# Recursive Sub-Division Technique for Higher-Order-Element 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. The technique is demonstrated with higher-order hexahedra and tetrahedra with Lagrangian polynomial basis functions. The results are compared to other techniques for visualization of higher-order element isosurfaces.**

[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.

## I. Introduction

The use of higher-order (greater than second order) 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 visions 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. While the algorithm should work for virtually any element type, this paper will focus on tetrahedral and hexahedral elements with Lagrangian polynomial basis functions.

## 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 linear elements. There are a couple of variations on subdivision technique. The simplest is to subdivide a 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 tesselate 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.

### B. Higher-Order Hexahedra and Tetrahedra Basis Functions

The method is currently applied to hexahedra and tetrahedra with Lagrangian basis functions. This leads to a relatively simple formulation for the basis functions. Consider first a quadratic (27-node) hexahedron. The element has a local coordinate system $(\xi, \varepsilon, \zeta)$ centered on the cell-centered node and ranging from -1 to 1 in each of the three coordinate directions.

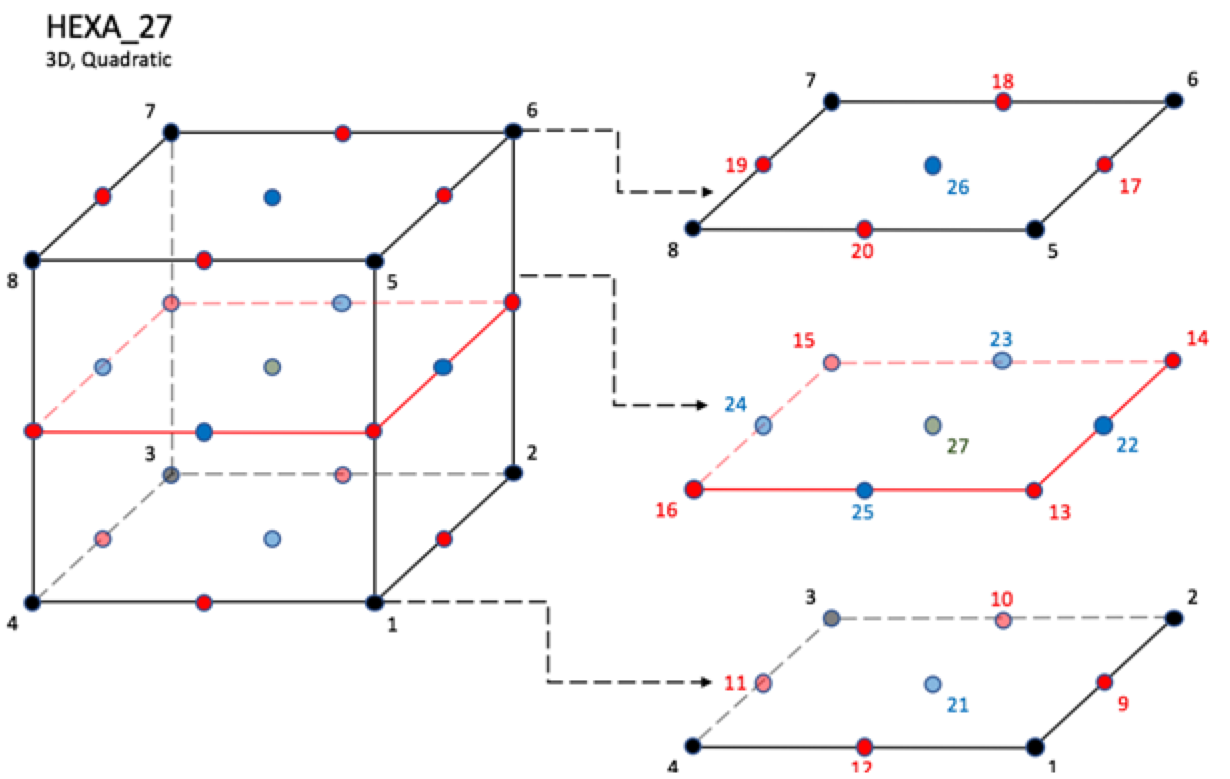


*Figure 1. 27-node hexahedron.*

If we consider a three-index node numbering

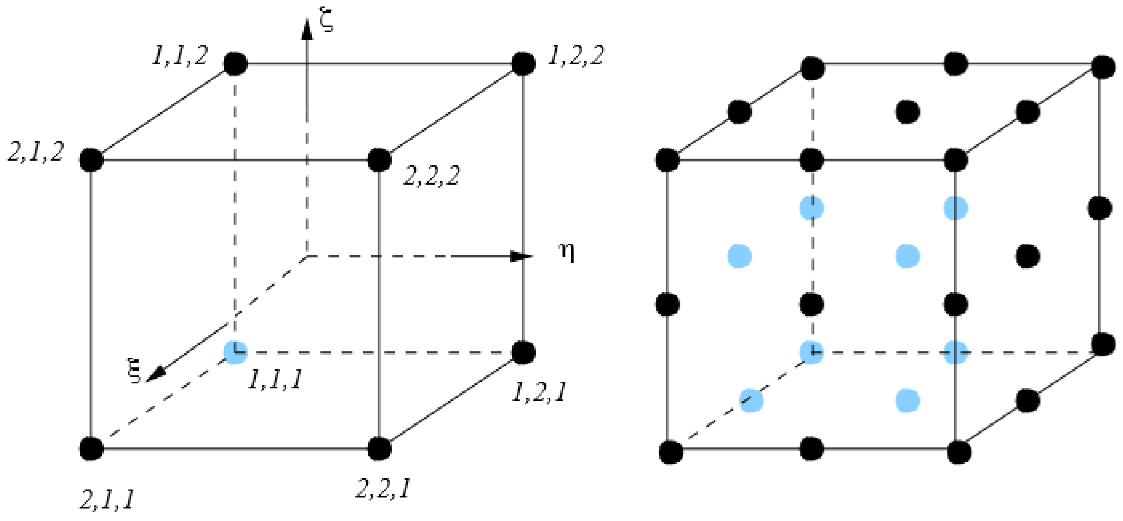


*Figure 2. Three-index node numbering.*

On this element, the solution is approximated by quadratic basis functions of the local coordinates

$$U(\xi,\eta,\zeta) = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} U_{i,j,k} N_{i,j,k}(\xi,\eta,\zeta)$$

where $N_{i,j,k}$ are the basis function that are one at the specified node and zero at the other nodes. For Lagrangian basis functions these are the product of the corresponding 1-D basis functions in each of the coordinate directions.

