A User's Guide to MSES 3.05

Mark Drela
MIT Department of Aeronautics and Astronautics
July 2007

This document describes the use of the MSES multielement airfoil design/analysis system.

Contents

1 Input files

1.1 blade.xxx — airfoil geometry and grid domain

This is a formatted file which is created by the user and is used in the initialization program **MSET** and the airfoil manipulation program **AIRSET** to define the airfoil shape. The "xxx" is a variable extension name which can be anything, and is generally used to distinguish the case being run.

Note: The format of the blade.xxx file has changed since MSES v 2.7. The change was done in order to simplify the positioning of the top/bottom walls in solid-wall cases, and also to allow the airfoil and the grid to be located anywhere in space. The "old" format of blade.xxx is still recognized by MSES v 2.9 for convenience. The old format is assumed if the second line contains five or more numbers rather than four.

The new blade.xxx file has the following structure.

```
NAME (not more than 32 characters)
XINL XOUT YBOT YTOP
X(1,1)  Y(1,1)
X(2,1)  Y(2,1)
X(3,1)  Y(3,1)
  .      .
  .      .
X(I,1)  Y(I,1)
999.  999.
X(1,2)  Y(1,2)
X(2,2)  Y(2,2)
X(3,2)  Y(3,2)
  .      .
  .      .
X(I,2)  Y(I,2)
999. 999.
X(1,3)  Y(1,3)
X(2,3)  Y(2,3)
X(3,3)  Y(3,3)
  .      .
  .      .
X(I,3)  Y(I,3)
  .      .
  .      .
```

where,

XINL is the X-location of the left grid inlet plane, in the same coordinate system as the airfoil coordinates.

XOUT is the X-location of the right grid outlet plane.

YBOT is the `Y`-location of the lowest grid streamline. It is also the location of the bottom wall in solid-wall cases.

YTOP is the `Y`-location of the topmost grid streamline. It is also the location of the top wall in solid-wall cases.

For a unit airfoil located between $x, y = 0, 0$ and $1, 0$, the recommended minimum values for the four grid-boundary locations are:

```
 XINL XOUT YBOT YTOP :   -1.75  2.75  -2.0  2.5
```

These are suitable for infinite flow at low Mach numbers and moderate positive C_L. For negative lift, the magnitudes of `YBOT` and `YTOP` should be swapped to allow for the vertical grid-shifting effect of the circulation.

At higher Mach numbers, both `YBOT` and `YTOP` should be increased somewhat in accordance with Prandtl-Glauert scaling

$$\texttt{YBOT YTOP} \sim 1/\sqrt{1 - M_\infty^2}$$

to allow for the lateral increase in the airfoil's compressible disturbance field. Similarly, cases with large C_L values should have both the horizontal and vertical extent of the grid increased roughly as

$$\texttt{XINL YINL YBOT YTOP} \sim \sqrt{C_L}$$

which allows for the larger radial extent of the airfoil's influence.

It must be stressed that the exact values of these grid parameters are not very important, since high-order vortex+doublet farfield representation makes the solution extremely insensitive to the location of the outer grid boundaries. This is clearly demonstrated in the separate "MSES Domain Size Sensitivity Study" document. However, the grid must always be large enough to completely contain any embedded supersonic zones.

X(1,1),Y(1,1) through to X(I,L),Y(I,L) are the coordinates of each airfoil element surface, starting at the trailing edge, going round the leading edge in either direction, then going back to the trailing edge. The individual elements are separated by a "999.0 999.0" line as indicated above. It is important that the elements be ordered top to bottom, so that in a typical multielement configuration, the slat would be first, the main element second, and so on.

It is strongly recommended that the airfoil be located in space near the origin. Although the airfoil's location doesn't really matter to the flow solver, the solid-wall boundary conditions with a nonzero angle of attack will rotate the entire outer grid boundary about the $x, y = 0.25, 0.0$ point. If the airfoil is at some distant location and the grid is rotated about the 0.25,0.0 point, the airfoil might try to migrate (correctly) out of the grid. Naturally, this will cause serious problems. If it is absolutely necessary to locate the airfoil far from the origin, then one should change the center of rotation, called XCENT,YCENT, from 0.25,0.0, to somewhere close to the airfoil. This can be done using program **MEDP** , described later.

MSES can model blunt trailing edge flows to a reasonable degree of accuracy, provided the trailing edge thickness is not too extreme. Up to a few percent of chord is generally acceptable. A blunt trailing edge is specified if the first and last points of an element do not exactly cooincide, so that the airfoil is "open". The trailing edge base spanning these points should be more or less at right angles to the airfoil's local camber line.

If the airfoil has a round trailing edge, it is strongly suggested that the circle be "cut off", leaving the airfoil open as just described. This will give a more realistic calculated blunt trailing edge flow.

A sharp corner in any element contour, such as one resulting from a flap cove, can be specified with two identical sucessive x, y coordinate pairs. Two separate splines are then used on each side of the corner. A corner is also significant to the **MSET** program, in that it indicates a point where separation is assumed when the streamline grid is initialized. A corner also indicates a trip location in **MSES** . This is done to simulate the nearly instantaneous transition which occurs in the free shear layer above the cove recirculation region. It is a good idea to force boundary layer transition (described later) at or just ahead of any such surface corner. Laminar separation off the corner, where the shape parameter $H = \delta^*/\theta$ can soar into the hundreds, will not be handled reliably by the free transition prediction method. It is better to trip the flow at the corner, which is the same as assuming instant transition in the separated shear layer.

Only one corner point per element side is currently allowed. Note that this differs from the single-element ISES code, where a doubled point implies a sharp leading edge. **MSES** does not allow sharp leading edges.

The first two lines in `blade.xxx` can be omitted, in which case the user will be prompted to enter the missing information from the keyboard.

1.2 mses.xxx — runtime parameters

This is the runtime-parameter input file which is used by the solver programs **MSES** , **MSIS** , **MPOLAR** , and **MPOLIS** . Again, the "xxx" is arbitrary and used to distinguish a particular case.

The `mses.xxx` input file contains the following information:

```
GVAR(1) GVAR(2) ...  GVAR(N)
GCON(1) GCON(2) ...  GCON(N)
MACHIN  CLIFIN  ALFAIN
ISMOM   IFFBC  [ DOUXIN  DOUYIN  SRCEIN ]    <-- optional
REYNIN  ACRIT  [ KTRTYP ]                    <-- optional
XTRS1   XTRP1  XTRS2  XTRP2  XTRS3  XTRP3 . . . .
MCRIT   MUCON
[ ISMOVE  ISPRES ]       <-- optional (see below)
[ NMODN   NPOSN  ]       <-- optional (see below)
[ ISDELH  XCDELH  PTRHIN  ETAH ]      <-- optional (see below)
```

where,

a) GVAR(1) ... GVAR(N) is a list of integers specifying the global variables to be used. The list of possible global variables is,

```
      3    DCIRC  far-field vortex strength
      4    DALFA  freestream angle of attack
```

```
 5    DSBLE  LE stagnation point for all elements,
             mass fraction for all-1 elements

 7    DDOUB  x,y-doublet,source strengths

 9    DPREX  grid-exit static pressure

10    DREYN  Reynolds number DOF

11    DPDX0  zeroth mixed inverse prescribed Pi DOF
12    DPDX1  first  mixed inverse prescribed Pi DOF
13    DPDD0  second mixed inverse prescribed Pi DOF
14    DPDD1  third  mixed inverse prescribed Pi DOF

15    DMASS  total mass flow
16    DMAS1  first mass fraction DOF above/below configuration

20    DMODn  modal geometry DOF flag    n = 1, 2, ... NMODN
30    DPOSn  element position DOF flag   n = 1, 2, ... NPOSN
```

b) GCON(1) ... GCON(N) is a list of integers specifying the global constraints to be used. The list of possible options is,

```
 3    Set LE Kutta condition for all elements
 4    Set TE Kutta condition for all elements

 5    Drive alpha  to  ALFAIN
 6    Drive CL     to  CLIFIN

 7    Drive x,y-doublets,source to farfield match
 8    Drive x,y-doublets,source to DOUXIN, DOUYIN, SRCEIN

10    Fix TE point

11    Fix left  endpoint of freewall segment
12    Fix right endpoint of freewall segment
13    Fix Pxx at left  endpoint of freewall segment
14    Fix Pxx at right endpoint of freewall segment

15    Drive Mach to MACHIN
16    Drive Mach*sqrt(CL) to MACHIN
17    Drive Reyn to REYNIN
18    Drive Reyn*sqrt(CL) to REYNIN

19    Drive mass-averaged inlet Pst to Pstinl

20    DMODn fixing constraint flag   n = 1, 2, ... NMODN
```

```
 30    DPOSn fixing constraint flag   n = 1, 2, ... NPOSN
 40    DMODn inverse constraint flag   n = 1, 2, ... NMODN
 50    DPOSn inverse constraint flag   n = 1, 2, ... NPOSN
```

c) MACHIN = inlet or freestream Mach number
$= \mathrm{M}/\sqrt{C_L}$ if constraint (16) is specified

CLIFIN = specified C_L (used only if C_L option invoked in b)

ALFAIN = freestream angle of attack in degrees
(used only if alpha option invoked in b)

d) ISMOM = 1 $\rightarrow$ use S-momentum equation
= 2 $\rightarrow$ use isentropic condition
= 3 $\rightarrow$ use S-momentum equation, with isentropic condition only near leading edge
= 4 $\rightarrow$ use isentropic condition, with S-momentum equation only where dissipation is active

IFFBC = 1 $\rightarrow$ use solid wall airfoil far-field BCs
= 2 $\rightarrow$ use vortex+source+doublet airfoil far-field BCs (best model for infinite domains)
= 3 $\rightarrow$ use freestream pressure airfoil far-field BCs
= 4 $\rightarrow$ use supersonic wave freestream BCs
= 5 $\rightarrow$ use supersonic solid wall far-field BCs

DOUXIN = specified x-doublet strength, $= D_x/V_\infty L^2$,
DOUYIN = specified y-doublet strength, $= D_y/V_\infty L^2$,
SRCEIN = specified source strength, $= \Sigma/V_\infty L$,
with L the length unit of input airfoil coordinates.
Used only if constraint (8) is specified, otherwise can be omitted from line.

e) REYNIN = Reynolds number $Re = \rho_\infty V_\infty L/\mu_\infty$, with L the length unit of input airfoil coordinates
$= Re/\sqrt{C_L}$ if constraint (18) is specified
= 0.0 $\rightarrow$ inviscid calculation
(restarting a viscous case with REYNIN = 0 "freezes" the BLs)

ACRIT = critical amplification factor "n" for e^n envelope transition model
= 9.0 is the standard model.

KTRTYP = 1 use envelope e^n method (default if omitted)
= 2 use frequency-tracking e^n method.

f) XTRSn = top (suction) surface transition strip x/chord location on element "n".

XTRPn = bottom (pressure) surface transition strip x/chord location on element "n".
Here, "x" is measured parallel to the element chord line which spans the element's widest dimension. The Orr-Sommerfeld e^n criterion is always active,
and free transition may occur upstream of XTRS or XTRP).

g)	MCRIT	= "critical" Mach number above which artifical dissipation is added
		= 0.99 usually for weak shocks
		= 0.90 for exceptionally strong shocks
	MUCON	= artificial dissipation coefficient (1.0 works well), (A negative value disables the 2nd-order dissipation. This is a "last-resort" option for difficult cases.)
h)	ISMOVE	= indicates what side(s) get altered in mixed-inverse cases
		= –1 $\rightarrow$ both element sides move opposite (camber preserved)
		= 0 $\rightarrow$ both element sides move together (thickness preserved)
		= 1 $\rightarrow$ upper side moves
		= 2 $\rightarrow$ lower side moves
		= ignored in direct cases
	ISPRES	= indicates the pressure BC to be imposed in mixed- or modal-inverse cases
		= 0 $\rightarrow$ Δp across element is imposed
		= 1 $\rightarrow$ upper side pressures imposed
		= 2 $\rightarrow$ lower side pressures imposed
		= ignored in direct cases
i)	NMODN	indicates that the n = 1,2,...NMODN modal geometry DOFs are selected only used if DMODn flag (20) is chosen as a DOF
	NPOSN	indicates that the n = 1,2,...NPOSN element position DOFs are selected only used if DPOSn flag (30) is chosen as a DOF
j)	ISDELH	Airfoil side to which Actuator Disk is attached Airfoil sides are numbered 1,2...top to bottom
	XCDELH	x/c location where AD is attached on side ISDELH
	PTRHIN	total pressure ratio p_{0_2}/p_{0_1} across AD
	ETAH	efficiency η_h of AD, which gives the total enthalpy ratio across the AD: $h_{0_2}/h_{0_1} - 1 = \frac{1}{\eta_h}\left[(p_{0_2}/p_{0_1})^{(\gamma-1)/\gamma} - 1\right]$

1.2.1 Farfield boundary conditions

As indicated by the various IFFBC options above, there is a wide variety of farfields which can be imposed. The most common is the infinite-flow farfield (IFFBC=2). The formulation uses the transformed Prandtl-Glauert coordinates

$$\begin{aligned}
\bar{x} &= \left[(x - x_o)\cos\alpha + (y - y_o)\sin\alpha\right]/\beta \\
\bar{y} &= -(x - x_o)\sin\alpha + (y - y_o)\cos\alpha \\
\bar{r} &= \sqrt{\bar{x}^2 + \bar{y}^2} \\
\bar{\theta} &= \arctan(\bar{y}/\bar{x}) \\
\beta &= \sqrt{1 - M_\infty^2}
\end{aligned}$$

centered on some singularity-location point x_o, y_o, aligned with the freestream direction, and scaled with β. The second-order farfield potential expansion is

$$\begin{aligned}\phi(x,y;M_\infty,\Gamma,\Sigma,D_x,D_y) \;&=\; -\frac{\Gamma}{2\pi}\bar\theta \;+\; \frac{\Sigma}{2\pi}\ln\bar r \;+\; \frac{D_x}{2\pi}\frac{\cos\bar\theta}{\bar r} \;+\; \frac{D_y}{2\pi}\frac{\sin\bar\theta}{\bar r} \\ &+\; \left(\frac{\Gamma M_\infty}{2\pi}\right)^2 \left[\frac{1}{4}\left(\frac{3-\gamma}{\beta}+\frac{\gamma+1}{\beta^3}\right)\frac{\ln\bar r\,\cos\bar\theta}{\bar r} \;+\; \frac{1}{16}\left(\frac{\gamma+1}{\beta}-\frac{\gamma+1}{\beta^3}\right)\frac{\cos 3\bar\theta}{\bar r}\right]\end{aligned}$$

which corresponds to a vortex, source, and two doublet components at x_o, y_o, with a higher-order correction for the dominant first Γ term. The strengths Γ, Σ ...are assumed to be normalized with V_∞. The location x_o, y_o, called XCENT,YCENT in the code, is set at $(0.25, 0.0)$ by default, but can be changed for any give case using program **MEDP** . This potential is used to obtain a local velocity and corresponding isentropic pressure

$$\begin{aligned}\vec V \;&=\; V_\infty\left[(\cos\alpha+\partial\phi/\partial x)\,\hat\imath \;+\; (\sin\alpha+\partial\phi/\partial y)\,\hat\jmath\right] \\ p \;&=\; p_{0_\infty}\left[1 \;-\; \frac{V^2}{2h_{0_\infty}}\right]^{\gamma/(\gamma-1)}\end{aligned}$$

This $\vec V$ is used at the inlet planes to set a streamline-direction boundary condition, and the p is used on the top and bottom streamlines to set a pressure boundary condition.

