2021

Recursive Sub-Division Technique for Higher-Order Pyramid and Prism Isosurface Visualization

VISUALIZING HIGHER-ORDER ELEMENTS

SCOTT T. IMLAY AND OTHERS

Recursive Sub-Division Technique for Higher-Order Pyramid and Prism Isosurface Visualization

Scott T. Imlay[1], Davide E. Taflin[2], and Craig Mackey[3]
Tecplot Inc., Bellevue, WA, 98006

Higher-Order finite-element CFD methods have the potential to reduce the computational cost to achieve a desired solution error. These techniques have been an area of research for many years and are becoming more widely available in popular CFD codes. CFD visualization software is lagging behind the development of higher-order CFD analysis codes. This paper discusses a technique for visualizing isosurfaces in higher-order element solutions with reduced memory usage. The technique recursively subdivides higher-order elements into smaller linear sub-elements where the isosurface can be extracted using standard marching-tets or marching-cubes techniques. Memory usage is minimized by discarding unneeded sub-elements. In a previous paper this technique was demonstrated with higher-order hexahedra and tetrahedra with Lagrangian polynomial basis functions. In this paper, the technique is extended to higher-order pyramids and prisms. The results are compared to other techniques for visualization of higher-order element isosurfaces.

I. Introduction

The use of higher-order (greater than second order) computational fluid dynamics (CFD) methods is increasing. Popular government and academic CFD codes such as FUN3D, KESTREL, and SU2 have released, or are planning to release, versions that include higher-order methods. This is because higher-order accurate methods offer the potential for better accuracy and stability[1], especially for unsteady flows. This trend is likely to continue.

Commercial visual analysis codes are not yet providing full support for higher-order solutions. The CFD 2030 vision states "…higher-order methods will likely increase in utilization during this time frame, although currently the ability to visualize results from higher order simulations is highly inadequate. Thus, software and hardware methods to handle data input/output (I/O), memory, and storage for these simulations (including higher-order methods) on emerging HPC systems must improve. Likewise, effective CFD visualization software algorithms and innovative information presentation (e.g., virtual reality) are also lacking." The isosurface algorithm described in this paper is the first step toward improving higher-order element visualization in the commercial visualization code Tecplot 360.

Higher-order methods can be based on either finite-difference methods or finite-element methods. While some popular codes use higher-order finite-difference methods (OVERFLOW, for example), this paper will focus on higher-order finite-element techniques. Specifically, we will present a memory-efficient recursive subdivision algorithm for visualizing the isosurface of higher-order element solutions. In a previous paper[5] we demonstrated this technique for higher-order tetrahedral and hexahedral elements with Lagrangian polynomial basis functions. In this paper we extend the algorithm to work with pyramid and prism elements with Lagrangian basis functions. The pyramid basis functions are more complicated because they are rational functions instead of simple orthogonal polynomials.

[1] Chief Technology Officer, P.O. Box 52708, Bellevue, WA, Senior Member AIAA.
[2] Senior Software Development Engineer, P.O. Box 52708, Bellevue, WA.
[3] Senior Research Engineer, Research, P.O. Box 52708, Bellevue, WA.

II. Approach

A. Related Work

Nearly all isosurfacing techniques for higher-order finite-elements involve subdivision of the higher-order element into a number of sub-elements which are then processed with standard linear methods. There are a couple of variations on subdivision technique. The simplest is to subdivide a specified number of times, adding nodes via interpolation using the element's basis functions, until the isosurface through the set of linear sub-elements sufficiently approximates the isosurface through the higher-order element. One example of this approach is the work of Remacle et. al.[3] where the local refinement is terminated when the "visualization error" is below a desired threshold. A second approach is by Thompson and Pebay[2] who first add nodes at minima and maxima within the element and on the element faces, and then tesselates the resulting existing and new nodes to get a linear subdivision. This technique is guaranteed to give a topologically correct isosurface, but the error of the isosurface may still be high and the cost of finding minima and maxima is non-trivial.

