High-order visualization with ElVis

J. Peiro, D. Moxey, B. Jordi, S.J. Sherwin, B.W. Nelson, R.M. Kirby and R. Haimes

Abstract Accurate visualization of high-order meshes and flow fields is a fundamental tool for the verification, validation, analysis and interpretation of high-order flow simulations. Standard visualization tools based on piecewise linear approximations can be used for the display of high-order fields but their accuracy is restricted by computer memory and processing time. More often than not, the accurate visualization of complex flows using this strategy requires computational resources beyond the reach of most users. This chapter describes ElVis, a truly high-order and interactive visualization system created for the accurate and interactive visualization of scalar fields produced by high-order spectral/*hp* finite element simulations. We show some examples that motivate the need for such a visualization system and illustrate some of its features for the display and analysis of simulation data.

1 Introduction

High-order simulations are now becoming increasingly common and codes for producing high-order solutions are advancing at a rapid rate but the tools which scientists and engineers use to visualise both their solution data and the underlying high-order geometry on which the problem is solved are still firmly embedded in the world of traditional linear finite elements. Whereas high-order methods typi-

J. Peiro, D. Moxey, B. Jordi and S.J. Sherwin
Department of Aeronautics, Imperial College London, UK
e-mail: j.peiro,d.moxey,b.jordi11,s.sherwin@imperial.ac.uk

B.W. Nelson and R.M. Kirby
School of Computing, University of Utah, USA
e-mail: bnelson,kirby@cs.utah.edu

R. Haimes
Department of Aeronautics and Astronautics, MIT, USA
e-mail: haimes@mit.edu

cally utilise non-linear polynomials (or other similar non-linear functions) to represent the solution which has been obtained on a given element, the vast majority of visualization software assumes that the data are provided in a linear format, and therefore a conversion is necessary before the data can be viewed.

At worst this poses a major challenge for the adoption of high-order methods and at best does not show the full potential of such methods, for both the developers and end-users of high-order software. From the perspective of a user, whilst it is certainly possible to obtain a visualization of a given simulation, the error that is present when the solution is interpolated to a set of linear functions may yield visualization artifacts which may or may not be present in the actual solution field. For developers this may result in an additional level of debugging which must be undertaken, since it is not always clear whether these errors occur from the numerical properties of a given scheme or indeed from a mistake within the code itself.

Additionally, from the perspective of mesh generation, it is clear that high-order meshes are not just desirable but necessary in order to obtain accurate and meaningful solutions. However, without the means of visualising the 'true' surface of a mesh and not its linear interpolation, it is either very difficult or impossible to predict where simulations may encounter issues when considering complex geometries.

Solutions to this problem do already exist. A naive solution is to simply oversample a mesh, using a large number of linear functions to represent the high-order data. This approach may also be considered in combination with an adaptive technique where the mesh is subdivided until a tolerance between the linear interpolation and high-order data is achieved. However, such approaches are expensive in terms of computational time meaning that they may not be interactive with the user, and additionally require large amounts of storage and memory in which to operate.

The purpose of this chapter is to give a brief overview of the Element Visualiser (ElVis) [7], which aims to address these issues through a novel approach that provides interactive and accurate visualization of high-order simulations[1]. The simulations are represented by spectral/hp finite element fields that are produced by the continuous or discontinuous Galerkin methods. ElVis was originally created by B.W. Nelson, as part of his PhD at the University of Utah [3]. Currently ElVis is actively maintained by R.M. Kirby at the University of Utah and R. Haimes at MIT, and has been the focus of our development efforts on high-order visualization throughout the course of IDIHOM. We outline the features of ElVis and give examples which demonstrate its advantages over traditional linear visualization software.

2 The importance of visualization accuracy

The main reason for the development of an accurate high-order visualizer is that linear approximations where high-order data is sampled onto linear primitives for visualization does not represent high-order data well and results in visualization

[1] The program is accessible at `www.sci.utah.edu/download/elvis`

errors. We illustrate this with two examples where we compare high-order and linear visualizations[2].

2.1 Visualization errors or artifacts?

We consider the simulation of the compressible viscous flow past a half delta-wing geometry with a symmetry plane running down the centre chord-line of the wing. The delta-wing geometry is linear and the computational mesh consists of 19410 quadratic tetrahedral elements with straight edges. The numerical solution has been obtained via a discontinuous Galerkin solver[3]. Figure 1 depicts a linear visualization of the solution on a plane cut located at the trailing edge of the wing. The two insets in the figure highlight areas where visualization errors or artifacts might be present. To determine the nature of these visual features requires an increase of the accuracy of the visualization.

