

Sketch-to-Solution: A Case Study in RCS Aerodynamic Interaction

Bil Kleb,^{*} Mark Schoenenberger,[†] Ashley M. Korzun,[‡] Michael A. Park,[§]
NASA Langley Research Center, Hampton, VA 23681, USA

Thanks to recent advances in the fields of anisotropic grid adaptation, error estimation, and geometry modeling, a sketch-to-solution work flow is now possible for viscous computational fluid dynamic (CFD) simulations. With this workflow, a CFD application engineer provides geometry, boundary conditions, and flow parameters; and the sketch-to-solution process yields a CFD simulation through automatic, error-based, grid adaptation.

To explore the benefits of this nascent capability, a conventional manual grid generation work flow is compared to this new automatic grid generation work flow for a given engineering question: What are the aerodynamic interactions caused by the reaction control system (RCS) on an entry vehicle? This case study indicates that while the automatic grid generation sketch-to-solution process is not yet mature, it is preferred over a manual grid generation work flow because it greatly reduces manual labor, eliminates many opportunities for human error, and provides grid sensitivity information.

Nomenclature

α	Angle of attack, deg.	$\mathcal{M}$	Manual grid generation workflow
$\mathcal{A}$	Automatic grid generation workflow	N	Number of nodes
C_D	Drag coefficient	N_E	Number of elements
C_{m_x}	Rolling moment coefficient	N_{Pr}	Number of prisms
C_{m_y}	Pitching moment coefficient	NPR	Nozzle pressure ratio (total/static)
C_{m_z}	Yawing moment coefficient	N_{Py}	Number of pyramids
C_X	Axial force coefficient	N_T	Number of tetrahedra
C_Y	Side force coefficient	NTR	Nozzle temperature ratio (total/static)
C_Z	Normal force coefficient	r_0	Initial stretching rate
f	Grid refinement factor	Re	Unit Reynolds number, M/ft
h_0	Wall-normal spacing, inches	$r_{\max}$	Maximum stretching rate
M	Mach number	T_0	Total temperature, °F

I. Introduction

DESPITE its success, viscous Computational Fluid Dynamics (CFD) has not reached the level of automation desired by practitioners, primarily due to the lack of robust, efficient grid generation and error-based grid adaptation. As the CFD Vision 2030 Study [1] states,

[U]ltimately the mesh generation process should be invisible to the CFD user or engineer. Given a suitable geometry representation and a desired level of solution accuracy, a fully automated meshing capability would construct a suitable mesh and adaptively refine this mesh throughout the solution process with minimal user intervention until the final accuracy levels are met. This process allows the user to focus on the final solution without concern for the construction and maintenance of the underlying mesh.

^{*}NASA Senior Researcher for Computational Aerothermodynamics, AIAA Associate Fellow.

[†]Research Engineer, Atmospheric Flight and Entry Systems Branch, AIAA Senior Member.

[‡]Research Engineer, Atmospheric Flight and Entry Systems Branch, AIAA Senior Member.

[§]Research Scientist, Computational AeroSciences Branch, AIAA Associate Fellow.

Meanwhile, Alauzet and Loseille [2] and Park et al. [3] summarize progress and future areas of research for solution-adaptive methods that includes the grid anisotropy necessary to resolve boundary layers; and an informal Unstructured Grid Adaptation Working Group (UGAWG) has been working to mature grid adaptation technology [4–8]. And as was shown by recent work in this area [9–13], the community is making great strides toward the goal of making the grids “invisible” for viscous CFD.

This paper seeks to take a given engineering question and see how well this nascent sketch-to-solution process performs compared to a conventional approach with manual grid generation.

II. Engineering Question

The engineering question for this case study is, “What are the aerodynamic interaction effects of reaction control system (RCS) jets for one possible configuration of the Mars Sample Return Lander (MSRL) [14] vehicle?” In this particular case study, a proposed wind tunnel entry of the "Option A(N1)" configuration with conical thrusters located near the heatshield is considered. This configuration is shown in Figure 1, and the conditions considered are given in Table 1 for which we consider the case for an RCS roll command, i.e., four thrusters (two pairs) firing to spin the capsule about the vehicle’s longitudinal axis.

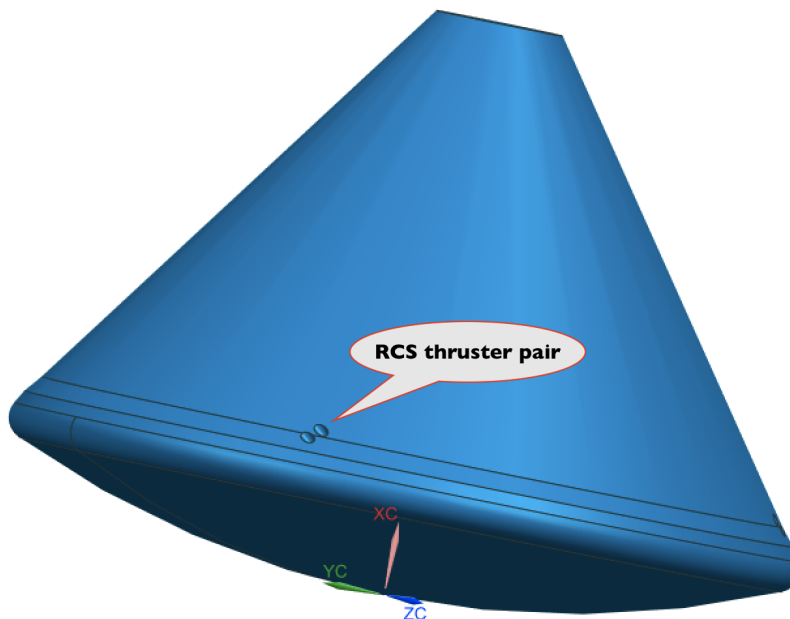


Fig. 1 Mars Sample Return Lander aeroshell geometry, option A(N1), with conical thrusters located near heatshield.

Table 1 Wind Tunnel conditions

M	Re , M/ft	T_0 , °F	α , deg.	NPR	NTR
2.399	3.000	125	-20	1588	1.9

III. Geometry Creation

This study began with a set of axisymmetric coordinates and parameters that defined the smooth-body MSRL capsule geometry and a table of thruster throat locations and pointing vectors (see Table 2).

Engineering Sketch Pad (ESP) [15], which includes the Open-source Constructive Solid Modeler (OpenCSM) [16], was used to turn these coordinates into a 3D solid model by way of a constrained 2D sketch and subsequent revolve.

Table 2 MSRL thruster throat locations and pointing vectors.

