J/kg·K, γ=1.4). The NORTH and SOUTH boundaries are specified as reflecting walls at radii r_o and r_i, representing the outer and inner radii of the vortex segment that is centred at node A. The WEST boundary has the specified inflow as a function of radius

$$\begin{aligned}
\rho(r) &= \rho_i \left[1 + \frac{\gamma-1}{2} M_i^2 \left\{ 1 - \left(\frac{r_i}{r} \right)^2 \right\} \right]^{\frac{1}{\gamma-1}} \quad , \\
p(r) &= p_i \left(\frac{\rho}{\rho_i} \right)^{\gamma} \quad , \\
u(r) &= u_i \frac{r_i}{r} \quad ,
\end{aligned}$$

with $r_o = 1.384 R_i$ and the properties at the inner radius being $M_i = 2.25$, $\rho_i = 1.0\,\mathrm{kg/m^3}$ and $p_i = 100\,\mathrm{kPa}$.

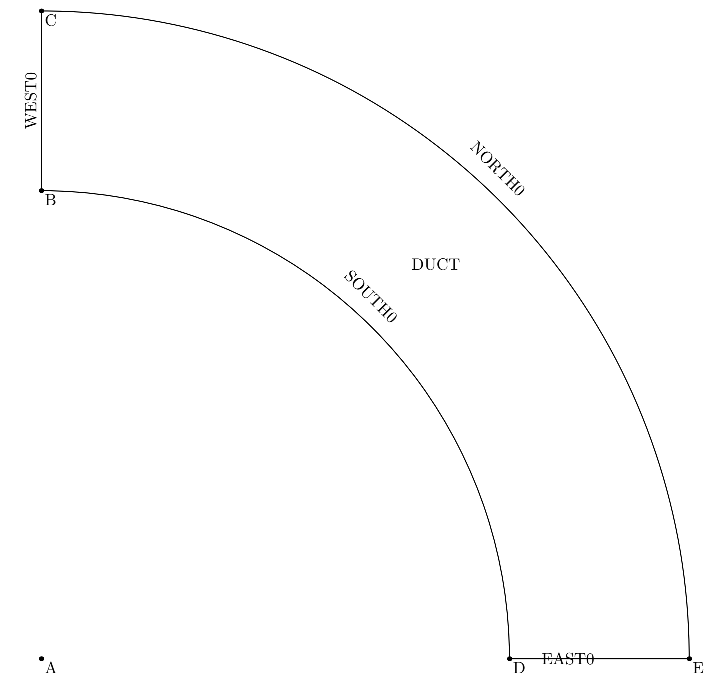


Figure 22: Schematic diagram of the first quadrant domain for the compressible-flow vortex.

Figures 23 through 25 show the radial distributions of flow properties and highlight some of the problems with the crude reflecting-wall boundary condition. Other than at the boundaries, there is close agreement between the analytic and numerical solutions. The errors at the inner and outer radii stand out clearly because we know that the trends of the flow property variations should continue at these boundaries and not mirror what is just inside the flow domain.

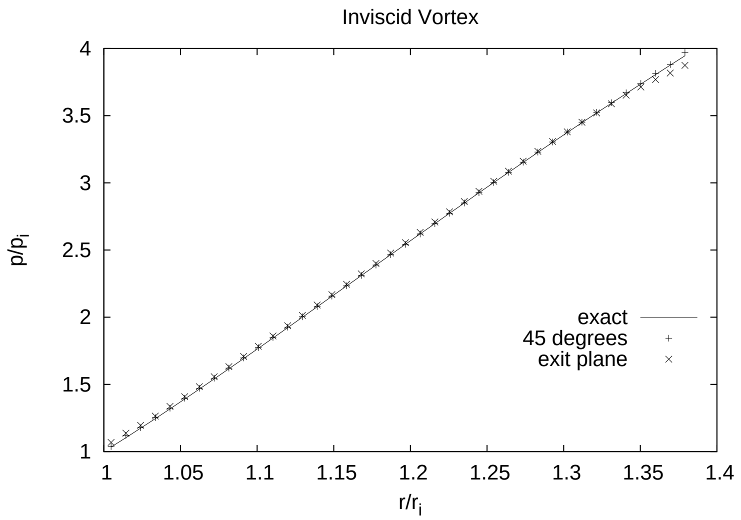


Figure 23: Radial distributions of pressure.