$$N_{i,j,k}(\xi,\eta,\zeta) = \bar{N}_i(\xi)\bar{N}_j(\eta)\bar{N}_k(\zeta)$$

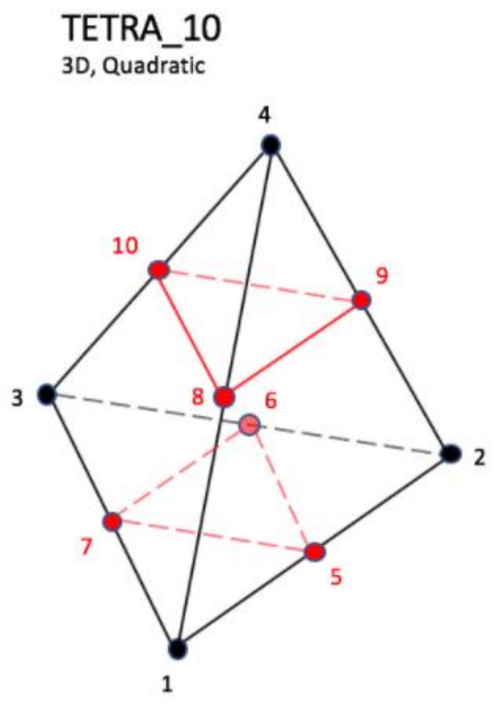
and

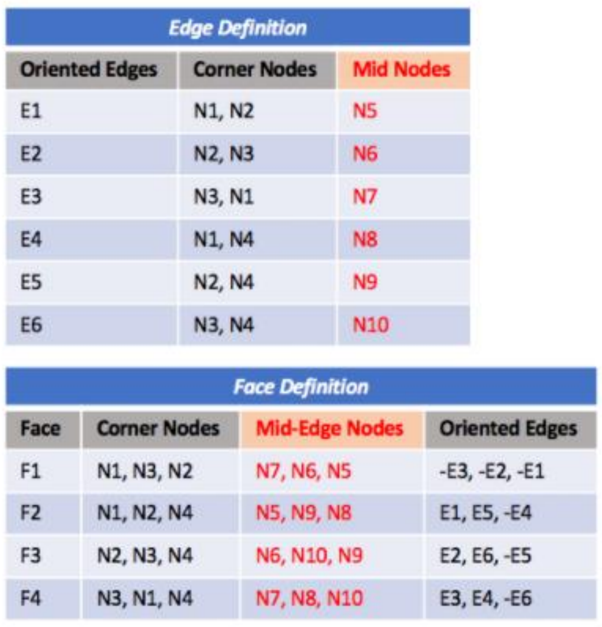
$$\bar{N}_1(\xi) = -\frac{\xi(1-\xi)}{2}$$
$$\bar{N}_2(\xi) = \frac{\xi(1+\xi)}{2}$$
$$\bar{N}_3(\xi) = (1-\xi^2)$$

Each increment in polynomial order adds a node in each of the coordinate directions.

The Lagrangian quadratic tetrahedron is shown in Figure 3.

TETRA_10





| Edge Definition | | |
|---|---|---|
| Oriented Edges | Corner Nodes | Mid Nodes |
| E1 | N1, N2 | N5 |
| E2 | N2, N3 | N6 |
| E3 | N3, N1 | N7 |
| E4 | N1, N4 | N8 |
| E5 | N2, N4 | N9 |
| E6 | N3, N4 | N10 |

| Face Definition | | | |
|---|---|---|---|
| Face | Corner Nodes | Mid-Edge Nodes | Oriented Edges |
| F1 | N1, N3, N2 | N7, N6, N5 | -E3, -E2, -E1 |
| F2 | N1, N2, N4 | N5, N9, N8 | E1, E5, -E4 |
| F3 | N2, N3, N4 | N6, N10, N9 | E2, E6, -E5 |
| F4 | N3, N1, N4 | N7, N8, N10 | E3, E4, -E6 |

*Figure 3. Lagrangian quadratic tetrahedron.*

The basis functions for each node are quadratic functions of $(\zeta_1, \zeta_2, \zeta_3, \zeta_4)$ defined by

$$\zeta_1 = \frac{V_{P234}}{V_{1234}}$$
$$\zeta_2 = \frac{V_{P134}}{V_{1234}}$$
$$\zeta_3 = \frac{V_{P124}}{V_{1234}}$$
$$\zeta_4 = \frac{V_{P123}}{V_{1234}}$$

Where $V_{P234}$, for example, is the volume of the sub-tetrahedra composed of nodes P (the point you are interpolating to), 2, 3, and 4.

## C. Subdivision into Linear Sub-Elements

The subdivision for each higher-order hexahedron is to

- break it into the logical set of linear sub-hexahedra,
- if the error too large create new edge, face, and volume nodes and subdivide as before
- repeat until the error is less than a predefined threshold.

For example, each 27-node quadratic hexahedron is divided into eight candidate sub-hexahedra. If the error is low enough assuming these hexahedra are linear, we are done. Otherwise, new edge, face, and cell centered nodes are created via interpolation by the basis functions and each sub-hexahedron is subdivided into eight sub-sub-hexahedra. The process continues until the desired error is achieved. This is essentially the same algorithm used by Remacle et. al.[3]

The unique feature of our algorithm is that we keep only the sub-hexahedra (or sub-sub-hexahedra, or sub-sub-sub-hexahedra, etc.) that contain the isosurface value. As a result, the added memory requirement may be orders of magnitude less than if the subdivided hexahedra had been saved for all elements.