Thruster	Throat Position			Thrust Vector		
	x , mm	y , mm	z , mm	$\hat{i}$	$\hat{j}$	$\hat{k}$
1A	1516.755	1521.476	1515.396	-0.26	0.673	-0.693
1B	1501.204	1590.870	1572.558	-0.26	0.673	-0.693
2A	-1516.755	1521.476	1515.396	0.26	0.673	-0.693
2B	-1501.204	1590.870	1572.558	0.26	0.673	-0.693
3A	-1516.755	-1521.476	1515.396	0.26	-0.673	-0.693
3B	-1501.204	-1590.870	1572.558	0.26	-0.673	-0.693
4A	1516.755	-1521.476	1515.396	-0.26	-0.673	-0.693
4B	1501.204	-1590.870	1572.558	-0.26	-0.673	-0.693

Total time for a novice user is on the order of an hour, while for an experienced ESP user, this task requires little more than the time to enter the numbers. Figure 2 shows the resulting geometry in ESP, and Appendix VI.A shows the OpenCSM file created by the interactive ESP session.

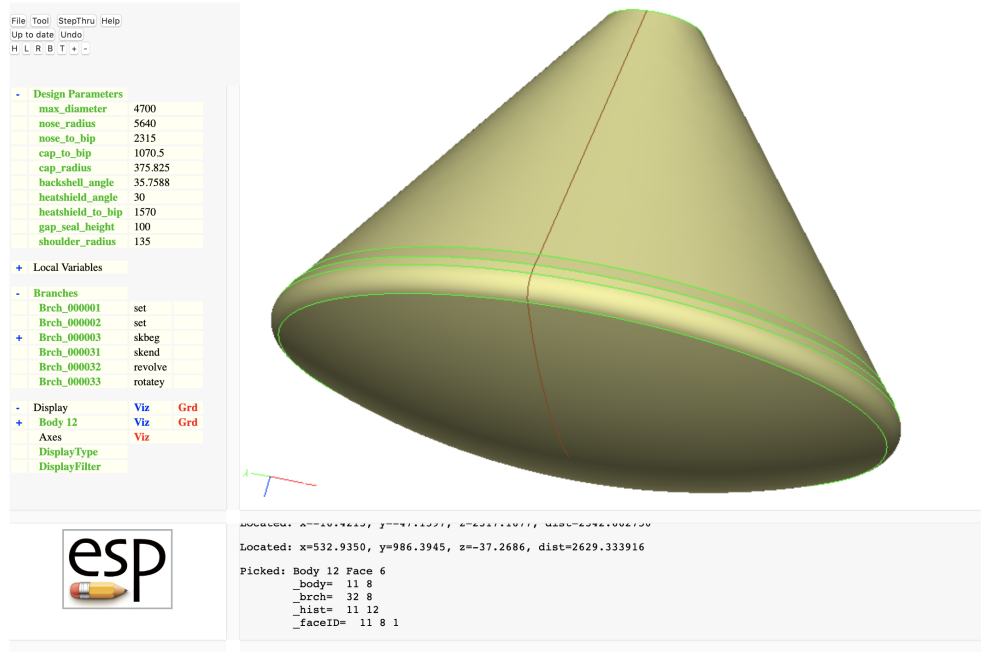


Fig. 2 Smooth-body MSRL geometry model created in ESP.

Adding the nozzles was accomplished by assigning a coordinate system for each thruster to the smooth capsule by creating a vector orthogonal to the first two via the cross product of the throat position and the nozzle pointing vector. Then, given the geometry for the conical wind tunnel model thrusters, each thruster was assigned a unique set of boundary condition tags, placed according to its coordinate system, and subtracted from the original smooth-body capsule geometry to produce the final capsule geometry with thrusters installed according to Table 2.

The OpenCSM file shown in Appendix VI.B was used to create a computational domain, place thrusters, assign boundary conditions, generate an EGADS [17] geometry file used by subsequent steps, and export a STEP file that was used as the starting point for the manual grid generation workflow discussed in the following section. Thanks to a spreadsheet and OpenCSM's ability to loop, this entire task took less than half an hour.

IV. Work Flows

The following two subsections outline the steps in each of the two work flows: manual grid generation and automatic grid generation. For both work flows, the FUN3D finite-volume flow solver [18] was used with the same settings: compressible, perfect gas with the Spalart-Allmaras (SA) turbulence model [19]. Comparisons between the two work flows are reserved for the following section.

A. Manual Grid Generation Work Flow

The components of manual grid generation work flow are shown in Figure 3. Starting with a problem definition, an initial grid is created, a flow solution is computed, and results are mined from the solution. In addition, a grid

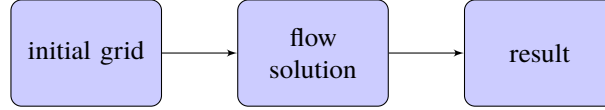


Fig. 3 Manual grid generation process.

sensitivity should be performed, which entails running a family of grids of various resolutions through this workflow. The components of the particular manual grid generation work flow are described in the following.

This instance of the manual grid generation work flow begins with receipt of a geometry file of the vehicle in STEP^a format. To be able use this geometry with the GridEx grid generation package [20, 21], the Siemens NX CAD software^b was used to translate this to a Parasolid geometry file in a matter of minutes (provided a license is available). In GridEx, the computational domain was created by subtracting the vehicle geometry from a sphere; and boundary face types are categorized by components and by whether they should have a boundary layer grid generated normal to them. Next, the surface grid is created by placing VGrid [22] spacing “sources” throughout the domain and visualizing their surface grid effects until the experienced operator is satisfied with the overall surface grid spacing. Sources can be points, lines, or simple volumes. For this geometry, a single strong source was placed near the centroid of the vehicle to resolve the surrounding volume and one at each nozzle throat to resolve the nozzle and the plume as seen in Figure 4. The

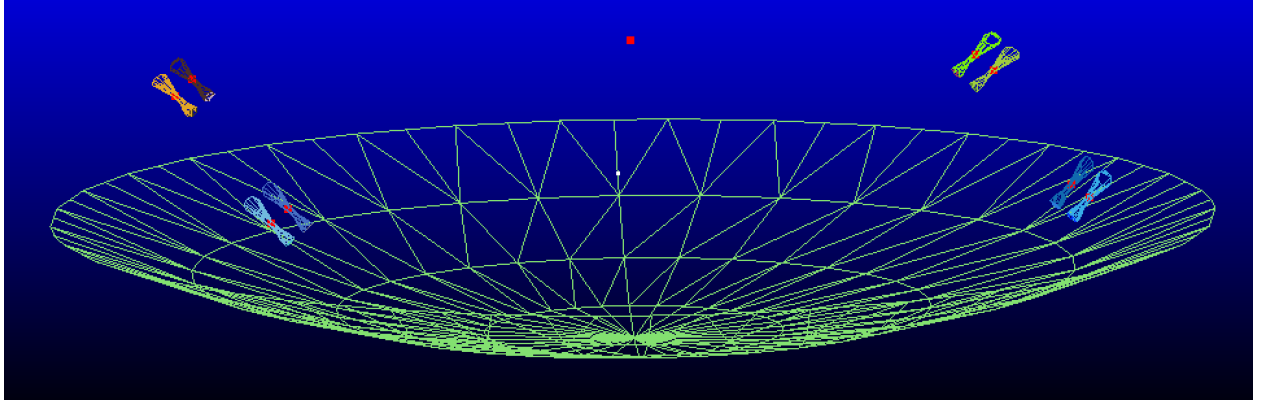


Fig. 4 The tiny red tetrahedra (▲) represent the approximate size and location of the VGrid spacing sources at the vehicle centroid and nozzle throats. Note: The very coarse tessellation shown is to provide a geometric reference and still be able to see the sources—it does not depict the final surface grid.