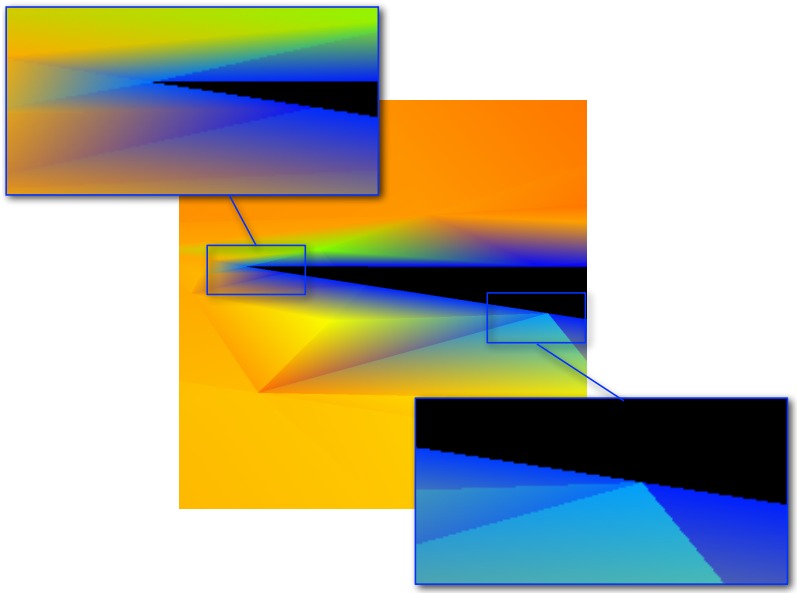

Fig. 1 Cut-surface at the trailing edge of a delta wing simulation displayed using a piecewise linear approximation. The two inset figures show enlargements of the areas where visualization errors or artifacts might have occurred.

[2] All the linear visualizations in this chapter have been performed with the code Visual 3 which is available at `raphael.mit.edu/visual3`

[3] Numerical solution computed by the Project X code: `raphael.mit.edu/projectx.html`

Focusing in the area near the edge of the wing, Figure 2 depicts a comparison of the linear visualization with a pixel exact display of the solution using ElVis which shows that the visual effect of the linear visualization is due to a lack of resolution of the boundary layer.

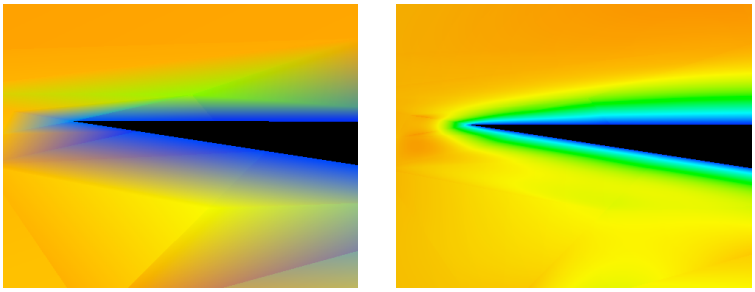

Fig. 2 Visualization errors: the linear visualization (left) shows apparent under-resolution of the boundary layer, whereas in reality the boundary layer is clearly present (right).

On the other hand, the comparison of linear and high-order visualizations in Figure 3 illustrate that the sharp changes observed in both are due to the discontinuous Galerkin approximation employed by the solver.

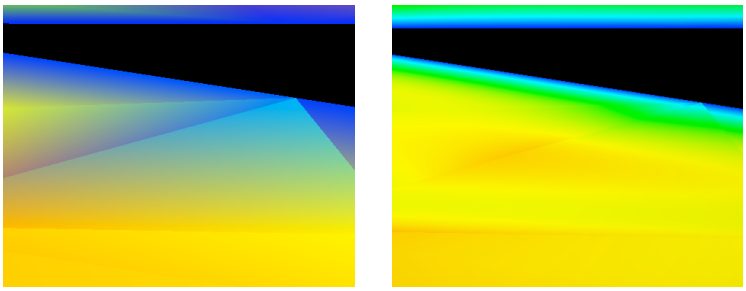

Fig. 3 The sharp changes observed in both the linear visualization (left) and the high-order visualization (right) are part of the piecewise discontinuous approximation of the solution.

2.2 Visualization for code debugging

Visualization tools can play a role in detecting bugs and errors within the solver. Figure 4 shows several visualizations of isocontours of the distance function $d(\vec{x})$, a

scalar function which defines the minimum distance between a point $\vec{x}$ and a surface S. This is an important function used in, for example, turbulence models which rely on this quantity to in order to estimate stresses on the flow due to viscous effects near walls. Errors in this function may, in the worst cases, substantially affect physical observations or indeed prevent the simulation from converging; in other cases however the effect can be minimal or non-existent.