A very similar subdivision algorithm is applied to higher-order tetrahedra, except that the subdivision is somewhat more complicated. Figure 4 shows how each 10-node quadratic tetrahedron can be subdivided into eight linear tetrahedra.

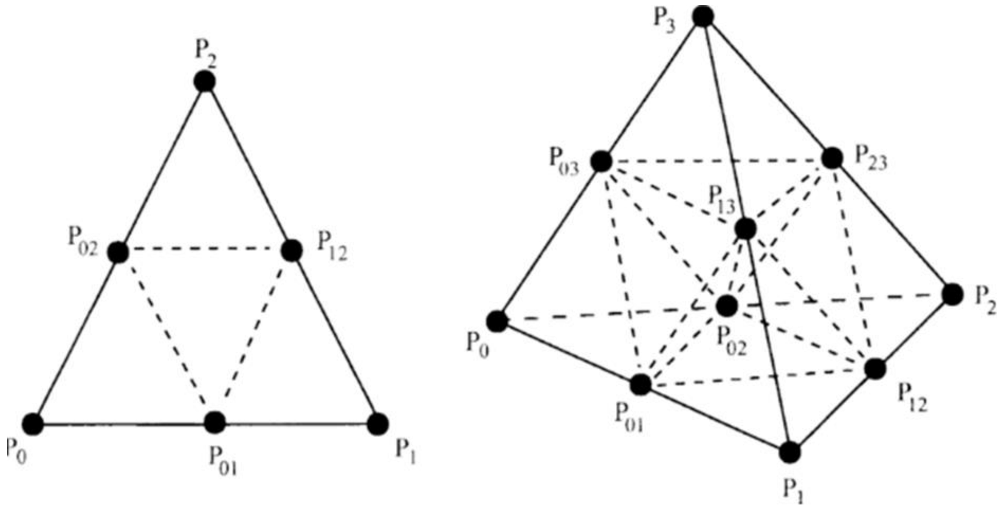


*Figure 4. 10-node quadratic tetrahedron subdivided into eight linear tetrahedra.*

## III. Results

To date, the new isosurface algorithm has been applied to arrays of 27-node quadratic hexahedra and 10-node quadratic tetrahedra with simple polynomial functions, and to a jet-flow CFD solution with 27-node quadratic hexahedra.

The first example is a spherical isosurface using both quadratic hexahedra and tetrahedra. Figure 5 show the results for an array of 27-node quadratic hexahedra. The upper left shows the nodal distribution, the upper right shows what you get if the extra nodes are ignored and the element is assumed to be linear. This is a common approximation when HOE results are visualized using tradition visual analysis codes. Another common approximation is to do the obvious subdivision on the 27-node element (breaking it into 8-linear cells) as shown in the lower left frame. The isosurface created using the new algorithm is shown in the lower right frame. It accurately represents the expected spherical surface.

