NASA/TM-20220017582/REV1

CAPSULE_GRID User Guide

David A. Saunders
Analytical Mechanics Associates, Inc.
Aerothermodynamics Branch
NASA Ames Research Center

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA

June 2024 Update

December 2022

Table of Contents

CAPSULE_GRID User Guide

David Saunders, AMA, Inc./TSA Branch at NASA Ames Research Center

Software utility *CAPSULE_GRID* and several related procedures that automate grid generation for computational fluid dynamics calculations about axisymmetric atmospheric entry vehicles are described as a supplement to the documentation within the source codes. This application program and its ancillary software utilities are all in the public domain at the web site https://software.nasa.gov/software/category/all/arc/1/cfdtools with the exception of the 2D hyperbolic volume gridding step for which scripts driving either *GRIDGEN*[1] or *HYPGEN*[2] are available. *CAPSULE_GRID* was developed at NASA Ames Research Center in support of the *DPLR*[3] real gas flow solver. Both forebodies (only) and full bodies are treated, for axisymmetric 2D flow calculations or for 3D calculations at angle of attack. The expected application is to the smooth outer mold line of a hypersonic atmospheric entry vehicle, for which a single layer of point-matched structured grid blocks normally suffices. Control files for a variety of representative cases are included with illustrations of results, and some insights into the subtleties of curvature-based grid point distribution are provided. Gridding of a capsule on a sting is supported. Additional procedures for treating arbitrarily long 2D wakes, prompted by asteroid studies and relevant to spacecraft such as Stardust, are also documented. Most recently, treating an *asymmetric* forebody (including an off-center nose, with or without an aft body) has been incorporated as an extra step.

I. Introduction

Generatrices and structured surface and computational volume grids for most of the typical (and not-so-typical) axisymmetric spacecraft capsule geometries (forebody alone or full body) can be generated rapidly with the combination of steps outlined in this user guide, the first step being named *CAPSULE_GRID*. The forebody may also be *asymmetric*: see section X for this recent option. The automated technique for replacing singular nose/tail portions of the interim revolved/spoked 3D surface grid by morphing a circular quadrant grid was originally developed for a heat-shield shape optimization method that allowed an off-center nose. Another exception to the usual axisymmetric assumption is a simulated drag skirt implemented originally for early studies of the ADEPT[4] (folding umbrella-like) concept.

A variety of computational **sphere-cone** generatrices may be constructed analytically from just cone angle and radius inputs (with options to round sharp corners on the aft body) or by redistributing points along an input discretized generatrix, either in one piece or as a pair of fore and aft segments appropriately blended. (An input generatrix may also be preserved exactly.) Several modules in the reusable `capsule.f90` package underlie analytic construction of likely capsule forebodies, including biconic heat shields and Apollo-type spherical-sections (via a cone angle specified as 0°). Gridding of a capsule on a shock tunnel sting is another *CAPSULE_GRID* option, along with a “ripple” option that, with an extra step, models the skin over toroids of a HIAD concept (hypersonic inflatable aerodynamic decelerator).

Curvature-based surface gridding is accomplished with a set of Fortran 90 subroutines driven by the *CAPSULE_GRID* main program, while 2D hyperbolic volume gridding is performed with available *GRIDGEN*[1] or *HYPGEN*[2] scripts followed by a *COMPRESS2D* step that adjusts radial grid line point counts and splits the grid into fore and aft blocks. Normally, initial 3D volume grids are produced by rotating the compressed 2D volume grid through 180° then using *RADIAL_INTERP* to impose the 6-, 12-, or 16-patch surface grid (output by

CAPSULE_GRID) on the revolved, spoked surface. (*RADIAL_INTERP* et al. are available from the same web site as *CAPSULE_GRID*.) The asymmetric option requires 3D volume grid generation, and a full-body case writes a single-patch form of the spoked surface grid because *HYPGEN* cannot handle the multipatch surface grid.

A flow chart of most of the sequences of steps leading to initial structured volume grids appears in Fig. 1, followed by a complete control input description. An outline of the history of *CAPSULE_GRID* indicates how the software has evolved out of the need to automate surface gridding for shape optimization purposes, and the key techniques employed are described in a Methodology section. Representative inputs and outputs are shown as a set of configurations treated in past entry vehicle analyses, and these should serve as starting points for future smooth OML (outer mold line) applications.

II. Work Flow

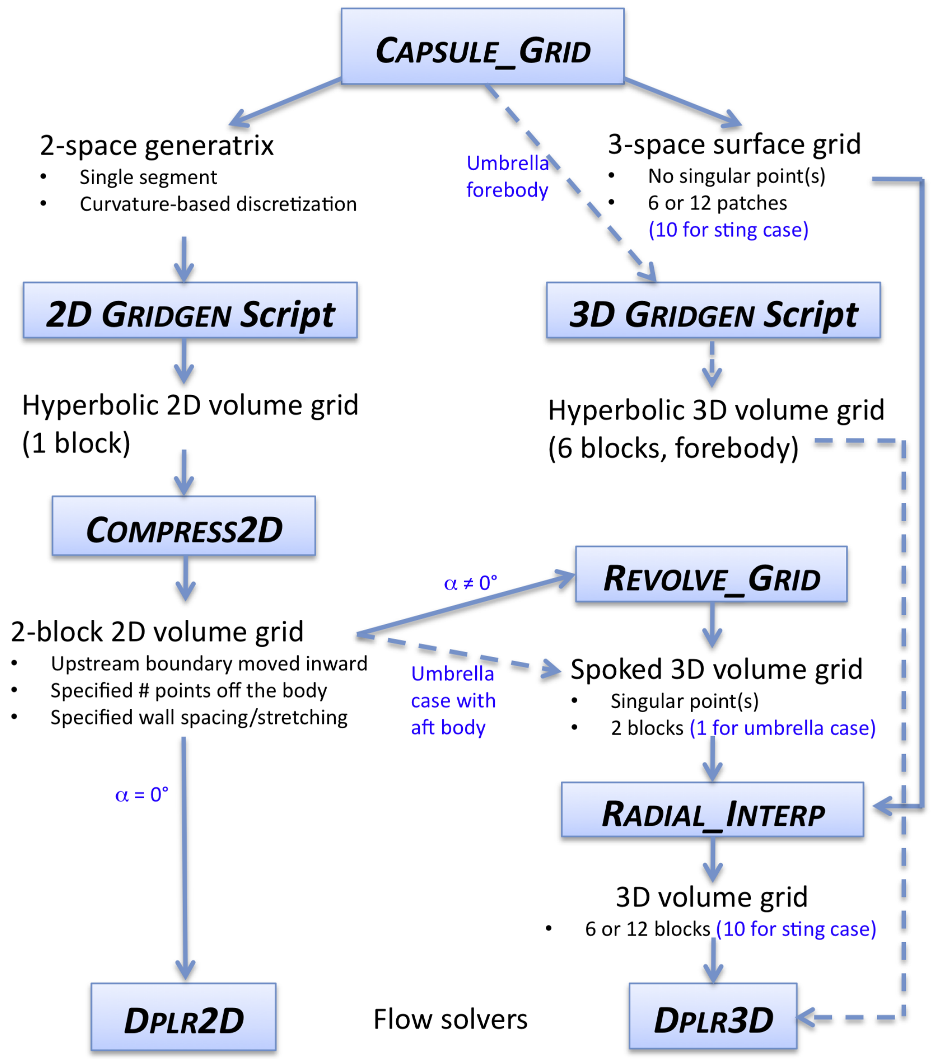


Figure 1: Utilities and scripts for automating axisymmetric capsule grid generation are applied in the indicated sequences. *HYPGEN* scripts may be used as alternatives to *GRIDGEN*. Additional procedures for arbitrarily long wakes are not shown. See the `/CAPSULE_GRID/Long-Wakes` directory and VIII (p. 40) for those. The asymmetric option cannot use a revolved 2D volume grid. A 3D *HYPGEN* script is used instead.

III. *CAPSULE_GRID* Inputs and Outputs

An essential ancillary input file named **`NosePatch61B.p3da`**[5] has been employed for all known uses of *CAPSULE_GRID*. This is an elliptically-smoothed 2D circular quadrant grid that is morphed twice and projected onto the nose surface (likewise for the aft body if there is one) to eliminate the singular point(s) produced by rotating the computed generatrix through 180°. A pair of Fortran namelists is used for the forebody and (optional) aft-body controls, in a single control file that is read as standard input. Namelists allow defaulting of control variables and are readily extended as further controls prove desirable. The inputs may appear in any order or be omitted altogether (implying use of relevant default values). Many of these inputs may not be used, depending on certain other inputs. For example, if `input_generatrix` is entered as a file name and not `'none'`, then inputs for defining the capsule analytically will be ignored. A complete example follows (for a 55° sphere-cone SPRITE configuration (Saturn PRobe Interior and aTmospheric Explorer) with aft body), including the variable descriptions with clarifications below the one-line entries. Plotted results appear in Figs. 2 and 3.

```
$FOREBODY_INPUTS
verbose = F,
aft_body_too = T,
asymmetric = F,
units_in_inches = F,
geometric_scale = 1.,
input_generatrix = 'none',
lower_generatrix = 'none',
spherecone = T,
numi_forebody = 201,
numj = 145,
x_nose = 0.,
r_nose = 0.,
radius_nose = 0.058166,
radius_base = 0.1778,
radius_shoulder = 0.014224,
radius_vertex = 99.,
half_cone_angle = 55.,
half_cone_angle_fore = 99.,
half_cone_angle_aft = 99.,
radius_cone_juncture = 99.,
skirt_angle = 15.,
skirt_length = 0.031214,
nose_patch_file_name = 'NosePatch61B.p3da',
output_generatrix = 'gen.dat',
surface_grid_file_name = 'surface_grid.g',
flat_blend_fore = T,
power_fore = 0.5,
ni_regrid_forebody = 33,
ripple_case = F,
ntoroids = 6,
peak_ripple = 0.05,
umbrella_case = F,
nedges = 8,
peak_deflection = 0.01,
rib_deflection = 0.,
frustum_radius = 0.,
resolve_the_ridges = T,
rib_thickness = 0.01,
round_the_ridges = T,
$END
```

```
$AFT_BODY_INPUTS
sting_case = F,
ncones = 3,
flat_blend_aft = T,
r_aft = 0., 0.160913, 0.0508, 0.0508,
cone_angle_aft = 15., 90., 0.,
rounding_mode = 2,
ds_round = 0., 0.00635, 0.00635, 0.00635,
ni_round = 0, 21, 21, 21,
power_aft = 0.5,
ng_split = 0,
nblend_fore_aft = 8,
numi_aft_body = 201,
ni_regrid_aft_body = 21,
x_parachute_cone = -99.,
i0_umbrella = 0,
i1_umbrella = 0,
i2_umbrella = 0,
sting_stretch_multiplier = 2.,
$END
```

```
!     CAPSULE_GRID control input descriptions, mostly from the main program but
!     with occasional additional tips; see further tips below these one-liners:
!
!     $FOREBODY_INPUTS
!     verbose = F,                 ! T invokes log file output from CURVDIS
!     aft_body_too = T,            ! Option to suppress the aft body
!     asymmetric = F,              ! T requires an input lower_generatrix
!     units_in_inches = F,         ! Option suited to arc-jets in the US of A
!     geometric_scale = 1.,        ! Option to scale the indicated geometry
!     input_generatrix = '...',    ! Full generatrix file; 'none' => analytic
!     lower_generatrix = '...',    ! If asymmetric = T, forward portion is used
!     spherecone = T,              ! T => half_cone_angle >= 0, else biconic
!                                  ! T includes CEV-type (see half_cone_angle)
!     numi_forebody = 201,         ! # grid points on output forebody generatrix
!     numj = 121,                  ! # spokes in body of revolution (4m + 1)
!     x_nose = 0.,                 ! Nose coordinate
!     r_nose = 0.,                 ! Nose coordinate; must be 0. if aft_body_too
!     radius_nose = 0.0889,        ! Radius of nose circular segment
!     radius_base = 0.1778,        ! Radius at maximum diameter
!     radius_shoulder = 0.00889,   ! Radius of shoulder circular segment
!     radius_vertex = 0.           ! Biconic option to round the cone juncture
!     half_cone_angle = 55.,       ! 0. => spherical section (Apollo/Orion-type)
!     half_cone_angle_fore = ..., ! Forward cone angle, if biconic
!     half_cone_angle_aft = ...,   ! Aft cone angle, if biconic
!     radius_cone_juncture = ..., ! Radius from Ox where cones meet, if biconic
!     skirt_angle = 35.,           ! Semi-angle if cone segment follows shoulder
!     skirt_length = 0.001,        ! Length in x, normally >= 0.; < 0. may be OK
!     nose_patch_file_name = '...' ! Grid quadrant used to remove singular point
!     output_generatrix = '...',   ! ... corresponding to the 3D surface grid
!     surface_grid_file_name = '.. ! Output 3D surface grid/no singular point(s)
!     flat_blend_fore = T,         ! T ==> extra blending between high/low curv.
!     power_fore = 0.5,            ! Initial exponent in curvature-based redist.
!     ni_regrid_forebody = 33,     ! 1:ni_regrid, 1:numj nose region is replaced
!     ripple_case = ...,           ! T => simulate inflatable toroid concept
!     ntoroids = ...,              ! # toroids modeled along cone segment
!     peak_ripple = ...,           ! Peak deflection between toroids
!     umbrella_case = ...,         ! T => polygonal forebody/optional deflectns.
!     nedges = ...,                ! # full polygon edges if umbrella_case
!     peak_deflection = ...,       ! > 0. => impose catenary-type deflections
!     rib_deflection = ...,        ! Option to bend ribs (+/- => concave/convex)
!     frustum_radius = ...,        ! 0. => start fabric/rib deflns. @ tngcy. pt.
!     resolve_the_ridges = ...,    ! T => adjust spacing either side of ribs
!     rib_thickness = ...,         ! Off-rib spacing to impose (<= input value)
!     round_the_ridges = ...,      ! T => loose local cubic spline rounding
!     $END
!

!     $AFT_BODY_INPUTS
!     sting_case = F,              ! Option to suppress closure of the aft body
!     ncones = 2,                  ! ncones >= 1
!     flat_blend_aft = T,          ! T ==> extra blending between high/low curv.
!     r_aft = 99., 0.12, 0.06,     ! Radii from fore/aft juncture (1:ncones+1)
!                                  ! r_aft(1) automatically matches forebody
!     cone_angle_aft = 40., 65.,   ! Half cone half angles (1:ncones); 0, 90 OK
!                                  ! > 90 and > 180 can also be OK
!     rounding_mode = 2            ! 1 => tangent length; 2 => radius; 0 = none
!     ds_round = 0., 0.003, 0.003, ! Tangent lengths or rounding radii >= 0.;
!                                  ! only ds_round(2:ncones+1) are looked at;
!                                  ! ds_round(ncones) < 0 flags special SRC case
!     ni_round = 0, 17, 17,        ! # uniform rounding circle arc points >= 0
!     power_aft = 0.5,             ! Initial exponent in curvature-based redist.
!     numi_aft_body = 201,         ! # grid points on aft body
!     sting_stretch_multiplier = 2., ! Applied to interim end-of-sting spacing
!     ng_split = 0,                ! Full input gen. index (fore/aft redist.)
!     nblend_fore_aft = 8,         ! # aft pts. used to match fore/aft spacing
!     ni_regrid_aft_body = 0,      ! Automated if analytic (tail cap index)
!     x_parachute_cone = ...,      ! > 0. means split patches 7-10 at this x
!     i0_umbrella = 0,             ! For fat aft body cases (extra morphing)
!     i1_umbrella = 0,             ! Inboard index defining aft spoke morph
!     i2_umbrella = 0,             ! Outboard index near shoulder likewise
!     $END
!
!  Clarification of Input Generatrix Option:
!
!     Since an input generatrix may or may not represent the desired point
!     distribution in the output surface grid, the work-around is as follows:
!
!     If the number of points found in the input generatrix matches the spec-
!     ified grid point count, numi_forebody [+ numi_aft_body - 1], then the
!     input points are used without change.  Otherwise, the curvature-based
!     redistribution applied to the analytic case is applied likewise, and
!     the number of grid points can be adjusted to suit the intended flow
!     calculation.
!
!  Clarification of Curvature-Based Controls:
!
!     Fore and aft body generatrices are now redistributed separately for all
!     cases except a full-body input generatrix case where the output number of
!     points,  numi_forebody + numi_aft_body - 1, matches the input number.
!     (Originally, a full body input generatrix was treated as a single curve.)
!     Curvature-based redistribution may be controlled with exponent inputs
!     power_fore and power_aft, which default to 0.5 and should be in the range
!     [0.1, 0.9], with 0.9 maximizing the curvature effect.  If the iterative
!     solution does not converge, the exponent is lowered by 0.1 until it does.
!
!  "Umbrella" Forebodies:  Two Types of Deflection
!
```

```
!     Originally, faceted outer forebodies were constructed with straight-line,
!     undeflected ribs matching the cone portion of the underlying sphere-cone
!     generatrix, and any deflections were confined to the panels between the
!     ribs.  The "peak_deflection" input controls these catenary-type surface
!     deflections in both surface directions.  Now, a second rib_deflection
!     input allows the ribs to be bent in either direction (negative input for
!     a convex/outwards rib bend).  The initial discretization of the generatrix
!     is bent with a catenary-type deflection either inwards or outwards.
!     Blending at the circular shoulder is an awkwardness treated as follows.
!     The outer end slope of the nominal catenary is used to locate the shoulder
!     circle point with matching slope.  Then the nominal catenary is morphed
!     via an arc-length-based technique to move its outer end from the tangent
!     point of the straight cone line to that matching-slope circle point.  The
!     resulting blend is not perfectly tangential, but should suffice for small
!     deflections.
!
!  Definition of Frustum Radius:
!
!     Originally, "umbrella" cases assumed that the ribs started at the sphere-
!     cone tangency point.  Now, ribs may be shorter than that, to model an
!     effectively larger nose cap that includes part of the cone.  The control
!     input "frustum_radius" defines the start of the ribs where any deflections
!     may be imposed out to the shoulder tangency location.  Defaulting this
!     frustum radius to zero leads to use of the original definition, namely the
!     sphere-cone tangency point.  To be precise, any rib bending will start at
!     the first point of the eventual (redistributed) generatrix that is at or
!     nearest to (but downstream of) the location specified by frustum_radius.
!
!  Clarification of Aft Body ncones and Collapsed Aft Ends:
!
!     To avoid confusion, ncones is expected to be 1 for a single frustrum,
!     2 for a typical biconic aft body, and so on.  Normally, these cases have
!     a finite vertical end segment in the generatrix, where a half cone angle
!     of 90 degrees is implied and ncones+1 radii > 0 should also be entered.
!     Internally, r_aft(ncones+2) = 0 will be assigned for arc length calcs.
!
!     Conversely, if the aft end collapses to a point, r_aft(ncones+1) = 0.
!
!  Aft Body Vertex Rounding Options (Analytic Cases Only):
!
!     [Note that forebody rounding is limited to the one vertex of a biconic,
!     and is handled in the capsules.f90 forebody package via the radius_vertex
!     parameter.  Aft bodies are more general and any rounding of vertices is
!     handled in this program.]
!
!     rounding_mode = 0 turns off any rounding of [aft body] vertices;
!                   = 1 means input ds_round values > 0 are treated as tangent
!                       lengths from the vertex to the underlying circle;
!                   = 2 means input ds_round values > 0 are treated as radii
!                       of the rounding circle; ALSO: ds_round < 0 means a
!                       special case of rounding for SRC-type capsules with a
!                       quarter-circle quadrant for the aft-most segment.
!
!     If ds_round > 0 is specified for a vertex, a circular arc with ni_round
!     uniform points from tangent point to tangent point is inserted.  After
!     all insertions, the new geometry definition is redistributed to the
!     original numi_aft_body points using curvature-based techniques.
!     A collapsed aft body may also be rounded similarly: it is treated just as
!     for other vertices, but only half the circular points are inserted.
```

```
!
!     Note that ds_round(1) and ni_round(1) are ignored (no rounding at the
!     forebody/aft body juncture), but zeros should be entered as place holders.
!
!  Special SRC (Sample Return Capsule)-type Option (Full Quadrant Aft Body):
!
!     Use of ds_round(ncones) < 0. is the flag for a special case where the aft
!     body ends with a full quadrant (or a little less) that leaves no room for
!     normal rounding at the second last vertex.  A vertical segment is also
!     expected to precede the last vertex to be rounded.  Where that meets the
!     large aft circle has to be rounded after the normal rounding, because it
!     involves the vertical segment and what is now a large circular segment,
!     not a straight segment that would remain after less severe rounding of
!     the last vertex.  (Hard to explain!)  If the SRC-type aft body is not a
!     full quadrant, some geometry and trigonometry is required to determine
!     the last two r_aft inputs and the last cone angle implied by rounding
!     of the last vertex with the large radius of the less-than-full quadrant:
!
!        Rb = ds_round(ncones+1)  ! Large aft body radius of near-quadrant
!        xe = x_aft(ncones+2)     ! End of body, presumably known a priori
!        xo = xe - Rb             ! Center of large aft circle
!        xv = x_aft(ncones)       ! Vertical segment abscissa
!        tc = arcsin ((xv-xo)/Rb) ! The angle needed for cone_angle_aft(ncones)
!        ri = Rb*cos(tc)          ! Intersection radius needed for r_aft(ncones)
!
!  Handling of Aft Body 0, 180 and 90 Degree Half Cone Angles:
!
!     Note that if cone_angle_aft(j) = 0 or 180, the analytic specification is
!     not well defined.  The work-around is to enter the length of the step (in
!     the x direction) via r_aft(j+1); this will be reset to r_aft(j) internally
!     by the program.  For the 180 degree case, make r_aft(j+1) negative.
!
!     An interior angle of 90 degrees is also permitted without special handling
!     other than avoiding the infinite tangent internally.
!
!  Skirt Angle = Aft Body Half Cone Angle > 90 With Skirt Length < 0 (SRC-type):
!
!     The SRC-type configuration also requires an aft body cone angle > 90, in
!     combination with an equal skirt angle and a negative skirt length.
!
!  Aft Body Angle > 180 For Open SPRITE/ADEPT-Type With Flexible TPS:
!
!     The forebody/aft body juncture may need to be placed aft of the true
!     forebody shoulder via choice of skirt_length > 0 with skirt_angle = 15
!     say), combined with initial cone angles of 15 (say), 90, and 195 (say).
!     Looking at a working example is recommended.
!
!  Aft Body With "Umbrella" Forebody:
!
!     Default i1_umbrella and i2_umbrella = 0 (or umbrella_case = F) initially.
!     This suppresses the option for morphing an ADEPT-type aft body to an
!     umbrella-type forebody.
!     Pick indices on the resulting axisymmetric aft body spokes that make
!     sense for arc-length-based morphing (defining the essentially straight
!     line portions).  Outboard of that, the shoulder is simply shifted.
!
!     N.B.: These indices should be found from patch 7 of the interim surface
!     grid, NOT from the single-patch spoked_surface_aft.g file.
!     Also: for a fat aft body, if i1_umbrella on the aft body corresponds to
!     a radius that is greater than radius_frustum (where the forebody faceting
!     starts), use 0 < i0_umbrella < i1 to control additional morphing between
!     i0 and i1 because of the aft-body x-shifts from i1:i2 that maintain the
!     fabric thickness.  (Ideally these shifts go to zero at i1, but may not
!     for a fat payload housing.)  Default:  i0_umbrella = i1_umbrella - 20
!
!  Non-Analytic Full Body Case:
!
!     The inputs for numi_forebody and numi_aft_body should allow for a common
!     point where the two portions meet.  E.g., if 401 points are read in the
!     input generatrix, then 201 and 201 are plausible fore- and aft inputs.
!     See also "Clarification of Input Generatrix Option" above concerning the
!     sum of numi_forebody + numi_aft_body - 1.
!
!     If curvature-based redistribution is indicated, this is now done in two
!     parts.  The input geometry is split at the point of maximum radius, but
!     this can be overridden by entering ng_split > 0.  Keep in mind that the
!     split should be at or slightly aft of the shoulder max. diameter, for BC
!     and fore/aft block plotting purposes.
```