nominal surface grid for this case is shown in Figure 5. Including a small contribution from the farfield boundary, the surface grid contains 170k nodes and 336k triangular faces. For a simple geometry like this in the hands of an experienced GridEx user, this surface gridding process takes 0.5–1.5 hours; and even for the most complex geometries (e.g., a full-protuberance Space Launch System vehicle stack), never more than a work day.

A batchEx script, the text equivalent of the interactive GridEx session, is created to specify boundary layer grid properties for the AFLR3 volume grid generator^c and so that the script can be used to generate a family of grids for

^aSTEP stands for the Standard for the Exchange of Product model data, ISO 10303.

^b<http://www.siemens.com/nx>, last accessed December, 2019.

^c<http://www.simcenter.msstate.edu>, last accessed December, 2019.

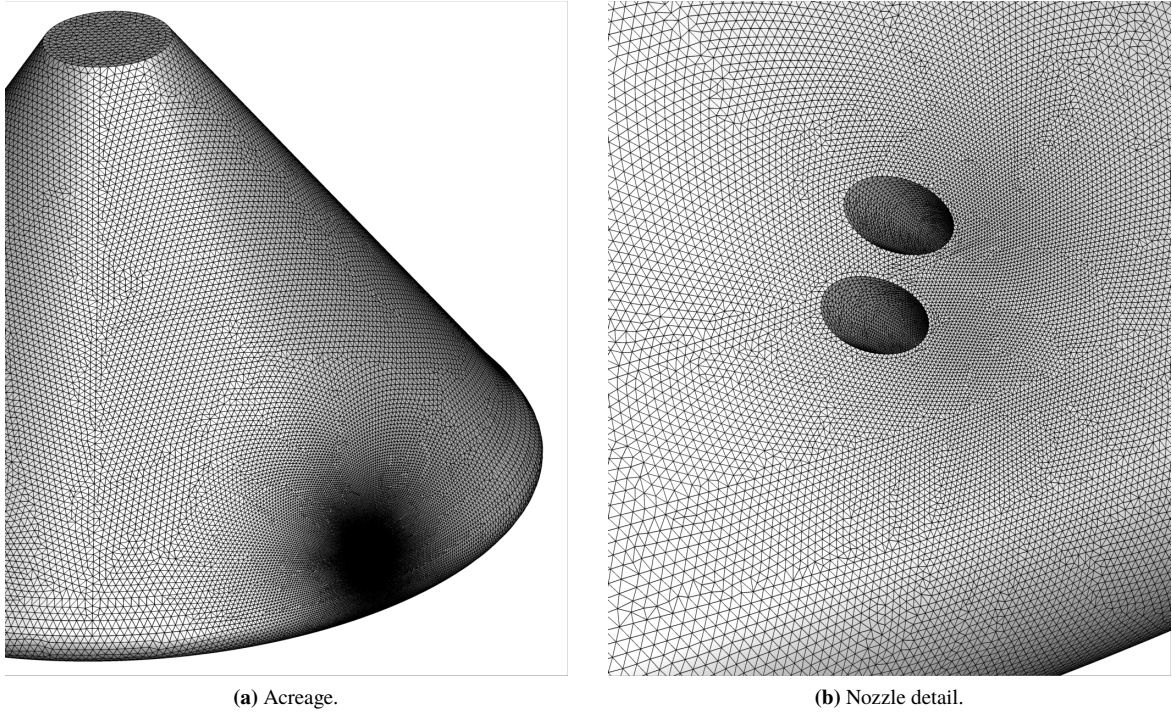


Fig. 5 Surface grid.

the grid sensitivity study by globally varying the spacing field request and the boundary layer growth factors in log space. Listing 1 shows excerpts of the batchEx script for this case and Table 3 shows the grid refinement statistics used for the grid sensitivity study. Volume grid generation is sequential and can take anywhere from 30 minutes to a few

Table 3 Grid refinement statistics.

f	N	N_T	N_{Py}	N_{Pr}	N_E	r_0	r_{max}	h_0
1.26	14.3M	51.4M	17.7k	16.8M	34.5M	1.10	1.19	1.26e-6
1.00	16.2M	54.5M	21.8k	20.9M	33.6M	1.08	1.15	1.00e-6
0.79	32.1M	109.2M	35.8k	40.9M	68.2M	1.06	1.12	0.79e-6
0.63	64.2M	217.1M	69.6k	82.8M	134.3M	1.05	1.09	0.63e-6

days depending on the final grid size and the number of spacing sources it needs to obey (the fewer the better). But unlike other grid generators that do not use the local reconnection algorithm, AFLR3 always has a valid volume grid throughout the grid generation process once it recovers the initial surface tessellation—unless of course, it exhausts the machine’s memory. With locally available machines, the practical upper limit is around 150M nodes.

One Achilles heel of this workflow is that one often does not discover problems in the underlying geometry until the volume grid generator attempts to recover the boundary tessellation when it begins to tessellate the volume. The most frequent issues are caused by

- 1) A face that is too small to be resolved by the requested spacing, causing the face to collapse into a zero-area triangle.
- 2) Pointed (singular) bodies of revolution with one or two faces, creating two triangles at the tip that collapse onto one another, no matter how small the spacing request.