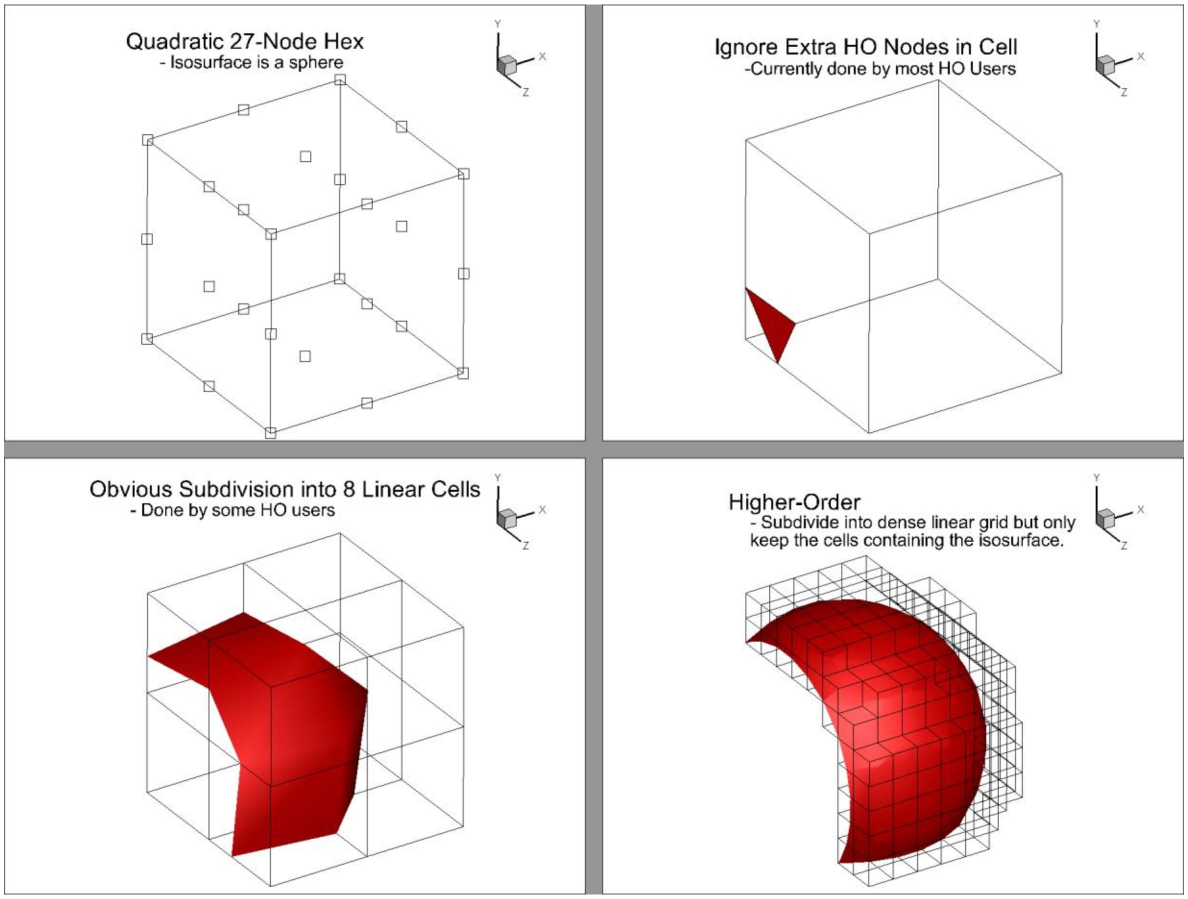

Quadratic 27-Node Hex
- Isosurface is a sphere
Ignore Extra HO Nodes in Cell
-Currently done by most HO Users
Obvious Subdivision into 8 Linear Cells
- Done by some HO users
Higher-Order
- Subdivide into dense linear grid but only keep the cells containing the isosurface.
X
Y
Z

*Figure 5. Results for an array of 27-node quadratic hexahedra.*

The same test was done with quadratic tetrahedra. Figure 6 compares the isosurface generated using the new algorithm at various levels of refinement to that created assuming a linear element (throwing away the extra elements). The linear tetrahedra case, throwing out the extra higher-order nodes, is clearly a poor representation of the actual spherical isosurface. Each successive refinement improves the isosurface and four levels of refinement is very close to the actual sphere. Note our selective algorithm discarded 98.7% of the created sub-cells and is very memory efficient.

Finally, we have one actual CFD results – vorticity isosurfaces for a jet. The grid is 15x15x15 27-node quadratic hexahedra. Figure 8 shows the results with the common practice of ignoring the higher-order nodes, the less common practice of subdividing the hexahedra into eight linear hexahedra so that the higher-order nodes are used, and the higher-order isosurface obtained with three levels of subdivision. In this case, only 132,889 sub-cells were needed – just 3.9% of the 3.375 million sub-cells that a non-selective subdivision all cells would have created.