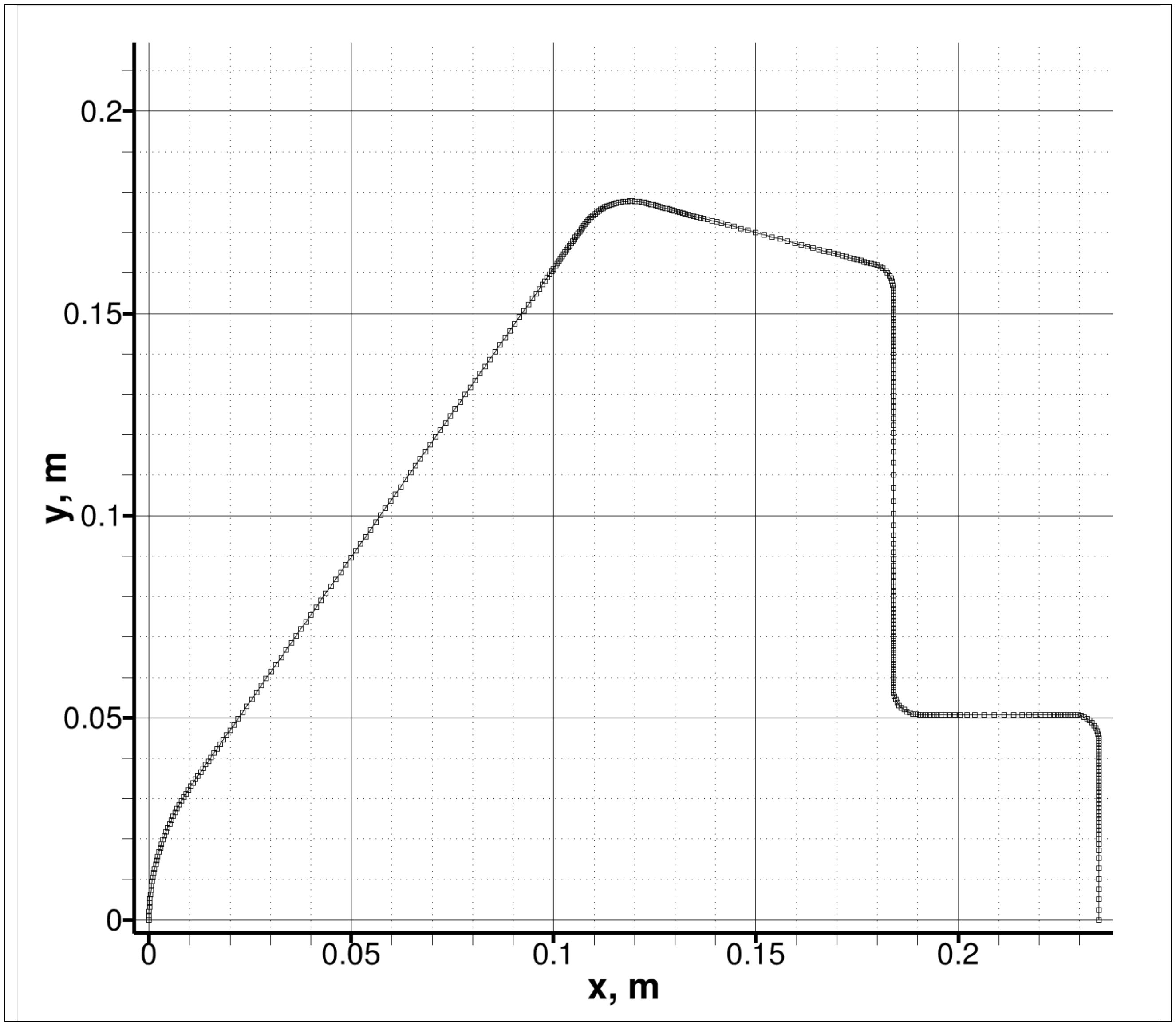


Figure 2: Computational generatrix produced by the above control file, for a 55° sphere-cone SPRITE configuration with a closed aft body. Note that *CAPSULE_GRID* forces uniform spacing towards the aft stagnation point because stretching towards the symmetry axis appears to affect *DPLR* results.

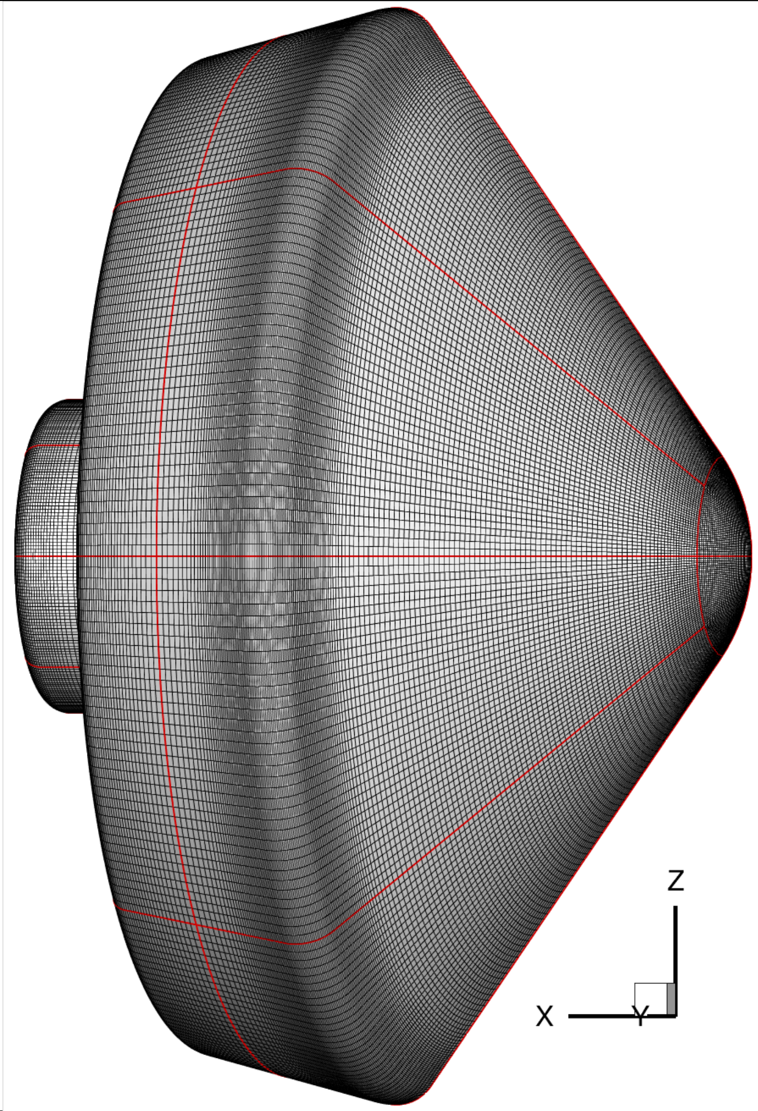


Figure 3: Computational surface grid produced by the above control file, for a 55° sphere-cone SPRITE configuration with a closed aft body.

IV. Background/Software Evolution

CAPSULE_GRID and its ancillary procedures have evolved from the *HEAT_SHIELD*, *FOREBODY_REGRID*, and *AFT_BODY* programs developed in the TSA (Aerothermodynamics) Branch at NASA ARC. *HEAT_SHIELD* performed constrained forebody shape optimization using modified Newtonian aerodynamics. It was initially adapted in 2000 by Dr. James Reuther for forebodies from portions of the aerodynamic shape optimization software he and the present author had worked on through the 1990s. (See, for example, https://www.researchgate.net/publication/24293150_Aerodynamic_Shape_Optimization_of_Complex_Aircraft_Configurations_via_an_Adjoint_Formulation.)

HEAT_SHIELD used curvature-based techniques and "sine bump" perturbing shape functions to construct optimized shapes that were, by design, not axisymmetric. The apex was permitted to move off-center, and the defining ribs or spokes were treated, in polar coordinates, much as wings with defining sections were treated in the earlier work. Asymmetry, however, has long proved of marginal interest to the reentry vehicle community, and an axisymmetric option (with a single generatrix to be optimized) was eventually incorporated in *HEAT_SHIELD*. Even more belatedly, treating the traditional analytic shapes (sphere-cones or biconics) was facilitated by constructing the generatrix as opposed to reading a defining section. Zero optimization iterations were commonly performed just to produce surface grids for the familiar geometries.

For CFD purposes, the resulting spoked surface grids (axisymmetric or not) had to be reworked to eliminate the singular point at the apex. Program *FOREBODY_REGRID* was written in 2008 to substitute circular quadrant nose patches for a specified forward portion of the spoked grid, using surface morphing and projection techniques. This early automation produced a 6-patch forebody topology that has served well for numerous reentry studies. In 2011, the program was extended to model forebodies with ribs (folded during launch) and flexible thermal protection on the conical portion of the underlying sphere-cones—the so-called "umbrella" configurations. Later in 2011, program *AFT_BODY* was first implemented to fill an obvious void. Any number of aft cone angles could be specified with associated radii and optional rounding of any of the vertices, enabling generation of quite a variety of axisymmetric shapes. Provision was made for blending with *FOREBODY_REGRID* results, but merging the fore- and aft body capabilities was clearly called for to reduce the number of steps in the automation, and the resulting *CAPSULE_GRID* has continued to evolve, with satisfactory handling of non-analytic sharp corners and transitions from zero curvature to high curvature among the challenges.

The chart above in Fig. 1 summarizes the axisymmetric volume gridding procedures that depend on the intended angles of attack and whether an aft body is present or not.

Most recently, the long-held assumption of axisymmetry has been generalized with an option to define a forebody with upper and lower generatrices that are blended from crown to keel to meet a symmetric aft body if one is present, on an Ox-centered circle. If `asymmetric = T`, the volume gridding requires a 3D script for *HYPGEN*.

Further highlights and user tips

As already indicated, forebodies are limited to sphere-cones, Apollo-type spherical sections, and biconics with optional rounding of the vertex where the biconic segments meet. Sphere-cones may be morphed into faceted "umbrella" ADEPT[4] configurations with optional deflections to model flexible TPS. Such *non-axisymmetric* cases affect the volume gridding steps, as will be explained below.

Joining a fore- and aft body is controlled by the forebody **skirt length** (normally an **x** distance, not an arc length distance along the forebody generatrix unless **skirt_angle** = 90°). The aft body radius (or diameter) at the juncture (⇔ first aft cone angle) is automatically derived from the last radius of the forebody. The grid spacing across the join is also blended automatically—see **nblend_fore_aft**.

SRCs (Sample Return Capsules) may need to invoke a specialized rounding option if the circular-section payload bay leaves no room for normal rounding of the second-last vertex in the generatrix. Enter a negative rounding radius for this workaround.

A **2-D volume grid** is generated hyperbolically with a *GRIDGEN* glf file (originally) or a *HYPGEN* script. This grid may then be compressed upstream for better proximity to the anticipated bow shock. *COMPRESS2D* also splits a full body case into two blocks for BC purposes, and options are provided for adjusting the wall spacing, the type of stretching, and the number of points in the off-wall direction. Recommendation: *Overdoing the compression step can be counterproductive.* A compressed grid is still not well aligned with the bow shock. This can give the flow solver convergence trouble because of strong shock effects. An initial solution on a larger upstream grid converges more readily, and the first alignment from that will be preferable to a misaligned solution on an overly-compressed grid, assuming one can be obtained. Suggestion: compress the upstream hyperbolic boundary to a high fraction of its initial distance from the nose. Actually, since any compression may impair the hyperbolic gridding result near parts of the surface, leaving the forward outer boundary where it is may actually be the best choice.

Normally, a 3-D volume grid is generated by rotating the (possibly compressed) 2-D grid through 180° (*REVOLVE_GRID*), thanks to the axisymmetry, then the resulting singular line(s) along the x-axis can be eliminated with a *RADIAL_INTERP* step, which efficiently imposes the topology of the multi-patch surface grid output by *CAPSULE_GRID* at all *k*-planes of the preliminary volume grid (thanks to the single layer of blocks), giving a 6-, 12-, or 16-block volume grid (or 10 blocks for **capsule-on-a-sting** cases, for which closure of the aft body is suppressed).

An **ADEPT** (**umbrella**-like) forebody needs a **3D *GRIDGEN* hyperbolic volume grid** script since it is not axisymmetric. This script for the 6-patch surface topology was developed in the *HEAT_SHIELD*/*FOREBODY_REGRID* era when asymmetry was commonplace. An analogous ***HYPGEN*** script had not been called for at the time of writing.

An **aft body** may also be appended to an **umbrella** forebody. Such a faceted, concave aft-body surface grid is produced by morphing the preliminary axisymmetric aft surface grid. Further, the associated full-body generatrix is used for an initial 2-D hyperbolic grid which can then be morphed to a 3-D full body volume grid as a special option in the *COMPRESS2D* step (no need for *REVOLVE_GRID* in this case). See pp. 22-28 for more on these deployable ADEPT configurations, including use of the control inputs **i1_umbrella**, **i2_umbrella**, and **i0_umbrella**. See also Lessons Learned on pp. 35-39.

V. Detailed Procedures

1. Edit a *CAPSULE_GRID* control file to suit the axisymmetric geometry desired. This file contains **two namelists** (for the forebody and an optional aft body) followed by a helpful legend extracted from the source code. It may have any name. A sample can be seen on pp. 6-11:

   ```
   [path]/cfdtools/CAPSULE_GRID/capsule.inp
   ```

   Also: **`ln -s [path]/cfdtools/CAPSULE_GRID/NosePatch61B.p3da .`**

2. Compute a **generatrix** (2-space, with z = 0. in column 3 for *GRIDGEN* purposes) and the associated 180° **3-space surface grid** by running *CAPSULE_GRID* like this:

   ```
   capsule_grid < capsule.inp > capsule.log
   ```

   - Use **`aft_body_too = T`** to include an aft body, and **`sting_case = T`** to model a capsule supported by a sting. Read the Lessons Learned below first.
   - Use **`umbrella_case = T`** to model a faceted/ribbed flexible-TPS case, which may or may not include an aft body. See pp. 22-25 for further details.
   - **Special Aft-Body Input Cases:** Note that aft-body cone angles of **0°** or **180°** do not allow the next aft-body radius input to define the extent of that cone segment. The workaround, for cone angle *j*, is to enter the *x* length of the segment via aft radius *j+1*; this will be reset to **`r_aft`** (*j*) internally. Also, if the cone angle is 180°, the input "radius" should be negative, since it represents an upstream step along the generatrix, parallel to the *x* axis.
   - The resulting generatrix has the specified number of points:

     ```
     numi_forebody [+ numi_aft_body - 1]
     ```
   - The fore- and aft-body portions are generated independently, each with optional curvature-based distributions, except that the initial cell size of an aft body matches the size of the last cell of the forebody if possible. [Note that curvatures are computed from normalized forms of the geometry so that the grid point distributions obtained are independent of the scale or units.]
   - If **`input_generatrix`** ≠ **`'none'`**, the indicated generatrix is read and turned into one with the specified number of points (curvature-based spacing) along with the associated 180° 3-space surface grid. Any analytic specifications are ignored in this case. [See page 24 for more on arbitrary generatrices.]
   - The 3-space surface grid is initiated as a spoked surface of revolution then regridded in the nose region with a pair of circular quadrant patches to remove the singular point. A closed aft body surface is regridded likewise The **ni_regrid** inputs are indices from the initial singular point, fore and aft, and should be adjusted for smooth blending of the grid spacing at the new boundaries.
   - The resulting surface grid has 6 patches (forebody only) or another 6 patches if an aft body is present, giving 12 patches, or 16 patches if **`x_parachute_cone`** > 0., or 6 + 4 = 10 patches for a capsule on a sting, which does not have a closed aft surface).
   - Note that the number of grid "spokes" specified (**`numj`**) must be of the form **`numj = 4m + 1`** because of this regridding.
   - Plot the 2- and 3-space surface grids and adjust the namelist inputs if necessary until the desired results are obtained. [Some aft bodies may need to be iterated on with the **`r_aft`** (:) radii inputs in order to match target x coordinates precisely, for instance.]

- The aft body control **ni_regrid_aft_body** can normally be defaulted by entering 0, but it may need to be overridden to achieve a desired (odd) point count in the main aft-body patches after the regridding. Blending of the spacing is insensitive to this.
- Sample axisymmetric surface grids are illustrated next before the volume gridding steps are outlined. Some non-axisymmetric grids may also be handled differently (see "umbrella" cases on pages 24-28). A capsule with sting is also shown on page 28.

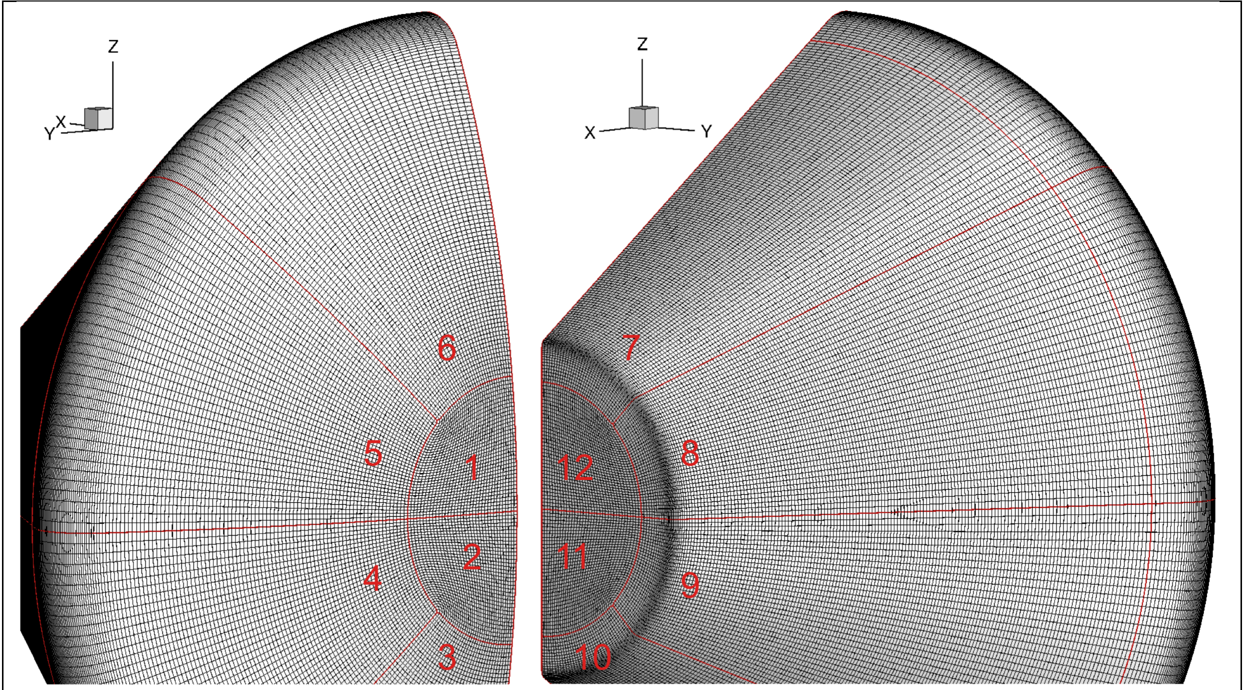


Figure 4: Representative 12-patch full-body surface grid (for the Orion/MPCV smooth OML with the default name surface_grid.g) produced by *CAPSULE_GRID* showing the same 6-patch topology on each of the fore- and aft bodies and the patch numbering. The singular points fore and aft have been eliminated from the underlying simple spoked 3-D grids via transforming the reference two-space circular quadrant patch (NosePatch61B.p3da) and projecting it on to the smooth outer mold line. Note curvature-based grid point distribution in the streamwise surface direction and uniform spacing in the azimuthal direction.