The first is often caused by two faces that should share an edge, but due to tolerances, may appear as two disjoint faces connected by an extremely thin face. This can often be cured by using a CAD system to “heal” or “repair” the geometry, but one must carefully monitor any geometry changes that might result. A more severe form of the first issue may be due to issues associated with the underlying surface parameterization, such as a fold or singularity. In these cases, often the

Listing 1 Excerpts of the batchEx script used to generate a family of grids by globally varying the spacing field request and the boundary layer growth factors in log space.

```

#!/bin/bash -uex

BASE=wt_8t-hs_msrl_opt_n_alhg

[..setup software environment..]

# AFLR3 Volume Grid Configuration
# Grid Type: 0 (ANISO=0, ISO=1, ISO90=2)
# Volume Growth: 0 (INTERP'D=0, MODERATE=1, HIGH=2, VERY_HIGH=3)
# Pentas: 0 (TRUE/FALSE)
# Initial Growth: 1.2
# Growth Multiplier: 1.05
# Max Growth: 2
# Spacing Type: 0 (ABSOLUTE=0, LAMINAR=1, TURBULENT=2)
# Spacing: 1e-05
# Spacing Multiplier: 1
# Re: 1e+06
# y+ (or Laminar Fraction): 1
# Custom: (null)
cat > .aflr3 << EOF
0
0
1
1.08
1.01
1.15
0
1.0e-6
1.01
0.0
1.0
-ogrid ${BASE}.lb8.ugrid
EOF

batchEx \
-parasolid \
-load ${BASE}.xmt_txt -v 1 \
-v 1 \
-slip farfield \
-f 1 -mv -1 \
-f 2 -mv -1 \
-noslip capsule \
-f 11 -mv -1 \
[...]
-noslip nozzle_6 \
-f 35 -mv -1 \
[...]
-slip plenum_1 \
-f 82 -mv -1 \
[...]
-model -1 \
-viscous 1 -stretch 0 \
-s 3 \
-m 2 \
-vsrc 1 1.5 0 0 0.030 1.0 0.005 0.18 0.01 0 \
-srange 1 1 -vsrc_extra 20 0 0 \
-vsrc 1 0.966 -2.051 -1.909 0.002 1.0 0.005 0.18 0.01 0 \
[...]
-vsrc 1 1.042 -1.962 1.928 0.002 1.0 0.005 0.18 0.01 0 \
-srange 2 -1 -vsrc_extra 6 0 0 \
-sdef 1 \
-smin 0.001 \
-bscale 1.0 \
-bc ${BASE}-s.mapbc \
-aflr3 ${BASE}-s \
-o ${BASE}-s

```

only recourse is to completely replace the offending face. The second issue can often be cured by splitting the revolved face into multiple faces that do not have diametrically opposed split lines and specifying a spacing small enough to resolve the resulting arcs with three or more points.

B. Automatic Grid Generation Work Flow

The components of the automatic grid generation work flow are shown in Figure 6. Starting with a problem definition, an initial grid is created and a flow solution is computed. The information from the flow solution is used to estimate error and specify a new grid resolution request. If the estimated errors are larger than limits specified by the practitioner, the current grid system is modified by grid mechanics to adhere to the grid resolution request. The process

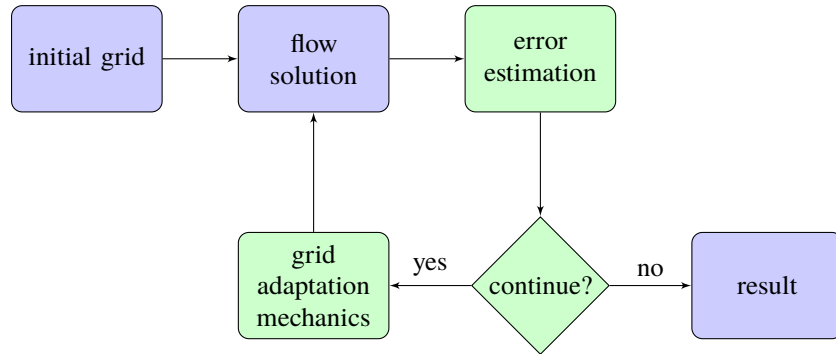


Fig. 6 Automatic grid generation process.

is repeated until an exit criterion is met (e.g., an accuracy requirement or a resource limit). Details of the particular sketch-to-solution work flow employed for this study are described in the remainder of this subsection.

In this case, the geometry is already in the native format for this automatic-grid workflow: an OpenCSM file that produces an EGADS geometry file. Kleb et al. [12] discuss options for other geometry starting points, e.g., a STEP

file, and contains a simple script used to complete the next two steps of:

- exporting a draft of the FUN3D viscous fluid solver [18] boundary conditions file (`.mapbc`),
- automatically creating a very coarse, surface-curvature-constrained, *isotropic* volume grid via `refine`^d and TetGen [23].

And so with geometry, initial grid, and boundary conditions in hand, all that remains is to configure the FUN3D flow solver via `fun3d.nml`, define the schedule and resources for the solution-adaptation cycle via `case_specifics`, and launch the solution process via the `f3d` utility that comes with FUN3D.^e

V. Comparisons

The observant reader may already be crying, "Foul!" as the Geometry Creation section resulted in the native input for the automatic grid generation work flow. And so to be fair when comparing the two work flows, the initial geometry translation from STEP to Parasolid in the manual grid generation work flow will not be included during comparisons.

But even this choice to use ESP for geometry work as opposed to the NX CAD software used in the manual grid generation workflow reveals an important difference: The first author has dabbled in NX for over a decade and still does not felt comfortable with it, whereas he felt comfortable using ESP within a few months. For example, the table of thruster positions and orientations was daunting for this casual NX user to implement; whereas, after creating a third coordinate vector via the cross product, it was a simple cut-and-paste operation in ESP that took less than 10 minutes total. In addition, ESP is very amenable to automation via application programming interfaces (APIs) [24, 25] and the text-based OpenCSM language.^f It is also unencumbered by the potential delays, cost, and scalability issues that come with network-licensed software.

Table 4 shows the cost comparison between the two grid generation work flows in terms of human time, computer time (CPU), and wall time, measured in hours. As anticipated, because the automatic grid generation work flow has fewer steps performed by a human, it has six times less cost in this area; and the potential for fewer human mistakes multiplies this savings. In this case, the manual workflow wins in terms of computer time. This is due to the lack of a strong solver as will be discussed later. The 25% more wall time for the manual grid generation method is largely due manual touch time; and note that if one had taken advantage of the automatically-generated grid work flow producing a sizable history of grids and the decreasing spread in their predictions at each grid level (as will be seen in Figure 7), the process might have been terminated at an order of magnitude fewer number of nodes.

Table 4 Grid generation work flow cost comparison as a function of tasks including limited manual grid generation sensitivity study.

Task	Human, hours		CPU, hours		Wall time, hours	
	$\mathcal{M}$	$\mathcal{A}$	$\mathcal{M}$	$\mathcal{A}$	$\mathcal{M}$	$\mathcal{A}$
Import geometry	0.1	0.1	-	-	0.1	0.1
Create domain	0.5	0.2	-	-	0.5	0.2
Assign BCs	0.2	0.1	-	-	0.2	0.1
Surface Grid	2.0	-	-	0.1	2.0	0.1
Volume Grid	0.5	-	4.0	0.3	6.5	0.3
FUN3D mapbc	0.1	0.1	-	-	0.1	0.1
FUN3D namelist	0.2	0.2	-	-	0.2	0.2
Job submission	1.0	-	-	-	1.0	-
Simulation	-	-	6000.0	18000.0	36.0	36.0
Tabulate results	0.5	-	-	-	-	-
Totals	4.6	0.7	6004.0	18000.4	46.6	37.1

$\mathcal{M} \equiv$ manual grid generation; $\mathcal{A} \equiv$ automatic grid generation

^d`refine` is developed by NASA and is available via <https://github.com/NASA/refine>, last accessed December, 2019.