We use recursive subdivision technique similar to Remacle et. al.[3] However, we further minimize the memory usage by discarding all sub-elements that don't contain the isosurface. This paper is an extension of the algorithm described by Imlay et. al.[5] to higher-order prism and pyramid elements.

B. Higher-Order Hexahedra and Tetrahedra Basis Functions

The Lagrangian basis functions for the hexahedral and tetrahedral element types were covered in a previous paper[5]. This paper describes the Lagrangian basis functions for the prism and pyramid element types.

Prism Basis Functions

The quadratic prism has 18 nodes as shown in figure 1: six nodes at the corners, nine nodes at the edges, and three nodes at the center of the quadrilateral faces. Figure 1 also shows the local coordinate system used to define the basis functions.

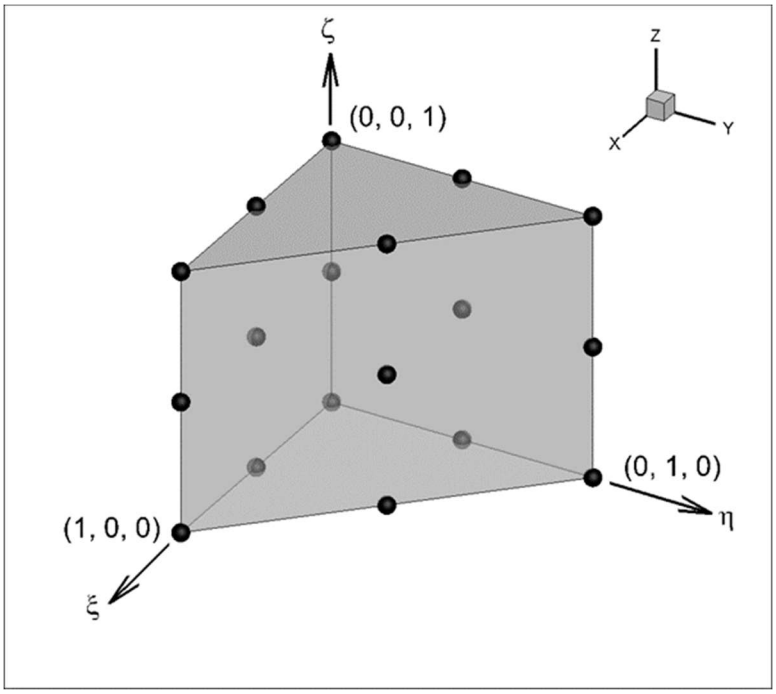

Figure 1. Quadratic Prism

On this element, the solution is approximated by the product of a quadratic triangle basis function in the (ξ, η) plane and a quadratic line basis function in the ς direction.

$$N_p^2(\xi, \eta, \varsigma) = N_t^2(\xi, \eta) N_l^2(\varsigma)$$

Look first at the triangle basis function. We assume a right triangle which can then be mapped into a general curved triangle. The local coordinates for the right triangle are shown in figure 2.