The essential inputs of the control file used to produce the above Orion result are shown below. Inputs not shown have been defaulted. The dimensions of the resulting twelve surface patches are also shown.

```
$FOREBODY_INPUTS                   $AFT_BODY_INPUTS              12
aft_body_too = T,                  ncones = 1,                     37   37    1
numi_forebody = 161,               r_aft = 0., 0.91168595,         37   37    1
numj = 145,                        cone_angle_aft = 32.5,          37  129    1
radius_nose = 6.03504,             rounding_mode = 2,              37  129    1
radius_base = 2.5146,              ds_round = 0., 0.0508,          37  129    1
radius_shoulder = 0.25146,         ni_round = 0, 21,               37  129    1
half_cone_angle = 0.,              numi_aft_body = 201,           165   37    1
skirt_angle = 32.5,                ni_regrid_aft_body = 37,       165   37    1
skirt_length = 0.15225564,         $END                           165   37    1
ni_regrid_forebody = 33,                                          165   37    1
$END                                                               37   37    1
```

37 37 1

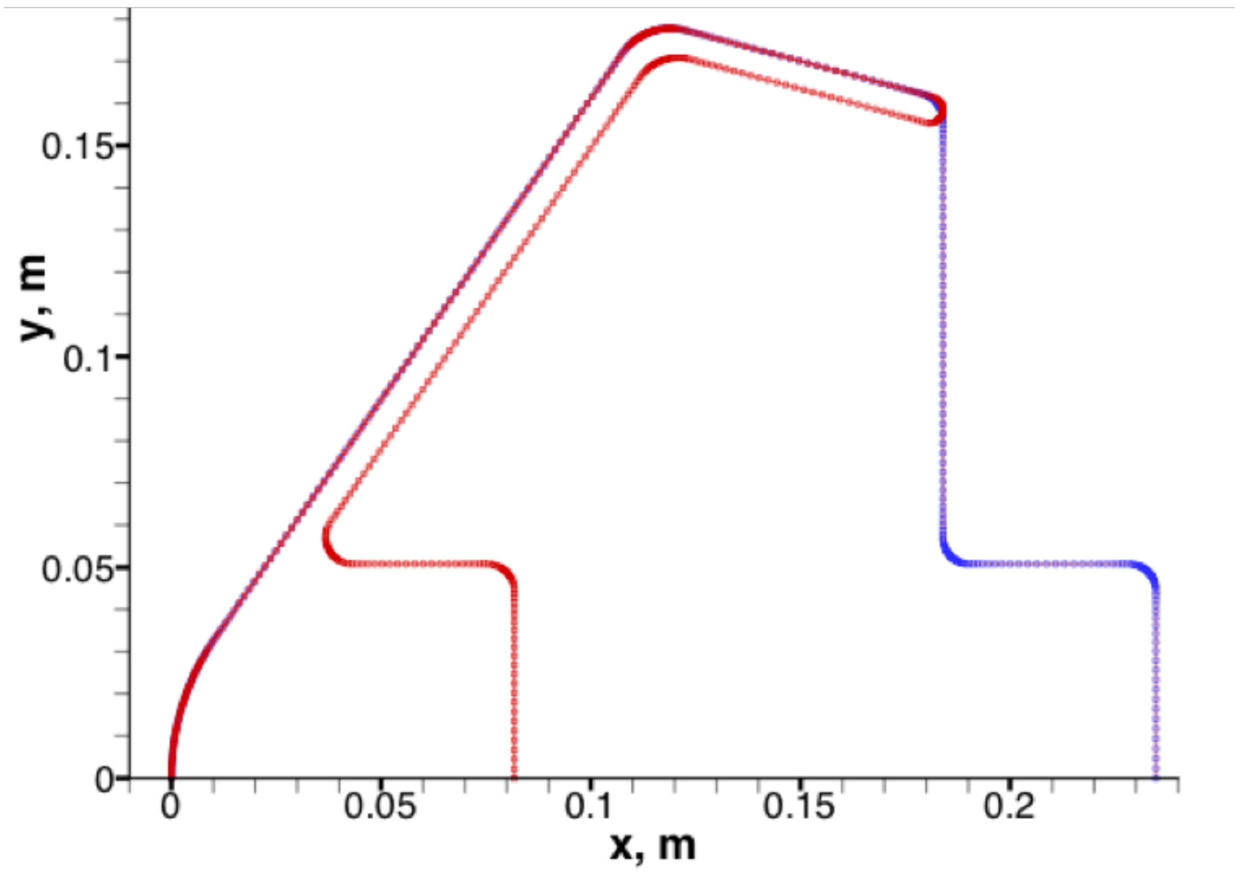


Figure 5a: Two sample full-body generatrices from *CAPSULE_GRID*: open (red) and closed SPRITE configurations with 14" diameter and 55° forebody half cone angle. The aft bodies are initially rectilinear, and the vertices are optionally rounded, either with specified radii or with tangent lengths from the underlying vertices. The point distributions on the discretized generatrices are curvature-based and independent of the scaling.

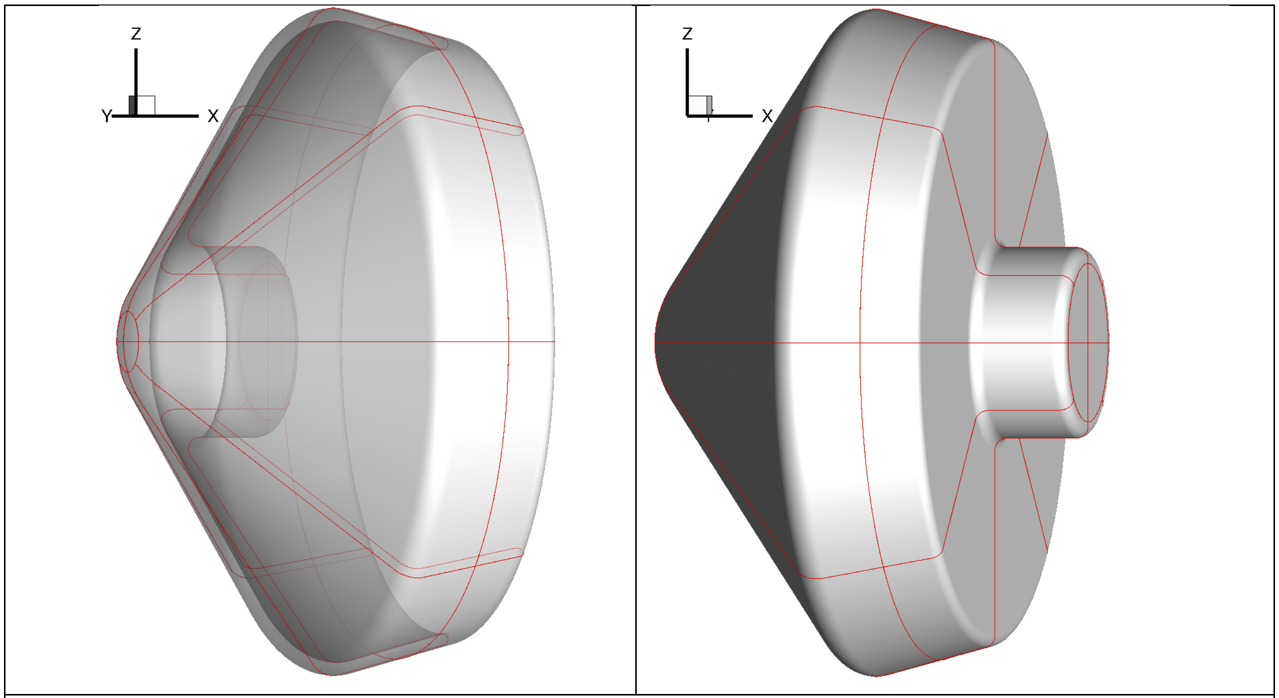


Figure 5b: Corresponding 3-D surface grids from *CAPSULE_GRID*: open aft body (translucent, left) and closed SPRITE configurations with 14" diameter and 55° forebody half cone angle.

3. When the *TECPLOT*able generatrix and 180° surface grid appear satisfactory (point counts appropriate for the flow solver block splitting, and continuous spacing across the nose patch boundaries, for instance), either the **`forebody-generatrix.dat`** or **`full-body-generatrix.dat`** form of the generatrix will be ready for 2-D volume gridding via an available *GRIDGEN* script (glf file) or *HYPGEN* script.

4. For *GRIDGEN*, copy **`[path]/cfdtools/CAPSULE_GRID/2D-forebody.glf`** or **`[path]/apps/cfdtools/CAPSULE_GRID/2D-full-body.glf`** to the working directory, and enter:

   ```
   gridgen -b 2D-forebody.glf    or
   gridgen -b 2D-full-body.glf
   ```

   One or two `*.glf` file parameters may need tweaking to give a satisfactory result. Be sure that the specified number of hyperbolic marching steps was actually performed (no ***GRIDGEN*** diagnostic if not) by checking the dimensions of the **`2D.p3da`** output grid. (This count need not be the eventual grid point count, thanks to the *COMPRESS2D* step.)

   N.B.: A sting case is treated as a forebody.

   For *HYPGEN*, copy **`[path]/apps/cfdtools/CAPSULE_GRID/capsule.sh`** and consult **hypgen.doc** from the same location. Enter ./capsule.sh to execute the multiple steps. Unlike the *GRIDGEN* approach, *HYPGEN* can control both the number of points off the surface and the length of the wake, although trial and error will likely be required.

5. ***COMPRESS2D*** will squeeze the upstream portion of the **`2D.p3da`** single-block hyperbolic 2-D volume grid and (in the full-body case) split it into 2 blocks for BC purposes. A rough shock stand-off distance at the nose (with margin) is prompted for as a multiple of the diameter. Enter the command **`compress2d`** and answer a few prompts to control

the compression (see Figs. 6a, 6b and the warning on p. 14), the limiting shock angle (forebody) or splitting (full body), the off-wall spacing and stretching (`ds2fr = 1`, probably), and the number of points in the off-wall *j* direction. Some trial and error is likely to be required, and taking notes on the controls tried is advised.

6. If α = 0 (axisymmetric flow), the result is ready for *DPLR2D*. For 3-D solutions at angles of attack (but not "umbrella" cases—see pp. 24-27), run ***REVOLVE_GRID*** and answer a few prompts,

7. A revolved 1-block (forebody) or 2-block (full body), 180° 3-D volume grid has singular point(s) at the nose [and tail] that are not viable with *DPLR*. Therefore, the final step is to impose the topology of the 6-, 10-, 12-, or 16-patch surface grid from *CAPSULE_GRID* on this volume grid via ***RADIAL_INTERP***[5], which efficiently produces a new volume grid from an existing volume grid and a new surface grid by taking advantage of the single layer of blocks most often employed with atmospheric entry vehicles. Copy a control file from here:

   `[path]/RADIAL_INTERP/radial_interp.inp`

   Edit this copy (keep the name), then enter **`radial_interp > radial_interp.log &`**. The resulting 6-, 10-, 12-, or 16-block volume grid is ready for *DPLR3D*. The *TEMPLATE* utility can help with preparation of the *DPLR* control file (and interface file, unless *FCONVERT*'s option to determine block interfaces is employed instead of using this file).

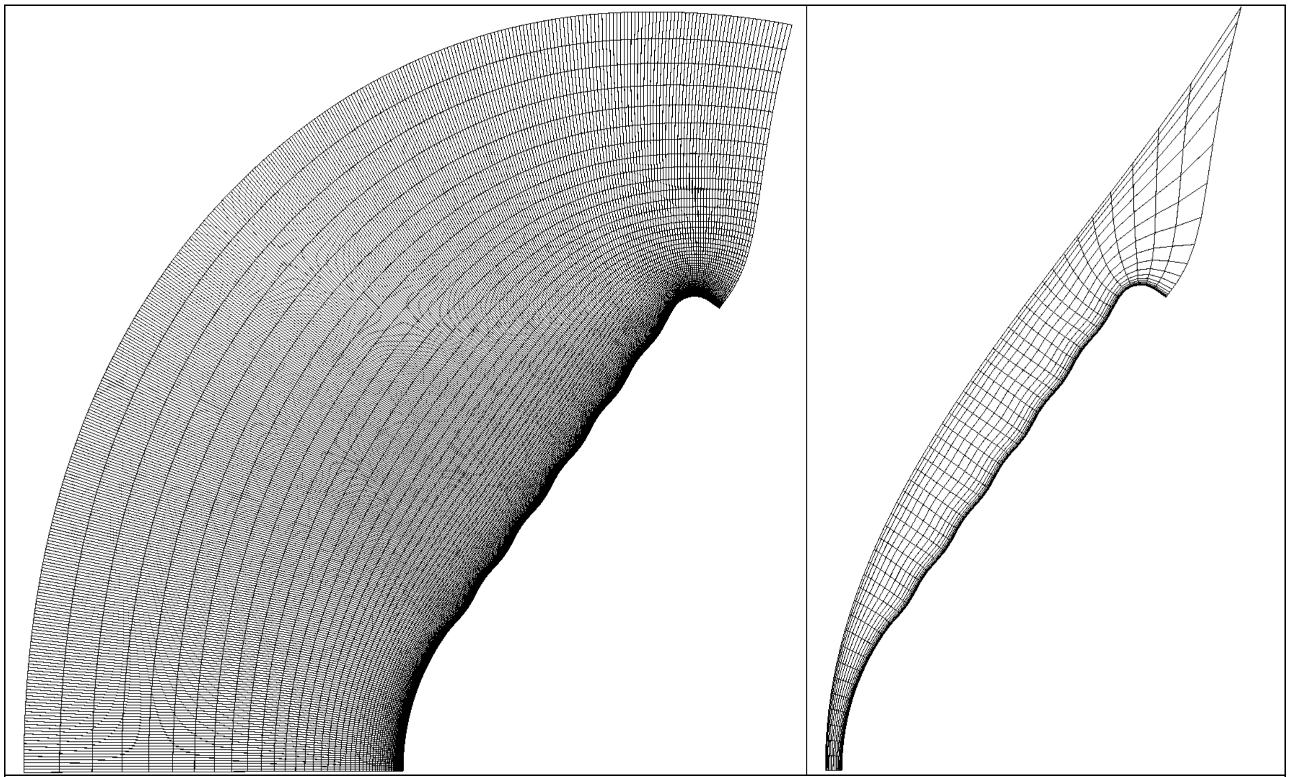
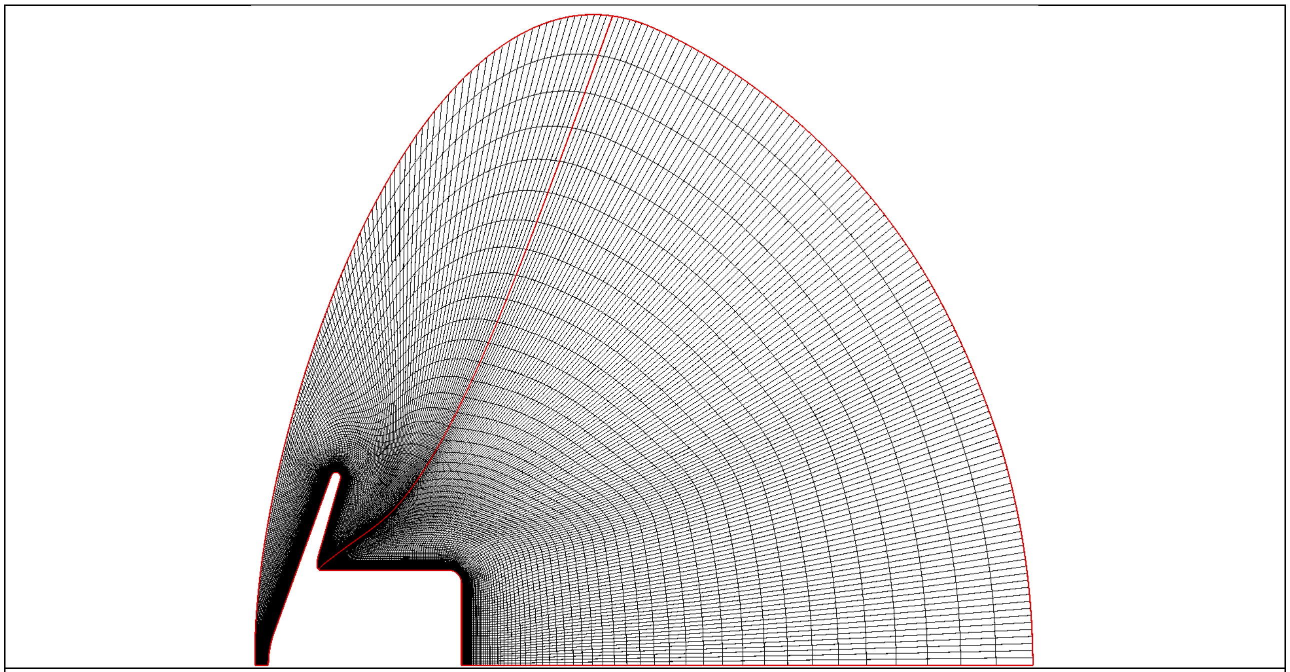


Figure 6: (a) Sample automated initial forebody 2-D volume grid (HIAD, 8.5 m diameter, 55° sphere-cone, 6 toroids with 1" deflections) before and after application of *COMPRESS2D* for axisymmetric calculations. Only every 8th grid line is shown in the compressed form. N.B.: Both compressions shown are now considered excessive. See the page 14 recommendations.



(b) Sample automated over-compressed full-body 2-D volume grid (ADEPT, 6 m diameter, 70° sphere-cone housing a 2 x 2.76 m cylindrical payload, with no faceting). N.B.: The initial 2-D hyperbolic volume grid should also have been grown further than shown—5-6 diameters aft of the nose is recommended, or possibly as much as 10 diameters if wake radiation calculations are in the picture.

Another sample control file for *CAPSULE_GRID* (corresponding to the closed SPRITE example illustrated above) appears below. Note that Fortran namelists traditionally start at or after column 2 as shown. Originally this had to do with using column 1 for carriage control and being able to read a written namelist, but modern compilers no longer seem to enforce this peculiar restriction. The trailing commas are also optional, and more than one entry may appear on a line, separated by commas.

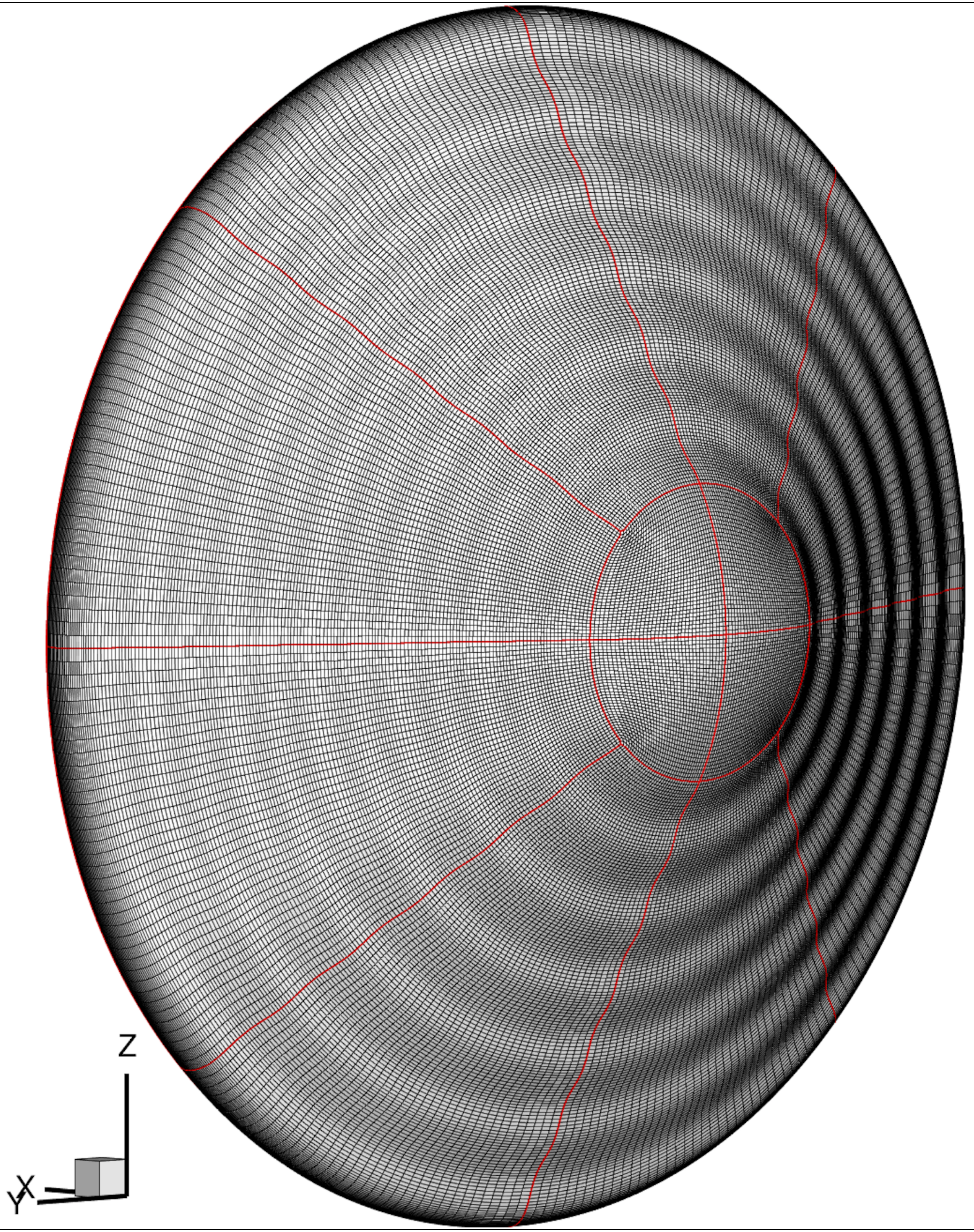
```
 $FOREBODY_INPUTS
 aft_body_too = T,
 units_in_inches = F,
 geometric_scale = 1.,
 input_generatrix = 'none',
 spherecone = T,
 numi_forebody = 201,
 numj = 145,
 x_nose = 0.,
 r_nose = 0.,
 radius_nose = 0.058166,
 radius_base = 0.1778,
 radius_shoulder = 0.014224,
 radius_vertex = 99.,
 half_cone_angle = 55.,
 half_cone_angle_fore = 99.,
 half_cone_angle_aft = 99.,
 radius_cone_juncture = 99.,
 skirt_angle = 15.,
 skirt_length = 0.031214,
 nose_patch_file_name = 'NosePatch61B.p3da',
 output_generatrix = 'gen.dat',
 surface_grid_file_name = 'surface_grid.g',
 power_fore = 0.5,
```

```
 $AFT_BODY_INPUTS
 sting_case = F,
 ncones = 3,
 r_aft = 0., 0.160913, 0.0508, 0.0508,
 cone_angle_aft = 15., 90., 0.,
 rounding_mode = 2,
 ds_round = 0.,0.00635,0.00635,0.00635,
 ni_round = 0, 21, 21, 21,
 power_aft = 0.5,
 numi_aft_body = 201,
 ni_regrid_aft_body = 21,
 sting_stretch_multiplier = 3.,
 ng_split = 0,
 i0_umbrella = 0,
 i1_umbrella = 0,
 i2_umbrella = 0,
 $END
```

```
ni_regrid_forebody = 33,
ripple_case = F,
ntoroids = 6,
peak_ripple = 0.0254,
umbrella_case = F,
nedges = 8,
peak_deflection = 0.01,
rib_deflection = 0.,
frustum_radius = 0.,
resolve_the_ridges = T,
rib_thickness = 0.01,
round_the_ridges = T,
$END
```