The singularity strengths Γ, Σ, D_x, D_y are treated as global variables in the overall Newton system. The Γ variable (3) is constrained with the TE Kutta condition (4). The Σ, D_x, D_y variables are all simultaneously selected with (7), and implicitly constrained with the three least-squares farfield-matching conditions (7).

$$\begin{aligned}\mathcal{J}(\Gamma,\Sigma,D_x,D_y,M_\infty\vec q(s)) \;&\equiv\; \frac{1}{2}\int\left|\vec V\times\vec q\right|^2 ds \\ \mathcal{R}_{D_x} \;\equiv\; \frac{\partial\mathcal{J}}{\partial D_x} \;&=\; \int\left|\vec V\times\vec q\right|\left|\frac{\partial\vec V}{\partial D_x}\times\vec q\right| ds \;=\; 0 \\ \mathcal{R}_{D_y} \;\equiv\; \frac{\partial\mathcal{J}}{\partial D_y} \;&=\; \int\left|\vec V\times\vec q\right|\left|\frac{\partial\vec V}{\partial D_y}\times\vec q\right| ds \;=\; 0 \\ \mathcal{R}_{\Sigma} \;\equiv\; \frac{\partial\mathcal{J}}{\partial\Sigma} \;&=\; \int\left|\vec V\times\vec q\right|\left|\frac{\partial\vec V}{\partial\Sigma}\times\vec q\right| ds \;=\; 0\end{aligned}$$

The integrals are carried over the top and bottom streamlines. If constraint (8) is selected instead, the singularity strengths are simply set explicitly,

$$\begin{aligned}\mathcal{R}_{D_x} \;&\equiv\; D_x \;-\; (D_x)_{\rm spec} \;=\; 0 \\ \mathcal{R}_{D_y} \;&\equiv\; D_y \;-\; (D_y)_{\rm spec} \;=\; 0 \\ \mathcal{R}_{\Sigma} \;&\equiv\; \Sigma \;-\; (\Sigma)_{\rm spec} \;=\; 0\end{aligned}$$

where the specified values are given by DOUXIN, DOUYIN, SRCEIN.

If the solid-wall IFFBC=1 option is chosen, the top and bottom streamlines are simply forced to be straight lines, tilted at the angle of attack α. The inlet and outlet flow is also imposed to have this same direction. This can model a solid-wall wind tunnel configuration.

If the constant-pressure IFFBC=3 option is chosen, the pressure all along the outer boundary is held at a constant p_∞, and the streamline shapes adjust as necessary. This can model a free-jet wind tunnel configuration.

There are also “unofficial” IFFBC options which can represent more elaborate situations as described below.

IFFBC=6: Similar to IFFBC=1, but enforces solid top and bottom walls of arbitrary shape $y(x)$, rather than the default flat walls. The wall shapes are specified in the `xywall.xxx` file, which is required. See the header of SUBROUTINE FFBC1 (in `ffbc.f`) for the file format.

IFFBC=7: Similar to IFFBC=2 or IFFBC=3, but enforces arbitrary top and bottom wall pressure distributions $C_p(x, y)$ along some paths $y(x)$ reasonably near the top and bottom grid boundaries. The intent is to match measured wall tunnel pressures, thereby removing the tunnel blockage uncertainty. This option requires a `ffbc.xxx` file to specify path shapes and pressure distributions. See the header of SUBROUTINE FFBC2 (in `ffbc.f`) for the file format. Some deviation from the specified $C_p(x, y)$ will be created so that the TE Kutta condition can be met. This deviation will be minimal if the $C_p(x, y)$ pressures correspond closely to the C_L value specified in `ffbc.xxx`. Again, see the header for a more detailed explanation.

1.2.2 Variable, Constraint indices

In general, any number of global variables can be specified for a given solution case, as long as the same number of global constraints are also specified. It must be kept in mind that specifying the leading edge Kutta condition (constraint 3) produces a number of constraints equal to the number of elements, since the Kutta condition is applied to each element. Likewise, specifying the leading edge movement variable (variable 5), automatically adds one internal additional variable consisting of the mass flow difference between the captured streamtubes on each side of each element after the first one. This mass flow difference must be a variable since it is not known a priori. In any case, the total number of global variables and constraints is reported by **MSES** before the solution is started. Whatever these are, it is only necessary that all the constraints together properly constrain all the variables. For example, the angle of attack variable ALFA (variable 4) can be constrained either by specifying it directly (constraint 5), or by specifying C_L (constraint 6) instead. However, the latter cannot be used near maximum C_L as the problem will be poorly conditioned.

The global-variable and global-constraint indices in lines a) and b) can be specified in any order. The only exception is that the modal geometry and element position global-variable indices (20,30) and their corresponding fixing constraints (20,30) must be placed after all the other indices (1 . . . 19). This restriction is imposed to allow the block solver SUBROUTINE SOLVE to omit solving for the modal and position global variable influence vectors for all but the final Newton iteration. This produces significant CPU time savings.

The DMODn (20) and DPOSn (30) variables do not necessarily have to be paired with their respective fixing constraints (20,30) or (40,50). Alternative constraints can be used as long as the overall system is well-posed. For example, one flap element rotation variable (DPOS1 say) can be used in conjunction with the specified-C_L constraint (6), with α being held fixed. Instead of varying α, the code will rotate the flap until the specified C_L is achieved. Alternatively, a camber mode variable (DMOD1, say) might be selected instead of the flap rotation mode. Again, the code will alter the camber until the prescribed C_L is achieved. In contrast, using a flap translation variable in conjunction with the specified-C_L constraint is likely to produce an ill-conditioned system, since C_L is only weakly influenced by element translation.

1.2.3 Fixed-lift Mach and Reynolds number variation

The fixed-lift Mach and Reynolds number constraints (16) and (18), which are typically used in conjunction with the global variables (15) and (10), automatically scale the Mach and Reynolds numbers as $1/\sqrt{C_L}$. This corresponds to a given aircraft wing undergoing trim speed (i.e. C_L changes) at fixed altitude. A typical application of these constraints would be to generate a fixed-lift polar to be used in generating an overall aircraft drag polar, or to create an airfoil polar on which the low-C_L Mach buffet boundary is clearly discernable. The latter application requires care at the lowest specified angle of attack, since there is a minimum α below which buffet sets in and no steady **MSES** solution exists. In fact, there are generally two solutions for a given α and $M\sqrt{C_L}$ — the usual attached-flow branch, and a post-buffet branch with massive separation. The latter (if it converges) has a very high C_D, and a lower C_L and higher M. At the minimum α, i.e. the buffet boundary, these two branches cooincide, and convergence becomes difficult. Close to this minimum α, the solution might jump to the post-buffet branch from which the code usually cannot recover (this crudely mimics the Mach-tuck effect). This situation is analogous to trying to specify C_L near $C_{L\max}$, where the code might jump to the post-stall condition. To calculate a fixed-lift polar close to the buffet boundary, it is necessary to approach it from above with quite small α decrements — $0.1°$ or less.

1.2.4 Momentum/Entropy conservation

The ISMOM flag controls whether and where S-momentum (streamwise momentum) or streamline entropy are conserved. The streamtube-cell equation residuals for each case are:

$$\text{ISMOM=1:} \qquad \mathcal{R}_1 \equiv \Delta p + \frac{m}{A}\Delta\tilde{q} - P_s = 0$$

$$\text{ISMOM=2:} \qquad \mathcal{R}_2 \equiv -p\,\Delta\tilde{S} = 0$$

where m is the streamtube's mass flow and A is the streamtube's cross-sectional area ($\mathcal{R}_1$ is slightly modified for the case of a curved streamtube). Dissipation is introduced in the form of an upwinded speed $\tilde{q}$ described in the next section. The upwinded entropy is defined in the standard manner, but using the upwinded speed.

$$\begin{aligned} \tilde{h} &= h_0 - \frac{1}{2}\tilde{q}^2 \\ \tilde{S} &= \ln\left[\frac{\left(\tilde{h}/h_{0_\infty}\right)^{\gamma/(\gamma-1)}}{p/p_{0_\infty}}\right] \end{aligned}$$

The changes $\Delta(\;)$ are along the streamwise direction in the cell, and P_s is the streamwise pressure force contribution to the cell from streamtubes above and below, arranged so that the net $\mathcal{R}_1$ equation is strongly conservative.

ISMOM=1 gives a standard momentum-conserving Euler solver, while ISMOM=2 gives the equivalent of a standard entropy-conserving Full-Potential solver (although the upwinding is based on the speed, not the usual density). ISMOM=3,4 are hybrids which attempt to make the use of the best features of the two types of solvers: correct Rankine-Hugoniot shock jumps of the Euler solver, and zero spurious total pressure loss of the Full-Potential solver.

For ISMOM=3, the S-momentum equation $\mathcal{R}_1 = 0$ is used everywhere except in a region near the leading edge, where the isentropic equation $\mathcal{R}_2 = 0$ is used instead. The size of the isentropic region is hard-wired into a data statement at the top of SUBROUTINE SETUP (in `setup.f`), and typically extends from the inlet plane to $\sim$ 10 cells downstream of the stagnation point, and 4 cells below and above the stagnation streamline. This should cover all typical airfoils, but can be changed if appropriate for an unusual case. **MPLOT** allows the display of the momentum-conserving region on the grid and flow contour plots, the remainder being the entropy-conserving region.

MSES v 2.8 was the first to incorporate the new ISMOM=4 option, which makes *all* cells isentropic except those where aritificial dissipation contributes significantly to total pressure losses (as described later). In practice, this is nearly the same as ISMOM=3, but it will always give momentum conservation at shocks, and hence is "safer" than ISMOM=3. It also doesn't rely on the ad-hoc hard-wired isentropic region, and hence is more automatic. Note also that ISMOM=4 is equivalent to ISMOM=2 for subcritical cases.

Because conserving total pressure is much more important than conserving momentum (except at shocks, of course), it is strongly recommended that ISMOM=3 or 4 be used for all flows. Even very small total pressure errors in the vicinity of the leading edge (where they are most likely to occur) can cause dramatic errors in lift and drag of a viscous case at near-stall conditions. This is because for a given imposed streamwise pressure gradient dp/ds, the edge velocity gradient du_e/ds which drives the viscous layers depends on the total pressure. Enforcing isentropy (even if only at the leading edge for ISMOM=3) avoids most such problems, and hence significantly reduces grid density requirements. With ISMOM=3, it is only necessary to ensure that no shocks traverse the isentropic region, otherwise incorrect Rankine-Hugoniot jumps will result and the wave drag will not be properly predicted. ISMOM=4 does not have this possible problem, and is recommended for general use.

1.2.5 Artificial dissipation

The artifical dissipation in **MSES** is a speed-upwinding formulation analogous to bulk viscosity. Instead of the actual speed q, the momentum and/or isentropy equations are defined using an upwinded speed $\tilde{q}$ defined by

$$\tilde{q}_i \;=\; q_i \;-\; \mu_i^{(1)} \left(q_i - q_{i-1} \right) \;+\; \mu_i^{(2)} \left(q_{i-1} - q_{i-2} \right) \frac{s_i - s_{i-1}}{s_{i-1} - s_{i-2}}$$

where i is the grid node index along a streamtube, s is the arc length along the streamtube, and $\mu_i^{(1)}$, $\mu_i^{(2)}$ are the first- and second-order dissipation coefficients. To maintain numerical stability and allow shock capturing, the following formulas for the dissipation coefficients are used.

$$\mu_i^{(1)} \;=\; \frac{C_\mu}{\gamma} \, (1 - M_{\text{crit}}) \; \log \left[1 + \exp \left(\frac{1 - 1/M^2}{1 - M_{\text{crit}}} \right) \right] \quad , \quad \text{with } M^2 = \tfrac{1}{2}(M_i^2 + M_{i-1}^2)$$

$$\mu_i^{(2)} \;=\; \begin{cases} \mu_i^{(1)} & ; \quad \text{2nd-order dissipation} \\ 0 & ; \quad \text{1st-order dissipation} \end{cases}$$

In the limit $M_{\text{crit}} \rightarrow 1$, the above formula asymptotes to

$$\mu \;\rightarrow\; \max \left[0 \;,\; \frac{C_\mu}{\gamma} \left(1 - 1/M^2 \right) \right]$$

which is the form indicated by a formal stability analysis, and is the dotted curve in Figure 1. For $M_{\text{crit}} < 1$, the full formula produces somewhat larger μ values, which asymptote rapidly to zero below

$M = M_{\rm crit}$. The intent of introducing $M_{\rm crit}$ is to provide a user-adjustable margin of safety. Figure 1 shows the variation of μ versus $M_{\rm crit}$ and the overall scaling constant C_μ. Previous MSES versions used the simpler form $\mu \sim \max(0, 1 - M_{\rm crit}^2/M^2)$, which has a slope discontinuity at $M = M_{\rm crit}$. The new form appears to have better shock propagating properties, most likely due to its exponential "tail" for subsonic Mach numbers.