^eNote: Examples of `fun3d.nml` and `case_specifics` are given in Kleb et al. [12].

^fWhile it might be even more convenient if an established language like Python could be used instead—see Refs. [26, 27].

How well did each answer the engineering question at hand, "To what extent did the RCS interact with the vehicle aerodynamics?" First, let's examine the trajectory of overall vehicle drag as a function of grid nodes, N , shown in Figure 7 where the sequence of manually-generated grids are shown as orange circles (○), and the automatically-generated

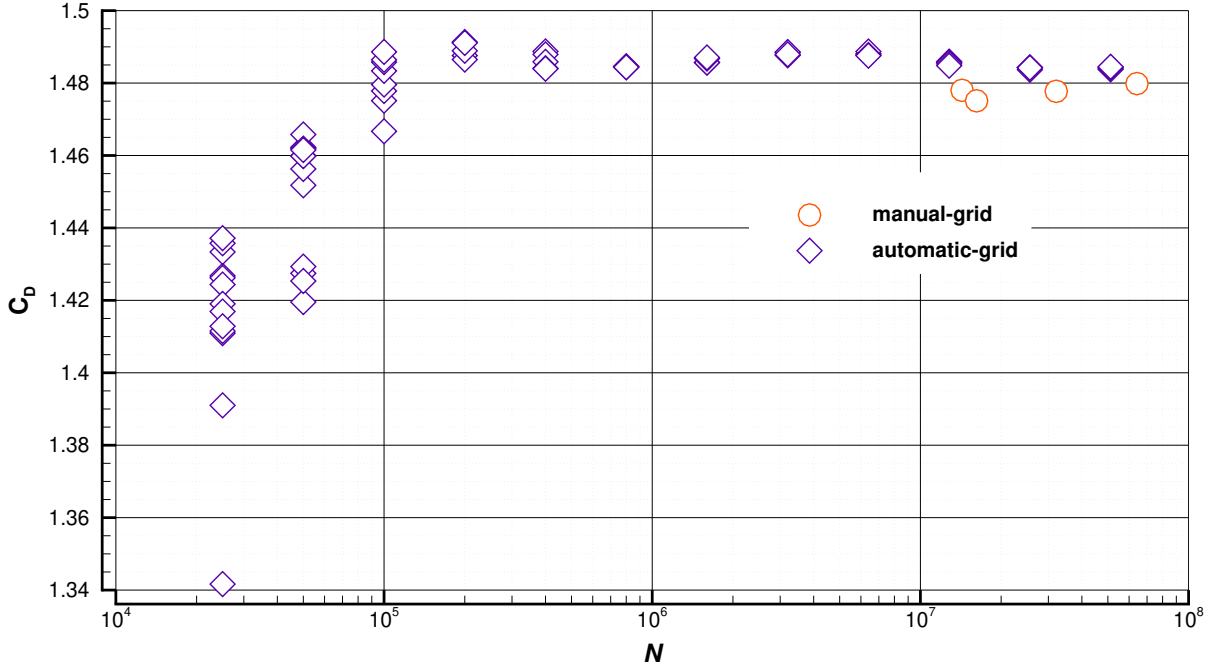


Fig. 7 Overall vehicle drag coefficient, C_D as a function of grid nodes, N .

grids are shown as violet diamonds (◇). The manually-generated grid sequence and the automatically-generated grid sequences appear to be converging to a similar drag value, but from opposite sides. The automatic grid generation trajectory also appears to approach the neighborhood of the asymptote with far fewer nodes, e.g., one-to-two orders of magnitude sooner in this case. This is partly due to the failure to manually produce a true family of grids on the coarser side of what the expert decided was a nominal resolution for this study and because grid sensitivity studies typically strive for finer meshes, not coarser ones—if resources allow. Meanwhile, the automatic process starts with far coarser grids because they are much cheaper and because it gives the adaptation process for freedom to seek the optimum node-distribution topology sooner.

For this study, the automatic grid generation work flow was configured to begin at 25,000 nodes and to monitor the drag coefficient from each solution, similar to that proposed in Ref. [10]. Specifically, the number of nodes is allowed to double when either:

- at least four solutions at a given number of nodes show a drag coefficient within 0.1% spread, or
- twenty grid-solution pairs have been computed at a given number of nodes.

As expected, initial coarser meshes show a wide variation in drag prediction from grid to grid; but as the number of nodes increases, the spread becomes less and typically only requires the minimum four grid-solution pairs before doubling the size of the grid. This staged grid generation process has many advantages.

First, by monitoring the spread in a quantity of interest at a given resolution and doubling based on tolerance (or reaching a limit), work is not wasted as would be the case if the number of iterations at a given number of nodes was fixed. This staged grid generation process also allows the user to monitor the quantity or quantities that are of most interest—see for example, Figure 3 in Ref. [12].

Second, because the solution from each previous grid is interpolated onto the new grid,[§] the process has built-in grid sequencing, that, when coupled with a Newton flow solver, can yield prodigious iterative convergence. Consider, for example Figure 8 from Ref. [12] that shows the two fold advantage of using a Newton solver over a typical defect-correction solver for grid sequences like these: 1) increased robustness and 2) increased efficiency.

[§]Grid sequencing could also be done with the manual grid generation work flow; but due to the extra manual effort involved, it is rarely done.

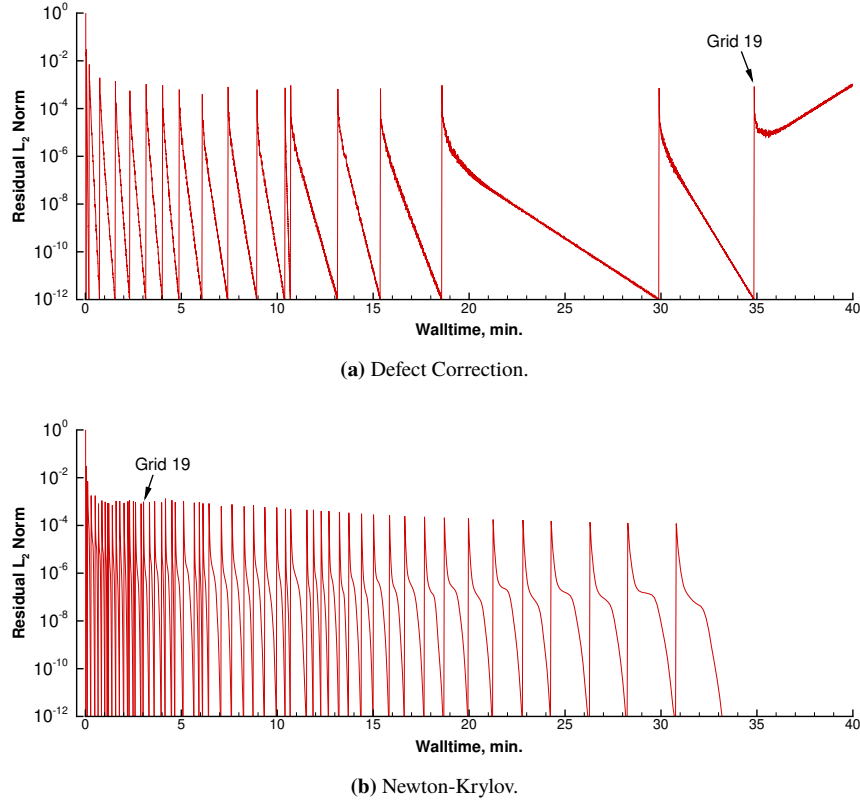


Fig. 8 Comparison of the robustness and cost for two solver approaches (from Ref. [12]).