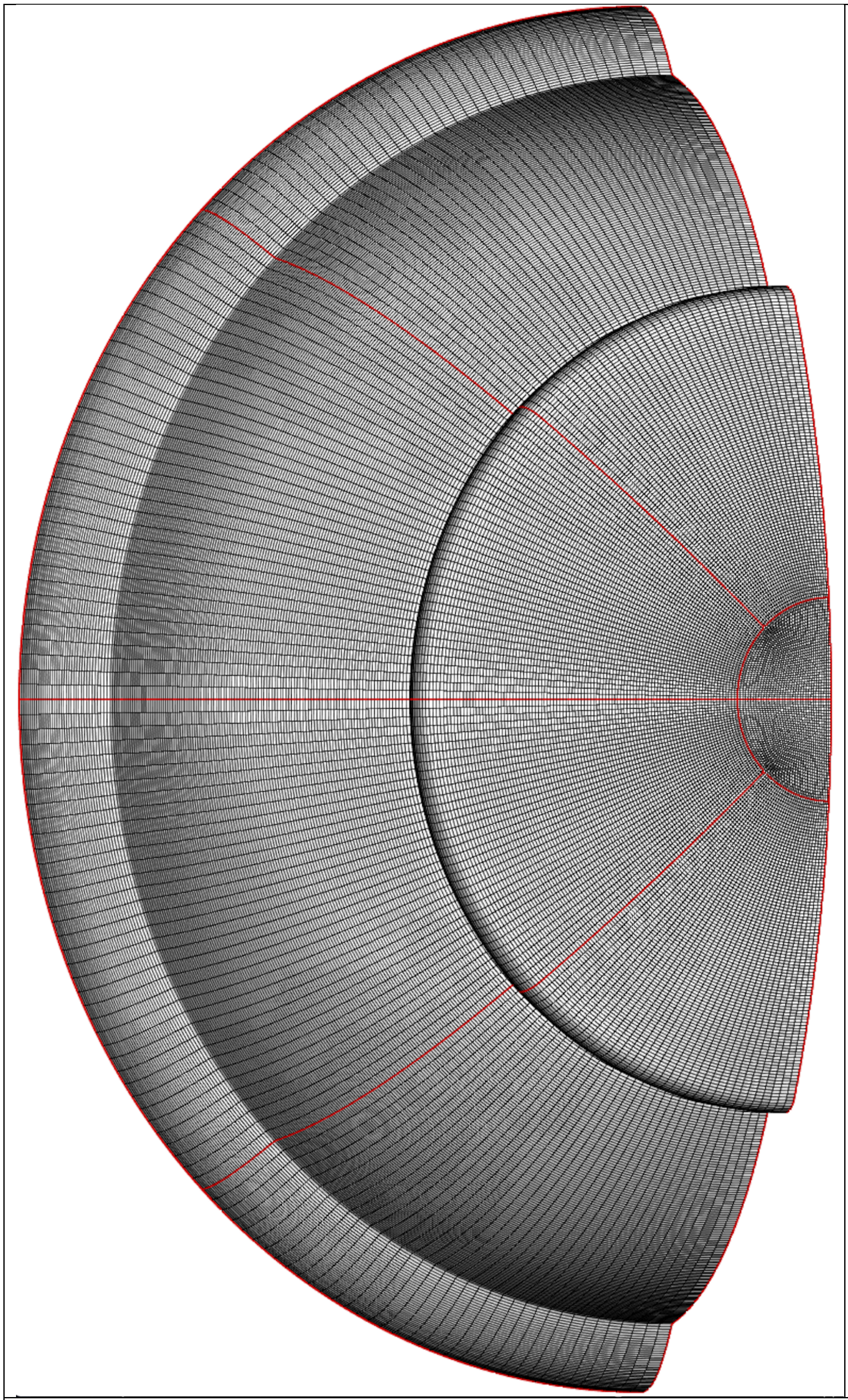
Further sample control files should be found in a subdirectory named `/CAPSULE_GRID` **`/Standard_Configurations`** or similar. Discussing them all is beyond the scope of this document, but they should include these cases, two of which appear in Figs. 7 and 8:

123CH_With_Sting (Orion shock tunnel model)	SIAD (Fig. 8)
CEV (aka MPCV and Orion)	SPRITE (Opem and Closed)
Genesis	SRC (Concave & Convex Aft Body)
HIAD-12-Meter (Fig. 7)	Sample_Return_Capsule (several)
Hemisphere	Sphere
Mars_Pathfinder	Stardust
Mars_Phoenix_Nonanalytic	Unit-Radius-Meteor
Pathfinder (slight variation)	Venus_45deg



```
 $FOREBODY_INPUTS
 aft_body_too = F,
 units_in_inches = F,
 geometric_scale = 1.,
 input_generatrix = 'none',
 spherecone = T,
 numi_forebody = 201,
 numj = 145,
 x_nose = 0.,
 r_nose = 0.,
 radius_nose = 3.0,
 radius_base = 6.0,
 radius_shoulder = 0.3,
 radius_vertex = 99.,
 half_cone_angle = 60.,
 skirt_angle = 30.,
 skirt_length = 0.1,
 nose_patch_file_name =
'NosePatch61B.p3da',
 output_generatrix =
'gen.dat',
 surface_grid_file_name =
'surface_grid.g',
 ni_regrid_forebody = 33,
 ripple_case = T,
 ntoroids = 6,
 peak_ripple = 0.05,
 umbrella_case = F,
 nedges = 8,
 peak_deflection = 0.01,
 resolve_the_ridges = T,
 rib_thickness = 0.01,
 round_the_ridges = T,
 $END
```

Figure 7: 12 m HIAD (Hypersonic Inflatable Aerodynamic Decelerator) forebody (`ripple_case = T`).



```
 $FOREBODY_INPUTS
 aft_body_too = F,
 units_in_inches = F,
 geometric_scale = 1.,
 input_generatrix =
'2d.siadE.line.p3da',
 spherecone = T,
 numi_forebody = 301,
 numj = 145,
 x_nose = 0.,
 r_nose = 0.,
 radius_nose = 0.058166,
 radius_base = 0.1778,
 radius_shoulder = 0.014224,
 radius_vertex = 99.,
 half_cone_angle = 55.,
 half_cone_angle_fore = 99.,
 half_cone_angle_aft = 99.,
 radius_cone_juncture = 99.,
 skirt_angle = 15.,
 skirt_length = 0.031214,
 nose_patch_file_name =
'NosePatch61B.p3da',
 output_generatrix = 'gen.dat',
 surface_grid_file_name =
'surface_grid.g',
 ni_regrid_forebody = 33,
 ripple_case = F,
 ntoroids = 6,
 peak_ripple = 0.05,
 umbrella_case = F,
 nedges = 8,
 peak_deflection = 0.01,
 resolve_the_ridges = T,
 rib_thickness = 0.01,
 round_the_ridges = T,
 $END
```

Figure 8: 8 m SIAD (Supersonic Inflatable Aerodynamic Decelerator) forebody.

VI. Methodology Discussions

Option to read a generatrix as opposed to constructing an analytic one

Quite often, `input_generatrix = 'none'` is appropriate. However, if a legitimate file name is entered here (probably a generatrix extracted from a CAD model), surface and volume grids can still be constructed from such a numerical definition. Two situations are handled:

If the requested number of grid points, `numi_forebody [+ numi_aft_body - 1]`, exactly matches the number of points found in the input generatrix, then those points are employed precisely. Otherwise, curvature-based redistribution to the specified number of points is applied the same way as it is applied to initial discretizations of analytic generatrices. In case the input definition contained some preferred point distribution, this option also saves the file **`proportionally_adjusted_gen.dat`** which can then be made use of in a second run of *CAPSULE_GRID*.

An input **full-body** generatrix, if not used precisely because of the requested point counts, is redistributed in either two parts or one part, depending on the input value of `ng_split`. Some aft bodies may have very small radii in the generatrix, resolution of which is more feasible with smaller systems of equations to solve iteratively during the curvature-based redistribution. The two parts are defined by some sensible index entered for `ng_split` in the aft-body namelist. Entering 0 defaults to the index of the maximum radius in the input generatrix. Use `ng_split < 0` to specify redistribution of a full-body input generatrix as a single piece. This will avoid any effects of blending the fore- and aft-body grid spacing across a split, and should behave well if the aft body does not have extremely small radii.

Further control is provided through inputs `power_fore` and `power_aft`, which default to the 0.5 value that was originally hard-coded. These exponents should be in the range 0.1 – 1.0, with higher values preserving more of the curvature-based spacing. If any curvature-based iteration does not converge, it is automatically repeated with the exponent reduced by 0.1, down to as low as 0., which corresponds to uniform spacing and should never fail (although it is unlikely to be useful except for plotting results).

Note that some generatrices can never be redistributed adequately using the plain curvature-based technique: they may contain straight line segments meeting at a vertex, meaning the computed curvature is zero on both segments except at the vertex. Such one-point spikes in the spacing shape function being used are almost certain to be missed, leading to uniform spacing across the vertex. However, a work-around has been incorporated to resolve such vertices by detecting the spikes and adjusting the curvature distribution without touching the geometry. See **Automated Clustering Around Sharp Vertices** on pp. 32-33**.**

Common mistakes with input generatrices are inadequate resolution and drastic changes in resolution. Be sure to cluster plenty of points around sharp or small-radius corners to avoid spline excursions during redistribution.

For additional discussion of curvature-based redistribution, see also **Further Notes on Curvature-Based Redistribution** and **Lessons Learned (1)** below.

Option to model deployable aeroshells (ADEPT "umbrella" configurations)

A specialized option activated by entering `umbrella_case = T` turns a plain sphere-cone forebody into a polygonal forebody with any number of edges or ribs. An aft body is also an option. The `nedges` input refers to the whole surface, although only the starboard half is constructed as usual. The y = 0 symmetry plane passes through the *middle* of the upper and lower edges so that it captures the maximum amount of deflection imposed via catenary-type bending in both the radial and azimuthal directions according to `peak_deflection` (fabric; > 0.) and `rib_deflection` (ribs; < 0. for convexity). Use `frustum_radius` to control the start of both types of deflection. This defaults to the sphere-cone tangency point; override the default for shorter ribs.

Rounding of the ridges along the ribs is an option, as is some added resolution of the ribs. If `resolve_the_ridges = T`, the spacing alongside each rib is adjusted via restretching in the transverse direction so that it does not exceed half the value of `rib_thickness` that is input. (Towards the apex, this spacing may be unavoidably smaller than [half] the specified thickness, which is really a spacing input.) Some possibilities appear in Figs. 9-12.

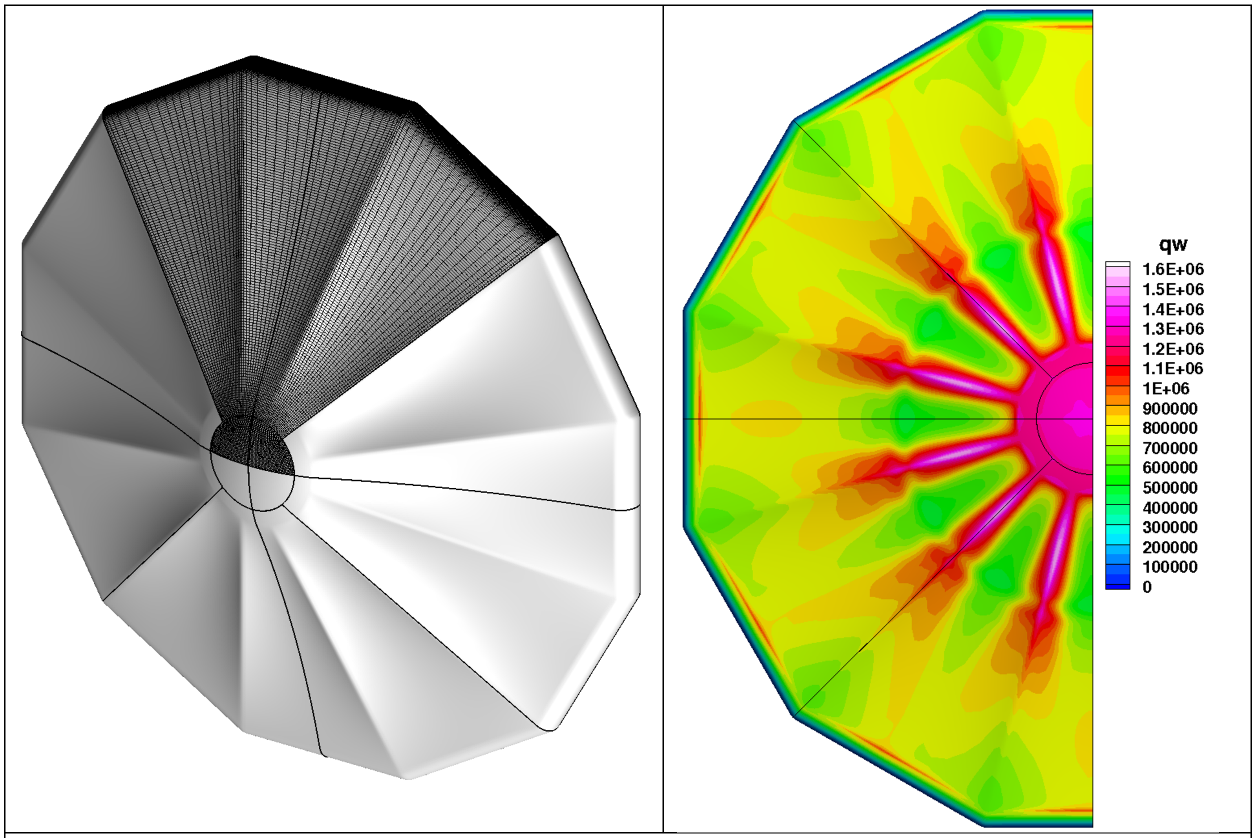


Figure 9: An 8-meter 70° sphere-cone forebody, modified by CAPSULE_GRID's "umbrella" option to simulate an ADEPT (Adaptive, Deployable Entry and Placement Technology) concept is shown (left) and solved for at the peak heating Mach 48 condition of a ballistic entry to Venus (laminar heat flux solution shown in W/m^2). The peak deflection was 10 cm. The initial 3-D hyperbolic volume gridding employed `heatshield-generator.glf` applied to the 6-patch regridded form of the surface that is output by CAPSULE_GRID. Cases with the aft body (see below) can morph the 2-D hyperbolic volume grid into a 3-D volume grid instead as part of the COMPRESS2D option. By default, deflections start at the sphere-cone nose tangency point. Use `frustum_radius` to grow the rigid nose cap on to the conical flank (i.e., to start the deflections further outboard).

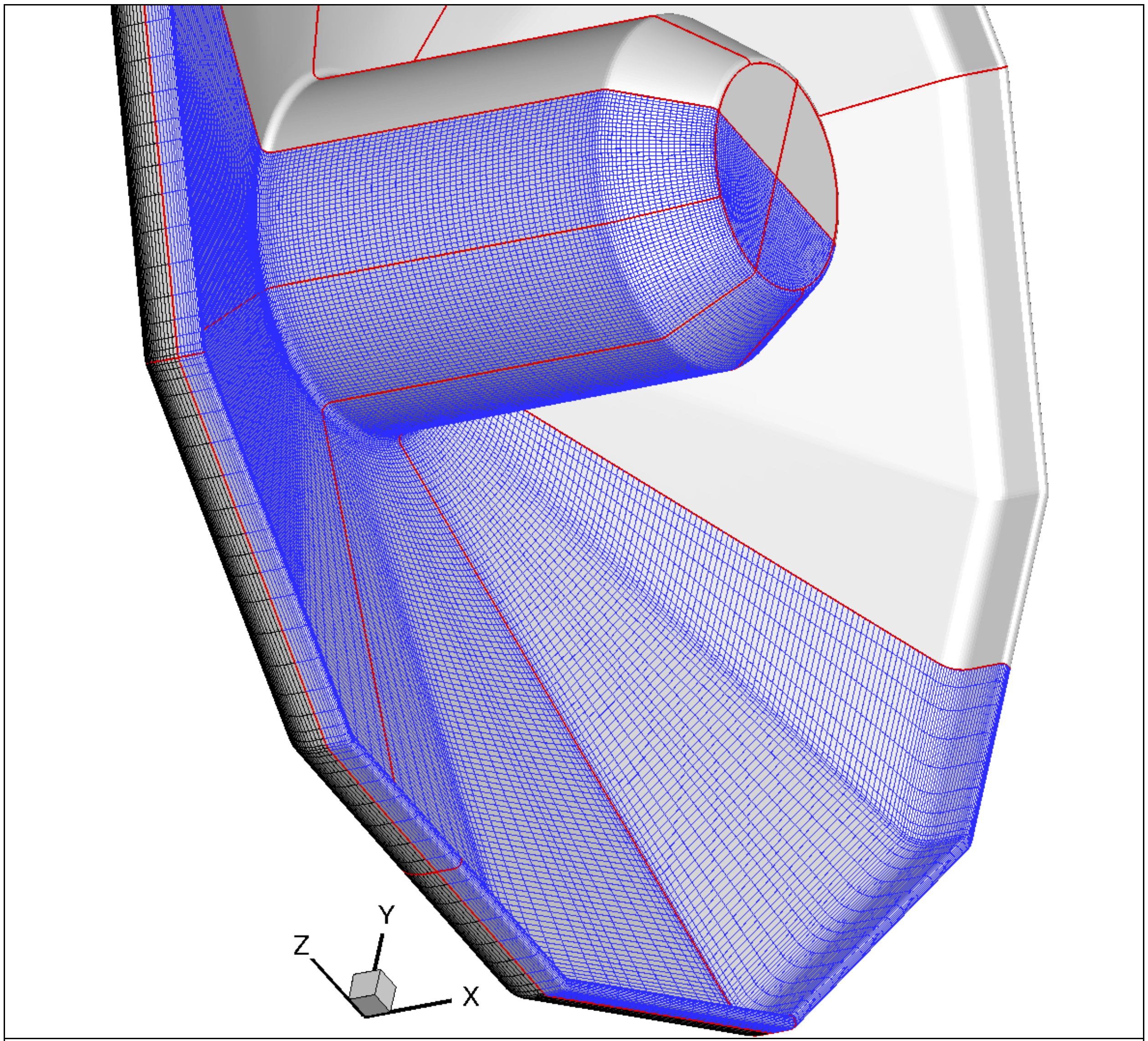


Figure 10: This 6-meter diameter, 12-ribbed full-body ADEPT "umbrella" case models the 5-cm rib thickness rather than the 5-mm flexible TPS thickness, in order to leave enough points to round the aft end of the shoulder. Achieving such a uniform TPS thickness requires careful choice of r_aft(:), by trial and error. Any deflections imposed on the forebody appear in the aft body too, via x shifts and morphing. Ribs have been resolved using 0.01 m for the `rib_thickness` input (= 2 x grid spacing along a rib; 0.02 was intended), and the rib ridges were rounded. They are now flattened instead.

Underlying forebody dimensions are 165 x 121, while the aft body is 258 x 121. The `i1-` and `i2_umbrella` values of 102 and 196 apply to patches 7-10 (239 x 31). Outer points 196 - 239 on each spoke of these patches are *shifted* from the interim axisymmetric aft surface to match the faceted forebody, avoiding geometric distortion. Then points 102 - 196 on each aft outer spoke are morphed as curves to match the shifted aft shoulder portion. For the record, the nose and aft-body quadrant patch inputs were `ni_regrid_forebody` = 33 and `ni_regrid_aft_body` = 20. The `i0_umbrella` control was not needed (slender payload bay inboard of the forebody transition to faceting).

Note that for the "umbrella" morphing to be feasible, the value of `numj - 1` must be divisible by both `4` and `nedges`. Also, the underlying geometry need not be a strict sphere-cone. For instance, the generatrix for the author's **Configuration Z3** (blended circle-ellipse-circle, with a convex flank) may also be morphed this way.