The second-order term in the formula for $\tilde{q}_i$ above is constructed to cancel the first-order term in the case of a linear variation of $q(s)$. Analytically, the net result of either the first-order or second-order upwinding is to add a term to the streamwise momentum equation

$$\frac{\partial p}{\partial s} + \rho q \frac{\partial \tilde{q}}{\partial s} = 0 \qquad \longrightarrow \qquad \frac{\partial p}{\partial s} + \rho q \frac{\partial q}{\partial s} = -\rho q \frac{\partial(\delta q)}{\partial s}$$

where $\delta q = \tilde{q} - q$ is the upwinding modification to the speed. This added term is typically dissipative, producing streamwise changes in the total pressure

$$\frac{1}{p_0}\frac{\partial p_0}{\partial s} = -\gamma M^2 \frac{1}{q}\frac{\partial(\delta q)}{\partial s}$$

and is the mechanism which allows captured shocks to generate total pressure loss.

In the ISMOM=4 option, the magnitude of the fractional *upwinded* total pressure change over a cell implied by a δq change is monitored:

$$\Delta\left(\ln \tilde{p}_0\right) \simeq \delta q \, \Delta\left(\frac{\rho q}{p}\right)$$

If the righthand side term is sufficiently negative, below some small tolerance $-\epsilon_p$ (typically at a shock), then $\mathcal{R}_1 = 0$ is used instead of $\mathcal{R}_2 = 0$, so that this total pressure loss is realized in the solution. The local velocity gradient is also examined, with acceleration favoring $\mathcal{R}_2$ and deceleration favoring $\mathcal{R}_1$. To minimize the tendency for non-convergent limit cycles, the equation switch is implemented as a continuous blend. The actual general equation solved with the ISMOM=4 option is

$$f\,\mathcal{R}_1 + (1-f)\,\mathcal{R}_2 = 0$$

where $0 \le f \le 1$ is a blending fraction which depends on the local magnitude of the total pressure loss, as well as the local speed gradient.

It should be pointed out that this seemingly ad-hoc blending of $\mathcal{R}_1$ and $\mathcal{R}_2$ is perfectly legitimate within the accuracy of the numerical scheme, since in smooth flow these two equations are equivalent to within $\mathcal{O}(\Delta s^2)$. Hence, away from shocks, any change in f will produce a solution change of at most $\mathcal{O}(\Delta s^2)$ — the same as the truncation error of the discretization scheme.

1.2.6 Artificial dissipation level selection

The magnitude of the upwinding (e.g. magnitude of δq) is controlled by the approximate threshold $M_{\rm crit}$ (MCRIT), and the weighting factor C_μ (MUCON). If MUCON is specified as negative, then $C_\mu = |\text{MUCON}|$, and $\mu^{(2)} = 0$.

Lowering $M_{\rm crit}$ and increasing C_μ both increase the amount of upwinding, but in different ways as shown in Figure 1. The effect of C_μ on the numerical normal-shock profile is shown in Figure 2. In general, $C_\mu \simeq 1$ gives the cleanest normal shocks. For oblique shocks, the effective coefficient is

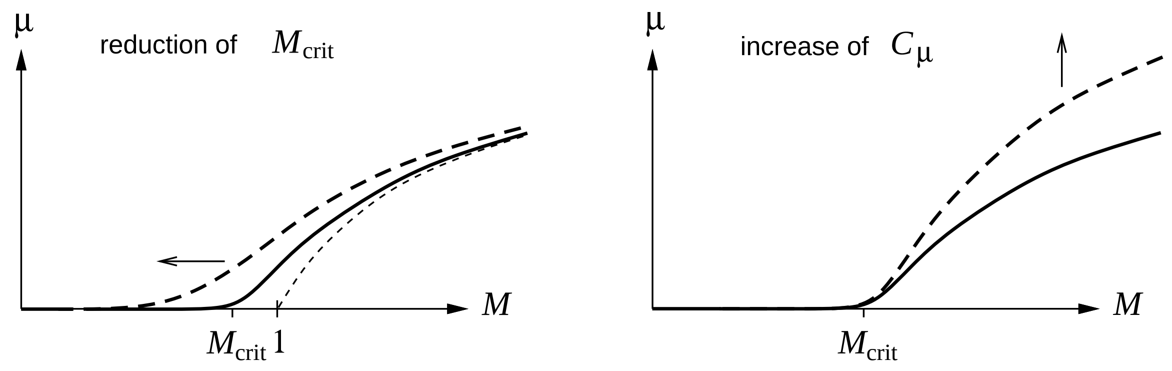


Figure 1: Effect of dissipation parameters M_{crit} and C_μ on dissipation level.

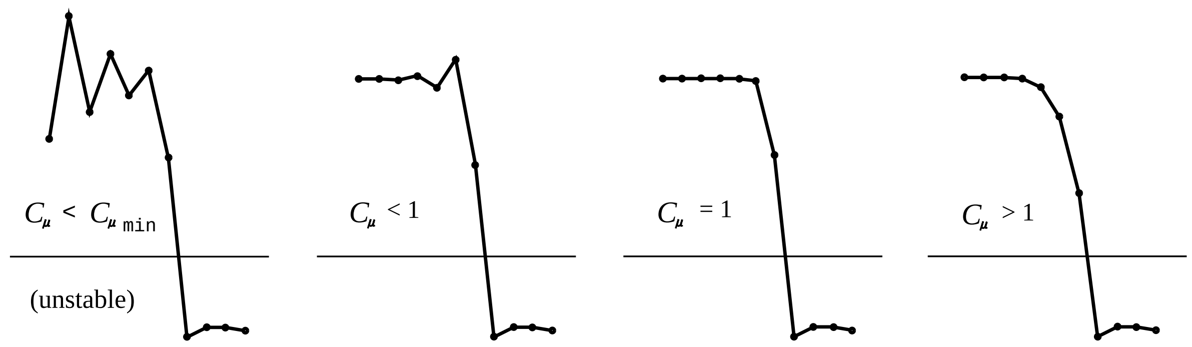


Figure 2: Effect of the dissipation weight C_μ on the numerical structure of a captured shock. Assumes $M_{\text{crit}} \simeq 1$.

$C_\mu / \cos^2\theta$, with θ being the shock angle. This favors somewhat smaller values of C_μ. The stability analysis indicates that in general the minimum allowable coefficients are

$$\begin{aligned} \text{1st-order dissipation:} \quad & C_{\mu_{\text{min}}} = 1/2 \\ \text{2nd-order dissipation:} \quad & C_{\mu_{\text{min}}} = 1/4 \end{aligned}$$

with $M_{\text{crit}} \leq 1$ being required in all cases. Violation of these thresholds will produce numerical instability and a nearly-singular Newton matrix.

Reduction of M_{crit} mainly has an effect where locally $M \simeq 1$. Specifically, the sharp subsonic recovery downstream of the shock is smeared if the post-shock Mach number falls significantly above M_{crit}, as shown in Figure 3. Hence, reduction of M_{crit} tends to smear weak shocks, but has little effect on strong shocks – a rather undesirable situation. It is therefore desirable to set M_{crit} just below 1.0 to give a small margin of safety against numerical instability.

The second-order dissipation substantially reduces spurious total pressure errors wherever the S-momentum equation is used, compared to the original first-order dissipation. It also gives crisper shocks for a given value of C_μ, gives more reliable shock loss results, and is nice in general. The benefits are greatest for weak oblique shocks, which tend to be quite heavily smeared with first-order dissipation.

First-order dissipation may also be required in certain cases where the second-order dissipation cannot maintain numerical stability even with a large C_μ. The most common source of difficulty is a normal shock traversing a sheared grid, especially when the local streamtube cell aspect ratio $A/\Delta s$ is large, as is typical in the outermost streamtubes. The shearing is typically caused by the quasi-normal

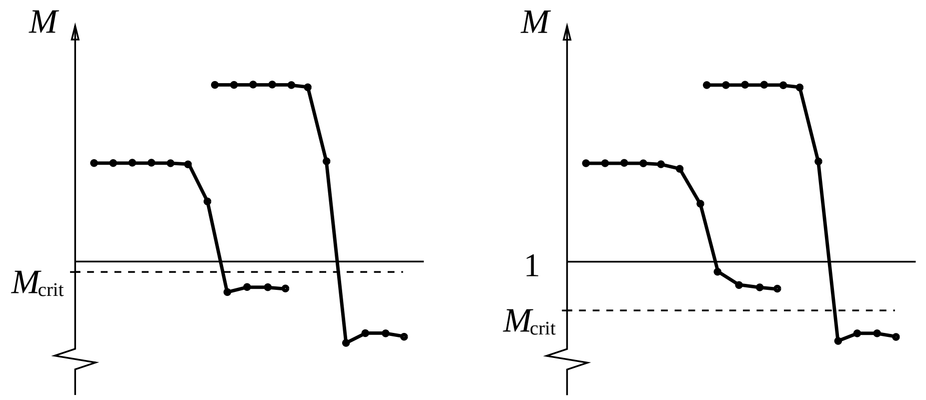


Figure 3: Effect of the dissipation threshold M_{crit} on the numerical structure of weak and strong captured shocks. Assumes $C_\mu \simeq 1$.

grid lines fanning out away from the airfoil. This fanning can reduced by increasing the "X-spacing parameter" in the **MSET** grid parameter change menu (described later). Increasing the number of points over the inlet and outlet streamlines also reduces this fanning out.

1.2.7 Dissipation enhancement during convergence

The movement of a captured shock tends to be a slow process, with the movement distance per Newton iteration being limited to the smeared shock's thickness. An ideally-resolved shock can therefore move at most one or two cells per iteration, which is deemed unacceptable. To speed up this process, **MSES** takes the liberty of temporarily reducing M_{crit} and/or increasing C_μ to smear the shock as much as possible so that it can move as fast as possible. The reduction of M_{crit} is especially effective here, since this smears the subsonic side of the shock, allowing it to move many cells per Newton iteration. The reduction is based on the maximum fractional density change from the previous Newton step,

$$\delta\bar{\rho} \;=\; |\, \delta\rho/\rho \,|_{\max}$$

which typically is large at a moving shock.

The temporary value used for M_{crit} is

$$\begin{aligned}
(M_{\text{crit}})_{\text{temp}} &= 0.75 \;+\; (M_{\text{crit}} - 0.75)\, \exp\Big(-r^2\Big) \\
\text{where} \quad r &= \frac{\delta\bar{\rho}}{\varepsilon} \\
\text{or} \quad r &= \frac{\delta\bar{\rho}^3}{\varepsilon\,(\delta\bar{\rho}^2 + (\varepsilon/4)^2)}
\end{aligned}$$

and $\varepsilon = 0.15$ is a reasonable threshold. When $\delta\bar{\rho}$ is large, the exponential is negligibly small, giving the decreased value of $(M_{\text{crit}})_{\text{temp}}$. As the shock reaches its destination, $\delta\bar{\rho}$ decreases, and $(M_{\text{crit}})_{\text{temp}}$ reverts to its prescribed value, which then sharpens the shock. The two forms for the argument r give nearly the same results, except very close to convergence where $\delta\bar{\rho}$ is small. The simple form for r gives $1 - \mathcal{O}(\delta\bar{\rho}^2)$ for the Gaussian, while the second more complicated form gives $1 - \mathcal{O}(\delta\bar{\rho}^6)$. Both forms give terminal quadratic convergence of the overall Newton scheme, but the second form has a larger basin of attraction and is preferred for this reason.

The "old" fractional density change $\delta\bar{\rho}$ is saved in the `mdat.xxx` file, so that the overall process is automated, and is not outwardly visible to the user even if the Newton iteration is halted and then restarted.

1.2.8 Mixed-Inverse flags

Note that the flags ISMOVE and ISPRES do not indicate the target element for Mixed-Inverse. That information is contained in `mdat.xxx`, and corresponds to the last-edited element in **MEDP** before `mdat.xxx` was written out.

1.2.9 Actuator Disk parameters

MSES has the ability to model a fan as an actuator disk, which introduces momentum and energy sources into the flow. The S-momentum residuals are modified to the following form:

$$\text{ISMOM=1:} \qquad \mathcal{R}_1 \;\equiv\; \Delta p \,+\, \frac{m}{A}\Delta\tilde{q} \,-\, P_s \,-\, \rho(\Delta h_0)_{AD} \,+\, p(\Delta S)_{AD} \;=\; 0$$

$$\text{ISMOM=2 :} \qquad \mathcal{R}_2 \;\equiv\; -p\,\Delta\tilde{S} \,+\, p(\Delta S)_{AD} \;=\; 0$$

The $(\Delta h_0)_{AD}$ and $(\Delta S)_{AD}$ quantities are nonzero only in cells containing the actuator disk, and are specified via the total-pressure ratio and the efficiency.

$$\begin{aligned}
\tau_{AD} &\equiv h_{0_2}/h_{0_1} \\
\pi_{AD} &\equiv p_{0_2}/p_{0_1} \qquad \text{(specified by PTRHIN)} \\
\eta_h &\equiv \frac{\pi_{AD}^{(\gamma-1)/\gamma} - 1}{\tau_{AD} - 1} \qquad \text{(specified by ETAH)} \\
(\Delta h_0)_{AD} &= \frac{h_{0_1}}{\eta_h}\left(\pi_{AD}^{(\gamma-1)/\gamma} - 1\right) \;=\; h_{0_2} - h_{0_1} \\
(\Delta S)_{AD} &= \frac{\gamma}{\gamma-1}\,\ln\left[\frac{h_{0_2}}{h_{0_1} + \eta_h(\Delta h_0)_{AD}}\right]
\end{aligned}$$

For low speed flows, it is sometimes more convenient to work with the total pressure rise coefficient ΔC_{p_0} rather than π_{AD}. The following relations are useful for translating between these two alternative parameters.

$$\begin{aligned}
\frac{p_{0_2} - p_{0_1}}{\frac{1}{2}\rho_\infty V_\infty^2} \;\equiv\; \Delta C_{p_0} &= (\pi_{AD} - 1)\,\frac{2}{\gamma M_\infty^2}\left[1 + \frac{\gamma-1}{2}M_\infty^2\right]^{\gamma/(\gamma-1)} \\
\pi_{AD} &= 1 \,+\, \Delta C_{p_0}\,\frac{\gamma M_\infty^2}{2}\left[1 + \frac{\gamma-1}{2}M_\infty^2\right]^{-\gamma/(\gamma-1)}
\end{aligned}$$

The relations above are exact. For low speed flows, $M_\infty^2 \ll 1$, they can be simplified by omitting the bracket term.

