PROGRESS IN AUTOMATION OF OVERSET STRUCTURED SURFACE GRID GENERATION

William M. Chan

NASA Ames Research Center

13th Symposium on Overset Composite Grids and Solution Technology, Mukilteo, Washington, October 17-20, 2016
OVERVIEW

- A brief history of current technology
- Starting point and objectives
- Overview of three main steps
- Algebraic step details
- Future plans for subsequent steps
- Summary and conclusions


### STRUCTURED OVERSET GRID CFD PROCESS

- **CAD Geometry**
  - Surface Grid Generation
  - Volume Grid Generation
  - Domain Connectivity

- **Flow Solution Computation**

- **Post-processing**
  - Aerodynamic loads
  - Flow field features

#### Grid Generation

- **Time**
  - 80+% (Domain decomposition, Grid point distribution)
  - < 5% (Surface mesh creation, Surface coverage)
  - 5-15% (Depends on algorithm/software)

#### Feature curves

#### Surface grids

C = number of components

G = number of grids

P = number of grid points
A BRIEF HISTORY

1980’s
Fortran programs
No tools
No GUls
C < 10
G < 10
P < 10M

1990’s
Chimera Grid
Tools, grid-centric
scripts, Gridgen,
Overture, C ~ 50
G ~100, P ~ 50M

2000’s
CGT Script Library,
parameterized,
component-centric
scripts, Glyph,
C ~ 100, G ~ 200,
P ~ 100M

2010’s
Parallelized using
team of workers,
C ~ 100’s, G ~ 1000,
P ~ 100’s M
CURRENT TECHNOLOGY LIMITATIONS AND FUTURE COMPUTATIONS

GUI limitations
- Steps not easily reproducible
- Not parameterized

Scripting approach limitations
- Require significant manual effort to build
- Script modification needed even for small topological changes

- Manual scripting approach (even with team) is no longer practical
- Difficult to achieve higher level of abstraction beyond component level
PREVIOUS WORK ON SURFACE GRID GENERATION AUTOMATION

Chan and Gomez
(AIAA Paper 99-3303)
Wrap around surface features using hyperbolic grids and spider web grids

Dannenhoffer and Haimes
(AIAA Paper 2011-3540)
Use feature trees from CAD solid models to build up from basic solid shapes
- Unstructured surface triangulation geometry representation derived from native CAD, STEP or IGES file with proper resolution of high curvature regions

- Discrete surface curves (subset of triangle edges)
  - Surface features (hard edges to be retained in computational grid)
  - Max curvature curves (e.g., leading edges)
  - Open boundaries
CURRENT MANUAL PROCEDURE

1. Identify algebraic and hyperbolic domains

2. Determine grid point distribution on curves:
   - uniform/stretched
   - end point and max spacing
   - match grid points on opposite boundaries

3. Generate algebraic grids by specifying bounding curves set

4. Generate hyperbolic grids by marching from initial curves using specified marching distance, initial and end spacing, max stretching ratio

5. Concatenate algebraic and hyperbolic grids where appropriate
OBJECTIVES

Develop algorithm/software for overset structured surface grid generation
- Significant manual effort reduction (near term)
- Full automation (long term)

REQUIREMENTS

- Capture essential surface features (sharp edges, max curvature lines such as leading edges) so that they lie on grid interior
- Distribute grid points to
  - resolve surface and flow features accurately
  - maintain smooth variation with low stretching ratio (trunc. error)
- Maintain NFRINGE points overlap between neighboring grids where NFRINGE is determined by flow solver differencing stencil

GOOD TO HAVE

- Compatibility of cell attributes in overlap regions
- As few domains as possible, but not critical since flow solver will break grids up for MPI runs
TOP LEVEL DEVELOPMENT PLAN

Build grids around surface features using
1. **Algebraic methods** (domains bounded by 4 curves)
2. Hyperbolic methods (domains bounded by 1 curve)

Fill remaining interior gaps on smooth regions of geometry with algebraic or hyperbolic grids
ALGEBRAIC STEP
Control Parameters

Classification Parameters

$\theta_{\text{sharp}}$: triangle edge is a sharp edge if angle between neighboring triangles unit normal is larger than $\theta_{\text{sharp}}$ (20 deg)

$\theta_{\text{tot}}$: curve is turning curve if the total turning angle $> \theta_{\text{tot}}$ (30 deg)

Grid Point Distribution Parameters

$N_{p,\text{min}}$ = minimum number of points on a curve (5)

$SR_{\text{max}}$ = max stretching ratio (1.2)

$\Delta s_{\text{max}}$ = max grid spacing (0.5% of global bounding box diagonal)

$\theta_{\text{ml}}$ = max local turning angle on turning curves (5 deg)
ALGEBRAIC STEP
1. Determine Curves Network Connectivity

- Split curve at interior point if necessary so that all curves only meet at end points
- Identify all junction points (where 3 or more curves meet)
- Store curve ID and end point for all curves meeting at junction