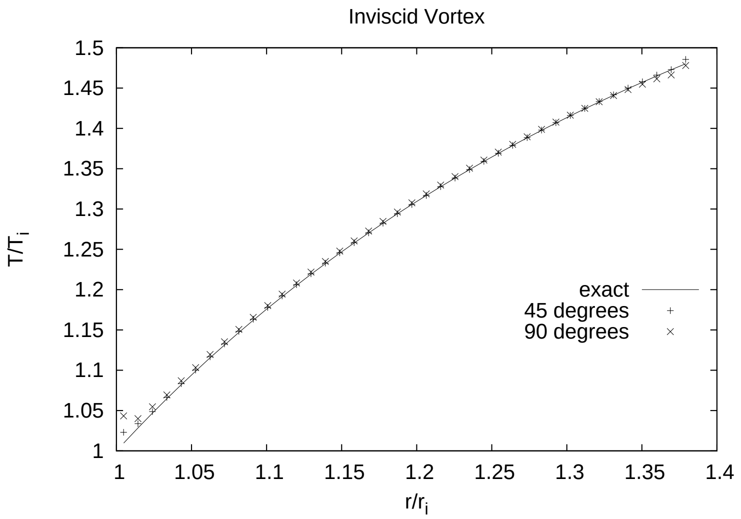


Figure 24: Radial distributions of temperature.