1.2.10 Input file examples

Viscous analysis for two-element case, specified angle of attack, vortex+doublets+source farfield, isentropic except near shocks, free transition (this is the most common case):

```
3 4 5 7
3 4 5 7
0.30  0.5   2.20      | MACHIN  CLIFIN  ALFAIN
4     2               | ISMOM   IFFBC
4.5E6 9.0             | REYNIN  ACRIT
1.0   1.0  1.0  1.0   | XTRS    XTRP    ...
0.99  1.2             | MCRIT   MUCON
```

Same as above, except isentropic condition used only near stagnation points. This doesn't rely on a shock finder, and hence is slightly more robust:

```
3 4 5 7
3 4 5 7
0.30  0.5   2.20      | MACHIN  CLIFIN  ALFAIN
3     2               | ISMOM   IFFBC
4.5E6 9.0             | REYNIN  ACRIT
1.0   1.0  1.0  1.0   | XTRS    XTRP    ...
0.99  1.2             | MCRIT   MUCON
```

Viscous analysis for two-element case, specified angle of attack, solid-wall farfield, fully isentropic, fixed transition on top side at 15% chord of element 1:

```
4 5 16
3 4 5
0.30  0.5   2.20      | MACHIN  CLIFIN  ALFAIN
2     1               | ISMOM   IFFBC
4.5E6 9.0             | REYNIN  ACRIT
0.15  1.0  1.0  1.0   | XTRS    XTRP   ...
0.99  1.0             | MCRIT   MUCON
```

Inviscid analysis for single-element case, farfield doublets+source held at current values, specified lift coefficient, isentropic only near leading edge, first-order dissipation:

```
3 4 5
3 4 6
0.70  0.5   2.20      | MACHIN  CLIFIN  ALFAIN
3     2               | ISMOM   IFFBC
0.0E6 9.0             | REYNIN  ACRIT
1.0   1.0             | XTRS    XTRP
0.99  -1.0            | MCRIT   MUCON
```

Inviscid analysis for single-element case, farfield doublets+source forced to zero, specified lift coefficient, isentropic only near leading edge, first-order dissipation:

```
3 4 5 7
3 4 6 8
0.70  0.5   2.20     | MACHIN  CLIFIN  ALFAIN
3     2    0. 0. 0.  | ISMOM   IFFBC  [ DOUXIN  DOUYIN  SRCEIN ]
0.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  -1.0           | MCRIT   MUCON
```

Viscous analysis for single-element case, with constant-lift Mach and Reynolds number variation ($M = 0.65/\sqrt{C_L}$, $Re = 2.0e6/\sqrt{C_L}$):

```
3 4 5 7 15 10
3 4 5 7 16 18
0.65  0.8   2.20     | MACHIN  CLIFIN  ALFAIN
4     2              | ISMOM   IFFBC
2.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  1.0            | MCRIT   MUCON
```

Mixed-Inverse inviscid camber design driven with upper surface pressures, specified angle of attack:

```
3 4 5 7 11 12
3 4 5 7 11 12
0.70  0.55  2.20     | MACHIN  CLIFIN  ALFAIN
4     2              | ISMOM   IFFBC
0.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  1.0            | MCRIT   MUCON
0     1              | ISMOVE  ISPRES
```

Modal inverse run with 12 shape mode DOFs and 2 position mode DOFs 2 position DOFs.

```
3 4 5 7 20 30
3 4 5 7 40 50
0.70  0.55  2.20     | MACHIN  CLIFIN  ALFAIN
4     2              | ISMOM   IFFBC
2.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  1.0            | MCRIT   MUCON
0     0              | ISMOVE  ISPRES (not used here)
12    2              | NMODN   NPOSN
```

Sensitivity run for viscous optimization using 12 mode DOFs and 2 position DOFs.

```
3 4 5 7 20 30
3 4 5 7 20 30
0.70  0.55  2.20     | MACHIN  CLIFIN  ALFAIN
4     2              | ISMOM   IFFBC
```

```
2.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  1.0            | MCRIT   MUCON
0     0              | ISMOVE  ISPRES (not used here)
12    2              | NMODN   NPOSN
```

Flap position mode DOF 31 (assumingly a rotation mode), driven to attain specified $C_L = 1.5$, at fixed $\alpha = 2.2°$.

```
3 4 5 7 30
3 4 5 7 6
0.20  1.5   2.20     | MACHIN  CLIFIN  ALFAIN
4     2              | ISMOM   IFFBC
1.0E6 9.0            | REYNIN  ACRIT
1.0   1.0            | XTRS    XTRP
0.99  1.0            | MCRIT   MUCON
0     1              | NMODN   NPOSN
```

2 Programs

For repetitive program execution, as typically occurs in an airfoil design session, it is convenient to use the simple shell script `mrun`. This is invoked with

```
 % mrun xxx
```

which then produces a simple menu for executing any of the programs, editing input files, etc. Alternatively, the programs can be executed manually as described below.

2.1 AIRSET — airfoil geometry manipulator

AIRSET is a menu-driven program which can be used to manipulate multi-element airfoil configurations, using a `blade.xxx` file or a bunch of separate element files as the input. Starting **AIRSET** with an argument:

```
 % airset xxx
```

will cause `blade.xxx` to be automatically loaded. It can also be loaded from the menu if the argument is not specified.

AIRSET permits actions such as splitting off a flap element and modifying the contour with the screen cursor. Likewise, element translation, rotation, scaling, etc., can be readily performed with immediate graphical feedback. Corner points can also be added or deleted. An inviscid panel method with a compressibility correction is also provided for a quick sanity check on any modification. Refer to the *AIRSET User Guide* manual for more details.

2.2 MSET — grid and flowfield initializer

MSET is the program which initializes the grid, the flowfield, and a variety of other variables. It is executed with the command

```
 % mset xxx yyy
```

with the second argument `yyy` optional. **MSET** reads the airfoil coordinate input file from `blade.yyy`, or `blade.xxx` if the `yyy` argument is omitted. The intent of the two arguments `xxx` and `yyy` is to allow one `blade.yyy` coordinate file to be used for many `xxx` flow cases without the need to create a `blade.xxx` file for each case. The output in any case is the main solution state file `mdat.xxx`.

New for **MSES** 3.04: If `blade.xxx` or `blade.yyy` does not exist, then **MSET** tries to read file `yyy` (i.e. without the `blade.` prefix). This allows using the common `.dat` airfoil file to set up a case. For example,

```
 % mset n12 naca0012.dat
```

will create the state file `mdat.n12`, using the airfoil coordinates in file `naca0012.dat`.

MSET also tries to read the gridding parameter save file `gridpar.xxx` (to be described shortly).

MSET is menu-driven to allow the user to iteratively generate a good initial grid by tweaking a small number of gridding parameters. The top level **MSET** menu is

```
   1  Specify alpha and generate streamlines
   2  Generate spacing and initialize grid
   3  Elliptic grid smoother
   4  Write  mdat.xxx

   5  Plot grid
   6  Plot Cp vs x/c

   7  Modify grid parameters
   8  Write  grid parameters to gridpar.xxx

   9  Change plot size
  11  Zoom
  12  Unzoom

  13  Write  blade.xxx
  14  Write  mses.xxx  with defaults
```

The generate the initial grid and write out the solution file, Options 1, 2, 3, 4 (in that order), must be issued as a minimum. The only exception is that Option 1 can be skipped if the angle of attack has been previously defined and written to the `gridpar.xxx` file, described shortly.

New for **MSES** 3.04: Options 13 and 14 conviently allow setting up an MSES run case using only a common `.dat` airfoil file, given as the 2nd Unix argument. For example,

```
% mset n12 naca0012.dat
  1
  2
  3
  4
 13
 14
```

will generate not only file `mdat.n12`, but also `blade.n12` and `mses.n12`.

2.2.1 Panel solution

Option 1 uses the specified angle of attack to generate a panel solution, which is then used to trace a pair of stagnation streamlines just above and below each element, as well as the upper and lower farfield streamlines. Iso-potentials emanating from all leading and trailing edges are also located. This divides up the domain into blocks, which are then automatically displayed in a plot. These blocks form the skeleton on which the grid is generated. The specified angle of attack here is of course somewhat arbitrary, since it is only used for initial grid generation. It is a good idea, however, to specify an angle of attack which avoids massive C_p spikes on any of the elements. This minimizes the start-up trauma with a subsequent viscous **MSES** solution. After Option 1 is issued, the C_p distributions can be immediately displayed with Option 6.

2.2.2 Initial surface gridding

Option 2 distributes grid nodes along the streamlines on the element surfaces (the total number of nodes on each surface may need to be changed first with option 7 described below). The local arc-length increment Δs between two surface grid nodes is determined from

$$\Delta s \sim \frac{1}{1 + a|\kappa|^b}$$

where κ is the local surface curvature. In regions of high curvature, the spacing is therefore smaller, depending on the curvature exponent b and the coefficient a. The exponent is specified directly from the menu described below. A large exponent ($b = 2$, say), makes the spacing small in high-curvature regions. A small exponent ($b = 0.05$, say), makes the spacing more nearly uniform everywhere. The curvature coefficient a is indirectly controlled by specifying Δs spacing at the leading edge (or stagnation point, to be more precise) of the element. Note that if κ is rather small at the stagnation point, the effect of a is largely shut off in the expression above. If necessary, a fudged additional curvature is added locally very near the stagnation point to allow the spacing requirement to be met. A message is printed when this action is taken. Fudged curvatures are also introduced near the trailing edge, and optionally at selected local-refinement zones on the upper and lower surfaces. The aim is also to control the local spacing.

For MSES v 2.9, rather than the actual local surface spacings Δs, the local/average spacing ratios $\Delta s/\Delta s_{\text{avg}}$ are specified instead. The average spacing is defined as $\Delta s_{\text{avg}} = s_{\text{side}}/N$, where s_{side} is 1/2 of the airfoil perimeter and N is the number of airfoil-side points. Specifying the ratio is more convenient than specifying Δs itself, since the former automatically adjusts the spacing if the number of points is changed.

The curvature exponent, and stagnation-point and trailing-edge spacing ratios and local-refinement parameters are altered from the following menu, which appears each time the spacing is generated and plotted.

```
   D sLE/dsAvg, dsTE/dsAvg  spacing ratios
   C urvature exponent
   U pper side spacing refinement
   L ower side spacing refinement

 Change what? (<Return> if spacing OK):
```

For each change request, the current values are displayed after the prompt. Selecting "`U`", for example, might produce

```
 Enter new upper s/smax limits, local/max density ratio  0.0100  0.0500   0.800
```

and just hitting <Return> will take the current values as the default input. One can change only some of the required three inputs by using commas. Entering

```
0.02
```

will only change the first value from 0.01 to 0.02, while entering

```
,,0.5
```

will only change the third value from 0.8 to 0.5. The first two "`s/smax`" values specify the fractional arc length from stagnation point to the trailing edge where the local refinement is to be placed. Tick marks inside the airfoil element contour indicate this fractional arc length in increments of 0.10. The local/max density ratio specifies the increase in local grid density over the max density which would occur in absence of the local refinement.

After the spacing parameters are altered, the new distribution is generated and displayed. The actual LE, TE, max, min, spacing ratios are also printed out. It must be mentioned that only the stagnation point spacing ratio "`dsLE/dsAvg`" can be controlled precisely with the input parameters. The other spacing ratios are approximate and may need to be iterated.

Once a good node distribution on each element is obtained, **MSET** proceeds to modify all the spacings to resolve any conflicts between adjacent elements. Basically, all the distributions are automatically fudged to make spacings on element surfaces facing each other match as closely as possible. This prevents massive grid shearing which would otherwise occur. No such action is necessary for single-element airfoils.

Once all the node distributions are finalized, intermediate streamline nodes are generated in the flowfield interior by simple linear interpolation from the stagnation and farfield streamlines. The resulting grid is not yet suitable for **MSES** calculations, but can be viewed with Option 5.

2.2.3 Grid smoothing

Option 3 invokes an elliptic SLOR grid smoother to "clean up" the linearly interpolated grid. This eliminates all kinks, overlaps, and also makes all the grid streamlines correspond to streamlines of incompressible inviscid flow. This is then an excellent initial guess for the **MSES** solver.

2.2.4 Initial solution file output

After the grid is smoothed, Option 4 can be issued to write out the initial solution file `mdat.xxx` which is then ready for the **MSES** solver. Before this is done, however, it is a good idea to view the grid with Option 5. Options 2,3 or 1,2,3 can then be repeated if necessary to obtain an acceptable grid before it is written out.