**Feature curves**
- Unsplit: 688
- Split: 1272

**Junctions**
- Deg 3: 610
- Deg 4: 149
- Deg 5: 8
- > Deg 6: 0

**Four-sided domains**: 311
ALGEBRAIC STEP
2. Determine Curve Attributes

- On sharp feature (yes – must preserve, control parameter $\theta_{\text{sharp}}$)
- On open boundary (yes – can only go one side for grid concatenation)
- Is a “short” curve (total arc length $\leq (N_{p_{\min}} - 1) \times \Delta s_{\text{max}}$)
- Is a “turning” curve (total turning angle $> \theta_{\text{tot}}$)
ALGEBRAIC STEP
3. Distribute Grid Points on Curves
3.1. Determine Distribution Type and Spacings

Turning curves or short non-turning curves:
Uniform spacing $\Delta s$ determined by $N_p_{\text{min}}$ or $\theta_{ml}$

Long non-turning curves:
Non-uniform spacing hyperbolic tangent distribution with specified
- max stretching ratio $SR_{\text{max}}$
- max interior spacing $\Delta s_{\text{max}}$
- spacing at end points $\Delta s_{e1}, \Delta s_{e2}$
- on geometry open boundary $\Delta s_e = 0.5 \Delta s_{\text{max}}$
- on geometry sharp edge with dihedral angle $\theta$
- use scale factor $\kappa(\theta)$ such that $\Delta s_e = \kappa(\theta) \Delta s_{\text{max}}$
  - concave corner ($\kappa > 1$, increases linearly with increasing concavity)
  - convex corner ($\kappa < 1$, decreases negative exponentially with increasing convexity)
ALGEBRAIC STEP
3. Distribute Grid Points on Curves
3.2. Match Grid Spacing at Junctions

Impose grid spacing continuity for matching curve pairs by taking minimum of predicted grid spacings.

Matching pair at junction
(compare curve end point unit vectors at junction)
ALGEBRAIC STEP
4. Identify Four-Sided Domains

Geometry surface and feature curves

Valid choice on surface

Invalid choice not on surface

Prefer more orthogonal option at corners

Not preferable
ALGEBRAIC STEP
5. Identify Adjacent Four-Sided Domains for Subsequent Concatenation

- Start with initial curve on open boundary or sharp edge bounding a four-sided domain
- Look for other four-sided domains that can be concatenated to initial or opposite curve
- Single direction sweep only

P1, P2, P3, P4 can be concatenated – impose same number of grid points on curves in concatenation direction
ALGEBRAIC STEP
6. Generate Surface Mesh Using TFI and Concatenate

Before concatenation: 38 patches
After concatenation: 16 patches
ALGEBRAIC STEP
7. Perform TFI Extension at Boundaries and Identify Initial Curves for Hyperbolic Marching (in progress)

Abutting patch boundaries after TFI

TFI extension at boundaries not on sharp edge

Initial curves on exposed boundaries for hyperbolic marching
TEST CASES
Landing Gear Strut

144 curves
32 TFI grids

<table>
<thead>
<tr>
<th>Mode</th>
<th>Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input prep. time</td>
<td>30 sec.</td>
</tr>
<tr>
<td>CPU time</td>
<td>3 sec.</td>
</tr>
</tbody>
</table>
TEST CASES
Attach Hardware

<table>
<thead>
<tr>
<th>Mode</th>
<th>Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input prep. time</td>
<td>30 sec.</td>
</tr>
<tr>
<td>CPU time</td>
<td>2.5 sec.</td>
</tr>
</tbody>
</table>

73 curves

14 TFI grids
TEST CASES
Feedline

126 curves

62 four-sided domains

2 TFI grids

<table>
<thead>
<tr>
<th>Mode</th>
<th>Manual</th>
<th>Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input prep. time</td>
<td>30 – 60 min.</td>
<td>30 sec.</td>
</tr>
<tr>
<td>CPU time</td>
<td>2 sec.</td>
<td>2.5 sec.</td>
</tr>
</tbody>
</table>
FUTURE PLANS FOR HYPERBOLIC STEP

Auto extract initial curves for hyperbolic marching
- Exposed boundary curves from algebraic step
- Unused original curves after algebraic step

Auto determine spatially variable marching
- distance
- direction
- Gap regions (if there are any) are not bounded by surface features
- Project reference surface vertices onto surface grids from algebraic and hyperbolic steps
- Identify vertices that fail to find donor stencil (orphan vertices)
- Identify orphan cells from orphan vertices on reference surface
- Build algebraic/hyperbolic grids over orphan cells
SUMMARY AND CONCLUSIONS

- A high level procedure has been designed for automation of overset structured surface grid generation
- Geometry starting point
  - unstructured surface triangulation representation
  - a set of surface feature curves that tesselates surface
- Approach: algebraic, hyperbolic, gap-filling steps
- Algebraic step
  - Auto curves network connectivity
  - Auto grid point distribution
  - Auto grid generation on four-sided domains
  - Auto concatenation sweep
  - Auto boundary extension (work in progress)
  - Significant time saving over current manual practice
- Future plan: automation of hyperbolic and gap filling steps