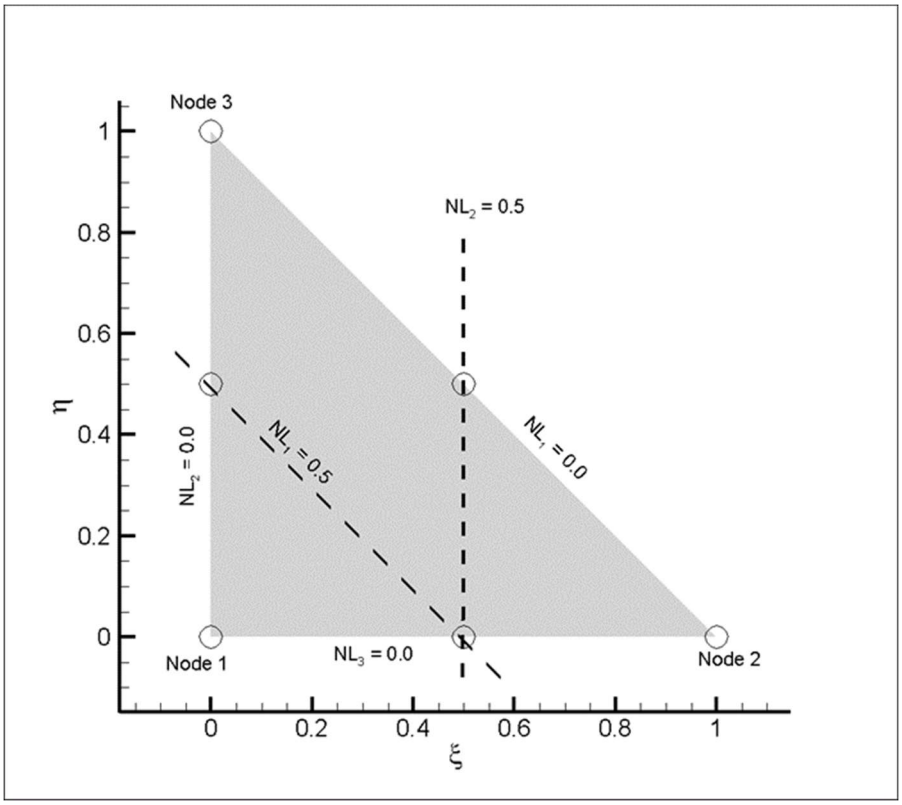

Figure 2. Triangle Face Basis Functions

The quadratic (p=2) basis functions, $N_t^2(\xi, \eta)$, are defined in terms of the linear (p=1) basis functions, $N_t^1(\xi, \eta)$, which will be described first. The linear basis functions for each node vary linearly from one at that node to zero at the other two nodes. So, for node 1, N_{t1}^1 is one at node 1 and zero at nodes 2 and 3. Mathematically:

$$\begin{aligned} N_{t1}^1 &= 1 - \xi - \eta \\ N_{t2}^1 &= \xi \\ N_{t3}^1 &= \eta \end{aligned}$$

The same rules apply for the quadratic basis function: the basis function for a node is one at the node and zero at all other nodes. This is satisfied by the equations:

$$\begin{aligned} N_{t1}^2 &= 2N_{t1}^1(N_{t1}^1 - 0.5) \\ N_{t2}^2 &= 2N_{t2}^1(N_{t2}^1 - 0.5) \\ N_{t3}^2 &= 2N_{t3}^1(N_{t3}^1 - 0.5) \\ N_{t4}^2 &= 4N_{t1}^1 N_{t2}^1 \\ N_{t5}^2 &= 4N_{t2}^1 N_{t3}^1 \\ N_{t6}^2 &= 4N_{t1}^1 N_{t3}^1 \end{aligned}$$

The quadratic basis functions for the line in the ς direction (figure 1) are as follows for nodes on the bottom ($\varsigma = 0$), middle ($\varsigma = {}^{1}/_{2}$), and top ($\varsigma = 1$) of the prism.

$$N_{lb}^2 = 2(1-\varsigma)(0.5-\varsigma)$$
$$N_{lm}^2 = 4\varsigma(1-\varsigma)$$
$$N_{lt}^2 = 2\varsigma(0.5-\varsigma)$$

The prism basis function for a node is the product of the triangle basis function for the node within the triangle with the appropriate line basis function (bottom, middle, or top).

Pyramid Basis Functions

The basis functions for the pyramid are more complicated than for the other element types. This is because the variation of the solution along the triangular faces must match the distribution along the faces of adjacent tetrahedra and, at the same time, the variation of the solution along the quadrilateral face must match the distribution along the face of an adjacent hexahedron. This can't be done with simple orthogonal polynomial basis functions, so we use rational polynomials for the pyramid basis functions. We use the basis functions of Chan and Warburton[4].