2.2.5 Grid parameters

Option 7 puts up the menu

```
    Current grid parameters:
 N    141   Airfoil side points (for chord = 1)          E
 E  0.400   Exponent for airfoil side points: n = N*chord
 I     35   Inlet  points on leftmost  airfoil streamline
 O     35   Outlet points on rightmost airfoil streamline
 T     19   Number of streamlines in top of domain
 B     15   Number of streamlines in bottom of domain
 M      7   Maximum streamlines between two elements
 X  0.850   X-spacing parameter
 A  2.500   Aspect ratio of each cell at stagnation point
```

```
    4.000   Angle of attack
 Change what (<cr> if done) ?
```

in which each gridding parameter is displayed and can be immediately altered (the angle of attack is altered with Option 1 at the main menu). The number of surface grid nodes allocated to each element is controlled with the two parameters labelled "`N`" and "`E`". Note that when `E` = 1.0, the number of grid points on each element is proportional to its chord, while if `E` = 0.0, all elements receive the same number of grid points. The X-spacing parameter "`X`" controls how much the quasi-normal grid lines spread out away from the airfoil. A larger value (0.8 ... 1.0) tends to make the grid more orthogonal (which is good), but may cause excessive bunching of the quasi-normal grid lines in high-lift cases (which is bad). Transonic flows are best run with a nearly-orthogonal grid, since grid shearing increases the amount of dissipation needed for stability.

The action of the remaining parameters is straightforward. It must be kept in mind that the point number parameters are only approximate, since **MSET** takes liberties in fudging point numbers to keep the grid shearing under control. Typically, they need to be iteratively adjusted.

If grid coarsening (described later) is to be performed, all the integer grid parameters above must be odd. Future MSES versions will probably automatically ensure this.

2.2.6 Grid parameter saving, recall

Once a good set of gridding parameters is obtained, including the spacing parameters generated with Option 2, they can all be saved to `gridpar.xxx` by specifying Option 8 at any time. If this file already exists, it is overwritten. `gridpar.xxx` will then be automatically read when **MSET** is executed again for that same `xxx` case, which causes all the gridding parameters to take on their saved values. This allows rapid generation of grids for cases which differ only slightly (e.g. flap deflection or angle of attack), since the same gridding parameters can then be used.

If `gridpar.xxx` has the angle of attack defined ($\alpha = 999.0$ implies it is not defined), then this will be used to generate a panel solution as soon as **MSET** is started. Option 1 in the main menu can then be skipped.

2.3 MSES — flow solver

MSES is the main program that solves the Euler equations. It has two input files, `mdat.xxx` and `mses.xxx`, and then writes the output file back to `mdat.xxx`. Thus the input file `mdat.xxx` can either be a restart file from an old calculation, or a brand-new file created by **MSET** .

2.3.1 Execution

MSES is run by typing

```
 % mses xxx n
```

with "`n`" (optional) being the maximum number of Newton iterations to be executed. If "`n`" is omitted, the user is asked for the number of iterations to be executed. If the user enters "n" then the program response is

0	writes solution to output file and terminates
n	performs n iterations and repeats the question

2.3.2 Coarsening/refining

MSES 2.95 has the capability to coarsen or refine the grid to possibly allow speeding up the convergence process. This feature is still experimental, and is currently entirely user-driven. It may become automated in future MSES versions.

Coarsening or refining is commanded by specifying "$-$" or "+" at the iteration-number prompt:

$-$	coarsens the grid, repeats question
$+$	refines the grid, repeats question

One can also specify the $\pm$ sign together with the number of iterations. Note that in this case, entering $+n$ is not the same as just n.

Coarsening is done by deleting every other grid point, so that the grid dimensions are halved. This reduces the CPU time per iteration by nearly a factor of 8, and also doubles the distance that a shock is propagated every iteration. Needless to say, coarsening is an extremely efficient way to obtain a reasonable flowfield solution estimate, especially for transonic cases. Subsequent refining interpolates the coarse solution back onto a fine grid, created by doubling the points in both directions. This is then an excellent starting point for reconvergence on the fine grid, which will now converge fairly quickly. Instead of just requesting $n = 15$ on the fine grid, one could request $n = -10$ and then $n = +5$, which will attain the solution in less than half the time. The same effect can also be achieved by running **MSES** in batch mode twice:

```
 % mses xxx -10
 % mses xxx +5
```

A flow case can be coarsened two or more times, but this is rarely advantageous.

Currently, coarsening is possible only if the number of grid lines over each grid segment is odd. The default gridding parameters in **MSET** for a single-element airfoil will always produce a grid which can be coarsened. The gridding logic for multi-element airfoils is relatively complicated, and currently **MSET** cannot guarantee that any particular multi-element grid can be coarsened. However, a grid can always be refined, provided the array dimensions are sufficiently large. Hence, a reasonable approach is to generate a coarse grid which is roughly half as dense as usual in each direction. After this is converged, it can be refined for the final solution.

2.3.3 Iteration convergence

After each iteration, **MSES** prints out a small table of numbers. These represent how much various parameters have changed (r.m.s. and maximum) during this iteration. The parameters are:

d(R)	density change
d(n)	grid node displacement (the grid position is part of the solution)
d(Gamma)	circulation (approx. $C_L/2$)
d(Alpha)	angle of attack

d(V)	all of the viscous variables, which are:	
	d(C)	max. Reynolds stress stress coefficient
	d(N)	transition amplification factor
	d(T)	momentum thickness
	d(D)	displacement thickness
	d(H)	shape parameter

Also displayed is the change in viscous C_D if a viscous case is being run. Convergence to plotting accuracy occurs when the changes drop to about 0.1×10^{-3} or so. If no separation is present, convergence to machine accuracy is achieved when the changes refuse to go down further with each iteration (about 0.1×10^{-5}). If significant separation is present, the final change magnitudes will be somewhat larger — about 0.1×10^{-4} or even 0.1×10^{-3}. This is due to the Newton matrix for such cases being relatively poorly-conditioned, which magnifies machine roundoff. Poor conditioning will also result from very low Mach numbers. It is recommended that a Mach number of at least 0.05 be used, and 0.10 is preferable if extensive separation is present or the case is near $C_{L\max}$, since this compounds the poor-conditioning problems. If this is a problem, **MSES** can be built in double precision by simply setting the appropriate flag in the `Makefile` (likewise for the plot library). The subcritical version **MSIS** , described later, does not have this machine roundoff problem, and can tolerate very small Mach numbers even in single precision.

The number of iterations specified in the command line or interactively is actually only a limit. **MSES** will terminate early if the r.m.s. changes drop below the convergence tolerance limits defined in file `EPS.INC`, and the maximum changes drop below 10× these limits.

2.4 MSIS — fast subcritical flow solver

Using **MSIS** instead of **MSES** is functionally equivalent to using **MSES** with ISMOM=2. It solves the Euler equations with the S-momentum equation replaced by an isentropic condition everywhere. The equation actually used is

$$\mathcal{R}_2 \;\equiv\; \tilde{S} \;=\; 0$$

This is a pointwise equation. When it is effectively eliminated from the system, the Newton matrix then has about half as many non-zero entries as with **MSES** , so **MSIS** is 2–4 times as fast. The input file `mses.xxx` is exactly the same as described above, except that ISMOM, MCRIT, and MUCON are not used. Also, **MSES** and **MSIS** can be run interchangeably with the same `mdat.xxx` file.

MSIS has another advantage in that it can calculate flows for extremely small freestream Mach numbers — $M_\infty \simeq 0.001$ or less (this feature is new for MSES v 2.9). The manner in which **MSIS** calculates the pressure from the streamtube area, mass flow and total enthalpy is not sensitive to machine round-off. **MSES** does have this sensitivity, and refuses to converge properly if $M_\infty < 0.10$ for viscous cases with separation, or if $M_\infty < 0.02$ for inviscid cases.

The **MSIS** executable image is formed by replacing `setup.o` with `setis.o`, and `fsolve.o` with `isolve.o` during linking. The isentropic version of the polar driver is called **MPOLIS** , and it is likewise linked with `setis.o` and `isolve.o` . Like **MSIS** , **MPOLIS** can calculate flows at very low Mach numbers.

Important: **MSIS** and **MPOLIS** can only be used (and are recommended) for subcritical cases. If any part of the flowfield in the converged solution or a transient becomes supersonic, **MSIS** or **MPOLIS** will fail. **MSIS** and **MPOLIS** also tend to have more start-up trauma for moderate-Mach cases (converged maximum local $M = 0.8$, say), with the initial local Mach tending to overshoot. An effective fix is to first perform one or two iterations with **MSES** , and then proceed with **MSIS** for all subsequent calculations.

2.5 MPOLAR — parameter-sweep calculation driver

MPOLAR is a version of **MSES** which conveniently sweeps through a range of a specified parameter, thus generating a polar curve. Because it takes full advantage of the quadratic convergence of the Newton method, using **MPOLAR** is much more efficient (and much easier!) than running a sequence of independent cases from scratch with **MSES** or **MSIS** .

Note that previous **MSES** versions had separate polar programs for alpha, CL, and Mach sweeps. These have been combined into the one current **MPOLAR** program.

In addition to the usual **MSES** input files, **MPOLAR** also requires an `spec.xxx` file, which contains the sequence of parameters, with an optional state file dump flag for each angle:

```
KSPEC
SPECIN(1)    [ KSENS(1) ]
SPECIN(2)    [ KSENS(2) ]
  .
  .
  .
SPECIN(NA)   [ KSENS(NA) ]
```

The first line indicates the index of the constraint whose parameter values are given in the subsequent lines. This constrain must also appear in the second line of the `mses.xxx` file. Currently allowed values for KSPEC, and corresponding SPECIN interpretations are

```
KSPEC  SPECIN
-----  ------
  5    ALFAIN
  6    CLIFIN
 15    MACHIN
 16    MACHIN  interpreted as MACH*sqrt(CL)
 17    REYNIN
 18    REYNIN  interpreted as REYN*sqrt(CL)
```

If the optional KSENS number on any line is nonzero, then a `mdat.xxx_n` file, with `n` = KSENS, will be written for that point after convergence as described later.

For any type of polar, it is best to make the parameter increment sufficiently small to give reliable reconvergence for each point. If `KSPEC = 5`, it is appropriate to make the α increment no larger than $0.5°$. For strong transonic cases which typically have large $dC_L/d\alpha$ values, an increment of $0.25°$ or even $0.1°$ may be appropriate. Although larger increments will result in less points to be computed, the number of iterations per point will increase such that little or no overall saving will result. Using small increments also gives a more robust sweep in nasty regions such as past $C_{L_{\max}}$. If the first point in the polar is a difficult case, it is suggested that that point be partially converged with **MSES** , so that the subsequent **MPOLAR** execution will not exceed its iteration limit. **MPOLAR** is executed by the command

```
 % mpolar xxx
```

and will perform all calculations without requiring further user input.

Coarsening/refining is currently not available with any of the polar driving programs. This may change in the future after some experience with these options is gained.

Everytime **MPOLAR** converges on a point to the tolerances in `EPS.INC`, it appends the integrated parameters to `polar.xxx`, appends the surface pressure and boundary layer variable distributions to `polarx.xxx`, and overwrites `mdat.xxx` with the converged solution. If KSENS for that point is nonzero and modal-geometry or element-position DOFs are active, then the sensitivity dump file `sensx.xxx_n` is also written out, with `n` = KSENS. The `sensx.xxx_n` files are used as input to the optimization driver program **LINDOP** .

If **MPOLAR** fails to converge on any one point, it will restart from the previously-converged point, and subdivide the offending SPEC increment. If that point fails, the SPEC increment will be subdivided further. If no convergence is achieved after five subdivisions, **MPOLAR** will terminate with a "Severe convergence problem detected" message. Failure to converge on any one point may be due to massive separation, or a very dramatic jump in a transition location. Occasionally, a limit cycle occurs, with a transition location oscillating back and forth. This is often caused by inadequate resolution near the transition point. Simply restarting **MPOLAR** may fix the problem, since such a restart will begin after the offending point if the preceding interval was subdivided previously.

If after **MPOLAR** execution it is found that the polar is not complete, points can be added to the `spec.xxx` file, and **MPOLAR** restarted. The new points will be automatically appended to the save and dump files. The `sensx.xxx_n` file counter `n` will also be properly restarted.

The results in `polar.xxx` can be perused directly, or plotted in pretty format using programs **PPLOT** or **SPLOT** . The results in `polarx.xxx` can only be viewed via the plotting program **MX-PLOT** .

With **PPLOT** , one can also plot two or more polars superimposed, and a file with comparison data can also be specified. The format of this file is as follows.

```
CD(1)  CL(1)
CD(2)  CL(2)
 .       .
 .       .
999.0  999.0
alpha(1)  CL(1)
alpha(2)  CL(2)
  .         .
  .         .
999.0   999.0
alpha(1)  Cm(1)
alpha(2)  Cm(2)
  .         .
  .         .
999.0   999.0
Xtr/c(1)  CL(1)
Xtr/c(2)  CL(2)
  .          .
  .        .
999.0   999.0
```

The number of points in each set (CD-CL, alpha-CL, etc.) is arbitrary, and can be zero. **PPLOT** is simply run from the terminal and prompts the user for the full input polar filenames and reference data filenames. The polar filenames do not need to have the "`.xxx`" name convention. **PPLOT** also tries to read the file `pplot.def`, which contains various plotting parameters. If `pplot.def` is not found, internal hard-wired defaults are used.

The program **MXPLOT** which plots the polar dump file is executed using

```
 % mxplot  <dump filename>
```

As with **PPLOT** , the usual "`.xxx`" name convention is not used here since more flexibility in naming the polar save and dump files is usually needed. **MXPLOT** is entirely menu-driven and requires no further explanation here.

MPOLAR requires that DALFA (4) be chosen as one of the global variables, and that the prescribed-α condition (5) be one of the global constraints. The input value ALFAIN in `mses.xxx` is ignored.

2.6 MPOLIS — fast subcritical parameter-sweep calculation driver

MPOLIS is the same as **MPOLAR** , except it's the equivalent of running **MSIS** rather than **MSES** over a parameter range. It uses the same `spec.xxx` parameter file.

