Parametric Representation and Recasting of Structured Meshes

Cameron Mackintosh, Konstantinos Vogiatzis

Introduction

External aerodynamics simulations often require excessively large meshes. Commercial mesh generation software is inherently limited in maximum mesh sizes due to memory and disk space requirements. This software can't always generate sufficiently fine meshes for a simulation but can usually generate coarse meshes that adequately represent the CAD geometry.

We introduce procedures to represent these coarse meshes with parametric surfaces (and we can trivially extend these procedures to support parametric volumes). These representations are significantly smaller than the coarse meshes, while also sufficiently representing the geometry. We then introduce procedures to generate arbitrarily fine meshes from these parametric representations. These procedures are computationally inexpensive and trivially parallelized across, allowing mesh operations (deformation, refinement, etc) to be distributed across many nodes.

These procedures are then written in Fortran 2003 for integration into the Ablation Fluid Dynamics Library.

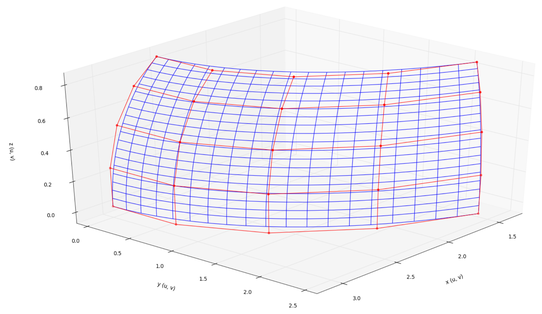

Fig 3. Bézier surfaces' control points fitted to coarse mesh

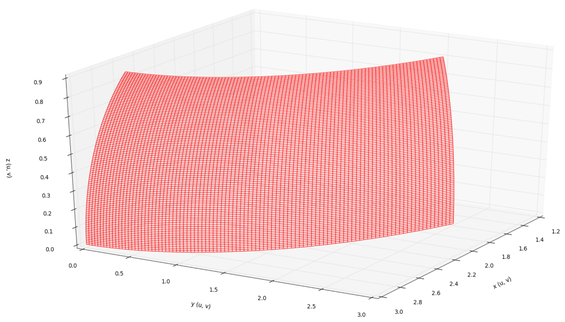

Fig 4. fine mesh generated from control points

Methods

We use Bézier surfaces as our parametric surfaces. A Bézier surface is defined in entirety by its matrices of control points. Points on the surface are represented by their computational space parameters. Any point's real-space coordinates can be calculated from the control points and the point's computational-space parameters via the de Casteljau algorithm or the less efficient Bézier equations (see Fig. 1 and Fig. 2).

We can fit a Bézier surface to a coarse mesh by assigning computational-space values to each of the mesh's points by arc length and solving the Bézier equations as an overdetermined system for the control points via least squares approximation. We can then generate new meshes by applying the de Casteljau algorithm to any computational-space distribution.

$$x(u,v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} (B_{n_u,i}(u) \cdot B_{n_v,j}(v) \cdot \mathbf{X_{i,j}})$$

$$y(u,v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} (B_{n_u,i}(u) \cdot B_{n_v,j}(v) \cdot \mathbf{Y_{i,j}})$$

$$z(u,v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} (B_{n_u,i}(u) \cdot B_{n_v,j}(v) \cdot \mathbf{Z_{i,j}})$$

$$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Fig 1. Bézier equations

$$x(t) = x_0^n(t)$$

$$y(t) = y_0^n(t)$$

$$z(t) = z_0^n(t)$$

$$x_i^k(t) = \begin{cases} (1-t)x_i^{k-1} + t x_{i+1}^{k-1}, & \text{if } k > 0 \\ \mathbf{x_i}, & \text{if } k = 0 \end{cases}$$

$$y_i^k(t) = \begin{cases} (1-t)y_i^{k-1} + t y_{i+1}^{k-1}, & \text{if } k > 0 \\ \mathbf{y_i}, & \text{if } k = 0 \end{cases}$$

$$z_i^k(t) = \begin{cases} (1-t)z_i^{k-1} + t z_{i+1}^{k-1}, & \text{if } k > 0 \\ \mathbf{z_i}, & \text{if } k = 0 \end{cases}$$

Fig 2. de Casteljau algorithm

The computational-space distributions used to generate the new meshes are not quite preserved in the meshes' real-space distributions. This is because the coarse mesh is discreet thus the arc length parameterization is approximate. We maintain a table that maps from the desired computational-space values to the required computational-space values to offset this parameterization error. We fit a Bézier curve to this table. We query the table by applying the de Casteljau algorithm (using the curve's control points) to our desired computational-space distribution to arrive at our required computational-space distribution. We then apply the de Casteljau algorithm (using the surface's control points) to this required computational-space distribution to arrive at our real-space distribution.

Results

We apply our procedures to a portion of a torus (as seen in Fig. 3 and Fig. 4). The initial, coarse mesh has dimensions 20x20. Our new, fine mesh has dimensions 100x100. We will use an arc length table of dimensions 50x50, an order of 5, and a uniform computational-space distribution.

	Average Segment Length Standard Deviation	Sum of Squared Distance to Geometry	Average User-space Runtime (seconds)
Bézier equations and no arc length table	3.015E-02	1.438E-02	2.966E-03
de Casteljau algorithm and no arc length table	3.042E-02	1.372E-02	2.856E-03
Bézier equations and linear arc length table	5.684E-07	1.248E-02	3.115E-02
de Casteljau algorithm and linear arc length table	5.44E-07	1.113E-02	3.107E-02
Bézier equations and fitted arc length table	2.446E-10	1.348E-02	6.820E-02
de Casteljau algorithm and fitted arc length table	2.239E-10	1.275E-02	5.304E-02

We can observe the accuracy improvements of using an arc length table, and even further so when the table is queried as a Bézier curve. The resulting efficiency decreases only slightly. We lastly verify our final code's computational time complexity, ensuring it matches the complexity of our procedures. It does.