Third, the goal of this particular grid-solution adaptation process is to minimize solution interpolation error; and as a result, it seeks the optimal grid point topology, or distribution, and not a sequence of uniformly-refined grids to show design order. As Diskin et al. [28] warn,

Establishing a design-order convergence for RANS solutions proved to be more difficult than was expected and could not be achieved using current grid-refinement guidelines emphasizing uniform refinement. Any discretization scheme needed grid adaptation near geometric singularities to establish the design-order convergence. This conclusion pertained not only for the specific RANS formulation but for any system of partial differential equations.

And finally, by being able to visualize the progression of quantity of interest (C_D in this case), the operator can decide to terminate the process due to resource constraints or comfort level. The potential downside with discretized nonlinear equations such as these, however, is that some important flow feature(s) might not yet have been resolved, and the asymptote could shift.

Table 5 shows a detailed comparison of force and moment coefficients for the positive roll command case as compared with the uncommanded case (jets off). The predictions agree with one another to the third decimal place, and both show the expected positive rolling moment. While examining the performance of the thrusters in more detail, however, the discharge coefficients were lower than expected for the automatic grid generation work flow case. The cause was found to be a lack of boundary layer resolution in the nozzle as shown in Figure 9a. Future work will seek to augment the adaptation process to assure appropriate boundary layer wall spacing will occur earlier in the process. As a workaround, the nozzle wall boundary conditions were changed to slip walls with the result shown in Figure 9b. This change yielded the expected discharge coefficient and still provided a very similar plume shape.

Finally, how do the two work flows compare against the CFD Vision 2030 Study's goal that, "the mesh generation process should be invisible to the CFD user or engineer"? As the name implies, the manual grid generation work flow requires considerable human attention in the course of generating the grid(s). For this case study, the human touch time was on the order of half a day, with another half a day for volume grid generation necessary to perform a

Table 5 Comparison of predicted vehicle forces and moments for RCS positive roll command and uncommanded.

Roll Thrusters	C_X	C_Y	C_Z	C_{m_x}	C_{m_y}	C_{m_z}	work flow
on	1.556	-0.006	-0.059	0.036	-0.003	0.000	manual grid generation, finest
on	1.558	-0.005	-0.059	0.034	-0.003	0.000	automatic grid generation, final
off	1.489	0.000	-0.058	0.000	-0.001	0.000	
on-off	0.069	-0.005	-0.001	0.034	-0.002	0.000	

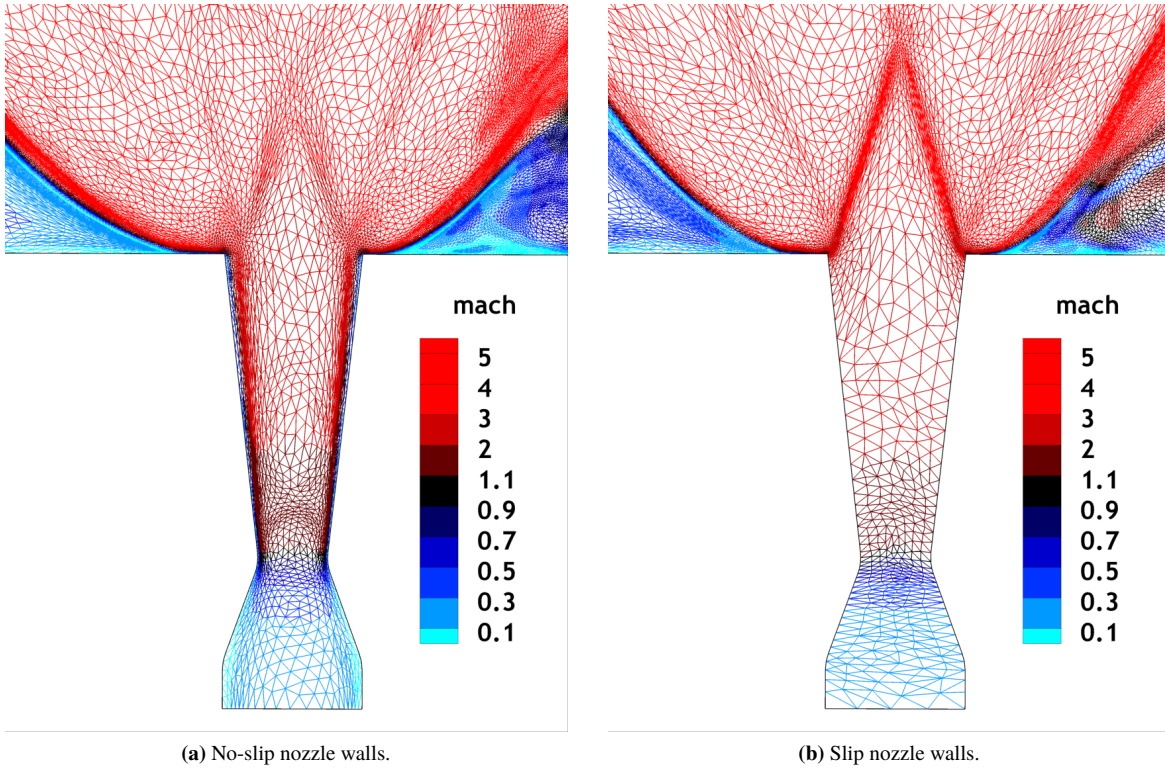


Fig. 9 Comparison of workaround for nozzle wall boundary condition.

cursory grid sensitivity study. As mentioned in the beginning of this section, there was zero human interaction with the grid generation and adaptation phases in the automatic grid generation work flow. The only grid-specific input was constraining the maximum number of nodes; or as Sir Arthur C. Clarke's Third Law posits, "Any sufficiently advanced technology is indistinguishable from magic."