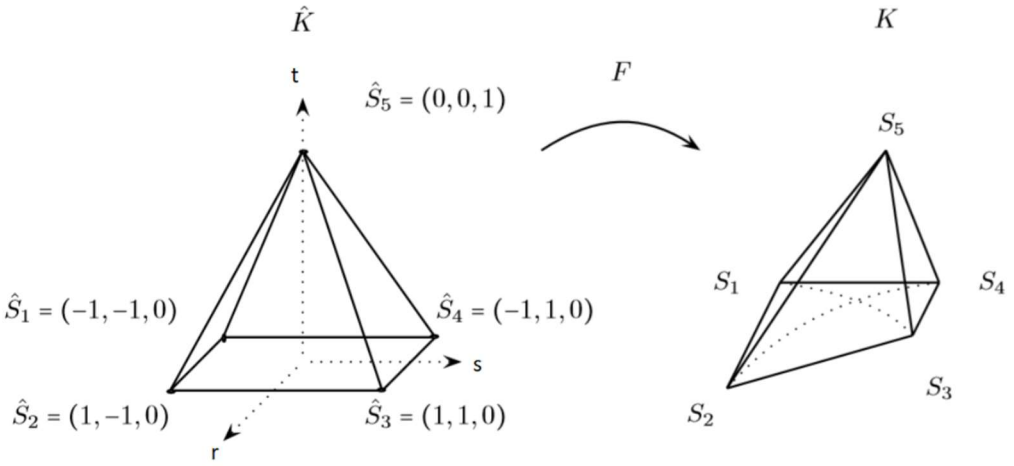

Figure 3. Pyramid Local Coordinates.

The local coordinates for the pyramid are shown in figure 3. Given this, the basis functions are

$$P_{ijk}(r,s,t) = P_i^{0,0}\left(\frac{r}{1-t}\right) P_j^{0,0}\left(\frac{s}{1-t}\right)(1-t)^c P_k^{2(c+1),0}(2t-1),$$

where

$$c = \max(i,j), \quad 0 \le i,j \le N, \quad 0 \le k \le N-c.$$

and $P_n^{a,b}(x)$ is the Jacobi polynomial of order n, orthogonal with respect to the weight $(1-x)^a(1+x)^b$. For a given N, the above procedure produces N_p distinct basis functions, where

$$N_p = \frac{(N+1)(N+2)(2N+3)}{6}$$

We may order these orthogonal functions arbitrarily as $\phi_j(r,s,t)$ from $j = 1,\ldots,N_p$.

The above are modal basis functions, meaning that the coefficients of the polynomial basis functions are not the solution at the nodes. We wish to have a nodal basis function.

Using the above basis functions, we may define the generalized Vandermonde matrix

$$V_{ij} = \phi_j(r_i, s_i, t_i), \quad i = 1, 2, \ldots, N_p$$

where (r_i, s_i, t_i) are a set of nodal points contained inside or on the boundary of the reference pyramid. Then, a nodal basis may be constructed using elements of the inverse of the Vandermonde matrix

$$\ell_i(r, s, t) = \sum_{j=1}^{N_p} (V^{-1})_{ij} \phi_j(r, s, t).$$

C. Subdivision into Linear Sub-Elements

The subdivision for each higher-order pyramid or prism element is as follows:

1. Break it into the logical set of linear sub-tetrahedra.
2. If the error too large, create new edge, face, and face nodes and subdivide the tetrahedron into 8 sub-tetrahedra as shown in figure 4. Interpolate the solution to these new nodes using the original prism or pyramid basis functions.
3. Repeat until the error is less than a predefined threshold.