2.7 MPLOT — solution plotter

MPLOT is the program which displays the solution in `mdat.xxx` at any time whether the solution is converged or not. It is executed by the command

```
 % mplot xxx
```

Note that if the solution in `mdat.xxx` is not converged, the results are physically meaningless. **MPLOT** has five major plotting options, selected from the top-level menu:

```
    1  Airfoil surface  plots
    2  Streamtube       plots
    3  Contour/grid     plots
    4  Outer streamline plots
    5  Wake profile     plots

    6  Zoom
    7  Unzoom
    8  Plotting options

    9  Re-read mdat.xxx
   10  Dump streamline grid
   11  Dump flowfield
   12  Dump BLs
```

Each command is largely self-explanatory. The "Outer streamline plots" option 4 is mainly intended to examine tunnel-wall pressures in solid-wall cases, although it can be used for any flow case. Outer-streamline velocities or pressures can be output from a converged `mdat.xxx` file by using the simple separate program **UEWALL** (see header in `uewall.f`).

2.7.1 BL variable plots

The "Surface plots" menu brought up by the top-level option 1 allows plotting of most of the airfoil surface and wake boundary layer variables:

```
  1    Mach vs x
  2      Cp vs x
                      --------------------
  3       H vs x
  4 top D,T vs x         BL data
  5 bot D,T vs x               on
  6      Cf vs x                airfoil
  7      Ue vs x                 and
  8    A/Ao vs x                  wake
  9      Ct vs x
 10  Rtheta vs x
 11   CDiss vs x
                      --------------------
 12   List CL, CD ...
 13   Change settings
 14   Hardcopy current plot
 15   Place symbol at specified grid location

 16   Change x-axis limits for element BL plots
 17   Change y-axis limits for current element BL plot
 18   Cursor blowup for current BL plot
 19   Reset x,y-axis limits for element BL plots

 20   Annotation menu
 21   Plotting options
 22   Specify/Change element for plotting
 99   Re-load mdat.xxx and replot
```

The `Mach` and `Cp` menu items (1 and 2), can display the distributions on one element, or all of them simultaneously. The `Cp` plot item 2 displays the extrapolated wall pressure p_w (described earlier). The "BL data" menu items (3 ... 11) display quantities for at most one element at a time. Items 3, 6 ... 11 show one variable on both sides of the element, while items 4,5 show δ^* and θ together for one side only. Items 4,5 also show the total (top + bottom) thicknesses for the wake as dotted lines, and also the inviscid-grid wall-offset distance Δn as a dashed line. Normally, $\Delta n = \delta^*$, and the two curves will overlay, but only if the case is fully converged. If the dashed Δn curve can be discerned, the case is not converged. Item 7 shows both the viscous edge velocity u_e as a solid line, and also the local inviscid velocity u_i defined at the displacement surface. There may be a visible difference between u_e and u_i if the case is not converged. Items 16,17,18,19 allow rescaling of the BL variable plot axes to zoom in on details of interest.

2.7.2 Force coefficients

The various forces and moments are computed in a number of ways. The x and y pressure and friction forces and the moment are computed for each element by direct integration over the surface.

$$\begin{aligned}
C_{x_p} &= \frac{2}{\rho_\infty V_\infty^2} \oint -p\, dy \\
C_{y_p} &= \frac{2}{\rho_\infty V_\infty^2} \oint p\, dx \\
C_{x_f} &= \frac{2}{\rho_\infty V_\infty^2} \oint \tau_w\, dx \\
C_{y_f} &= \frac{2}{\rho_\infty V_\infty^2} \oint \tau_w\, dy \\
C_m &= \frac{2}{\rho_\infty V_\infty^2} \oint \{-p\,[(x-x_o)\,dx + (y-y_o)dy] \;+\; \tau_w\,[(x-x_o)\,dy - (y-y_o)dx]\}
\end{aligned}$$

The total lift and moment, and the friction drag are then determined by summing over the elements, and projecting the lift and friction drag along the freestream axes.

$$\begin{aligned}
C_L &= \sum \left[(C_{x_p} + C_{x_f})\,\cos\alpha \;-\; (C_{x_p} + C_{x_f})\,\sin\alpha\right] \\
C_{D_f} &= \sum \left[C_{x_f}\,\cos\alpha \;+\; C_{y_f}\,\sin\alpha\right] \\
C_M &= \sum C_m
\end{aligned}$$

The pressure drag is not computed via the surface pressures, since this tends to be rather inaccurate.

The total drag is computed by performing the equivalent of a wake survey over the exit plane of the grid. The last computational point of each element's wake contains the wake exit quantities

$$\rho_{e_\mathrm{exit}} \qquad u_{e_\mathrm{exit}} \qquad p_{e_\mathrm{exit}} \qquad \theta_\mathrm{exit} \qquad \delta^*_\mathrm{exit}$$

This station is nearly at the freestream pressure, $p_{e_\mathrm{exit}} \simeq p_\infty$, but for maximum accuracy, especially on small grids, it is extrapolated to the exact p_∞ condition using the von Karman equation for a wake.

$$\begin{aligned}
d\left(\rho_e u_e^2 \theta\right) &= \delta^*\; dp \\
d\,\ln \rho_e u_e^2 \theta &= \frac{H}{\gamma M_e^2}\, d\,\ln p \\
(\rho V^2 \theta)_\infty &= \rho_{e_\mathrm{exit}} u_{e_\mathrm{exit}}^2 \theta_\mathrm{exit} \left(\frac{p_\infty}{p_{e_\mathrm{exit}}}\right)^{H_\mathrm{avg}/\gamma M_\infty^2} \\
\text{where} \quad H_\mathrm{avg} &= \frac{1}{2}\,(H_\mathrm{exit} \;+\; H_\infty) \\
\text{and} \quad H_\infty &= 1 \;+\; (\gamma-1)M_\infty^2
\end{aligned}$$

The extrapolated momentum defects are then summed over the elements to get the total viscous drag.

$$C_{D_v} = \frac{2}{\rho_\infty V_\infty^2} \sum (\rho V^2 \theta)_\infty$$

The wave drag is determined by computing the momentum defect of all the streamtubes in the inviscid part of the flow. This requires sampling the exit plane inviscid velocity and pressure

$$q_{\text{exit}} \qquad\qquad p_{\text{exit}}$$

and isentropically extrapolating this velocity to freestream pressure to get $q_{+\infty}$.

$$\begin{aligned} p_{\text{exit}} \left(1 - \frac{q_{\text{exit}}^2}{2h_{0_\infty}}\right)^{-\gamma/(\gamma-1)} &= p_\infty \left(1 - \frac{q_{+\infty}^2}{2h_{0_\infty}}\right)^{-\gamma/(\gamma-1)} \\ q_{+\infty} &= \left\{ 2h_{0_\infty} \left[1 - \left(\frac{p_\infty}{p_{\text{exit}}}\right)^{(\gamma-1)/\gamma} \left(1 - \frac{q_{\text{exit}}^2}{2h_{0_\infty}}\right)\right]\right\}^{1/2} \end{aligned}$$

The wave drag is then computed by integrating over all the streamtubes, each with mass flow dm.

$$C_{D_w} = \frac{2}{\rho_\infty V_\infty^2} \int (V_\infty - q_{+\infty})\, d\dot{m}$$

Unlike for all the other force components, this drag cannot be assigned to the individual elements.

The total drag coefficient is finally obtained by adding the viscous and inviscid momentum defects

$$C_D = C_{D_v} + C_{D_w}$$

and finally the pressure-drag component is indirectly obtained by subtracting off the friction drag.

$$C_{D_p} = C_D - C_{D_f}$$

Note that all the force coefficients are normalized with the freestream dynamic pressure $\rho_\infty V_\infty^2/2$. The absence of the usual chord normalization means that they are instead normalized by the same length unit L as used for the input airfoil coordinates. Hence, they will be the usual C_L, C_D, etc. only if the airfoil coordinates in `blade.xxx` have a unit chord. Boundary layer thicknesses such as δ^* and θ are also displayed without normalization, so in effect they are in this same length unit L.

The skin friction coefficient C_f (item 6) and dissipation coefficient $C_{\mathcal{D}}$ (item 11) are currently normalized with the freestream quantities:

$$C_f = \frac{2\,\tau_w}{\rho_\infty V_\infty^2} \qquad\qquad C_{\mathcal{D}} = \frac{\mathcal{D}}{\rho_\infty V_\infty^3} = \frac{1}{\rho_\infty V_\infty^3} \int_0^{y_e} \tau \frac{\partial u}{\partial y}\, dy$$

The alternative normalization is with the boundary layer edge quantities, e.g.:

$$C_f = \frac{2\,\tau_w}{\rho_e u_e^2} \qquad\qquad C_{\mathcal{D}} = \frac{\mathcal{D}}{\rho_e u_e^3}$$

which are standard in boundary layer theory, but are inconvenient for plotting since C_f and $C_{\mathcal{D}}$ are then strongly singular at a stagnation point. The latter definitions are currently commented out in SUBR. INIT in `mplot.f`, and can be easily activated if needed.

2.7.3 Individual TS wave amplitude plots

MSES uses the "envelope method" for transition prediction, which is a simplified version of the e^n method. Instead of tracking the Tollmien-Schlichting (TS) wave amplitudes for many individual frequencies (as in the e^n method), the envelope method determines for each surface point the amplitude of whatever frequency happens to be most amplified at that point. This is a great simplification, but involves some approximations which have the greatest impact in flows where the shape parameter varies rapidly. To permit checking of the validity of the results of the envelope method, the `A/Ao vs x` (item 8) in the surface plots menu displays the amplitude of both the envelope as well as the individual frequencies which *would have* been predicted by the e^n method. Ideally, the envelope curve just touches the largest of all the individual frequency curves at each location. Any deviation from this ideal usually lies within the uncertainty in the specified critical amplification factor "n".

Activation of the TS wave amplitude plotting requires the appropriate definition of the OSOBJ variable in the `Makefile`. The disadvantage of this feature is that the necessary storage of the Orr-Sommerfeld database (read in once from `oshai.dat`) makes the **MPLOT** executable image quite large.

2.7.4 Graphics library

The graphics library employed by all plotting programs is Xplot11 (`libPlt.a`), which is aimed at driving X-terminals. It also allows a PostScript file `plot.ps` to be generated from the current visible screen plot at any time. Both color and black&white output is supported. More information on this plot library is in the `../plotlib/Doc` description file.

The screen graphics window initially is sized to cover roughly 70% of the screen. This is controlled by the variable SCRNFR which is hardwired to a default value of 0.7 in all the plotting programs, and can be changed if desired. It can also be made negative to put the default window in Portrait form, although most of the plotting programs are tailored for the Landscape format.

The X11 window normally has a reverse (black) background. Standard-video, with a white background, can be selected or unselected by setting the Unix shell variable as follows.

```
 % setenv XPLOT11_BACKGROUND white
 % unsetenv XPLOT11_BACKGROUND
```

This can also be done with a menu selection in the shell script `mrun`. Reverse-video has no effect on screen colors. It also has no effect on PostScript output at all, which is always black-on-white or color-on-white.

2.8 MEDP — specified-pressure editor

MEDP is an interactive menu-driven program used to modify data in `mdat.xxx`. **MSES /MEDP** is really an airfoil <u>redesign</u> system rather than a pure design system, and it requires that `mdat.xxx` contain a previously-converged case.

2.8.1 Data modification

When **MEDP** is executed with

```
 % medp xxx
```

it first tries to read `mdat.xxx`, and then displays its top level menu:

```
  1   Edit Cp distributions
  2   Re-read mdat.xxx
  3   Write   mdat.xxx

  4   Write airfoil coordinate file

  5   List flow conditions and forces
  6   Change MSES flow parameters
  7   Plot size change

  8   Write CPwall file
  9   Read  CPspec file
 10   Write Mwall file
 11   Read  Mspec file
 12   Read  new airfoil coordinate file
```

The purpose of most of these options is to insert and extract data from the solution. Option 6, for example, allows the modification of certain parameters not normally accessible through the usual input file `mses.xxx`.

The primary purpose of **MEDP** , however, is to insert specified pressure (or Mach) distributions into `mdat.xxx` for inverse calculations. Options 8,10 can be used to write out the current distributions. These can then be edited and read back in with Options 9,11, and subsequently saved to `mdat.xxx` with Option 3.

The much more user-friendly way to modify the C_p distributions is via Option 1, which puts the user into another menu:

```
   I nitialize CPspec <== CPwall for target element
   M odify Cp
   S lope-matching at input-Cp endpoints (toggle)
   D emark inverse segment
   P lot Cp vs s/smax
   G rid-plotting toggle
   V ector Cp plot
   B lowup
   R eset to original size
   H ardcopy current plot
   N ew target element
```

which is used to perform the specified-C_p modifications interactively via the cursor as described below.

2.8.2 Specified-C_p editing

The specified pressure array `CPspec` read in from `mdat.xxx` is originally zeroed out in **MSET** , so if this is a first inverse editing session, the user must select option `I` to initialize `CPspec` to the current wall pressure coefficient arrays `CPwall`. Option `M` will then allow the user to "edit" `CPspec` to his liking with the screen cursor. This can be done repeatedly if needed. At the completion of each `M` command, the cursor-specified C_p points are splined, and the piece is grafted into the existing `CPspec` distribution. The splined piece can have natural spline endpoints, or values and slopes which match the current `CPspec`. The latter case is the default, and can be toggled with the `S` command.

If a Mixed-Inverse case is to be run with **MSES** , then inverse target segment endpoints will need to be specified with option `D`. In the default (unmarked) case, the target segment endpoints are just downstream of the front stagnation point, and at the rear trailing edge point. For cases with fairly blunt leading edges, the left segment endpoint should not be placed too close to the stagnation point. At near-stagnation pressures, the surface deformation is very sensitive to small pressure changes. If a pressure slightly above stagnation is specified, then the calculation will fail, since then there is no possible solution. With successive Newton iterations, the surface will severely indent and then tangle hopelessly. In essence, modifying the geometry near a stagnation point via the surface pressure is ill-conditioned and should be avoided.