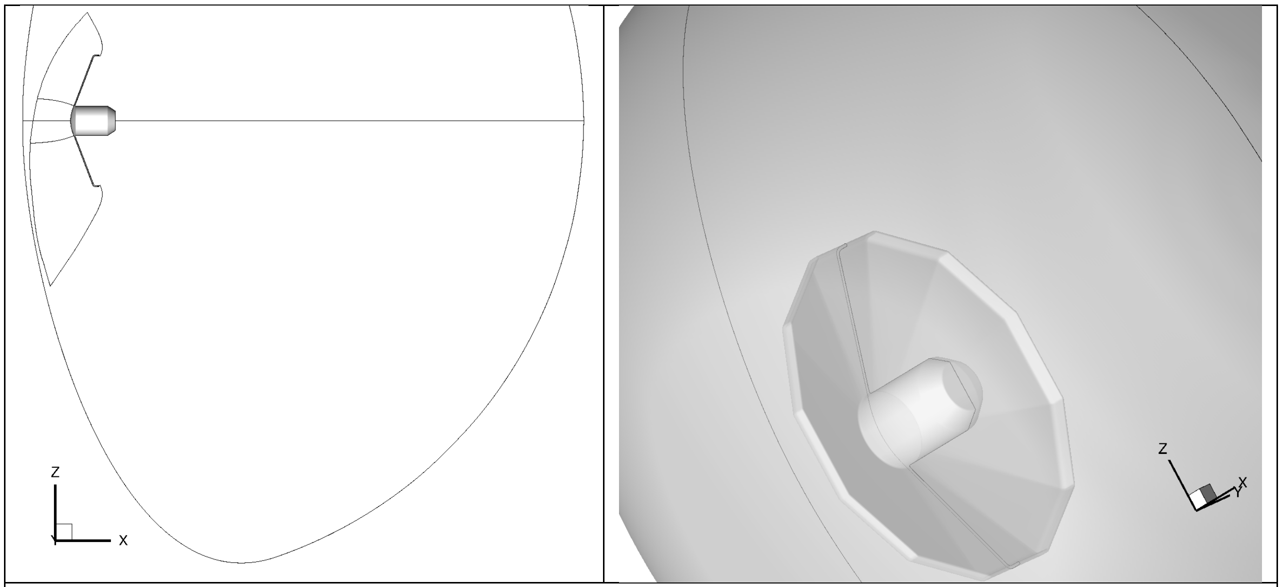


Figure 11: The same 6-meter diameter, 12-ribbed full-body ADEPT "umbrella" case is shown after the *COMPRESS2D* step. It helps to have a forebody-only shock-aligned grid for comparison.

On the left, the symmetry plane of the compressed/morphed 3-D volume grid is shown for a low Mach number case (large stand-off distance upstream) at angle of attack.

Note the rule of thumb for the distance of the downstream boundary from the nose: about 4 diameters. This has to be controlled in the 2-D hyperbolic gridding script (number of steps and/or initial step).

On the right, the inner and outer *k*-planes of the morphed 3-D volume grid from the specialized *COMPRESS2D* option are shown, prior to the final *RADIAL_INTERP* step that imposes the topology of the 12-patch surface grid from *CAPSULE_GRID* onto every *k* plane.

Note that the surface faceting and any fabric deflections mean the radial lines of the volume grid constructed this way are not all truly orthogonal to the surface, although the discrepancies should be small. For the case shown, an effort to construct a 3-D volume grid directly in *GRIDGEN* proved unsuccessful, and this prompted the 2-D morphing option.

The following Fig. 12 illustrates the need for introducing the `i0_umbrella` control, as well as the usage of the earlier `i1_umbrella` and `i2_umbrella` controls. In the presence of forebody faceting (and possibly fabric and rib deflections), an axisymmetric aft body must be adjusted to preserve the thickness of the fabric being modeled. The recommended gridding approach is to turn off faceting initially (`umbrella_case = F`) and probe surface patch 7 along the centerline to choose the needed aft body indices. Here, `i1_umbrella` and `i2_umbrella` should define the essentially straight portion of each outer aft spoke. The outboard portion from `i2_umbrella` to `ni` is first shifted to match the faceted forebody without distortion. Then the portion from `i1_umbrella` and `i2_umbrella` is morphed to follow the shifted shoulder portion. One or two further steps are still required, though.

Constant fabric thickness is imposed as follows: For each spoke grid line, the thickness is taken to be the difference in **x** between the aft body at **i2_umbrella** and the corresponding **x** of the forebody at the same radius. This difference is x is then imposed on the aft body spokes from **i1_umbrella** to **i2_umbrella**.

One more step may still be needed, because the x shift (zero at **i2_umbrella**) may not be zero at **i1_umbrella**. If the faceting starts nearer the axis (**radius_frustum**) than the upstream radius of the aft payload housing (corresponding approximately to the radius at **i1_umbrella**), then the transition to the payload bay requires another morph between indices **i0_umbrella** and **i1_umbrella**, where the default for **i0_umbrella** is **i1_umbrella - 20**, which may be overridden with a nonzero input.

Be sure to check **i0/i1/i2_umbrella** if **numi_aft_body** **or** **ni_regrid_aft_body** needs changing.

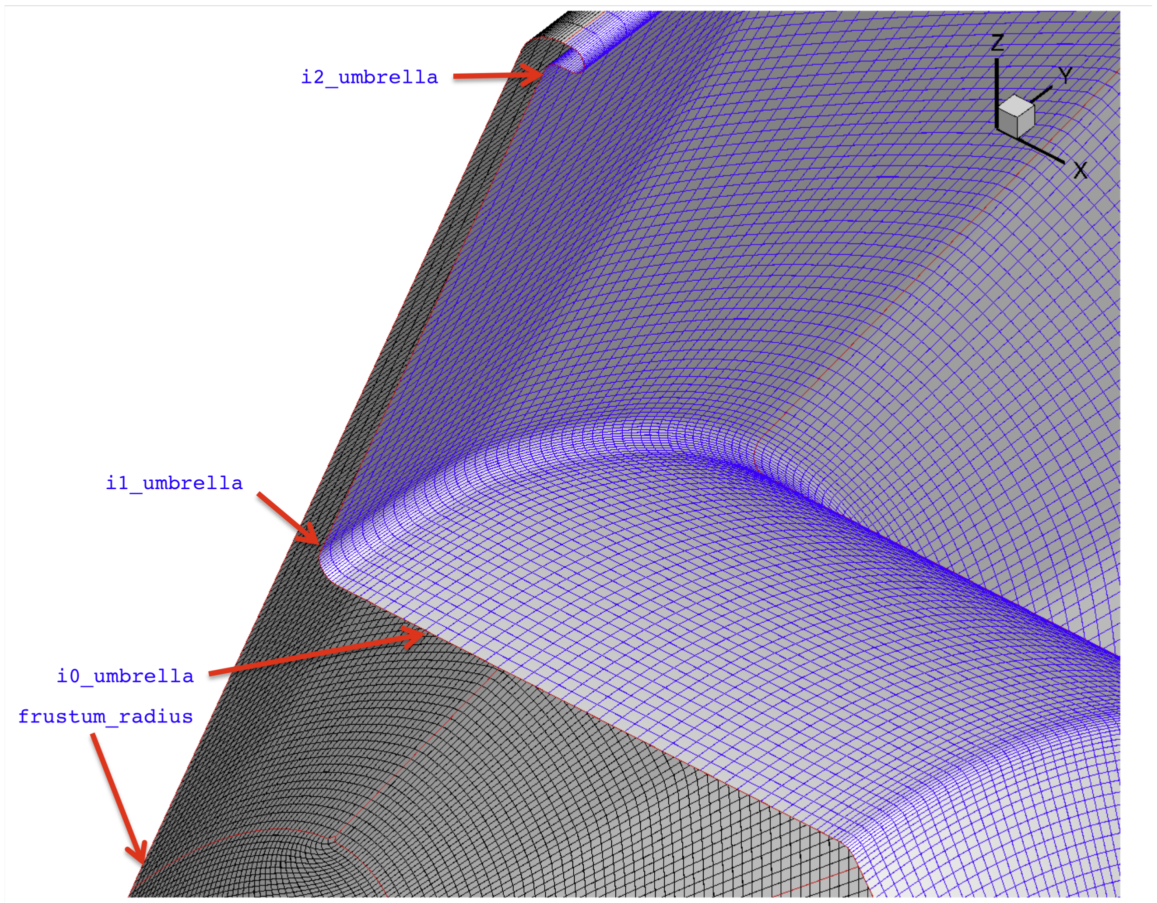


Figure 12: If the forebody is faceted, with possible fabric and rib deflections, the aft body (initially axisymmetric) must be adjusted to conform to the forebody. The procedure is outlined above, and the figure clarifies the meaning of the various control inputs required.

In this example, the transition to faceting begins well inside the radius of the payload bay, requiring a second morphing of each spoke between indices i0_umbrella and i1_umbrella. Moving i0_umbrella further aft (smaller) improves the correction.

Option to model a shock tunnel sting

A capsule supported in a ground test section with a sting can be modeled using an option (**`sting_case = T`**) that suppresses closure of the aft body and restretches the generatrix points along the sting (which are essentially uniformly spaced in the preliminary curvature-based distribution). A scaled Orion MPCV test article (using **`geometric_scale = 0.05050505`**) can be modeled with the aft body controls shown below Fig. 13:

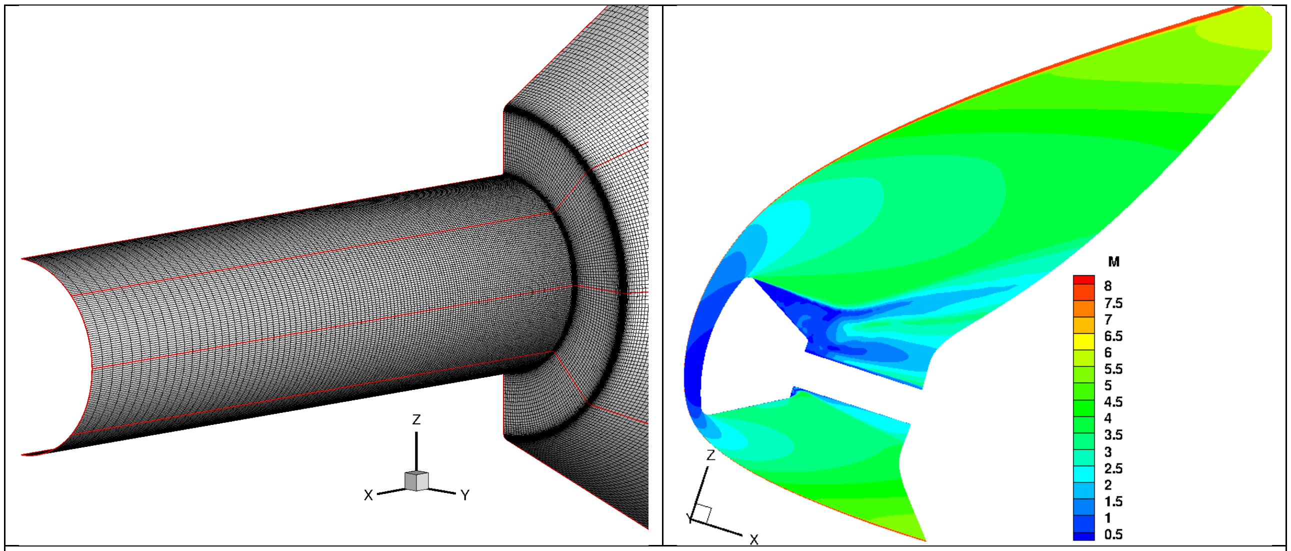


Figure 13: Use of `sting_case = T` suppresses closure of the aft body. The standard curvature-based spacing on the sting portion of the generatrix (mostly uniform) could be retained via `sting_stretch_multiplier = 0.`, but `3.0` was used here to increase the sting spacing with x. Any nonzero multiplier is applied to the interim end-of-sting spacing for the restretching. The symmetry plane Mach contour plot shows a simulation of a shock tunnel test, with +18° angle of attack.

```
$AFT_BODY_INPUTS
 sting_case = T,
 ncones = 3,
 cone_angle_aft = 30.0, 90., 0.,
 r_aft = 0., 1.05852395, 0.62865, 4.31798,
 rounding_mode = 2,
 ds_round = 0., 0.0508, 0.02, 0.0,
 ni_round = 0, 31, 31, 0,
 numi_aft_body = 297,
 ni_regrid_aft_body = 0,
 sting_stretch_multiplier = 3.,
 $END
```

Additional discussion of the curvature-based grid point distribution scheme appears on the following pages. **Lessons Learned (1)** was prompted by experience with extremely fine sting-case grids, but is relevant to any case with an aft body. However, note that it was written prior to **Further Notes on Curvature-Based Redistribution** below. These notes include the realization that the geometry should be normalized before performing curvature-based calculations. This is now automatic. (Curvature is extremely dependent on data scaling.)

Further Notes on Curvature-Based Redistribution

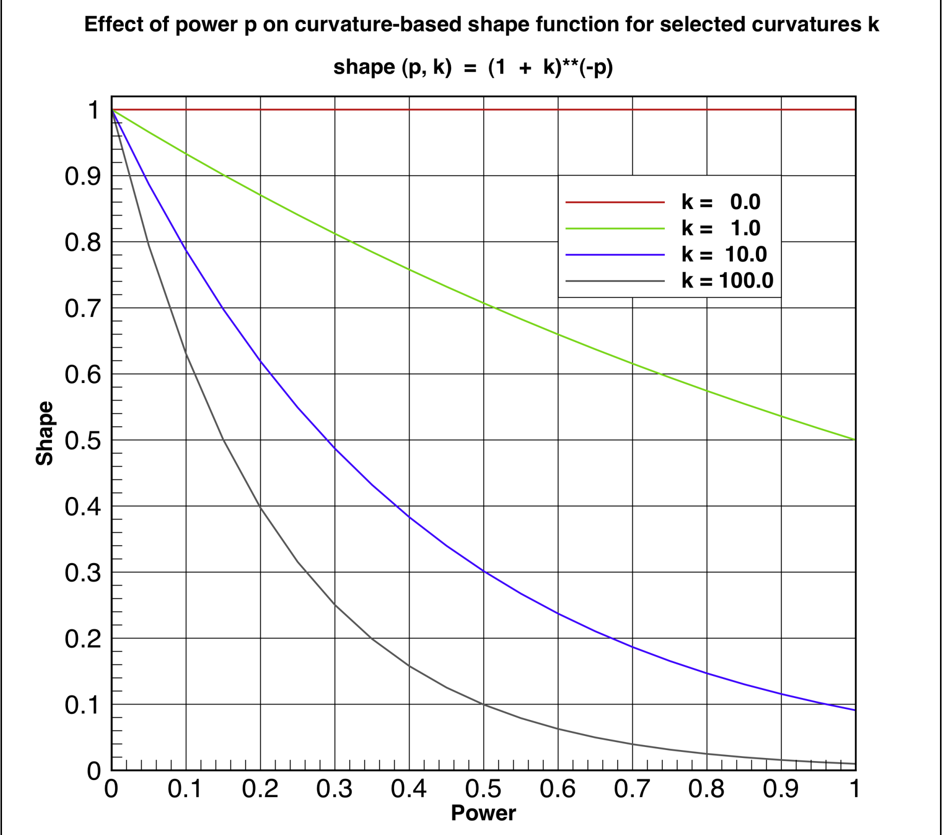
The generatrix for simulating a subscale hemisphere model fired in a ballistic range is discussed here to provide an understanding of the difficulties of blending straight-line/zero-curvature/infinite-radius-of-curvature segments with any other segments, and of the curvature-based shape function that underlies the present technique. The relevant circular quadrant has a radius of exactly 1 once it is normalized to the unit square as part of the calculations, while the portion of the aft body that is treated as a forebody skirt has zero curvature. Only the forebody part of the generatrix needs to be discussed. This is carried around the slightly rounded shoulder as part of blending the three segments, although a skirt length of 0 can also produce acceptable results. Note also that when **`skirt_angle = 90°`** (only), **`skirt_length`** is in the vertical direction, not in the x direction as it normally is.

A hemisphere (Fig. 12) is an extreme case of the Apollo-type shape (**`half_cone_angle = 0`**), signaled by **`radius_base = radius_nose`**. This has to be treated as a special case in the analytic construction because the maximum radius (R_{base}) cannot equal R_{nose} unless $R_{shoulder}$ = 0. In the latter case, we would ***not*** recommend zero **`skirt_length`**, since the shoulder would be sharp but with largely uniform spacing there. [See the following sub-section for automated clustering at sharp vertices in the fore- or aft-body generatrix.]

```
$FOREBODY_INPUTS
aft_body_too = T,
input_generatrix = 'none',
spherecone = T,
numi_forebody = 221,
numj = 121,
radius_nose = 0.01429,
radius_base = 0.01429,
radius_shoulder = 0.0002,
half_cone_angle = 0.,
skirt_angle = 90.,
skirt_length = 0.002,
nose_patch_file_name = 'NosePatch61B.p3da',
power_fore = 0.5,
ni_regrid_forebody = 33,
$END

$AFT_BODY_INPUTS
ncones = 1,
r_aft = 99., 0.,
cone_angle_aft = 90.,
rounding_mode = 2,
ds_round = 0., 0.004,
ni_round = 0, 21,
numi_aft_body = 111,
ni_regrid_aft_body = 33,
$END
```

Smoothing of the shape function distribution, $\mathbf{s_i = 1 / (1 + \kappa_i)^p}$, $\mathbf{p \in [0, 1]}$, is essential for smoothly-varying grid spacing. *CAPSULE_GRID* also smooths the resulting redistributed arc-lengths. Heuristics are unavoidable. The present numbers of explicit smoothing iterations are `MAX (9, MIN (NDAT/20, 20))` (shape function) and `MAX (5, MIN (NDAT/20, 20))` (redistributed arc lengths). The redistributed generatrix points are (barely) visible in green in the second half of Fig. 14.



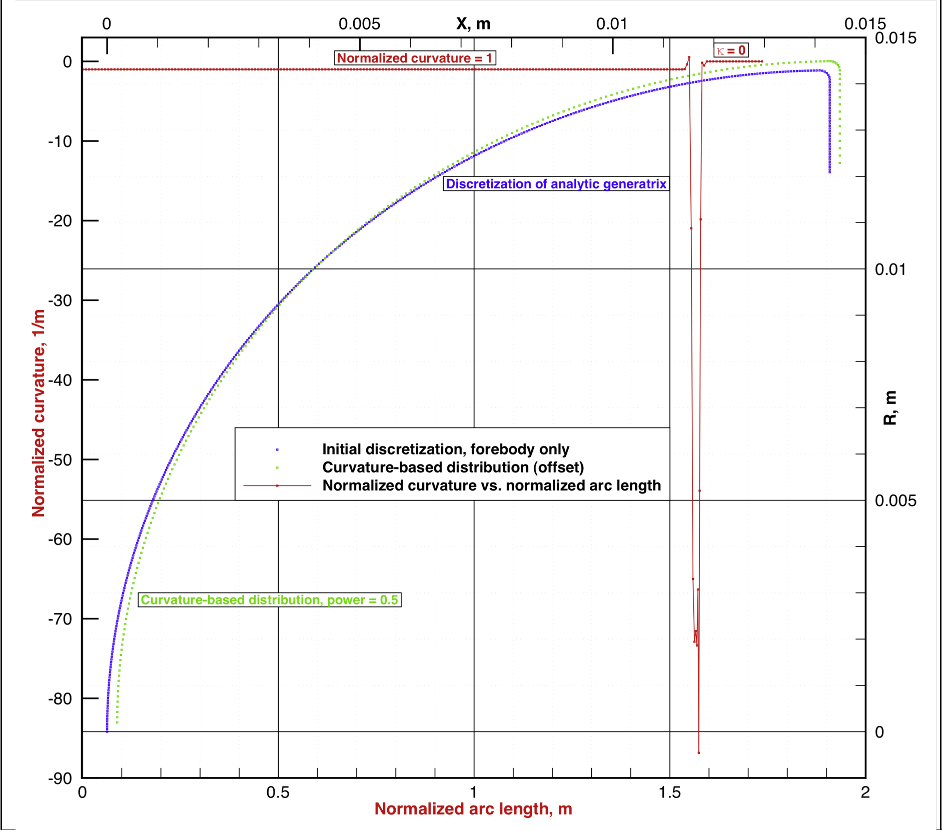


Figure 14:

For a handful of curvature values, the effect of the power p in the curvature-based shape function

$$\frac{1}{(1+\kappa)^p}$$

is shown at left. The grid spacing achieved is always proportional to this shape function. A value of 1 (κ = 0) everywhere would correspond to uniform spacing for any value of p.

The (normalized) hemisphere case involves the first two curves and (essentially) the fourth at the rounded shoulder.

Note that if p = 1, the nose spacing will be exactly half that of the nose circle (0.5/1), which is too different. A better compromise is p = 0.5, where we see the ratio is √2/1 ~ 0.707, and ~0.1 on the shoulder.

The lower figure shows details of the raw discretized generatrix, the redistributed grid points, and the (normalized) curvature data that produced them with p = 0.5.

Automated Clustering Around Sharp Vertices

To the plain curvature-based gridding scheme, a sharp vertex is effectively invisible because it produces a 1-point spike in curvature too narrow to be resolved in the associated redistribution shape function, and the result is essentially uniform spacing in the vertex region where clustering is normally desirable. Such clustering has been automated in *CAPSULE_GRID*. The strategy is to detect the 1-point spikes heuristically (knowing that the geometry has been normalized), and then to broaden and shorten them with further heuristics. Left and right-sided shape functions (from earlier aerodynamic shape optimization work), possibly of different heights, are substituted for each spike without touching the discretized geometry. Any such vertex needs to be safely away from an end of the generatrix, to allow room for broadening of the associated curvature spike. Figures 15-16 show an example of automated broadening and moderation at three vertices.