7.1 .sit file

```
# vtx.sit
# Inviscid supersonic vortex -- flow in a bend.
```

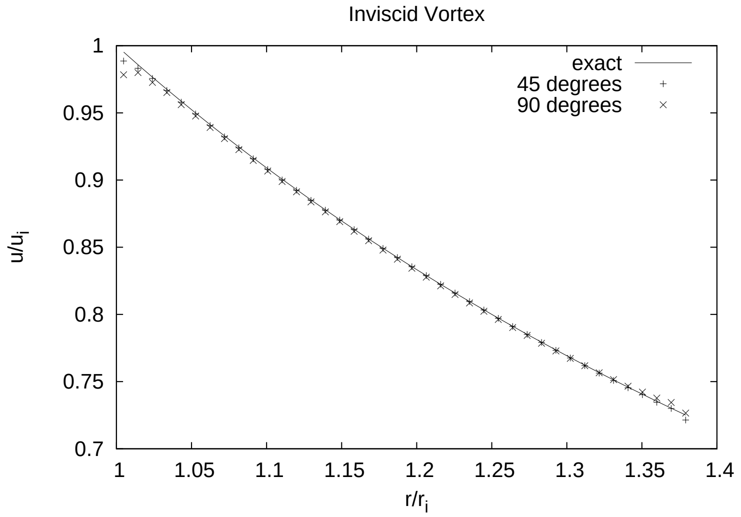


Figure 25: Radial distribution of circumferential velocity.

```
# Set up a single, curved block.
BEGIN_GEOMETRY
   NODE a 0.0   0.0
   NODE b 0.0   1.0
   NODE c 0.0   1.384
   NODE d 1.0   0.0
   NODE e 1.384 0.0

   ARC  bd b d a
   ARC  ce c e a
   LINE bc b c
   LINE de d e

   # Define the boundaries
   POLYLINE north0 1 + ce
   POLYLINE east0  1 + de
   POLYLINE south0 1 + bd
   POLYLINE west0  1 + bc
END_GEOMETRY

BEGIN_FLOW
   # Gas and flow properties
   GAS_TYPE perf_air_14
   # The following are not really important because the
   # actual data will be taken from the profile.dat file.
   GAS_STATE initial 100.0e3    0.0 0.0 348.43  1.0
   GAS_STATE inflow  1.000e3 841.87 0.0 348.43  1.0

   # Set the boundary discretisation
   DISCRETISE north0 80 0 0 0.0
   DISCRETISE east0  40 0 0 0.0
   DISCRETISE south0 80 0 0 0.0
   DISCRETISE west0  40 0 0 0.0

   # Inflow and outflow boundaries
   BOUNDARY_SPEC west0 STATIC_PROF
   BOUNDARY_SPEC east0 SUP_OUT 300.0

   BLOCK duct  + north0 + east0 + south0 + west0
   FILL_BLOCK duct  initial
END_FLOW

BEGIN_CONTROL
```

```
    TITLE Inviscid vortex flow around a bend.
    # We need to set CASE_ID 0 to get the data read from profile.dat.
    CASE_ID 0

    FLUX_CALC  ausmdv
    MAX_TIME   20.0e-3
    MAX_STEP   6000
    TIME_STEP  1.0e-6
    DT_PLOT    5.0e-3
END_CONTROL

# Name the output files and build them.
BEZIER_FILE  vtx.bez
PARAM_FILE   vtx.p
MPOST_FILE   vtx.mpost
MPOST_SCALES 0.1 0.1
BUILD

EXIT
```

7.2 Shell scripts

```
#! /bin/sh
# vtx_run.sh
# Exercise the Navier-Stokes solver for the inviscid vortex test case.
# It is assumed that the path is set correctly.

# Generate the Bezier, Input parameter and MetaPost files from the Script File.
# scriptit.exe < vtx.sit > vtx.log
scriptit.tcl -f vtx.sit -do-mpost > vtx.scriptit-log
mpost vtx.mpost

# Generate the inflow profile.
awk -f make_profile.awk

# Generate the Grid and Initial Solution Files.
mb_prep.exe -f vtx

# Integrate the solution in time.
time mb_cns.exe -f vtx

echo At this point, we should have a computed solution in vtx.s
```

```
#! /bin/sh
# vtx_plot.sh
# Exercise the Navier-Stokes solver for the inviscid vortex test case.
# It is assumed that the path is set correctly.

# Extract the solution data and reformat.
mb_post.exe -fp vtx.p -fg vtx.g -fs vtx.s -fo vtx -t 20.0e-3 -generic

# Pick up the reformatted data and make a contour plot.
mb_cont.exe -fi vtx.gen -fo vtx.ps -var 6 -ps -colour \
   -xrange 0.0 1.5 0.5 -yrange 0.0 1.5 0.5

# Extract radial profiles at 45 degrees and at 90 degrees from the inlet.
mb_prof.exe -fp vtx.p -fg vtx.g -fs vtx.s -fo vtx_profile_45.dat \
   -t 20.0e-3 -xline 0 40
awk -f extract_radial.awk vtx_profile_45.dat > radial_profile_45.dat

mb_prof.exe -fp vtx.p -fg vtx.g -fs vtx.s -fo vtx_profile_90.dat \
```

```
   -t 20.0e-3 -xline 0 80
awk -f extract_radial.awk vtx_profile_90.dat > radial_profile_90.dat

# Generate postscript plots of the radial profiles.
gnuplot radial_profile.gnu

echo At this point, we should have a plotted the solution
```