VI. Concluding Remarks

To explore the benefits and pitfalls of the nascent sketch-to-solution automatic grid generation capability, this paper compared a conventional manual grid generation work flow to this new automatic grid generation work flow for a given engineering question, "What are the aerodynamics interactions caused by the reaction control system on an entry vehicle?" This case study indicates that while the automatic grid generation sketch-to-solution process is not yet mature (e.g., the leeside nozzle boundary layer grid was slow to come in), it is preferred over a manual grid generation work flow because it greatly reduces manual labor, thereby eliminating many opportunities for human error; it has built-in grid sequencing and grid sensitivity; and with some adjustments, it shows the promise of reaching the correct answer with far fewer grid points.

Future work consists of finding better ways to manage the overall grid node budget, or in other words, seeking poka-yoke^h mechanisms that either guard against failures or bring attention to them if they occur.

Acknowledgments

The authors would like to acknowledge: John Van Norman (Analytical Mechanics Associates, Hampton, VA) for uncovering past Mars RCS analysis artifacts; Kyle Thompson, Bill Wood, and Steve Bauer (NASA Langley) for manuscript reviews; John Dannenhoffer (Syracuse University) and Bob Haines (MIT) for help with ESP, EGADS, and OpenCSM; Jan-Reneé Carlson (NASA Langley) for FUN3D's propulsion infrastructure; the FUN3D team (NASA Langley) for continued development and support; and the mid-range compute cluster team (NASA Langley) for ready access to high performance computing.

This work was supported by the NASA Langley Research Center Engineering and Research Directorates; the NASA Science Mission Directorate (SMD) Mars Sample Return Lander (MSRL) project; and the Aeronautics Research Mission Directorate (ARMD) Transformational Tools and Technologies (TTT) project,

Appendices

Configuration files used by some of the work flows explored above.

A. Smooth-body MSRL OpenCSM file

```
# Mars Sample Return Lander
# Option N configuration
# (A1: 2019-10-17)
# with 100mm tall fairing over thermal seal joggle
# at heatshield/backshell juncture
# Dimensions: mm

# BIP = backshell interface plane (origin)
# z is from origin through nose

despmtr max_diameter      4700
set      max_radius       0.5*max_diameter
despmtr  nose_radius      5640
despmtr  nose_to_bip      2315
despmtr  cap_to_bip       1070.5
despmtr  cap_radius       375.825
despmtr  backshell_angle  35.758805246
despmtr  heatshield_angle  30
despmtr  heatshield_to_bip 1570
despmtr  gap_seal_height  100
set      gap_seal_to_bip  heatshield_to_bip-gap_seal_height
despmtr  shoulder_radius  135

skbeg    0    0    0    1
skvar    xy    "    0.0;    0.000;    0.0;\
          1070.5;    0.000;    0.0;\
          1070.5;    375.825;    0.0;\
          -1470.0;    2205.316;    0.0;\
          -1570.0;    2287.329;    0.0;\
          -1647.2;    2331.913;    0.0;\
          -1838.3;    2269.319;    44.9;\
          -2315.0;    0.000;    120.5;"

skcon     X    1    -1    0
skcon     Y    1    -1    0
skcon     H    1    2    0
skcon     L    1    2    cap_to_bip
skcon     V    2    3    0
skcon     L    2    3    cap_radius
skcon     I    3    4    180-backshell_angle
skcon     W    4    1    gap_seal_to_bip
skcon     W    5    1    heatshield_to_bip
skcon     I    5    6    180-heatshield_angle
skcon     T    6    -1    0
skcon     R    6    7    shoulder_radius
skcon     Y    6    7    max_radius-shoulder_radius
skcon     T    7    -1    0
skcon     R    7    8    nose_radius
skcon     H    8    1    0
skcon     L    8    1    nose_to_bip
skcon     P    8    -1    0
linseg    ::x[2]    ::y[2]    0
linseg    ::x[3]    ::y[3]    0
linseg    ::x[4]    ::y[4]    0
linseg    ::x[5]    ::y[5]    0
linseg    ::x[6]    ::y[6]    0
arc        ::x[7]    ::y[7]    0    ::d[7]    xy
arc        ::x[8]    ::y[8]    0    ::d[8]    xy
linseg    ::x[1]    ::y[1]    0
skend     0
revolve    0    0    0    1    0    0    360
rotatey    90    0    0

end
```

^hPer Ref. [29], poka-yoke (ポカヨケ) is a Japanese term that means “mistake-proofing” or “inadvertent error prevention.” A poka-yoke is any mechanism in any process that helps an operator avoid (yokeru) mistakes (poka). Its purpose is to eliminate product defects by preventing, correcting, or drawing attention to human errors as they occur.

B. Domain, Thrusters, and BCs OpenCSM file

```
# LRC UPWT 6"-diameter Option N-A1 model with 8 thrusters near heatshield

despmtr model_diameter          6
despmtr farfield:zcent_in_diameters -1.2
despmtr farfield:radius_in_diameters 2.5

# create farfield domain and assign BCs
sphere 0 0 farfield:zcent_in_diameters*model_diameter farfield:radius_in_diameters*model_diameter
attribute bc $farfield

# load capsule and assign BCs
udprim csm filename $msrl_opt_n_a1-101719.csm
attribute bc $capsule

# thruster locations and orientations where (0,0,0) corresponds to throat location
#
#      x0      y0      z0      dx1      dy1      dz1      dx2      dy2      dz2
csystem tpos1 " 1516.755; 1521.476; 1515.396; -0.26; 0.673; -0.693; -2074.244; 657.108; 1416.360;"
csystem tpos2 " 1501.204; 1590.870; 1572.558; -0.26; 0.673; -0.693; -2160.804; 631.469; 1423.936;"
csystem tpos3 " -1516.755; 1521.476; 1515.396; 0.26; 0.673; -0.693; -2074.244; -657.108; -1416.360;"
csystem tpos4 " -1501.204; 1590.870; 1572.558; 0.26; 0.673; -0.693; -2160.804; -631.469; -1423.936;"
csystem tpos5 " -1516.755; -1521.476; 1515.396; 0.26; -0.673; -0.693; 2074.244; -657.108; 1416.360;"
csystem tpos6 " -1501.204; -1590.870; 1572.558; 0.26; -0.673; -0.693; 2160.804; -631.469; 1423.936;"
csystem tpos7 " 1516.755; -1521.476; 1515.396; -0.26; -0.673; -0.693; 2074.244; 657.108; -1416.360;"
csystem tpos8 " 1501.204; -1590.870; 1572.558; -0.26; -0.673; -0.693; 2160.804; 631.469; -1423.936;"

# from 4700 mm to model diameter/units
scale model_diameter/4700

# import (long) conical thruster geometry
udprim csm filename $prototype3l.csm
store thruster

# mark thruster BCs, position, and subtract from capsule
patbeg it 8
  restore thruster
  attribute bc !$thruster_+it
  select face 6
  attribute bc !$plenum_+it
  applysys !$tpos+it
  subtract
patend

# create computational domain
subtract

# orient with x-axes running from nose to backshell
rotatey -90 0 0

# move origin from backshell interface plane to nose
translate model_diameter*2315/4700 0 0

dump wt_8t-hs_msrl_opt_n_a1.egads
dump wt_8t-hs_msrl_opt_n_a1.stp

end
```