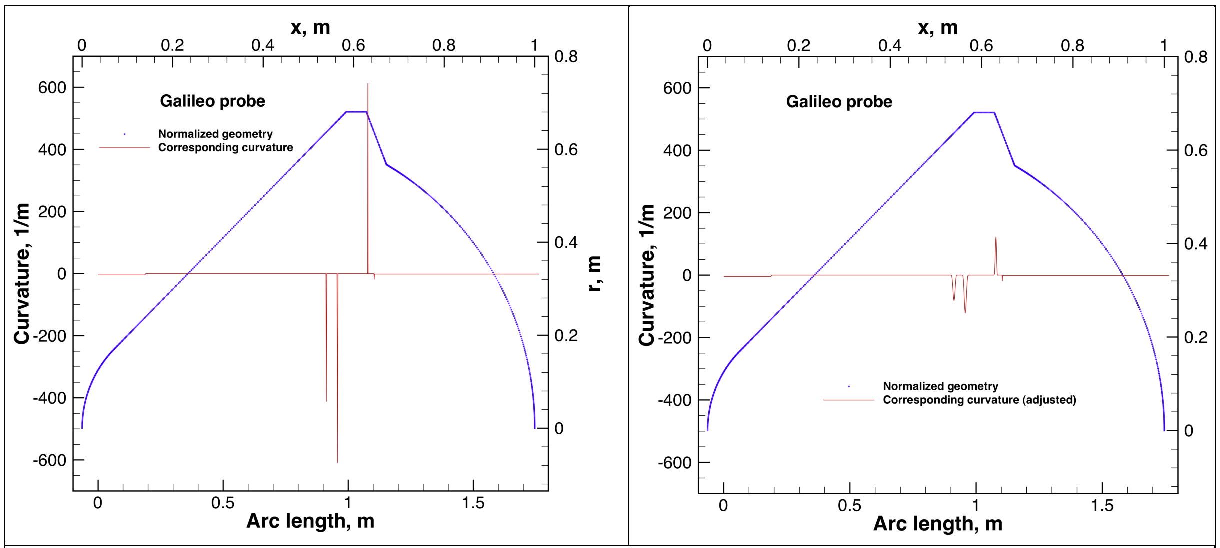


Figure 15: Three sharp vertices in this (blue) generatrix produce large curvature spikes (red, left) that have been automatically moderated in the right-hand figure. The `ng_split` control has been set to split the input generatrix at about half way between the 2nd & 3rd vertices, although `ng_split = -1` could also have been used to redistribute the generatrix as one piece, not two as shown here.

```
$FOREBODY_INPUTS
aft_body_too = T,
input_generatrix = 'HiResGeneratrix.dat',
numi_forebody = 201,
numj = 121,
nose_patch_file_name = 'NosePatch61B.p3da',
output_generatrix = 'gen.dat',
surface_grid_file_name = 'surface_grid.g',
power_fore = 0.4,
ni_regrid_forebody = 33,
$END

$AFT_BODY_INPUTS
numi_aft_body = 201,
ng_split = 349,
power_aft = 0.4,
ni_regrid_aft_body = 35,
$END
```

The essential controls are shown at left. The input generatrix has 599 points. Note that `power_fore` and `power_aft` are below the default of 0.5 as a further way to moderate the spikes.

Surface grids corresponding to both the spiky and the adjusted curvature distributions are shown on the next page.

This automated clustering towards sharp vertices is now standard.

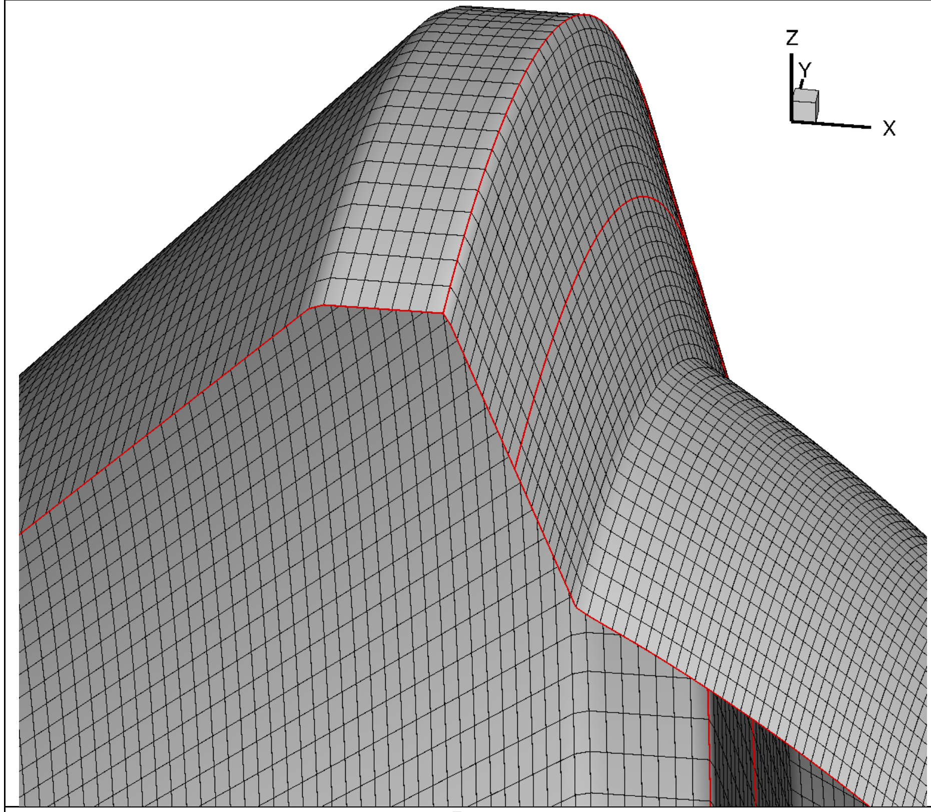


Figure 16a:

Plain curvature-based gridding across sharp vertices in this Galileo generatrix cannot see the 1-point curvature spikes, leading to essentially uniform spacing, which is normally not what is desired.

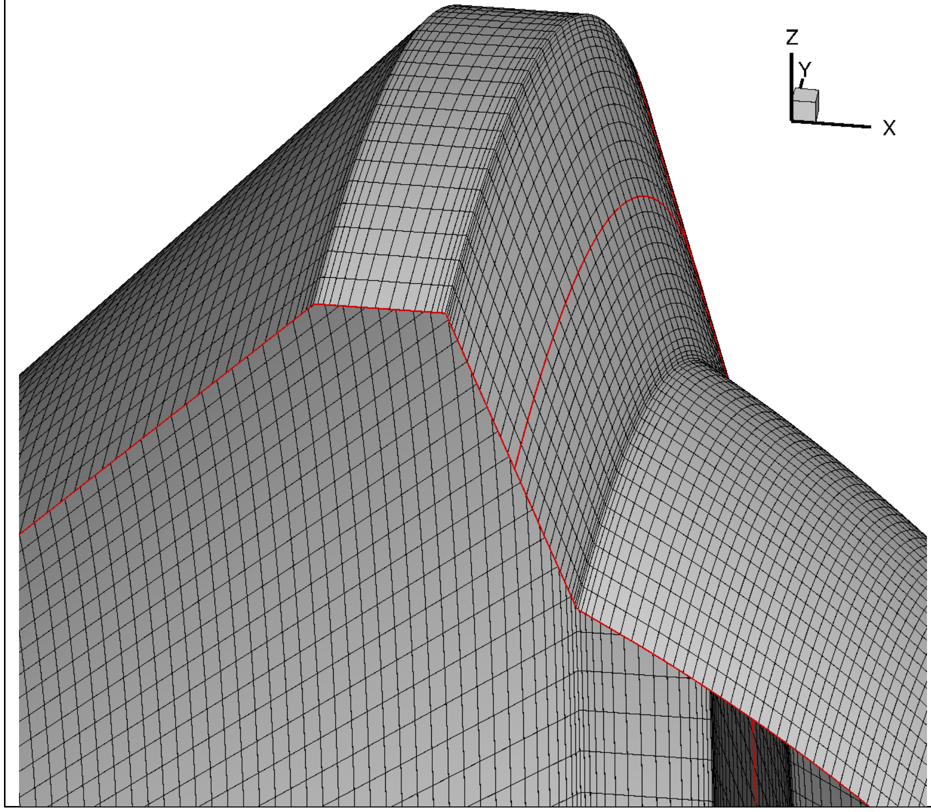


Figure 16b:

Automated vertex detection and artificial adjustment of the curvature distributions (fore and aft separately here) produces the point clustering that is normally preferred. Note where the full input generatrix has been split between vertices 2 and 3. The initial part of the aft portion of the gridded generatrix is also reblended to match spacing at the juncture.

Avoid such blending by entering instead `ng_split = -1` and perhaps adjusting `numi_forebody` and `numi_aft_body` (not shown).

Generatrix Cell-Spacing Growth Rate Distribution

As an aid to assessing grid quality, *CAPSULE_GRID* now writes the ancillary file **`generatrix-growth-rates.dat`** in this *TECPLOT*able format:

```
#      s    growth              x             y    i
0.00000  1.00000  0.0000000E+00  0.0000000E+00    1
0.00353  1.00000  2.7973901E-05  3.5257613E-03    2
0.00705  1.00000  1.1196602E-04  7.0506333E-03    3
0.01058  1.00000  2.5192920E-04  1.0573727E-02    4
 :        :        :              :               :
 :        :        :              :               :
1.62320  1.00000  9.2591010E-01  1.0438579E-02  398
1.62668  1.00000  9.2596004E-01  6.9592440E-03  399
1.63016  1.00000  9.2599002E-01  3.4796793E-03  400
1.63364  1.00000  9.2600000E-01  0.0000000E+00  401
```

The cell growth rate distribution for the above Galileo case with automated vertex capturing is shown below in Fig. 17. Such sharp features make staying within the desirable window of ~[`0.8`, `1.2`] difficult. More grid points or smaller **`power-`** controls could help. [Growth rate at point *i* in arc lengths $s_{1:n}$ is defined as $(s_{i+1} - s_i) / (s_i - s_{i-1})$, $i = 2{:}n\text{-}1$.]

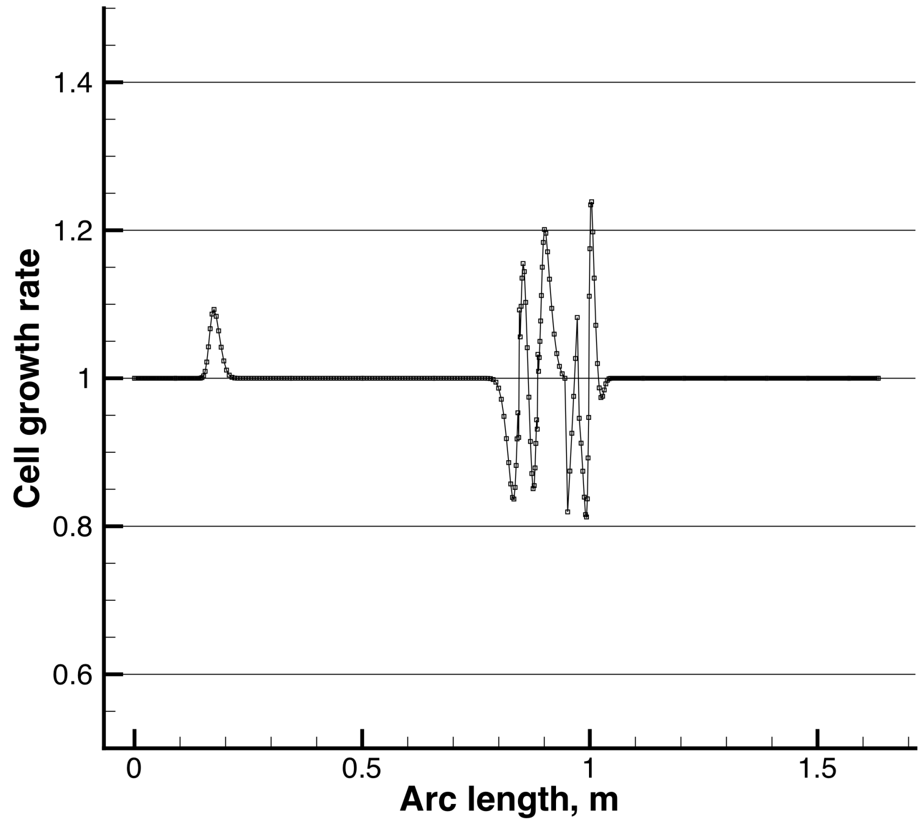


Figure 17: The cell growth rate distribution for the preceding Galileo generatrix shows that the surface grid, redistributed in two parts here, is still less than ideal. Sharp vertices are better avoided if possible. [Actually, redistributing to the same total number of points but as one part rather than two, via `ng_split = -1`, improves the result shown and avoids the blending of spacing across the split that is done otherwise. But converging with large numbers of points may not be possible.]

VII. Lessons Learned

(1) Explicit Aft-Body Vertex Rounding With Too Many Points

A pitfall has been observed when the surface grid density for a sting case (nominally about 65,000 x 161 volume points) was being doubled and doubled again. Equal scaling of the relevant integer inputs (`numi_fore_body`, `ni_regrid_forebody`, `numj`, and the similar aft-body controls) is not quite the right thing to do: the `ni_round` inputs are an exception. These control the number of points on each circular arc used to round each aft-body vertex before the resulting discretization of the generatrix is redistributed based on curvature. It turns out that too-large values for `ni_round` can overwhelm the smoothing of the plain curvature-based shape function employed, so the blending between the extremes of tight curvature in the filleted corners and no curvature between the vertices is not as smooth as expected. The Fig. 18 plots of intermediate quantities not normally displayed help explain why:

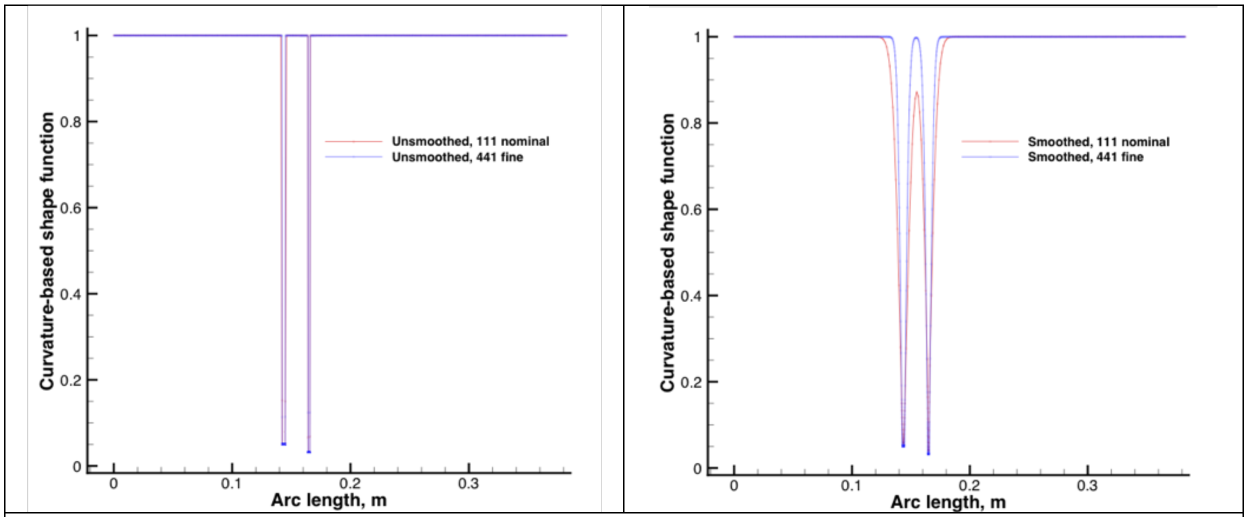


Figure 18: The data for these plots were extracted from the aft-body portion of the sting case shown on page 29. The plain shape function used to control spacing with respect to arc length *s* is *(1 + κ(s))*$^{-e}$, where the exponent is typically 0.5. Grid point spacing will be proportional to the shape function after it is smoothed, however many grid points are specified. On the left, before smoothing, the first segment with shape function value 1 corresponds to the zero-curvature aft cone. The first plunge corresponds to the aft body shoulder (high curvature => small spacing); the middle segment with value 1 corresponds to the vertical aft-body segment (zero curvature); the second plunge is from the filleted corner at the start of the sting (even tighter spacing), and the last segment with value 1 is from the sting itself. On the right, the results of smoothing are shown for both the nominal density and the 4x density. The relative spacing at the rounded vertices for both densities indicate some points in the corners will be as tightly spaced as they would be if the unsmoothed shape function had been used. The blending is also more abrupt for the fine (blue) case, which used 4 x 21 → 81 for the `ni_round` inputs. A preferable result for the fine grid is shown on the next page, obtained by specifying a more reasonable value of 31 for these inputs in the fine grid case, because the number of smoothing iterations is heuristically set at `numi_aft_body/8` or about 40 and 160 respectively (with a minimum of 5). 81 points at constant small shape value is simply too many for the smoothing of the fine grid spacing. [# smoothing iterations has since been retuned for normalized data. See p. 30.]

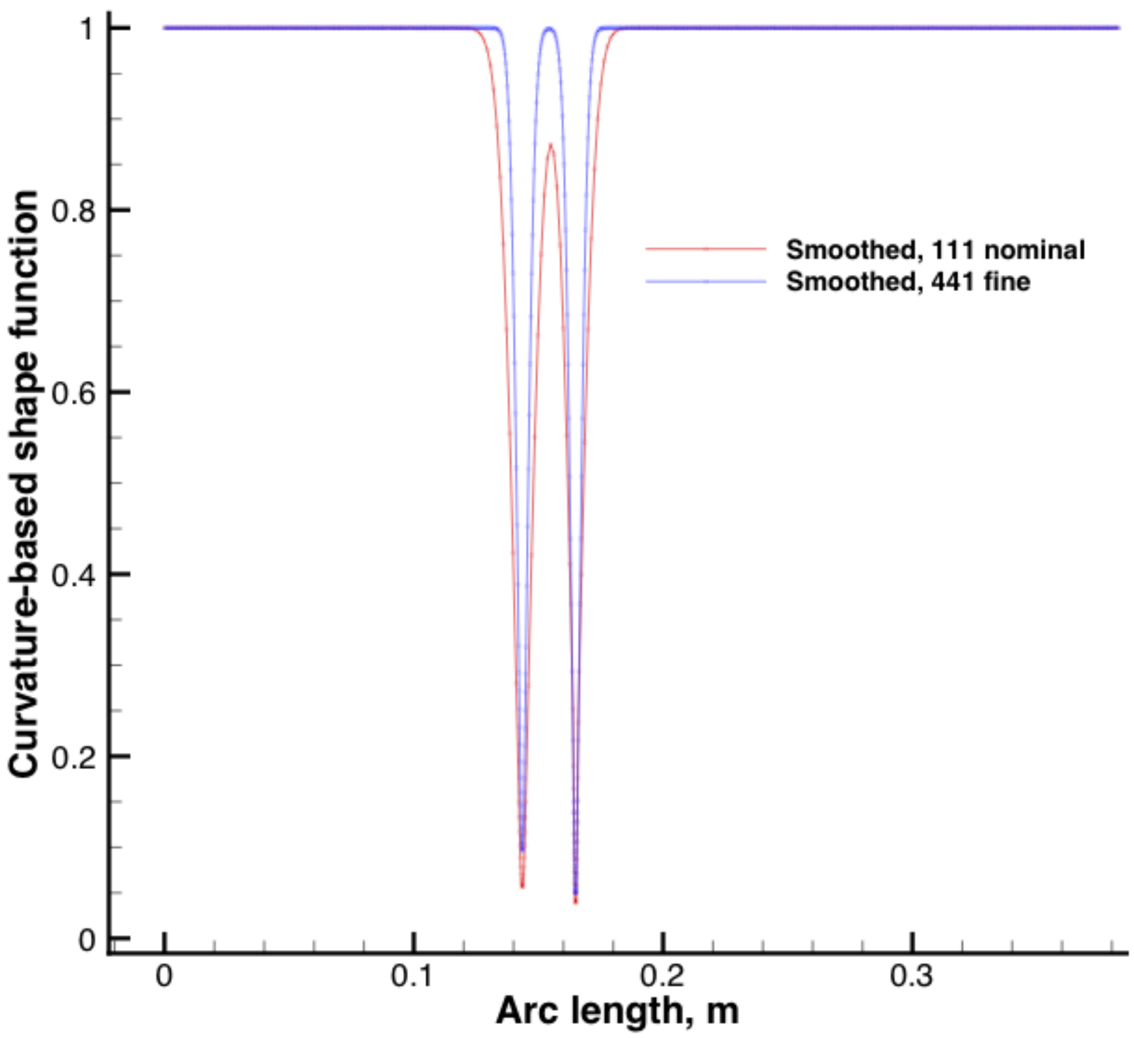


Figure 19: This smoothed curvature-based shape function for the above-mentioned fine grid used `ni_round = 31` instead of `81`, giving better blending of the final generatrix grid points at the rounded vertices as explained earlier. Observe that the blue shape function does not plunge so low as in the previous figure. [Note too that *CAPSULE_GRID* also specifies additional smoothing of the arc-length spacing after the initial curvature-based redistribution. But we digress.] The point being made in this lesson learned is to pay attention to the `ni_round` inputs. Count the number of points on each rounded vertex after a first try, then adjust these inputs to be roughly those counts. This gives the redistribution adequate geometry resolution without thwarting the smoothing needed to overcome the extremes of curvature that are typical of aft bodies.

Note that since this experience was documented, the curvature-based calculations have been revised to work with a normalized form of the geometry (larger data range transformed to [0,1]), which has the effect of making the resulting grid point distributions independent of the scaling and the units. See the **Notes on Curvature-Based Redistribution** on preceding pages, written subsequently. Less smoothing is now specified in the automated scheme. The above lesson learned is retained nevertheless to address the issue of rounding vertices: do not specify too many rounding points. See also the discussions of sharp vertices earlier and in the next paragraphs.

(2) **Sharp Vertices in the Generatrix**

While 2-D hyperbolic volume gridding generally produces tolerable results at sharp vertices in the generatrix, the *COMPRESS2D* step where the number of radial points and/or the wall spacing are adjusted can undo the smoothing inherent in the hyperbolic grid and give unacceptably angular slope discontinuities in the grid lines parallel to the surface.

This has prompted two further redistribution options in *COMPRESS2D* (other than constant wall spacing): (1) apply the same *relative* spacing (which may still be unacceptable unless very little compression of the upstream boundary is applied), and (2) preserve the initial wall spacing while changing the number of points and/or the stretching.