7.3 Notes

- This simulation reaches a final time of 20 ms in 5081 steps and, on a Celeron 2.4 Ghz system, this takes 3 min, 54 s of CPU time.
- The inflow that was applied to the WEST boundary as a STATIC_PROFile was generated with the following AWK script and written to the file `profile.dat`. MB_CNS looks for this file when the STATIC_PROF boundary condition is used. See comments in the `init_profile_data()` function in the C-module `cns_bc.c` for details of the expected file format.

```
# make_profile.awk
# Set up an inflow profile for the inviscid vortex case
# PJ, 20-Feb-01
#
function pow( base, exponent ) {
   # print base, exponent
   return exp( exponent * log(base) )
}

BEGIN {
   Rgas   = 287       # J/kg.K
   g      = 1.4       # ratio of specific heats

   n      = 40
   r_i    = 1.0       # metres
   r_o    = 1.384
   dr     = (r_o - r_i) / n

   # Set flow properties ar the inner radius.
   p_i    = 100.0e3                  # kPa
   M_i    = 2.25
   rho_i  = 1.0                      # kg/m**3
   T_i    = p_i / (Rgas * rho_i)     # K
   a_i    = sqrt( g * Rgas * T_i )   # m/s
   u_i    = M_i * a_i                # m/s
   # print p_i, M_i, rho_i, T_i, a_i, u_i

   # Generate the profile along the radial direction.
   print n > "profile.dat"
   for ( i = 1; i <= n; ++i ) {
      r   = r_i + dr * (i - 0.5)
      # print "i=", i, "r=", r
      u   = u_i * r_i / r
      t1  = r_i / r
      t2  = 1.0 + 0.5 * (g - 1.0) * M_i * M_i * (1.0 - t1 * t1)
      rho = rho_i * pow( t2, 1.0/(g - 1.0) );
      p   = p_i * pow( rho/rho_i, g )
      T   = p / (rho * Rgas)
      print p, u, 0.0, T > "profile.dat"
      print r/r_i, p/p_i, u/u_i, 0.0, T/T_i > "radial_profile_0.dat"
   } # end for
```

```
}
```

- The plots were generated via the following scripts

```
# extract_radial.awk
# Extract the radial profile data from mb_prof.exe generated files.
BEGIN{
   r_i = 1.0; p_i = 100.0e3; u_i = 841.87; T_i = 348.43;
}

$1 != "#" {
   x = $1; y = $2; p = $7; u = $4; v = $5; T = $10
   r = sqrt( x * x + y * y )
   speed = sqrt( u * u + v * v )
   print r/r_i , p/p_i , speed/u_i , 0.0 , T/T_i
}
```

```
# radial_profile.gnu

set term postscript eps enhanced 20
set output "radial_profile_p.eps"
set title "Inviscid Vortex"
set xlabel "r/r_i"
set ylabel "p/p_i"
# set yrange [1.0:4.5]
set key 1.35, 2
plot "radial_profile_0.dat" using 1:2 title "exact" with lines, \
     "radial_profile_45.dat" using 1:2 title "45 degrees", \
     "radial_profile_90.dat" using 1:2 title "exit plane"

set term postscript eps enhanced 20
set output "radial_profile_u.eps"
set title "Inviscid Vortex"
set xlabel "r/r_i"
set ylabel "u/u_i"
# set yrange [0.7:1.0]
set key
plot "radial_profile_0.dat" using 1:3 title "exact" with lines, \
     "radial_profile_45.dat" using 1:3 title "45 degrees", \
     "radial_profile_90.dat" using 1:3 title "90 degrees"

set term postscript eps enhanced 20
set output "radial_profile_T.eps"
set title "Inviscid Vortex"
set xlabel "r/r_i"
set ylabel "T/T_i"
# set yrange [1.0:1.7]
set key 1.35, 1.2
plot "radial_profile_0.dat" using 1:5 title "exact" with lines, \
     "radial_profile_45.dat" using 1:5 title "45 degrees", \
     "radial_profile_90.dat" using 1:5 title "90 degrees"
```

8 Pressure on a flat-faced cylinder

This example models the bar gauge type of pressure sensor as used in the expansion-tube facilities. It also shows the application of a multiple-block grid to describe the flow domain (Figure 26) around a flat-faced cylinder whose axis is aligned with the free-stream flow direction. The free-stream Mach number is 4.76 to match one of the higher Mach number conditions reported in Ref.[8].