If a Modal-Inverse or optimization case is to be run, then the geometry modes and their endpoints must be specified in file `modes.xxx` (described in the Optimization section below).

If a screen cursor is not available, then top-level option 8 (or 10) can be used to dump the surface C_p's (or equivalent Mach numbers) into a formatted scratch file, which can then be edited manually, or preferably with the user's own software. This file can then be read back into **MEDP** with Option 9 or 11. The scratch file also contains the indices of the target element and the requested inverse segment endpoints. However the `CPspec` distribution is generated in **MEDP** , it must be written out with the `mdat.xxx` file so that **MSES** or **MSIS** can access the information. This is done with top-level Option 3.

It must be mentioned here that in MSES v 2.9 all inverse calculation operations are performed using the pressure p_i defined at the displacement streamline, which is what was done in all previous MSES versions. Hence, `CPwall` actually represents p_i and not p_w, and in viscous inverse cases one should not be surprised if `CPwall` for the two airfoil sides does not match up at the trailing edge, since the Kutta condition is imposed on p_w and not p_i.

3 Inverse Calculations

MSES or **MSIS** are configured for an inverse case by selection of the appropriate global variables and constraints. How the specified pressures are used, and how the airfoil is to be modified in the **MSES** inverse calculation, are selected via the variables ISMOVE and ISPRES in `mses.xxx`. The Mixed-Inverse option requires that the global variables (11) and (12) and the global constraints (11) and (12) be chosen in addition to the usual analysis-case variables and constraints. If desired, variables (13) and/or (14) can also be specified, with corresponding constraints (13),(14). The global variables (11). . . (14) are coefficients $A_1 \ldots A_4$ which multiply intentional"error" terms added to the specified wall pressure $p_{\rm spec}$. These error terms are necessary since it is not possible to specify an arbitrary pressure distribution and obtain a closed airfoil (Lighthill's theory). The final surface pressure after an inverse calculation is

$$p_i(s) \;=\; p_{\rm spec}(s) \;+\; A_1 f_1(s) \;+\; A_2 f_2(s) \;+\; A_3 f_3(s) \;+\; A_4 f_4(s)$$

where $f_n(s)$ are prescribed shape functions defined in SUBROUTINE INIT. The new degrees of freedom A_1 and A_2 allow the imposition of geometric closure constraints at each of the two inverse-segment endpoints:

$$\Delta n = 0$$

The optional degrees of freedom A_3 and A_4 allow imposition of pressure curvature at the two segment endpoints:

$$\frac{d^2 p_i}{ds^2} = \frac{d^2 p_{\rm spec}}{ds^2}$$

The latter produces a smoother geometry at the inverse segment endpoint(s), but results in a greater difference between `CPspec` and the final converged C_p. This can be useful in some cases, but is rarely required. The error terms added to $P_{\rm spec}$ will be minimized if modifications to `CPspec` are done so that the lift is approximately preserved.

3.1 Fixed α versus C_L

It is suggested that a fixed angle of attack (constraint (5)) be used with Mixed-Inverse cases when the inverse segment covers all or mostly all of an airfoil side. Since α and one of the added pressure DOFs have nearly the same effect on the surface C_p, leaving them both as free variables makes for a poorly-conditioned problem.

3.2 Separation in inverse cases

One important point to remember when performing viscous Mixed-Inverse cases is that there must not be any separation present within the target segment. Such a calculation is equivalent to imposing a pressure (or u_e) distribution on a separated boundary layer, which is physically ill-posed. The result will be an inability of the code to converge properly, and will likely produce a very irregular airfoil geometry in the separated region. A reasonably good solution to this problem is to temporarily set the Reynolds number REYNIN in `mses.xxx` to zero when converging the Mixed-Inverse case. This “freezes” the boundary layer displacement thickness and sidesteps the physical ill-posedness problem. After the inverse problem converges, the Reynolds number can be restored and the case immediately reconverged as a regular analysis case (this will retain the new modified airfoil geometry). This technique has been found to be very satisfactory in practice.

3.3 Modal-Inverse

The Modal-Inverse method restricts changes in the airfoil to be a sum of some specified set of geometry modes. If the modes are smooth, then the modified airfoil is guaranteed to be smooth as well. Modal-Inverse is quite robust, and is particularly useful in flows with shocks and/or separation, where the Mixed-Inverse method might produce and irregular airfoil shape or simply fail due to ill-posedness. On the other hand, Modal-Inverse will match `CPwall` to `CPspec` in only a least-squares sense, and is intended for changing the overall airfoil shape, rather than removing small geometric defects.

The Modal-Inverse option requires that any combination of the global variables (21-29) be chosen. For each global variable, its corresponding global constraint (41-49) must be specified. One can also use the “shortcut” variable (20) and constraint (40) in conjunction with a specified NMODN. The geometry mode shapes are set up in FUNCTION GFUN (in `gmodes.f`), and can be altered as desired. See the Optimization section below for further info on the mode shapes.

It must also be mentioned that the Modal-Inverse option does not allow a full linearization of all the governing equations, with the result that the convergence rate will be somewhat more sluggish than usual. Nevertheless, only a few more iterations relative to a usual analysis or Mixed-Inverse case will be required. On the other hand, a viscous Modal-Inverse case can typically handle limited separated regions within the target segment. Is is not necessary to temporarily "freeze" the boundary layers as with Mixed-Inverse.

3.4 Inverse-case convergence tolerance

It is rarely necessary to converge a Mixed-Inverse or a Modal-Inverse case down to the usual convergence tolerance used for analysis cases. If the iterations are halted before full convergence is reached, then the new airfoil shape is still quite usable — it might have been changed 95% of the way towards the "correct" geometry rather than the 99.999% obtainable with full convergence. This slight difference is irrelevant in actual iterative design applications, since the "correct" geometry rarely turns out to be exactly what is required.

3.5 Inverse Design Session

Below is a sample inverse design calculation sequence, starting from the seed case `xxx`. Program executions as well as option selections within **MEDP** are shown.

```
% mset xxx
% mses xxx                      (usual analysis run)
% medp xxx
    1 Edit Cp distributions
         I nitialize CPspec
         M odify Cp
         B lowup                (optional)
         M odify Cp             (repeat as needed)
         .
         .
         D emark inverse segment
         return
    3 Write mdat.xxx
    0 Quit
% edit mses.xxx                 (add 11,12 to input lines 1 and 2)
% mses xxx                      (inverse run)
% mplot xxx                     (optional)
% medp xxx
    1 Edit Cp distributions
         I nitialize CPspec     (optional)
         M odify Cp
         .
         .
         return
    3 Write mdat.xxx
    0 Quit
```

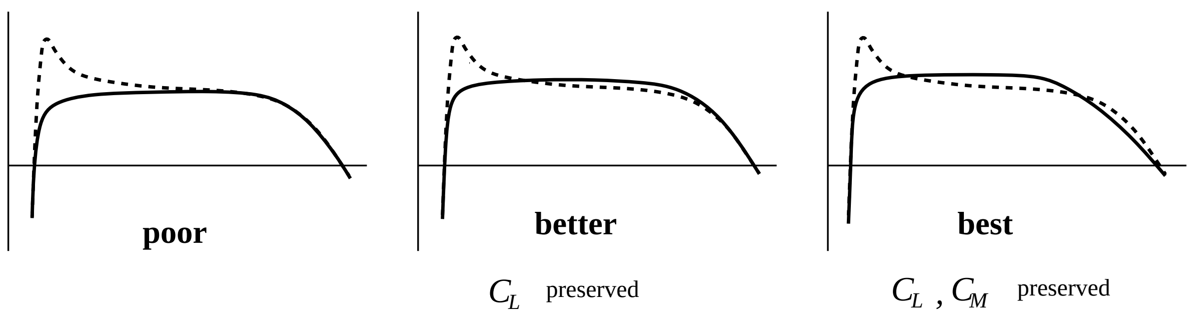


Figure 4: Poor and effective C_p modifications to remove spike (dashed line).

```
% mses xxx                          (inverse run)
% medp xxx
      4 Write airfoil coordinate file   (if satisfied with design)
```

The sequence above can be repeated as often as needed. The C_p plot in **MEDP** displays both the current surface pressures and the specified pressures. Any difference after convergence is due to the "error" terms which were added to attain closure at the segment endpoints.

The example above shows a Mixed-Inverse calculation. A Modal-Inverse calculation would be done by adding 21,22,. . . to input line 1 in `mses.xxx`, and adding 41,42,. . . to input line 2 instead of the 11,12 indicated.

MPLOT can of course be used anytime to examine the design in more detail. The final option 4 in **MEDP** saves the modified coordinates as a standard `blade.xxx` file.

3.6 Redesign rules of thumb

When the prescribed C_p distributions are edited in **MEDP** , it is useful to try to minimize the inevitable modifications caused by the modes $A_1 f_1(s)$, $A_2 f_2(s)$ . . . that **MSES** will add to enforce geometric closure. In general, the best type of C_p modification incurs minimal C_L and C_M changes, as shown in Figure 4. In the sketch labeled "poor", the new specified C_p (solid line) is likely to be substantially altered by the modes, likely causing geometry glitches at the blend points which will need further repair. In the sketch labeled "best", the mode modification is likely to be minimal.

This behavior is consistent with Lighthill's constraints, one of which states that the "average" surface speed, in some weighted-integral form, cannot be changed. The "poor" modification clearly tries to violate this constraint and will suffer a large mode correction. The other Lighthill constraints are moment-type constraints, and again the "poor" modification tries to violate this the most. It is effective to think of the C_p editing as a process where a fixed area under the curve is merely redistributed, preferably with some degree of left-right symmetry.

4 Optimization

Optimization capabilities are no longer part of MSES 2.9 itself, but are now implemented in the external interactive design/optimization driver **LINDOP** . To make use of **LINDOP** requires the specifying some number of geometry deformation modes as additional global unknowns (21-29), and specifying their corresponding fixing constraints (21-29). In addition, element position modes (31-39) and their fixing constraints (31-39) can also be specified. These translate/rotate the selected element(s) as rigid bodies. In either case, **MSES** sets all the geometry and position mode amplitudes to zero during a calculation, but it will still calculate the <u>sensitivities</u> of various quantities such as C_L, C_D, etc. to the mode displacements. These sensitivities are written out to the unformatted file `sensx.xxx`, which is then read in by **LINDOP** .

As mentioned earlier in the `mses.xxx` file description section, an alternative to specifying the individual geometry mode flags (21-29) is to specify the single flag (20), and choose the number of modes with NMODN. In the same manner, the (30) flag can be used in conjunction with NPOSN.

4.1 Geometry mode input file `modes.xxx`

All the geometry deformation modes selected in `mses.xxx` must be described in the file `modes.xxx`. This has the following format.

```
KDOF(1)  IMODE(1)  GWT(1)  ST1(1)  ST2(1)  IEL(1)
KDOF(2)  IMODE(2)  GWT(2)  ST1(2)  ST2(2)  IEL(2)
   .        .        .       .
   .        .        .       .
KDOF(N)  IMODE(N)  GWT(N)  ST1(N)  ST2(N)  IEL(N)
```

where,

a) KDOF ties the mode shape to one of the modal geometry unknowns chosen (DMODn = 21 . . . 20 + NMODN). KDOF = 1 ties it to DMOD1, KDOF = 2 ties it to DMOD2, etc. A mode can be composed of any number of disconnected pieces. Each piece of the mode is described on a separate line, each beginning the same KDOF value.

b) IMODE specifies the mode (or mode piece) shape. The shapes are implemented in FUNCTION GFUN or variants thereof (in `gmodes.f`). The particular shapes currently implemented are:

IMODE	mode shape
$n = 1 \ldots 19$:	$\sin(n\pi s/s_{\max})$
$n = 20$:	Local bump at leading edge, sized by local curvature
$n = 21 \ldots 40$:	$T_{n-20}(s/s_{\max})$
$n = 41 \ldots \infty$:	$\sin^a[\pi(s/s_{\max})^b]$
	$\sin^a[\pi(1 - s/s_{\max})^b]$

T_n are Chebyshev polynomials, modified to be zero at the mode piece endpoints $s/s_{\max} = 0, 1$. These are a good alternative to the simple sine waves ($n = 1 \ldots 19$), since they give more resolution at the endpoints. A good first choice for mode shapes is to use a reasonably large number of the Chebyshev modes (IMODE = 21, 22, . . .) over the entire upper and/or lower surface. This allows nearly arbitrary geometry variations. The mode shapes selected can be plotted in **LINDOP** .

c) GWT is the mode-piece scaling factor. If a mode consists of only one piece, GWT has no effect, as it merely rescales the mode amplitude variable. However, it is needed if the geometry mode is composed of two or more pieces, and each piece must be scaled differently. An example is a pure camber mode, where two identical shapes are placed on opposite sides of the airfoil, and GWT is specified as +1.0 and -1.0 for the two pieces (a positive mode displacement is taken as outward normal to the surface of the airfoil). GWT is also significant in that it will alter the convergence behavior (but not the final answer) of the steepest-descent optimization process. In this respect, it has been found advantageous to set the GWT factors for all the modes so that the mode derivatives with respect to ST are all comparable in magnitude.

d) ST1, ST2 are the mode endpoint locations on the airfoil. These are the normalized arc lengths s/s_{side} from the airfoil nose (not the stagnation point like with Mixed-Inverse!), to the trailing edge. For example, specifying ST1, ST2 = 0.0, 0.5 will result in the mode extending over approximately the front half of the airfoil. There is also the option to specify ST1 and ST2 as element x/c values. The necessary code is presently hibernating in SUBROUTINE GNSET (in `gnset.f`) as comments, and only needs to be enabled.

e) IEL specifies the target element on which the mode piece acts.