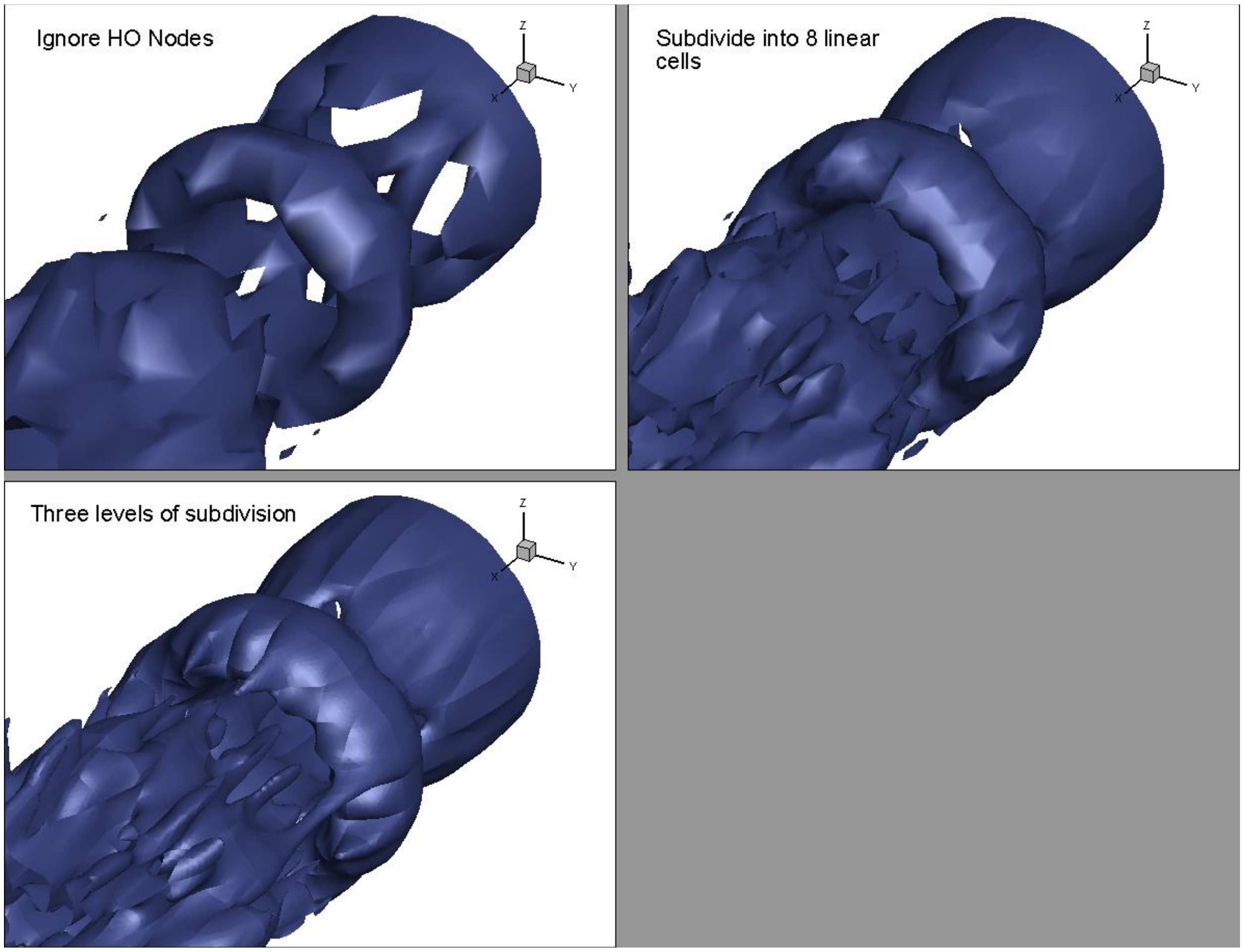


*Figure 8. Higher-order results compared with the common practice of ignoring the higher-order nodes.*

## IV. Conclusions

A new recursive subdivision algorithm has been developed to compute isosurfaces for higher-order element solutions. 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. The benefits of the new algorithm were also demonstrated for higher-order hexahedral CFD results.

## V. 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.