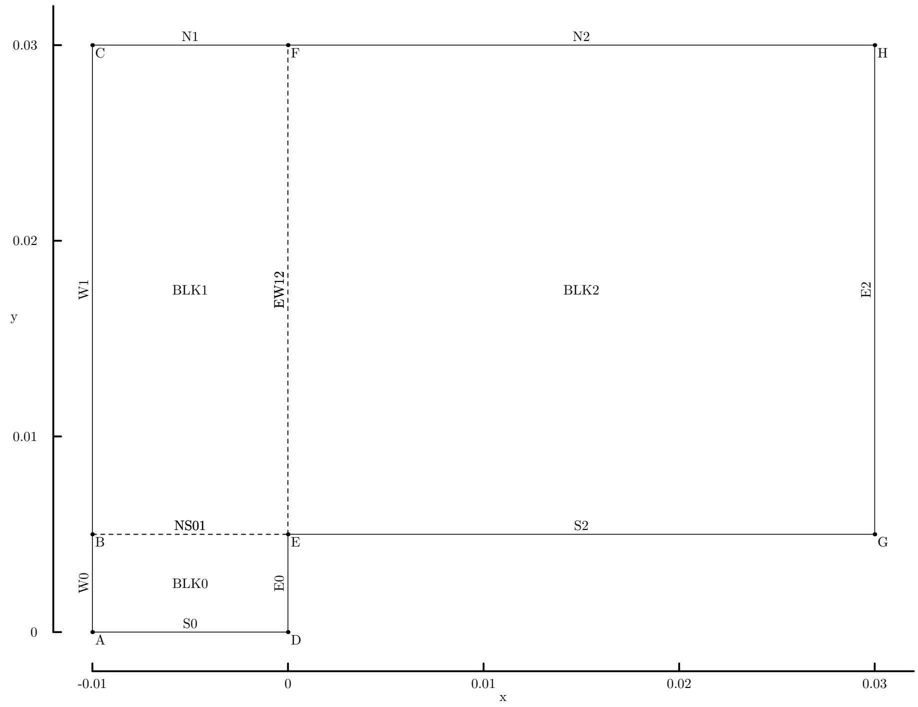


Figure 26: Schematic diagram of the full flow domain around the flat-faced cylinder.

The simulation is started with low pressure stationary gas throughout the domain and the inflow conditions are applied to the west boundaries of blocks 0 and 1. After allowing 50 μs for the flow to reach steady state, the pressure distribution throughout the domain is shown in Fig. 27. The stand-off distance of 2.814 mm was determined by searching for the pressure jump along the row of cells adjacent to the centreline.

Figure 28 shows the distribution of pressure across the face of the cylinder. The simulation data agrees closely with Kendall's measurements except in the region the sharp corner where there is inadequate resolution and an absence of viscous effects in the simulation.