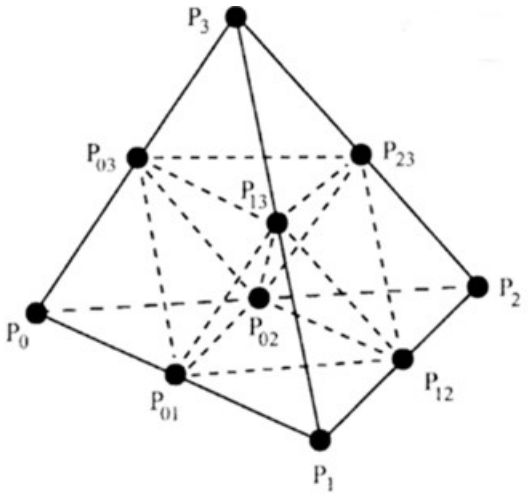

Figure 4. Subdivision of Quadratic Tetrahedron

While the process is the same for pyramids and prisms, the initial subdivision is different.

Prism Subdivision:

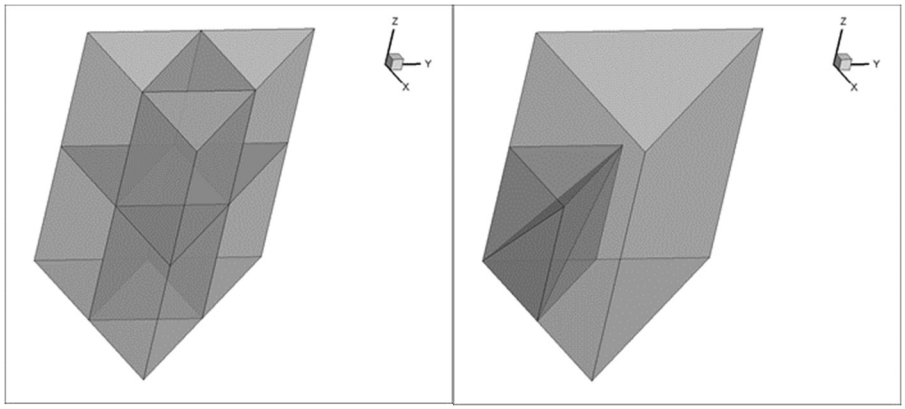

Figure 5. Subdivision of Quadratic Prism into Linear Tetrahedra

The quadratic prism has nodes at the corners, edge mid-point, and the center of the quadrilateral faces. It can be subdivided naturally into eight linear prisms as shown in figure 5, and each sub-prism can be subdivided into three linear tetrahedra, yielding 24 linear sub-tetrahedra for each quadratic prism.

Pyramid Subdivision:

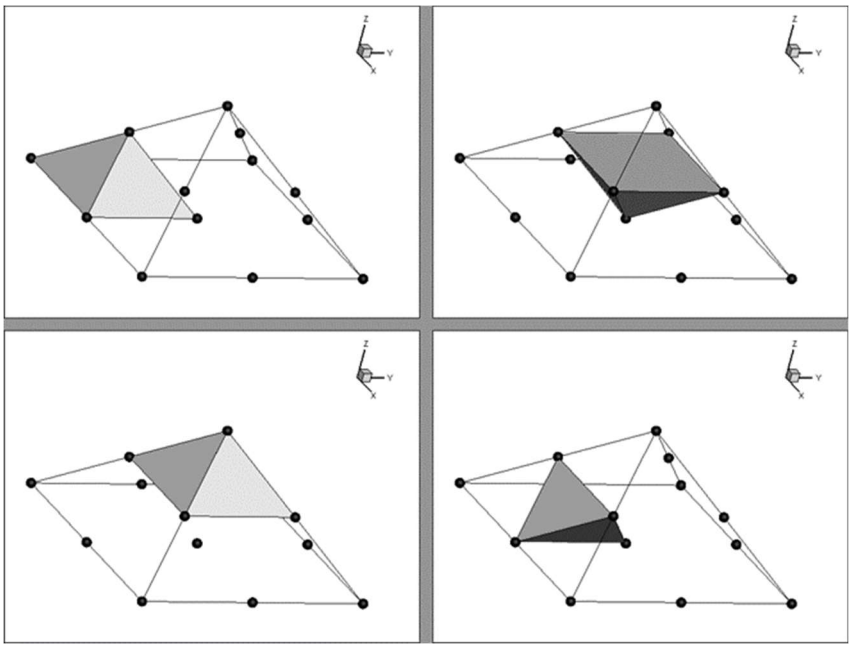

Figure 6. Subdivision of Quadratic Pyramid into Linear Tetrahedra

As shown in figure 6, the quadratic pyramid naturally subdivides into eight linear pyramids and four linear tetrahedra. Each linear pyramid can be further subdivided into two linear tetrahedra, yielding a total of twenty sub-tetrahedra for the quadratic pyramid.