The best choice (given that modeling of sharp corners may not be essential anyway) is to avoid sharpness in the first place. All analytic aft-body vertices may be rounded explicitly if requested, while cusps in non-analytic input generatrices are now resolved as well as possible via automated clustering discussed earlier, and this helps the *COMPRESS2D* step as well.

(3) Short Shoulder Segment and Related Umbrella Issues

The ADEPT ("umbrella") variation illustrated in Fig. 11 has such a short initial aft-body segment that there is no room to redistribute that portion of the aft part of the generatrix to match the last spacing of the forebody portion. A skirt length of 0. initially contributed to the spacing mismatch, because the forebody distribution saw no zero-curvature segment aft of the shoulder circle that was terminated at the maximum diameter. A **`skirt_length`** of just `0.0015` (for a 6 meter diameter forebody) helped to blend the fore- and aft-body curvature-based redistributions of the initial discretizations. For a double-density form of this grid, **`skirt_length = 0.012`** was found to give an adequate blend. Trial and error is unavoidable.

The complete standard-resolution control file for this case is shown below. Note that **`numj = 181`** was a compromise between adequate overall resolution and resolution of the 2 in/0.05 m rib width: **`numj - 1`** had to be divisible by **`nedges = 12`**. With **`frustum_radius = 0.75`** (fairly long ribs), raising **`numj`** to **`193`** would have meant more of the inboard rib thickness was less than specified, and the nose patches would have needed to be larger (**`ni_regrid_forebody`**) exceeding **`frustum_radius`** and further affecting the spacing along the ribs.

```
$FOREBODY_INPUTS
aft_body_too = T,
input_generatrix = 'none',
spherecone = T,
numi_forebody = 209,
numj = 181,
x_nose = 0.,
r_nose = 0.,
radius_nose = 1.5,
radius_base = 3.0,
radius_shoulder = 0.09,
radius_vertex = 99.,
half_cone_angle = 70.,
half_cone_angle_fore = 99.,
half_cone_angle_aft = 99.,
radius_cone_juncture = 99.,
skirt_angle = 0.0,
skirt_length = 0.0015,
nose_patch_file_name = 'NosePatch61B.p3da',
output_generatrix = 'gen.dat',
surface_grid_file_name = 'surface_grid.g',
ni_regrid_forebody = 61,
umbrella_case = T,
nedges = 12,
peak_deflection = 0.0,
rib_deflection = 0.0,
frustum_radius = 0.75,
resolve_the_ridges = T,
rib_thickness = 0.05,
round_the_ridges = T,
$END

$AFT_BODY_INPUTS
ncones = 6,
r_aft = 0., 0.0435, 2.95, -0.0732, 1.5525, 1.1319865, 0.42926,
cone_angle_aft = 0., 90., 180., 110, 0., 57.4106,
rounding_mode = 2,
ds_round = 0., 0.0245, 0.0245, 0.04, 0.04, 0.04, 0.04,
ni_round = 0, 17, 17, 25, 17, 25, 25,
numi_aft_body = 260,
ni_regrid_aft_body = 20,
i0_umbrella = 105,
i1_umbrella = 135,
i2_umbrella = 195,
$END
```

Note that **`round_the_ridges = T`** now really means "flatten the rib ridges" (a simple switch from loose to tight local spline fits), meaning three points at each *i* station define the ribs along *j* lines. There is no way to put more than three points on the flat ribs (which appeared less of an issue with the original rounding). Inevitably, there are limits to this kind of automation, which should nevertheless serve for preliminary design studies.

(4) Two-Stage Gridding of Forebodies With No Skirt[#] or as a Full Body Initially

If a sphere-cone forebody grid terminates at the maximum diameter point on the shoulder, the 2-D hyperbolic volume gridding step tends to produce unsatisfactory outflow boundaries, no matter what the "splay" setting is in the **`2D-forebody.glf`** script. One simple work-around is to build the grid in two stages. [A second possibility is mentioned following Fig. 20.]

First, carry the shoulder round further by specifying a short skirt length and a sizeable skirt angle. Then the resulting hyperbolic volume grid contains radial lines off the shoulder near the desired cut-off that are much better suited for the no-skirt outflow boundary. Once such an augmented 2-D grid has been compressed and revolved, a 3-D volume grid for the no-skirt case at angle of attack is readily produced by imposing the no-skirt surface grid on the skirted 3-D volume grid with *RADIAL_INTERP*. Figure 20 illustrates symmetry-plane results for a small forebody case being modeled in the CUBRC shock tunnel. (If only ballistic entry is being simulated, the *RADIAL_INTERP_2D* analogue would serve the same purpose.)

[#] But see the N.B. on the next page.

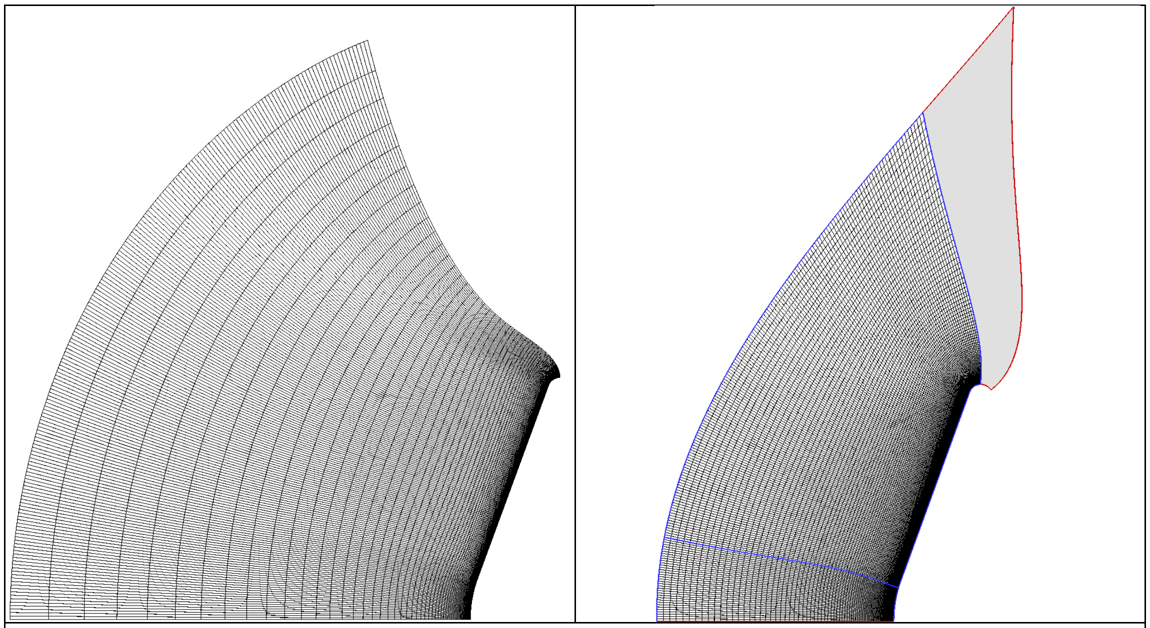


Figure 20: An automated *GRIDGEN* hyperbolic volume grid for a 70° sphere-cone forebody cut off at the maximum diameter (left, no aft skirt) has an unsatisfactory outflow boundary. A work-around is to include a fuller shoulder and short skirt initially (red, right), then impose the no-skirt surface grid on the resulting (compressed/revolved) volume grid to give the desired 3-D volume grid (blue, shown with lower half omitted). The grid is for simulating flow on an 8-inch diameter model in the CUBRC shock tunnel at angle of attack.

```
$FOREBODY_INPUTS
aft_body_too = F,
units_in_inches = T,
spherecone = T,
numi_forebody = 216,
numj = 145,
x_nose = 0.,
r_nose = 0.,
radius_nose = 2.0,
radius_base = 4.0,
radius_shoulder = 0.2,
skirt_angle = 50.,
skirt_length = 0.02,
nose_patch_file_name = 'Nose ...’,
output_generatrix = 'gen.dat',
surface_grid_file_name = 'surf...’,
ni_regrid_forebody = 35,
$END
```

```
$FOREBODY_INPUTS
aft_body_too = F,
units_in_inches = T,
spherecone = T,
numi_forebody = 193,
numj = 145,
x_nose = 0.,
r_nose = 0.,
radius_nose = 2.0,
radius_base = 4.0,
radius_shoulder = 0.2,
skirt_angle = 0.,
skirt_length = 0.0,
nose_patch_file_name = 'Nose ...,
output_generatrix = 'gen.dat',
surface_grid_file_name = 'surf...’,
ni_regrid_forebody = 35,
$END
```

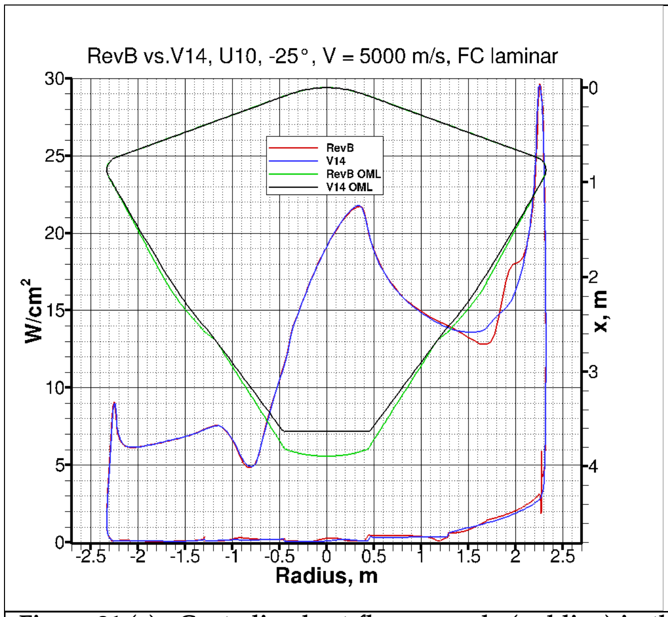
(b) *CAPSULE_GRID* control files for two-stage gridding of the above no-skirt case are shown as for running in different directories. Changing the output names would allow both stages to be performed in the same directory. The right-hand controls produce the 3-D surface grid that can be imposed on on the revolved hyperbolic grid resulting from the left-hand controls.

N.B.: Terminating a forebody grid at the maximum diameter is normally not recommended, in case radiation lines of sight off the shoulder are desired later. The red grid above is therefore normally preferable to the blue grid. [A more serious issue to beware of with forebody grids ending too abruptly is that the "sonic bubble" in the flow solution may not close, violating the supersonic outflow boundary condition.]

Alternative Forebody Grid Option: Add any reasonable aft body to the generatrix, build a full-body hyperbolic volume grid (less troublesome), and use COMPRESS2D to split it such that block 1 can be extracted and used for the forebody.

(5) Effect of Skewed Radial lines in the Hyperbolic Volume Grid

When the aerothermal database for a near-definitive RevB configuration of the Mars SRL (Sample Retrieval Lander) capsule was being calculated, solutions at the highest of three angles of attack (-25°) were marred by persistent heat flux anomalies in the vicinity of the stagnation point. Most noticeable as "wiggles" in the centerline heat flux distribution, these anomalies were much less prevalent at the same flight conditions for an earlier V14 geometry with the same 70° sphere-cone forebody. Another symptom observed was the abnormally poor alignments of the outer grid boundaries with the bow shocks, in spite of careful use of standard practices. Indeed, perceptible indentations were observed in the aligned outer boundaries opposite the stagnation regions. See Fig. 21.



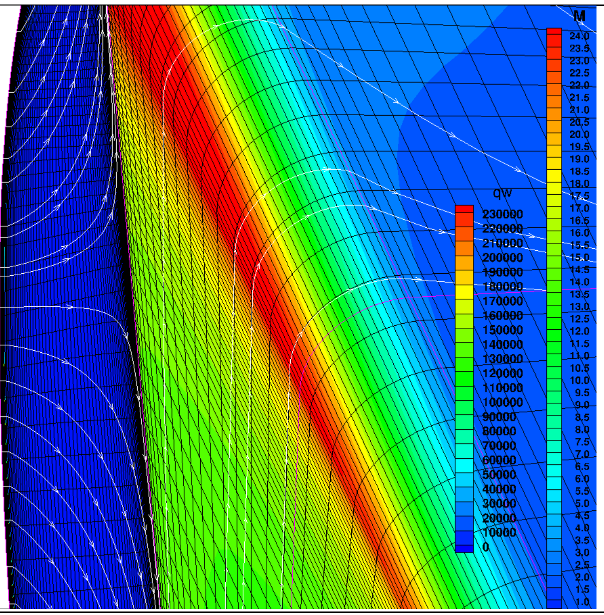


Figure 21 (a): Centerline heat flux anomaly (red line) in the stagnation region typical of what was obtained for the SRL RevB geometry at -25° angle of attack but not for the earlier V14 geometry with the same forebody.

(b): An indentation in the outer grid boundary is (barely) perceptible opposite the stagnation region in this pitch-plane plot of the shock-aligned solution for SRL Rev B at -25° at the same U10 flight condition. Note the skewed radial grid lines in the same region.

The most likely cause was suspected to be the point distribution on the conical flank towards the shoulder. Experience shows the advisability of spreading point clustering on the shoulder well forward onto the outer portion of the conical part of the generatrix. Indeed, the RevB generatrix proved inferior to that of V14 in this respect, so the cone portion of the RevB generatrix was redistributed to increase the clustering towards the shoulder. Initially, at the condition chosen for further investigation, the corresponding aligned volume grid was derived from the existing aligned volume grid and the improved surface grid using the *RADIAL_INTERP* technique. The resulting flow solution proved no better, and an extra shock alignment step did not help either (not shown).

Colleague Kaelan Hansson next wondered if the aft-facing step at the shoulder seal (not present in the V14 generatrix) might be adversely affecting the hyperbolic volume grid produced by *HYPGEN*. Redoing the volume gridding on the improved RevB generatrix did reveal the root cause of the heat flux anomaly: not the shoulder seal step, but *skewed radial grid lines* off the surface in the stagnation region at the -25° condition. The culprit is really the smoothing step(s) considered essential as part of the shock alignment process. This is easier to explain for the double smoothing option within *DPLR*. [Arc length *changes* (only) were originally smoothed because smoothing of full radial grid line arc lengths was deemed likely to introduce surface geometry artifacts into the aligned outer grid boundary. After smoothing of both full arc lengths and arc length changes were made options in *DPLR*, it became generally accepted that smoothing of both (double smoothing) produced the best results.]. It is now understood that if the initial radial grid lines produced by *HYPGEN* are not essentially perpendicular to the heat shield surface, the skewed lines will have longer arc lengths than if they were not skewed. Those lengths are therefore shortened more than they should be during the smoothing step (and the same reasoning applies to smoothing of arc length changes). This explains the indentation seen in the aligned outer boundary of the volume grid, and the resulting flow disturbance at the shock travels to the surface, producing disturbances in the heat flux distribution (stagnation region wiggles in the centerline plot, and a less obvious effect on the surface heat flux contours).

Note that initial use of *RADIAL_INTERP* to produce a volume grid matching the revised generatrix and surface grid actually thwarted the surface redistribution because that technique explicitly preserves the radial grid lines of the original volume grid.

Results from the less-skewed volume grid after realigning from scratch are shown in Fig. 22.

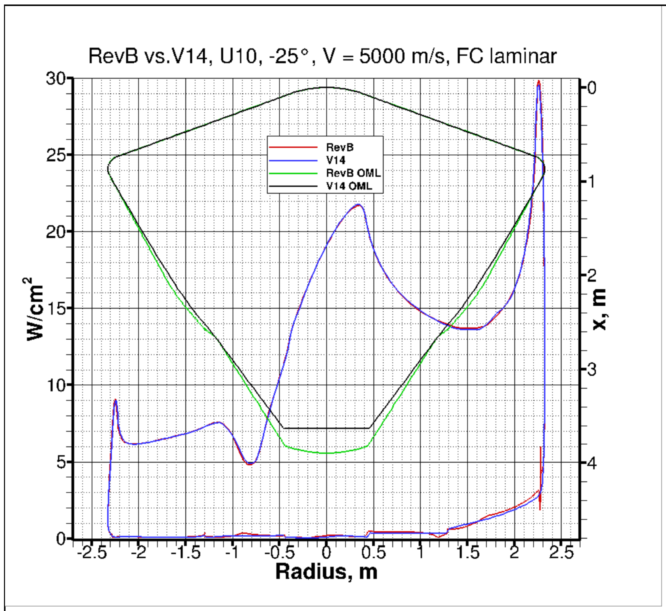
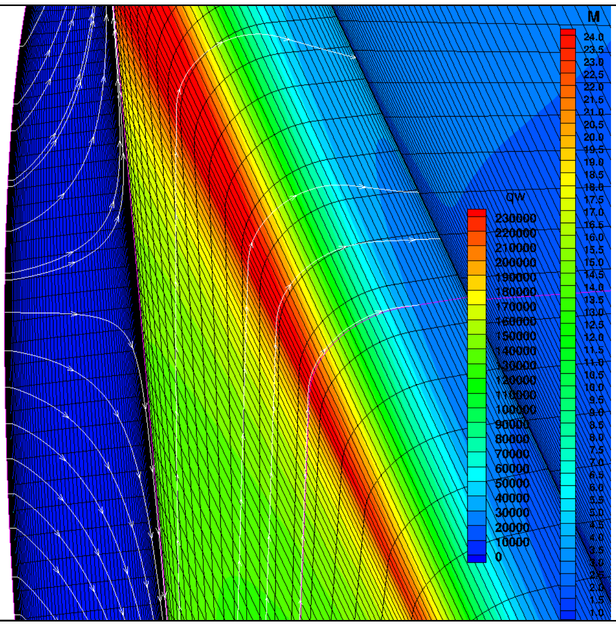
	

Figure 22 (a): Centerline heat flux with no anomaly in the stagnation region following redistribution of the heat shield generatrix, recalculation of the hyperbolic volume grid, and realignment from scratch. SRL RevB and V14 configurations at -25° angle of attack.	**(b): Improved SRL RevB surface and volume grid: the forebody radial grid lines are essentially orthogonal throughout, and the shock boundary is smoothly convex as it should be.**

The lesson learned here is to pay careful attention to possibly skewed radial grid lines in the hyperbolic volume grid that can affect shock alignment and hence produce surface heat flux anomalies.

VIII. Long Wake Procedures

Real gas flow calculations on idealized hypervelocity asteroids prompted a means of extending the computational grid for the wake much further than the standard procedures are intended for (with a single layer of grid blocks around the body). A longer wake is desirable for the associated radiation calculations. The *CAPSULE_GRID* directory should contain **`/Long-Wakes`** with ancillary utilities *EXTEND_WAKE*, *ALIGN_WAKE*, and *PRESERVE_-BODY_FLOW*. *EXTEND_WAKE* converts a standard 2-block aligned grid to a 5-block topology that enables the wake to be extended arbitrarily via linear extrapolation of the wake outer boundary. After the grid has been extended, the flowfield on the new topology can be initialized with *FLOW_INTERP_2D* (on cell-centered state variable data) then reconverged. That extended grid can be improved with *ALIGN_WAKE*, which reads the grid along with a (vertex-centered) function file containing Mach number. The shock boundary in the wake block 5 of this grid is improved by estimating the shock edge and fitting a parabola to it. This process can also be repeated following reconvergence, in which case the wake may be stretched even further in conjunction with *FLOW_INTERP_2D*. The converged flow solution of blocks 1-4 can optionally be preserved precisely with *PRESERVE_BODY_FLOW* to avoid unnecessary interpolation errors. Figure 23 illustrates application of the above procedures to an initial 2-block flow solution for the Hayabusa 2 sample return capsule, producing a 10-diameter wake (which the standard 2-block topology actually handles reasonably well. But asteroids are more extreme, and 50-diameter wakes have been produced this way).

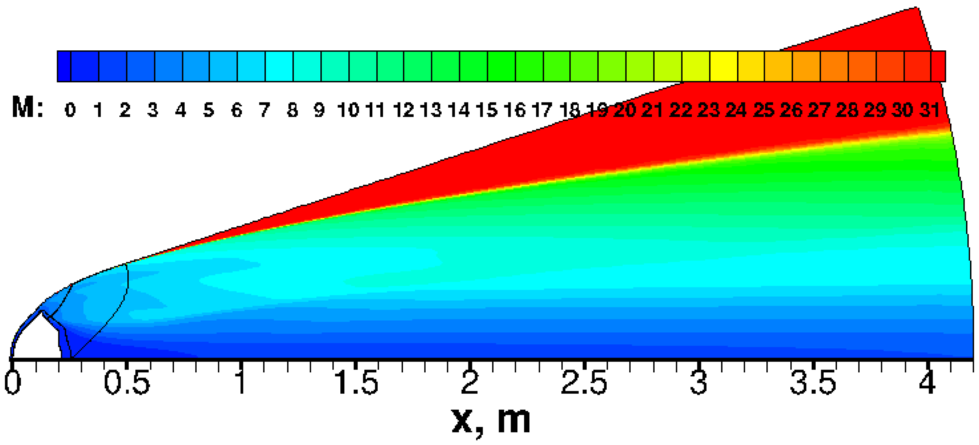
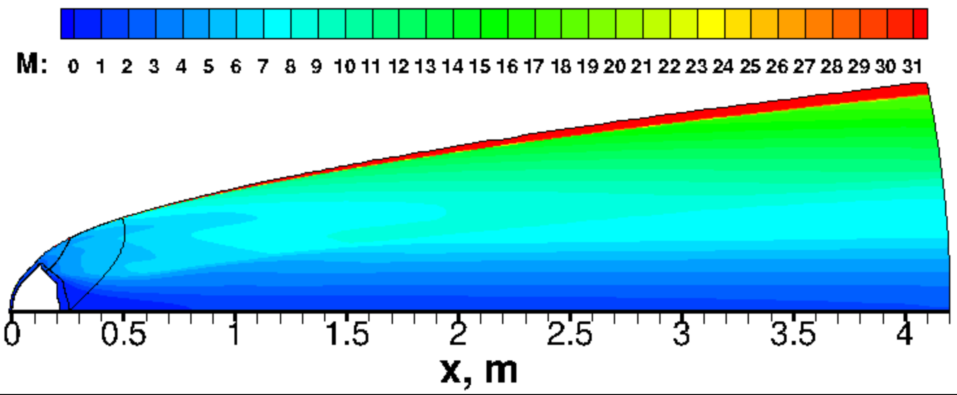


Figure 23. (a) *EXTEND_WAKE* result. Blocks 1 and 3 are in the boundary layer; blocks 2 and 4 are outboard of those, and block 5 can be extended indefinitely. *FLOW_INTERP_2D* initialized the flowfield for *DPLR2D*.