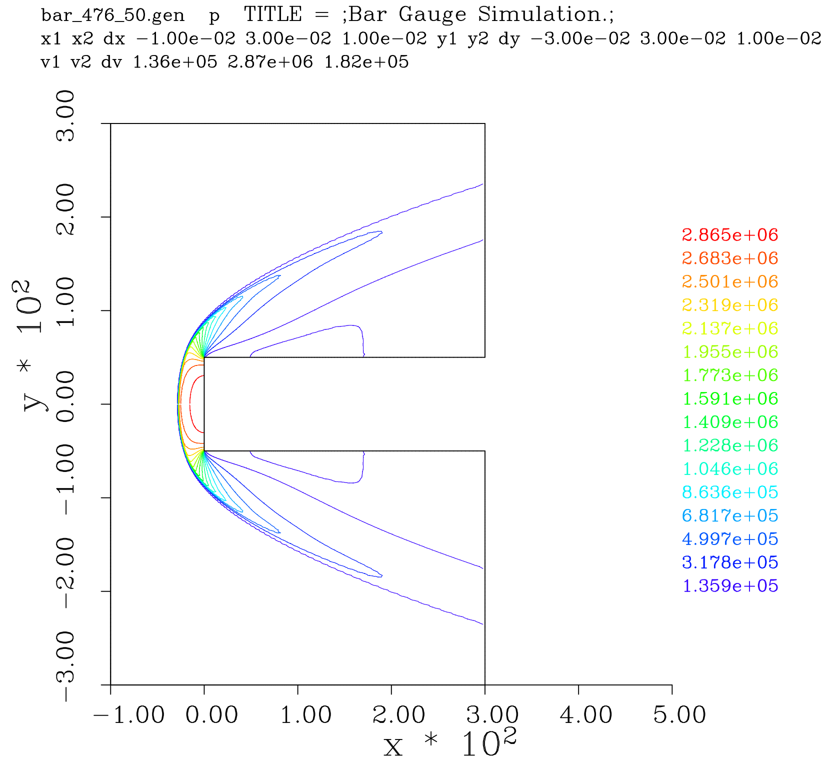


Figure 27: Pressure contours within the flow domain at 50 μs.

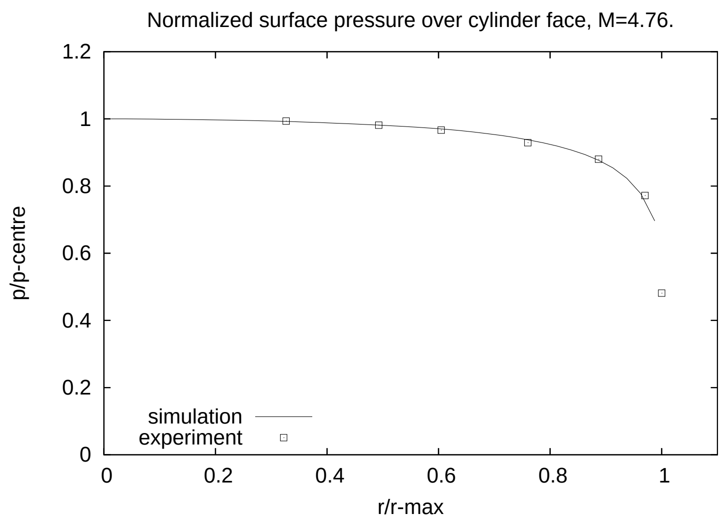


Figure 28: Normalised pressure across the face of the cylinder compared with experimental measurements [8].