III. Results

The new isosurface algorithm has been applied to arrays of 18-node quadratic prisms and 14-node quadratic pyramids to produce isosurfaces of simple polynomial functions.

The example is a spherical isosurface using both quadratic prisms and pyramids. The quadratic prism result is shown in figure 7. This is with just 54 quadratic prism elements and four levels of subdivision. Note that the selective subdivision only needed 2,353 linear tetrahedra whereas the full subdivision would have created 663,552 tetrahedra – a factor of 280 reduction in memory.

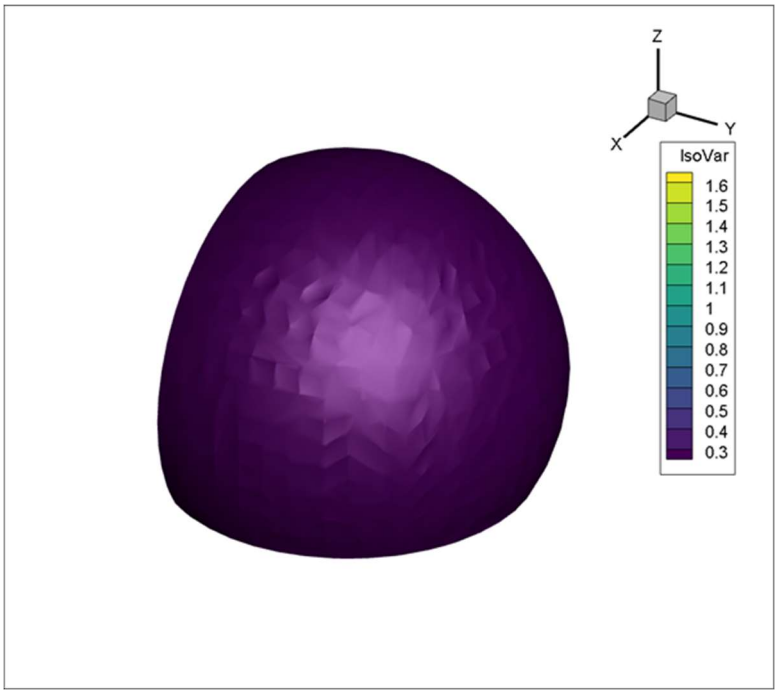

Figure 7. Isosurface for Quadratic Prisms with 4 Levels of Selective Subdivision

The higher-order prism isosurface algorithm also works for curved elements as shown in figure 8.