4.2 Element position mode input file `moves.xxx`

The `moves.xxx` file has the following format.

```
KDOF(1)  GWT(1)  XT(1)  YT(1)  IEL(1)
KDOF(2)  GWT(2)  XT(2)  YT(2)  IEL(2)
   .        .       .      .
   .        .       .      .
KDOF(N)  GWT(N)  XT(N)  YT(N)  IEL(N)
```

where,

a) KDOF ties the position mode to one of the element position variables chosen (DPOSn = 31 . . . 30 + NPOSN). KDOF = 1 ties it to DPOS1, KDOF = 2 ties it to DPOS2, etc. KDOF should not be repeated on different lines.

b) GWT specifies whether this is a translation or rotation mode.

GWT = 0	translation mode
GWT $\neq$ 0	rotation mode of GWT radians per unit mode change

c) XT, YT are either the displacement vector components (if GWT = 0), or the rotation center absolute coordinates (if GWT $\neq$ 0).

d) IEL specifies the target element

The mode amplitudes DMODn and DPOSn are read in from a file `params.xxx`, which is generated by **LINDOP** . This conveys the optimization changes to **MSES** for implementation. This file also contains the flow parameters `ALFAIN`, `CLIFIN`, `MACHIN`, `REYNIN`, which will overwrite the values in `mses.xxx`. A message is printed when this occurs to warn the user.

4.3 Multi-point optimization setup example

Problem statement:

Minimize the C_D of an airfoil weight-averaged over the two points

1. $C_L = 0.80$, $M = 0.75$, $Re = 5 \times 10^6$
2. $C_L = 1.00$, $M = 0.65$, $Re = 5 \times 10^6$

using four upper-surface perturbation modes, one lower-surface perturbation mode, and one camber-line perturbation mode. The objective function to be minimized is

$$\mathcal{F} \;\equiv\; 3C_{D_1} \;+\; 2C_{D_2}$$

Problem setup:

The two points will have the extensions `xxx_01` and `xxx_02`. Examples of the two input files which implement the above flow conditions and declare the mode shape variables are:

File `mses.xxx_01`:

```
3 4 5 7 20
3 4 6 7 20
0.75   0.80  0.0   | MACH   CL    ALFA
 3     2           | ISMOM  IFFBC
5.0E6  9.0         | RE     ACRIT
0.1    0.1         | XTRS   XTRP
0.97   1.0         | MCRIT  MUCON
1      1           | ISMOVE ISPRES
6      0           | NMODN  NPOSN
```

File `mses.xxx_02`:

```
3 4 5 7 20
3 4 6 7 20
0.65   1.00  0.0   | MACH   CL    ALFA
 3     2           | ISMOM  IFFBC
5.0E6  9.0         | RE     ACRIT
0.1    0.1         | XTRS   XTRP
0.97   1.0         | MCRIT  MUCON
1      1           | ISMOVE ISPRES
6      0           | NMODN  NPOSN
```

The airfoil geometry in the two state files `mdat.xxx_01` and `mdat.xxx_02` must of course always be the same. It is simplest to use **MSET** with `blade.xxx` to create one `mdat.xxx` file, and then copy it to `mdat.xxx_01` and `mdat.xxx_02`. To ensure that the airfoil shape remains the same during optimization, the `modes.xxx_nn` files must be identical. For example:

Files `modes.xxx_01` and `modes.xxx_02`:

```
1    1     1.0  0.0  1.0  1    | Camber-line  mode , DOF 21, mode shape 1
1    1    -1.0  0.0 -1.0  1    |   "
2    5     1.0  0.0  1.0  1    | Upper-surface mode, DOF 22, mode shape 5
3    6     1.0  0.0  1.0  1    | Upper-surface mode, DOF 23, mode shape 6
4    7     1.0  0.0  1.0  1    | Upper-surface mode, DOF 24, mode shape 7
5    8     1.0  0.0  1.0  1    | Upper-surface mode, DOF 25, mode shape 8
6    2     1.0  0.0 -1.0  1    | Lower-surface mode, DOF 26, mode shape 2

KDOF IMODE GWT  ST1  ST2  IEL
```

The mode weights of ±1.0 in the third column are appropriate if all the modes shapes as defined in FUNCTION GFUN have comparable derivatives. These weights can be effectively changed later by rescaling in **LINDOP** .

The optimization process is started by first converging the two `xxx_01` and `xxx_02` cases:

```
 % mses xxx_01
 % mses xxx_02
```

One optimization cycle then consists of executing **LINDOP** , and reconverging all cases with **MSES** again:

```
 % lindop xxx
 % mses xxx_01
 % mses xxx_02
```

This is repeated as long as deemed necessary. Of course, **MSIS** can be used instead of **MSES** if appropriate.

If many `xxx_nn` cases are involved, it is convenient to use the case-sequence shell script `mseq` to converge all of them. For example,

```
 % mseq mses xxx 1 10
```

is equivalent to

```
 % mses xxx_01
 % mses xxx_02
     .
     .
 % mses xxx_10
```

The isentropic variant `msis` can be specified in the first argument to `mseq`:

```
 % mseq msis xxx 1 10
```

The *LINDOP User's Guide* discusses the optimization procedures in much greater detail.

5 General hints

5.1 CPU times

Using **MSES** or its variants requires substantial computational resources, especially memory. A single solution will require anywhere between 2 Newton iterations for a subcritical inviscid case, to ~ 15 or 20 iterations for a separated viscous case. For a relatively large grid size, 280×50 say, each iteration might require about 20 seconds on a high-end workstation. This translates to an execution time somewhere between 30 seconds to 10 minutes. A more-typical (220×40) grid for a two-element airfoil will require about 10 seconds per iteration. As mentioned earlier, the subcritical code versions will require about 1/2 to 1/4 of these times. Also, fewer elements require fewer grid streamlines, which can drastically reduce computational time. In general, solution time scales as $I \times J^3$, where I is the number of quasi-normal lines, and J is the number of streamlines. Thus, streamwise resolution is relatively cheap, but normal resolution is expensive. Fortunately, **MSES** is extremely insensitive to normal resolution, while streamwise resolution may have a strong impact if small viscous features (e.g. separation bubbles) are present.

As mentioned previously, grid coarsening/refining can be very useful in reducing the overall CPU time. It is especially effective in transonic cases with propagating shocks, and with low Reynolds number cases with propagating separation bubbles.

On Unix machines with modest cache memory sizes, it *might* be beneficial to make the physical array dimensions IX,JX,NBX, (in `STATE.INC`) as small as possible. This will reduce the likelyhood of cashe misses during matrix solution and possibly give a significant CPU saving. Of course, reducing the array dimensions may have no effect on run times at all.

5.2 Sequence of solutions

When a <u>sequence</u> of solutions is needed, the CPU time per solution can drop substantially due to the quadratic convergence property of the Newton method used by **MSES** . Once a converged solution is obtained, convergence to a nearby solution after a small α or $M_\int$ (say) change is quite rapid, requiring only 3 or 4 additional iterations. **MPOLAR** takes full advantage of this property.

While the Newton solution method is very efficient for converging small parameter tweaks, it intensely dislikes large changes. Trying to reconverge a solution after a drastic parameter change, such as going from $\alpha = 0$ to $\alpha = 10$ degrees, is definitely not a good idea, especially if a viscous case is involved. Clearly, runs should be sequenced so that such large changes are avoided. **MPOLAR** should be used whenever possible, since in addition to making an efficient gradual sweep in α, it is able to squeeze one "free" Newton iteration out of stored residual information everytime α is changed.

5.3 Starting difficult viscous cases

Some viscous cases can be more effectively computed if the case is first partially converged as an inviscid case (REYNIN set to zero). This is especially true for transonic cases, or if the case to be converged corresponds to a significantly different lift than what was used by **MSET** to generate the initial streamline grid. Solid-wall (IFFBC=1) and free-jet (IFFBC=3) cases will benefit the most from this procedure, since **MSET** assumes an infinite flow (IFFBC=2) to generate the initial grid, which can be quite different from the final converged grid.

5.4 More MSES Hints

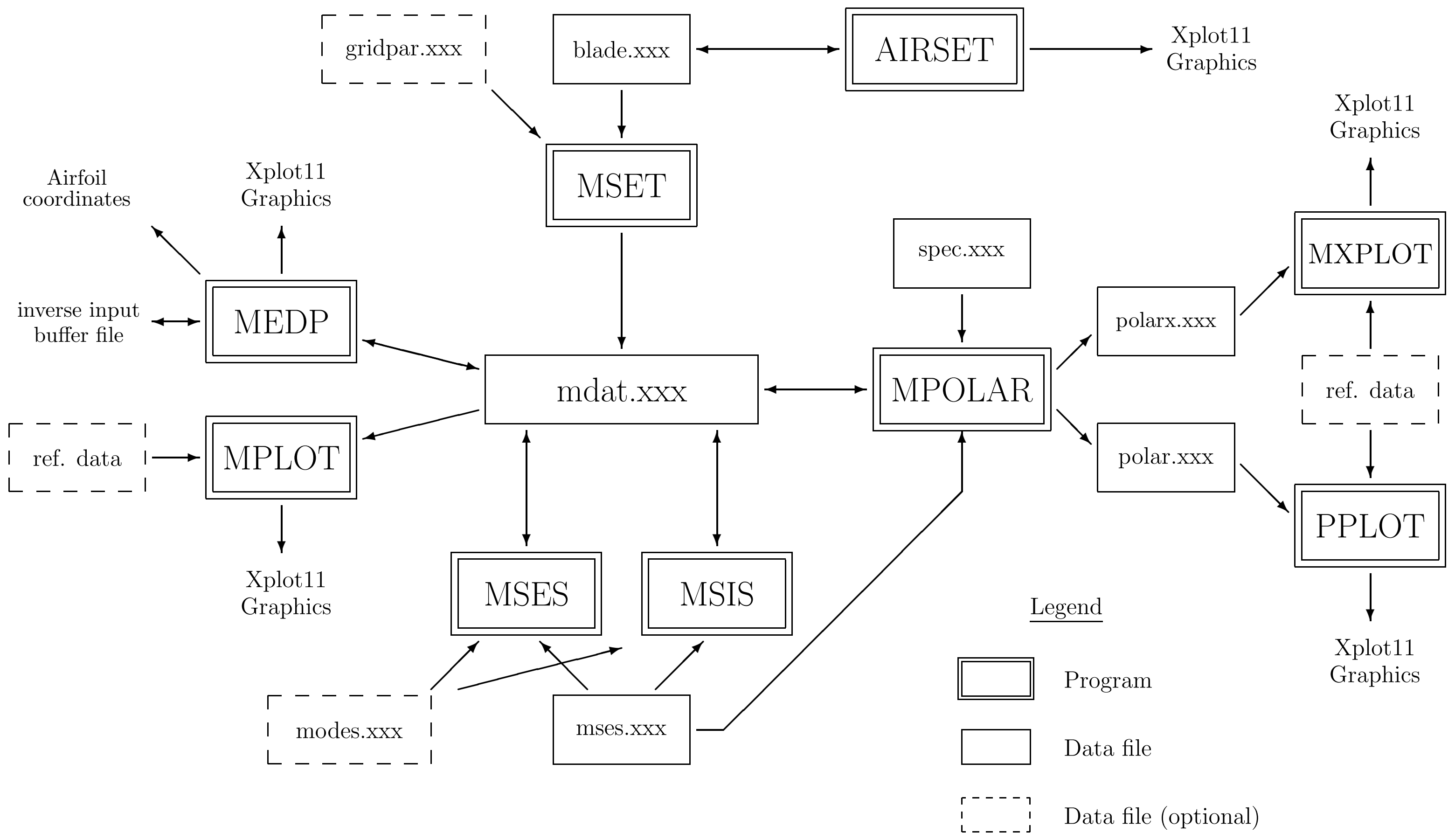
1. Any separation bubble must be well-resolved, especially on the suction side, since what goes on at the bubble has a strong impact on C_D and $C_{L\max}$. The default grid is usually adequate for most cases, but maybe not if the bubble is close to the leading edge and very small in streamwise extent. A NACA 0010, near $C_{L\max}$, for example, has a small but intense bubble near the leading edge which requires an unusually fine grid spacing there. Moderate Reynolds numbers (1-3 million, say) require the finest grid, since the bubbles are then still important, but very small. Streamwise grid spacing is "cheap", increasing the solution time only linearly, so it may be simplest to increase the grid point number parameter `N` in **MSET** to 150 or more. Inadequate bubble resolution results in a "ragged" or "scalloped" C_L vs C_D drag polar curve, so this is easy to spot.

2. If the initial grid was generated at an α which differs from the specified α by more than a few degrees, it is best to partially converge the case with **MSES** in the inviscid mode, until the specified α is reached. This is particularly true if free transition is used and large transition-point movement is expected. This is rarely necessary with fixed-transition cases.

3. If one of the elements has a cove region (where the flap is nested when not deployed), it is again best to first analyze the configuration at a representative angle of attack, and approximately determine the point of separation before the cove. Full convergence is not necessary here. This point can have a corner added (manually or in **AIRSET**). This will signal **MSET** to have the initial grid leave the surface here and thus simulate a massively separated region.

4. For a large α polar sweep, it is best to grid the configuration at an α near the middle of the range, and calculate the polar in two pieces — one going up in α, the other going down. It may be necessary to increase the local grid node concentration above and below the leading edge, so that as these points get dragged onto the leading edge, the resolution there remains reasonably good. An alternative approach is to generate two grids — one near each end of the α sweep. The polar calculation then proceeds in two pieces from each end to the middle.

MSES Roadmap


gridpar.xxx
blade.xxx
AIRSET
Xplot11 Graphics
MSET
Xplot11 Graphics
Airfoil coordinates
Xplot11 Graphics
spec.xxx
MXPLOT
inverse input buffer file
MEDP
polarx.xxx
mdat.xxx
MPOLAR
ref. data
ref. data
MPLOT
polar.xxx
PPLOT
Xplot11 Graphics
MSES
MSIS
Legend
Xplot11 Graphics
Program
modes.xxx
mses.xxx
Data file
Data file (optional)