8.1 .sit file

```
# bar_476.sit
# Bar gauge (or flat-faced cylinder) M=4.76, ideal air

BEGIN_GEOMETRY
   NODE a -10.0e-3   0.0
   NODE b -10.0e-3   5.0e-3
   NODE c -10.0e-3  30.0e-3
   NODE d    0.0       0.0
   NODE e    0.0       5.0e-3
   NODE f    0.0      30.0e-3
   NODE g   30.0e-3   5.0e-3
   NODE h   30.0e-3  30.0e-3

   # Lines that run vertically.
   LINE ab a b
   LINE bc b c
   LINE de d e
   LINE ef e f
   LINE gh g h

   # Lines that run horizontally.
   LINE ad a d
   LINE be b e
   LINE cf c f
   LINE eg e g
   LINE fh f h

   # Define the boundaries that will be used to
   # build the blocks.
   POLYLINE ns01 1 + be
   POLYLINE s0   1 + ad
   POLYLINE w0   1 + ab
   POLYLINE e0   1 + de

   POLYLINE n1   1 + cf
   POLYLINE ew12 1 + ef
   POLYLINE w1   1 + bc

   POLYLINE n2   1 + fh
   POLYLINE e2   1 + gh
   POLYLINE s2   1 + eg
END_GEOMETRY

BEGIN_FLOW
   # Gas and flow properties
   GAS_TYPE PERF_AIR_14
   GAS_STATE inflow    100.0e3 1653.0 0.0  300.0 1.0
   GAS_STATE initial      30.0    0.0 0.0  300.0 1.0

   # Set the boundary discretisation before building the blocks
   DISCRETISE ns01 120 0 0 0.0
   DISCRETISE s0   120 0 0 0.0
   DISCRETISE w0   40 0 0 0.0
   DISCRETISE e0   40 0 0 0.0

   DISCRETISE n1   120 0 0 0.0
   DISCRETISE ew12 80 1 0 1.2
   DISCRETISE w1   80 1 0 1.2

   DISCRETISE n2   120 1 0 1.1
   DISCRETISE e2   80 1 0 1.2
   DISCRETISE s2   120 1 0 1.1

   BOUNDARY_SPEC w0 SUP_IN inflow
   BOUNDARY_SPEC w1 SUP_IN inflow
   BOUNDARY_SPEC e2 SUP_OUT
```

```
   # Define blocks
   BLOCK blk0 + ns01 + e0   + s0   + w0
   BLOCK blk1 + n1   + ew12 + ns01 + w1
   BLOCK blk2 + n2   + e2   + s2   + ew12

   CONNECT_BLOCKS blk0 north blk1 south
   CONNECT_BLOCKS blk1 east blk2 west

   # Assign the initial gas states
   FILL_BLOCK blk0 initial
   FILL_BLOCK blk1 initial
   FILL_BLOCK blk2 initial
END_FLOW

BEGIN_CONTROL
   TITLE Bar Gauge Simulation.
   CASE_ID 0

   AXISYMMETRIC
   FLUX_CALC Adaptive

   MAX_TIME   50.0e-6
   MAX_STEP   15000
   TIME_STEP   2.0e-8
   DT_PLOT     5.0e-6
   DT_HISTORY 0.5e-6

   HISTORY_CELL blk0 120  1
   HISTORY_CELL blk0 120  5
   HISTORY_CELL blk0 120 10
END_CONTROL

# Name the output files and build them.
BEZIER_FILE bar_476.bez
PARAM_FILE  bar_476.p
MPOST FILE  bar_476.mpost
MPOST SCALES 5.0 5.0
MPOST XAXIS -10.0e-3 32.0e-3 10.0e-3 -2.0e-3
MPOST YAXIS 0.0e-3 32.0e-3 10.0e-3 -12.0e-3
BUILD

EXIT
```

8.2 Shell scripts

```
#!/bin/sh
# bar_476_run.sh
# Exercise the Navier-Stokes solver for
# Mark Sutcliffe's bar gauge.

# It is assumed that the path is set and that
# the script file "bar_476.sit" has been correctly written.

# Stage 1:
# Generate the input parameter and Bezier files
# (bar_476.p and bar_476.bez respectively)
# from the script file.
# A record of the transactions is recorded in the log file
# "bar_476.log" so that, if anything goes wrong,
# you can browse the log file and diagnose the problem.

# scriptit.exe < bar_476.sit > bar_476.log
scriptit.tcl -f bar_476.sit -do-mpost > bar_476.log
mpost bar_476.mpost
```

```
#  Stage 2:
#  Pick up the input parameter and Bezier files
#  and generate the grid and initial solution files
#  (bar_476.g and bar_476.s0).
#  Write these files as binary data.

mb_prep.exe -wb -f bar_476

#  Stage 3:
#  Pick up the grid and initial solution files as
#  binary data and integrate the solution in time.
#  The solution data at later times will be written
#  to the solution output file "bar_476.s"

time mb_cns.exe -rb -wb -f bar_476

#  Finished:
```

```
#!/bin/sh
# bar_476_plot.sh

# Stage 4: Extract particular times from the solution set
#
mb_post.exe -rb -fp bar_476.p -fg bar_476.g -fs bar_476.s \
-t 50.0e-6 -fo bar_476_50 -generic

# Stage 5:
# Pick up the reformatted data and make a contour plot.

mb_cont.exe -fi bar_476_50.gen -fo bar_476_50.ps -ps -colour -var 6 \
-mirror -xrange -0.010 0.030 0.010 -yrange -0.030 0.030 0.010

# Finished:
```

```
# bar_476_profile.sh
# Extract the flow data across the face of the bar gauge.

mb_prof.exe -fp bar_476.p -fg bar_476.g -fs bar_476.s -fo raw_profile.dat \
   -t 50.0e-6 -rb -xline 0 120

awk -f normalize.awk raw_profile.dat > norm_profile.dat

gnuplot <<EOF
set output "bar_476_norm_p.eps"
set term postscript eps 20
set xrange [0:1.1]
set yrange [0:1.2]
set title "Normalized surface pressure over cylinder face, M=4.76."
set xlabel "r/r-max"
set ylabel "p/p-centre"
set key bottom left
plot "norm_profile.dat" using 1:2 title "simulation" with lines, \
     "kendall_profile.dat" using 1:2 title "experiment" with points 4
EOF
```

```
#!/bin/sh
# bar_476_standoff.sh

mb_prof.exe -rb -fp bar_476.p -fg bar_476.g -fs bar_476.s \
```

```
    -fo bar_476_stag_line.dat \
    -t 50.0e-6 -yline 0 1

awk -f locate_shock.awk bar_476_stag_line.dat
```