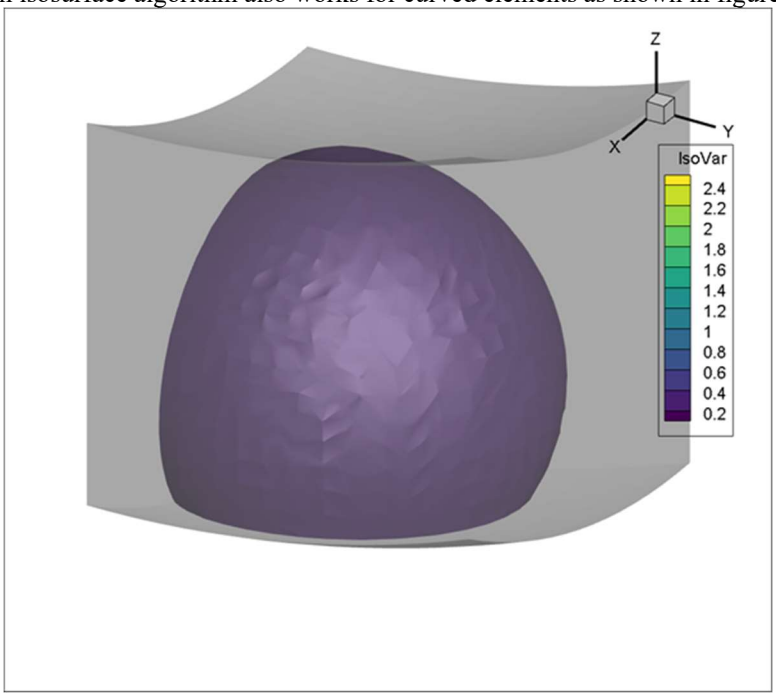

Figure 8. Isosurface from Curved Quadratic Prisms

For the same spherical isosurface, the quadratic pyramid isosurface is shown in figure 9. This is with just 54 quadratic prism elements and four levels of subdivision. Note that the selective subdivision only needed 3,804 linear tetrahedra whereas the full subdivision would have created 1,658,880 tetrahedra – a factor of 436 reduction in memory.

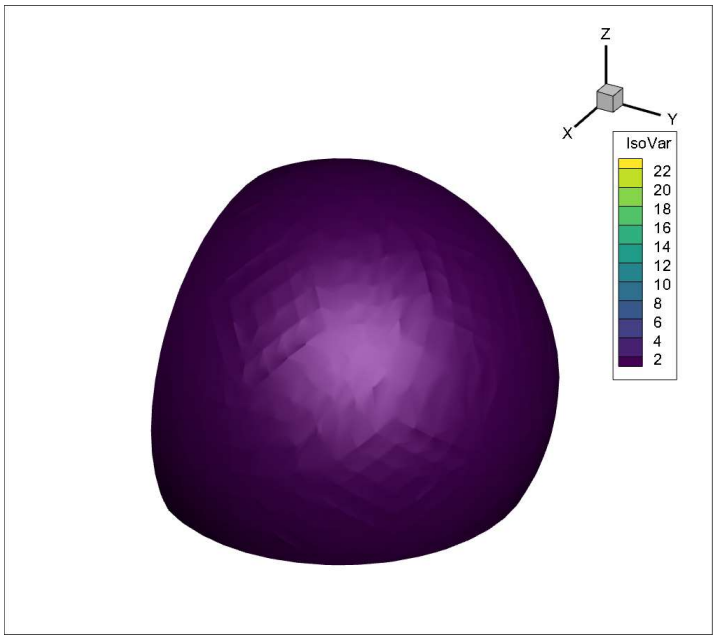

Figure 9. Isosurface for Quadratic Pyramids with 4 Levels of Selective Subdivision

IV. Conclusions

A recently developed recursive subdivision algorithm[5] to compute isosurfaces for higher-order element solutions has been extended to work with higher-order pyramid and prism element types. The algorithm minimizes memory usage by keeping only sub-elements that contain the isosurface. The benefits of the algorithm have been demonstrated for Lagrangian quadratic elements and simple functions resulting in spherical isosurfaces.

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 Data Science," *NASA/CR-2014-218178*, 2014.

[2]Thompson, D.C., and Pebay, P.P., "Visualizing Higher Order Finite Elements: Final Report," SAND2005-6999, Nov. 2005.

[3]Remacle, J.-F., Chevaugeon, N., Marchandise, E. and Geuzaine, C., "Efficient visualization of high-order finite elements," Int. J. Numer. Meth. Engng, Jul. 2006.

[4]Chan, J. and Warburton, T., "A Comparison of High-Order Interpolation Nodes for the Pyramid," SIAM J. Sci. Computing, Dec. 2014.

[5]Imlay, S., Taflin, D., and Mackey, C., "Recursive Sub-Division Technique for Higher-Order-Element Isosurface Visualization," to be presented at AIAA AVIATION Forum, Jun. 2020.