References

- [1] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA CR-2014-218178, Langley Research Center, Mar. 2014. doi:2060/20140003093.
- [2] Alauzet, F., and Loseille, A., "A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics," *Computer-Aided Design*, Vol. 72, 2016, pp. 13–39. doi:10.1016/j.cad.2015.09.005, 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.
- [3] Park, M. A., Krakos, J. A., Michal, T., Loseille, A., and Alonso, J. J., "Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Toward CFD Vision 2030," AIAA Paper 2016–3323, 2016.
- [4] Ibanez, D., Barral, N., Krakos, J., Loseille, A., Michal, T., and Park, M., "First Benchmark of the Unstructured Grid Adaptation Working Group," *Procedia Engineering*, Vol. 203, 2017, pp. 154–166. doi:10.1016/j.proeng.2017.09.800, 26th International Meshing Roundtable, IMR26, 18-21 Sept. 2017, Barcelona, Spain.

- [5] Park, M. A., Barral, N., Ibanez, D., Kamenetskiy, D. S., Krakos, J., Michal, T., and Loseille, A., “Unstructured Grid Adaptation and Solver Technology for Turbulent Flows,” AIAA Paper 2018–1103, 2018.
- [6] Tsolakis, C., Chrisochoides, N., Park, M. A., Loseille, A., and Michal, T., “Parallel Anisotropic Unstructured Grid Adaptation,” AIAA Paper 2019–1995, 2019.
- [7] Park, M. A., Balan, A., Anderson, W. K., Galbraith, M. C., Caplan, P. C., Carson, H. A., Michal, T., Krakos, J. A., Kamenetskiy, D. S., Loseille, A., Alauzet, F., Frazza, L., and Barral, N., “Verification of Unstructured Grid Adaptation Components,” AIAA Paper 2019–1723, 2019.
- [8] Park, M. A., Kleb, B., Jones, W. T., Krakos, J. A., Michal, T., Loseille, A., Haines, R., and Dannenhoffer, J. F., III, “Geometry Modeling for Unstructured Mesh Adaptation,” AIAA Paper 2019–2946, 2019.
- [9] Frazza, L., Loseille, A., and Alauzet, F., “Unstructured Anisotropic Mesh Adaptation for 3D RANS Turbomachinery Applications,” AIAA Paper 2019–3059, 2019.
- [10] Alauzet, F., and Frazza, L., “3D RANS Anisotropic Mesh Adaptation on the High-Lift Version of NASA’s Common Research Model (HL-CRM),” AIAA Paper 2019–2947, 2019.
- [11] Belme, A., Alauzet, F., and Dervieux, A., “An a priori Anisotropic Goal-Oriented Error Estimate for Viscous Compressible Flow and Application to Mesh Adaptation,” *Journal of Computational Physics*, Vol. 376, 2019, pp. 1051–1088. doi: 10.1016/j.jcp.2018.08.048.
- [12] Kleb, B., Park, M. A., Wood, W. A., Bibb, K. L., Thompson, K. B., Gomez, R. J., III, and Tesch, S. H., “Sketch-to-Solution: An Exploration of Viscous CFD with Automatic Grids,” AIAA Paper 2019–2948, 2019.
- [13] Park, M. A., Kleb, B., Anderson, W. K., Wood, S. L., Balan, A., Zhou, B. Y., and Gauger, N. R., “Exploring Unstructured Mesh Adaptation for Hybrid Reynolds-Averaged Navier-Stokes/Large Eddy Simulation,” SciTech 2020, American Institute of Aeronautics and Astronautics, Reston, VA (submitted for publication).
- [14] Muirhead, B. K., and Karp, A., “Mars Sample Return Lander Mission Concepts,” *2019 IEEE Aerospace Conference*, 2019, pp. 1–9. doi:10.1109/AERO.2019.8742215.
- [15] Haines, R., and Dannenhoffer, J. F., III, “The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry,” AIAA Paper 2013–3073, 2013.
- [16] Dannenhoffer, J. F., III, “OpenCSM: An Open-Source Constructive Solid Modeler for MDAO,” AIAA Paper 2013–701, 2013.
- [17] Haines, R., and Drela, M., “On The Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design,” AIAA Paper 2012–683, 2012.
- [18] Biedron, R. T., Carlson, J.-R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., Rumsey, C. L., Thomas, J. L., Thompson, K. B., and Wood, W. A., “FUN3D Manual: 13.6,” NASA TM-2018-220416, Langley Research Center, Oct. 2019.
- [19] Spalart, P. R., and Allmaras, S. R., “A One-Equation Turbulence Model for Aerodynamic Flows,” *La Recherche Aéronautique*, Vol. 1, 1994, pp. 5–21.
- [20] Jones, W. T., “An Open Framework for Unstructured Grid Generation,” AIAA Paper 2002–3192, 2002.
- [21] Jones, W. T., “GridEx – An Integrated Grid Generation Package for CFD,” AIAA Paper 2003–4129, 2003.
- [22] Pirzadeh, S. Z., “Progress Toward a User-Oriented Unstructured Viscous Grid Generator,” AIAA Paper 96–31, Jan. 1996.
- [23] Si, H., “TetGen: A Quality Tetrahedral Mesh Generator and 3D Delaunay Triangulator,” WIAS Technical Report 13, 2008.
- [24] Edward Alyanak, R. D., Haines, R., Dannenhoffer, J. F., III, Bhagat, N., and Allison, D., “Multi-fidelity Geometric-centric Multi-disciplinary Analysis for Design,” AIAA Paper 2016–4007, 2016.
- [25] Bryson, D. E., Haines, R., and Dannenhoffer, J. F., III, “Toward the Realization of a Highly Integrated, Multidisciplinary, Multifidelity Design Environment,” AIAA Paper 2019–2225, 2019.
- [26] Mukundakrishnan, B., Rajmohan, N., Rajnarayan, D. G., and Fugal, S., “A Script-Based CAD System for Aerodynamic Design,” AIAA Paper 2019–3069, 2019.

- [27] Durscher, R., and Reedy, D., “pyCAPS: A Python Interface to the Computational Aircraft Prototype Syntheses,” AIAA Paper 2019–2226, 2019.
- [28] Diskin, B., Thomas, J. L., Rumsey, C. L., and Schwöppe, A., “Grid-Convergence of Reynolds-Averaged Navier–Stokes Solutions for Benchmark Flows in Two Dimensions,” *AIAA Journal*, Vol. 54, No. 9, 2016, pp. 2563–2588. doi:10.2514/1.J054555.
- [29] *Toyota Production System: Beyond Large-Scale Production*, Productivity Press, 1988.