8.3 Notes

- The simulation reaches a final time of 50 μs in 2951 steps and, on a Celeron 2.4 Ghz system, this takes 19 min, 54 s of CPU time. This is equivalent to 16.9 μs per cell per predictor-corrector time step.

- The surface pressure is normalised with respect to the stagnation pressure after the bow shock, using the following AWK script.

```
# normalize.awk
# Normalize the surface pressure over the centreline static pressure.
BEGIN {
  p_centre = -1.0;
}

$1 != "#" {
  p = $7;
  r = $2;
  if (p_centre < 0.0) p_centre = p;
  print r/0.005, p/p_centre;
}
```

- Along a row of cells that have been extracted using `mb_prof`, the shock is detected using the following AWK script.

```
# locate_shock.awk

BEGIN {
    p_old = 0.0;
    x_old = -2.0;
    p_trigger = 200.0e3;
    shock_found = 0;
}

$1 != "#" { # for any non-comment line
    p_new = $7;
    x_new = $1;
    # print "p_new=", p_new, "x_new", x_new
    if ( p_new > p_trigger && shock_found == 0 ) {
        shock_found = 1;
        frac = (p_new - p_trigger) / (p_new - p_old);
        x = x_old + frac * (x_new - x_old);
        print "shock located at x = ", x
    }
    p_old = p_new;
    x_old = x_new;
}
```

```
END {
    if ( shock_found == 0 ) {
        print "shock not located";
    }
    print "done."
}
```

References

[1] B. B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[2] P. A. Jacobs. Single-block Navier-Stokes integrator. ICASE Interim Report 18, 1991.

[3] Ames Research Staff. Equations, tables and charts for compressible flow. NACA Report 1135, 1953.

[4] K. Sawada and E. Dendou. Validation of hypersonic chemical equilibrium flow calculations using ballistic-range data. *Shock Waves*, 11:43–51, 2001.

[5] J. D. Anderson. *Hypersonic and High Temperature Gas Dynamics.* McGraw-Hill, New York, 1989.

[6] L. H. Back, P. F. Massier, and H. L. Gier. Comparison of measured and predicted flows through conical supersonic nozzles, with emphasis on the transonic region. *A.I.A.A. Journal*, 3(9):1606–1614, 1965.

[7] A. Aftosmis, D. Gaitonde, and T. S. Tavares. Behaviour of linear reconstruction techniques on unstructured meshes. *A.I.A.A. Journal*, 33(11):2038–2049, 1995.

[8] J. M. Kendall. Experiments on supersonic blunt-body flows. Progress Report 20-372, Jet Propulsopn Laboratory, California Institute of Technology, Pasadena, California., February 1959.