(b) *ALIGN_WAKE* has improved the wake boundary. *FLOW_INTERP_2D* initialized the flowfield; *PRESERVE_BODY_FLOW* (optionally) transferred the flow for blocks 1-4 in Fig. 21a to the (identical) blocks 1-4 here before running *DPLR2D*.

IX. Splitting Surface Patches at a Parachute Cone Juncture

Configuration studies for the Mars Sample Return Lander mission required a certain (noncatalytic) boundary condition (BC) on the aft body parachute cone and a supercatalytic or fully catalytic BC on the remainder of the backshell. A further common requirement is to allow for coarsening the surface grid twice: one coarsening of an initial fine grid by a factor of 2 in the surface index directions for production flow calculations, and a second coarsening ("down-sequenced" grid) to speed an initial unaligned flow calculation from scratch. The surface patch dimensions all need to be of the form *4n+1* to enable these two levels of coarsening. Trial-and-error runs with adjusted generatrix point counts and tweaked `ni_regrid_aft_body` inputs have proved problematic. This situation prompted a *CAPSULE_GRID* option to place an aft surface patch split at an exact *x* location (`x_parachute_cone` > 0. input) with point counts either side adjusted to the form *4n+1* with the relative point spacing of the interim aft blocks 7-10 preserved as well as possible in the output 16 blocks of the split surface grid. Figure 24 shows a representative example.

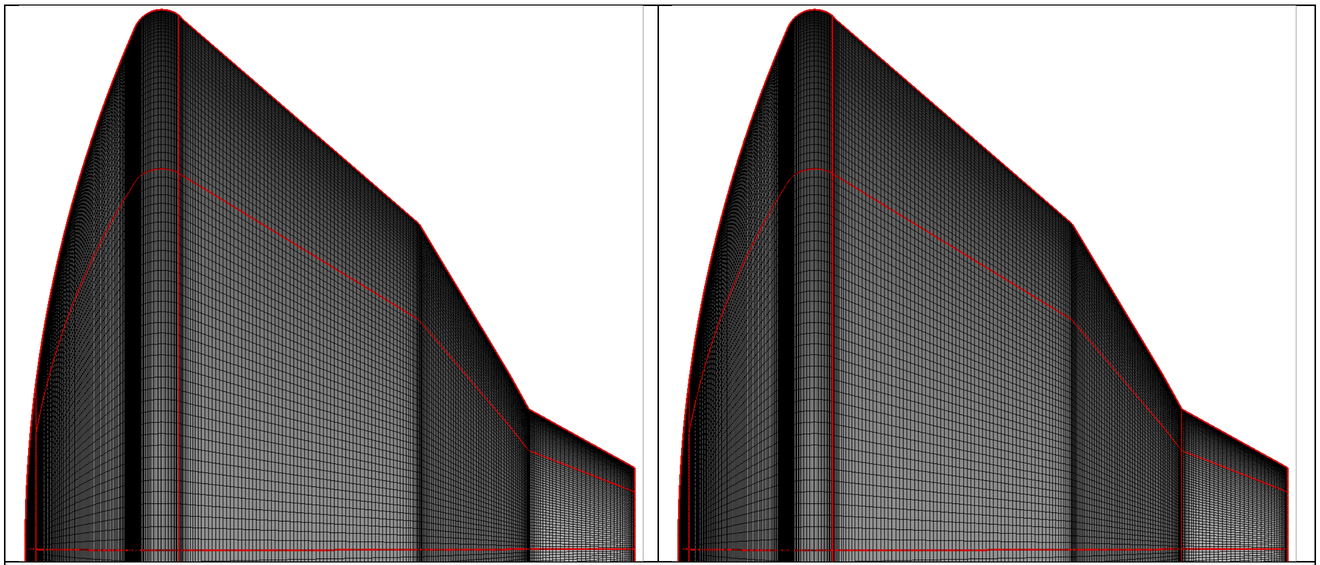


Figure 24: Option to split aft surface patches 7-10 of 12 (left) at a specified axial location (`x_parachute_cone > 0.`) to form 16 patches (right). The split patches have dimensions of the form 4n+1 for grid coarsening purposes, while capturing the specified x exactly. This automated splitting facilitates use of a different flow solver boundary condition on the parachute cone (typically noncatalytic).

X. Asymmetric Forebody Option

One configuration considered for the Mars Sample Return Lander in 2021 was defined by an upper generatrix that bulged forward starting from an above-center nose, along with a traditional 70° sphere/cone-type lower generatrix. Blending of the two cross-sections by means of linear combinations weighted according to azimuthal angle was initially implemented with a new blending subroutine, an ad hoc driving program, and reuse of the long-idle *FOREBODY_REGRID* utility. This capability is now an **`asymmetric = T`** option in *CAPSULE_GRID*. It has been installed mostly as one extra step following generation of the symmetric result that is needed if an aft body is present. The interim forebody surface is overwritten with the blended surface. Additionally, the spoked forms of the fore- and aft body are written as a single surface patch, as required by a 3D form of the *HYPGEN* script.

Figure 25 shows the geometry that prompted the **asymmetric = T** option. The strategy is to run *CAPSULE_GRID* in axisymmetric mode first with the upper generatrix in order to obtain a gridded form of that generatrix. A second run uses this gridded form exactly by means of precise choices for **numi_forebody**, **numi_aft_body**, and **ng_split**, along with input **lower_generatrix = '...'**. The forward portion of the lower section directly below points `1 to` **ng_split** of the upper section is automatically interpolated to the upper x coordinates in order to facilitate the linear combinations of vertical distances from the nose that constitute the initial blending (weighted according to spoke azimuthal angles at 1° intervals). Matching the aft body that is assumed to be symmetric is achieved by morphing each spoke so that its outermost point lies exactly on the appropriate perfect circle centered on Ox.

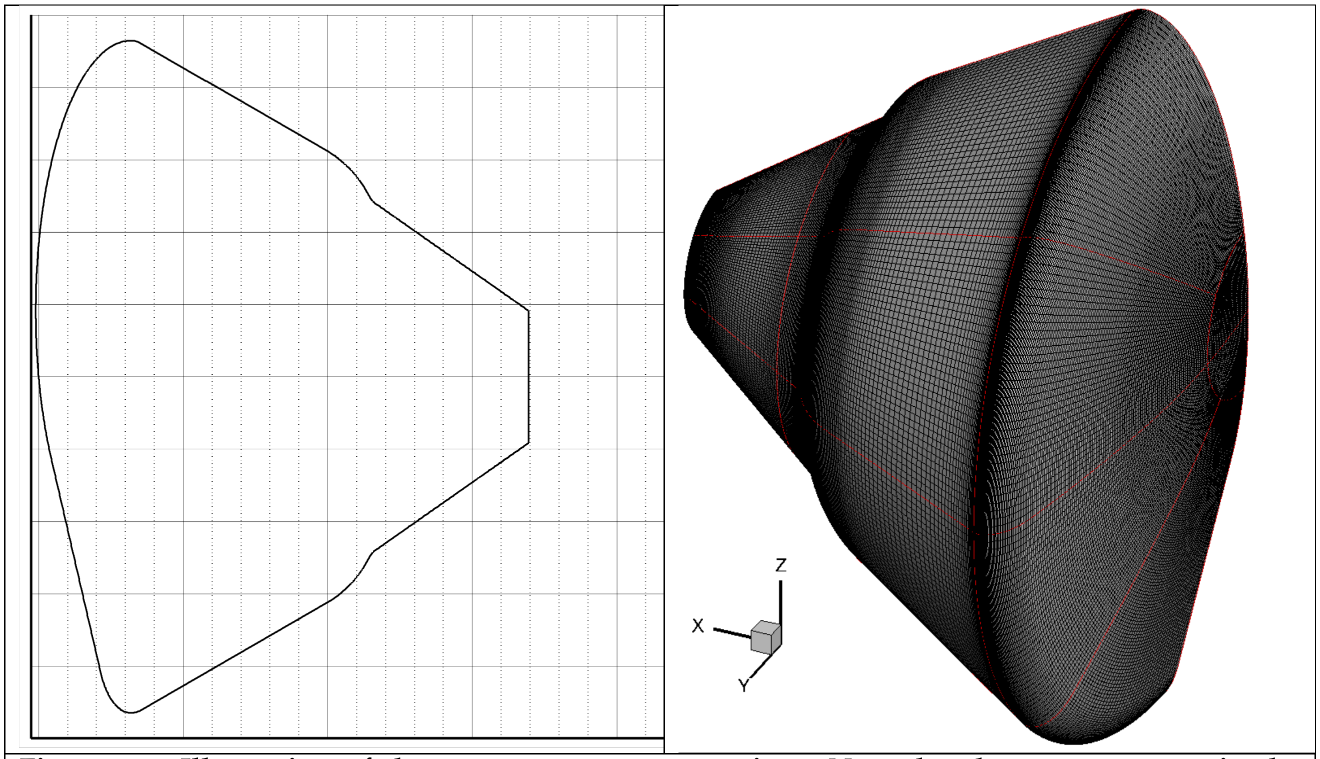


Figure 25: Illustration of the asymmetric = T option. Note the above-center nose in the cross-section. The aft-body is assumed to be axisymmetric. A single-patch spoked form of the surface grid is written as spoked_surface.gu, from which *HYPGEN* produces a single-block volume grid. The patched grid on the right is then imposed on the volume grid via *RADIAL_INTERP*.

The control file below should be used twice, first with **asymmetric = F** and a suitably dense form of **input_generatrix** (specially near the sharp corner) in order to produce the desired gridded form of the upper cross-section. The gridded **output_generatrix** would be **'gen-sym.dat'**. Then a second run with the shown controls produces the asymmetric result shown. Here, **output_generatrix** is not meaningful and so it is not written. The single-patch spoked form of the surface to be read by *HYPGEN* has the hard-coded name **spoked_surface.gu**.

```
$FOREBODY_INPUTS
verbose = F,
asymmetric = T,
aft_body_too = T,
units_in_inches = F,
geometric_scale = 1.,
input_generatrix = 'gen-sym.dat',
lower_generatrix = 'v21-lower.dat',
spherecone = T,
numi_forebody = 201,
numj = 145,
nose_patch_file_name = 'NosePatch61B.p3da',
output_generatrix = 'gen-full.dat',
surface_grid_file_name = 'split_grid.g',
flat_blend_fore = T,
power_fore = 0.5,
ni_regrid_forebody = 37,
$END

$AFT_BODY_INPUTS
flat_blend_aft = T,
power_aft = 0.7,
ng_split = 177,
nblend_fore_aft = 50,
numi_aft_body = 245,
ni_regrid_aft_body = 22,
x_parachute_cone = 2.525,
$END
```

XI. Concluding Remarks

The author's most recent gridding experience has involved steps at the shoulder seal that defeat use of curvature-based redistribution. The reason is that the top of a step needs to be the `ng_split` index for BC reasons, and there tends to be little or no curvature forward of the step. This means poor clustering where it is needed to resolve the step corner. The same is true at the bottom of the step, where straight-line segments are likely either side of a corner. The work-around is to cluster the generatrix as needed for a computational surface grid, then use *CAPSULE_GRID*'s option to preserve that input generatrix exactly by specifying `numi_forebody = ng_split` and `numi_forebody + numi_aft_body - 1 =` number of input generatrix points. A good choice for fine-grid resolution of a sharp corner is `1.e-6` m spacing either side of a corner (including, say, at a parachute lid corner).

The *REDISTRIBUTE_XY* utility applied multiple times to appropriate partial line segments is one way to achieve the desired rediscretization of a given generatrix. A second utility, *MORPH_LINE_SEGMENT*, has also been implemented to facilitate imposing the relative point distribution of an existing line segment on to another line segment, with the option to change the point count.

More recently, a table-driven scheme has been implemented to automate turning a CAD-based generatrix into surface grid form all at once instead of one line segment at a time. A utility named *GEN_TO_GRID* can discretize the line segments connecting critical points all at the same time, and a second iteration after a reasonable guess at good control inputs will normally suffice to produce the point counts and end-point spacings needed in a gridded form of the generatrix that *CAPSULE_GRID* can then preserve as described above. *GEN_TO_GRID* is appropriate when multiple features in the geometry such as shoulder seal steps and parachute cone block splits and lid corners need to be captured precisely. The control file looks this (entered as the first command line argument—not stdin):

```
kermit-xyz-meters.dat
-1 = high-end clustering (ds2); 0 = uniform; 1 = low-end clustering (ds1);
 2 = 2-sided clustering (ds1 and ds2);
 3 = curvature-based redistribution; ds1, ds2 = power, ismooth = 0.5, -3 say
 4 = redistribute with same relative distrib. as the following file(s).
RevB.111.fore.dat
  x                 y                 type     npts     ds1     ds2
0.0               0.0                    4      321      99      99
0.9967            2.28519                0       13   0.001   0.001
0.9967            2.273786485            2      169   0.001   0.004
2.576036319133804  1.292507132000739     2      105   0.004   0.002
3.475056007817630  0.4412387103716517    2       39   0.002   0.004
3.553832557629430  0.05508164159823618  0       15      99      99
3.555000000000000  0.000000000000000    99       99      99      99
```

Line 1 is the dataset consisting of two or three x, y[, $z = 0.$] columns defining the geometry precisely (probably discretized by a CAD system). Following the type 4 header line should be as many files as type 4 choices, and those names need to be cut and pasted.

Following further descriptive header lines, the table needs to be numeric and complete, so unused entries on the last line are entered as 99 to indicate "irrelevant." These critical points and the details of how to discretize the line segments between them are expected to be ordered nose to tail.

A consequence of the option to split the initial aft grid blocks at an *x* where a parachute cone starts (for catalytic BC reasons) is that point counts on either side of the split need to be of the form *4n+1* for grid sequencing reasons. This should be kept in mind if any of the above ancillary utilities are being used. Otherwise, *CAPSULE_GRID* may perform local spline redistribution, and any careful preceding adjustment of point counts (possibly to match the counts of an existing generatrix) may be undone unless those adjustments are mindful of a desired `x_parachute_cone`. The lesson learned is that it may be folly to match point counts to a preceding generatrix, because a parachute cone split can undo the matching with no easy work-around. A very first unaligned solution on a new geometry may not be able to use a saved unaligned solution from a previous geometry, in which case either grid sequencing or brute force interpolation of the state variables is the unavoidable choice. Do make a practice of saving an unaligned solution in case it can be used either to start a nearby condition or to start a solution on another geometry.

Also, always be sure to converge an unaligned solution well enough for the wake part of the shock to be fully developed. This may still be difficult or impossible at very low Mach numbers, meaning an initial alignment with *DPLR* may move parts of the downstream grid boundary that should stay untouched. This is a difficult tolerance issue, and it can be considered more cosmetic than consequential.

One final tip: be sure to cluster adequately on the outboard end of a conical flank near the shoulder. *CAPSULE_GRID* will normally do this, but not necessarily if an exact input generatrix is to be retained. Higher than expected shoulder heat flux spikes can be explained by inadequate clustering towards the shoulder and consequent increased dissipation.

Figure 24 illustrates some of the issues with a step at the shoulder seal location. Splitting the fore- and aft bodies at the top of the step means curvature-based redistribution does not see the sharp corner, so a properly discretized generatrix should be produced one way or another (such as with the ancillary utilities mentioned above) and employed exactly.

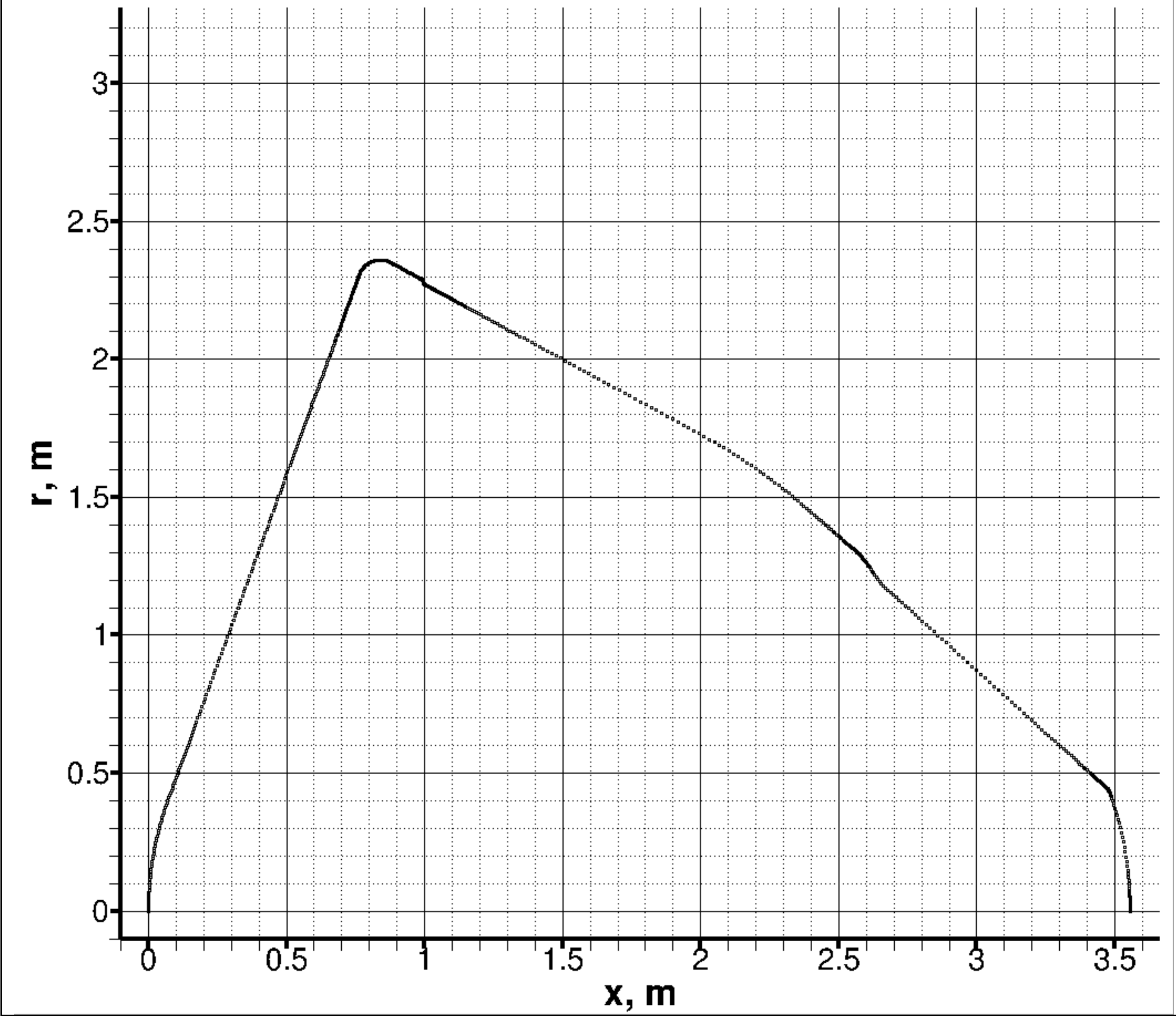


Figure 24: Illustration of good discretization of a generatrix to be preserved exactly. The resulting fine grid (`1.e-3` spacing at sharp corners) is suited to coarsening by a factor of 2 for production-level flow calculations, as well as further coarsening to speed an initial unaligned solution, even if a parachute cone split has been specified. See Lessons Learned (5) on pp. 40-41 about adjusting the forebody discretization to avoid possible skewing of the radial lines in the hyperbolic volume grid.

XII. Acknowledgments

CAPSULE_GRID and its supporting utilities were developed under multiple contracts with the Aerothermodynamics Branch at NASA Ames Research Center, through ELORET, Inc., ERC, Inc., and AMA, Inc. The most recent contract number is NNA15BB15C (ESTRAD). Some of the underlying numerical techniques were implemented under contracts with the Applied Aerodynamics Branch through Sterling Software, Inc., pre-1999. The support of these Branches and contractors over a decades-long career at Ames is gratefully acknowledged by the author. Hyperbolic volume gridding scripts by Todd White (Entry Systems and Vehicle Development Branch at Ames) represent a crucial contribution, and Todd's enthusiastic moral support has been much appreciated. Note that these scripts are for *GRIDGEN* which has been superseded by *POINTWISE*, but at the time of writing, no script translations have been performed. Instead, NASA's *HYPGEN* from the Chimera Grid Tools package has filled the void, and advice from its coauthor William Chan is thankfully acknowledged. For all things aerothermodynamic, Dr. Dinesh Prabhu (AMA, Inc.) has been and continues to be a tower of strength within the Ames Entry Systems and Technology Division. No one else has calculated more capsule flow solutions more knowledgeably than Dinesh, so if a dedicatee for this software is appropriate, he is the most deserving.

XIII. References

1. *GRIDGEN*: Commercial grid generation software superseded in 2016 by *POINTWISE*. Pointwise, Inc., 213 South Jennings Avenue Fort Worth, TX 76104.
2. *HYPGEN*: Hyperbolic grid generator, NASA Technical Memorandum 108791, William Chan, Ing-Tsau Chiu, and Pieter G. Buning, NASA Ames Research Center, 1993. Also: https://www.nas.nasa.gov/publications/software/docs/chimera/index.html
3. *DPLR*: Real gas flow solver, https://software.nasa.gov/software/ARC-16021-1A.
4. ADEPT: Adaptable Deployable Entry and Placement Technology, https://gameon.nasa.gov/projects-2/deployable-aeroshell-concepts-and-flexible-tps/.
5. Dr. James L. Brown, NASA Ames Research Center: Elliptic quarter circle grid file via private communication.