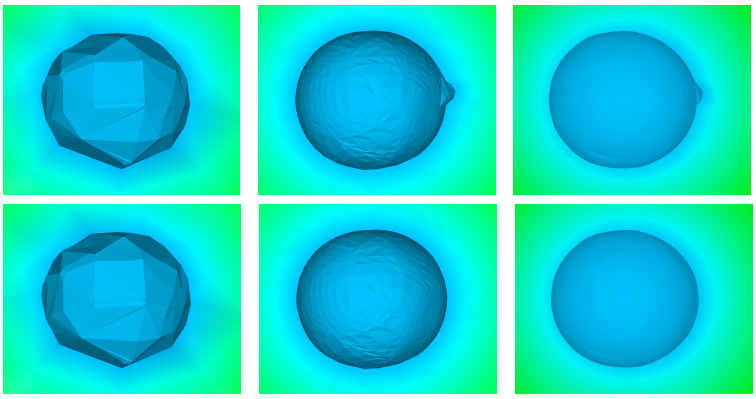

Fig. 4 visualizations indicating a bug relating to the distance function. The top row shows a version of the code where the bug is present, and the bottom where the bug is fixed. From left-to-right: linear interpolation of elements, linear interpolation with one level of refinement, ElVis. Note that the linearly interpolated elements without refinement appear identical.

Figure 4 shows how the use of linear visualization tools can prevent the detection of bugs in distance calculation routines. The top row shows results with a code which has a bug in its distance calculation routine, and the bottom where the offending bug has been fixed. On the left-hand side we see that the visualization is sufficiently coarse that no problem is detected even after the bug has been fixed. When the linear mesh is refined (middle), shows a small protrusion – however, the shape can still be inferred to be reasonable given the original linear image. ElVis on the other hand shows an obvious protrusion from an otherwise smooth surface. This example clearly demonstrates how using only linear tools to visualise data can lead to issues not being detected.

3 ElVis and its main features

The previous examples have shown that linear visualization tools operating on high-order data can introduce errors. The main motivation for the development of a truly high-order visualizer is the realization that these errors can be minimized if we take into account our knowledge of the data. In recent years, a series of tools have been

developed in an attempt to address the lack of appropriate visualization software for high-order data. A key issue with these tools is that they have been developed with the presumption of a single type of basis function to be used in each element. Whilst a conversion of high-order data may be possible from one data format to another if each tool assumes a polynomial expansion, the wide variety of non-linear basis functions that can be employed in high-order methods means that this is not always possible, and indeed for users can be a time-consuming task.

A further issue with existing visualization software is that if often comes with the drawback of high computational cost. Whilst it is clear that high-order visualization demands additional processing when compared to linear interpolation, existing software can require the use of expensive custom-made hardware in order to produce visualizations in reasonable amounts of time for users, particularly for large three-dimensional problems. Since interactivity is a key part of the visualization process, this is a significant problem which must be overcome.

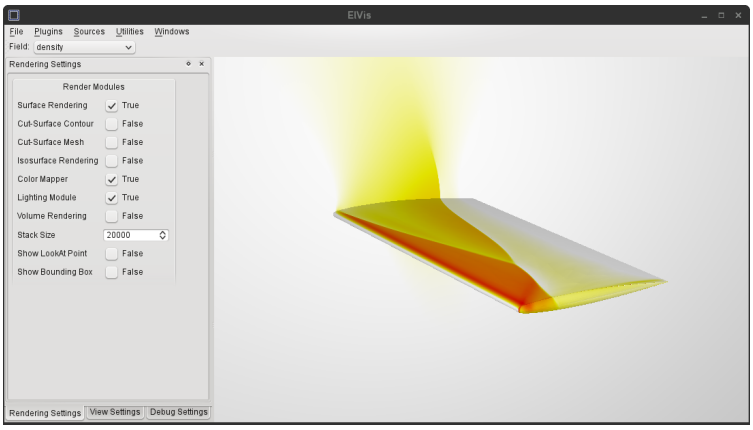

Fig. 5 ElVis' graphical user interface, with a visualization of density on an ONERA M6 wing.

ElVis has been designed to address each of these issues. Firstly, its design is deliberately modular, with the intention that the code responsible for visualization should be separated from the implementation and definition of the high-order basis functions. The way that this is achieved is through the use of an extensible plugin architecture that provides a simplified interface to the high-order data which exposes a minimal amount of functionality required to generate a visualization. This makes ElVis broadly applicable to a wide range of tasks in high-order visualization. The main modules of ElVis are: a *Plugin Module* that handles communication between ElVis and the simulation package, a *Visualization Module* that performs the visualization of the high-order data and a *Data Exploration Module* that gives users the ability to interactively explore the high-order data. The graphical user interface for the visualization and data exploration modules of ElVis is shown in Figure 5.

In order to realise the goal of performance and interactivity, ElVis utilises NVIDIA GPU hardware in combination with CUDA and the OptiX ray-tracing engine in order to provide real-time visualizations. Once data is available within ElVis, either through a data conversion or runtime plugin, selected GPU algorithms for the most common tasks of cut-surfaces/contour lines, volume rendering and isosurface generation have been generated in such a way as to obtain the required visualization accuracy from the underlying high-order data.

3.1 High-order elements

The numerical solution in ElVis is represented by high-order spectral/*hp* finite element approximations via continuous or discontinuous Galerkin methods. For these methods, the computational domain is tessellated into elements. The four basic element types are the hexahedron, prism, tetrahedron, and pyramid. These elements are defined through a mapping, $T : R^3 \rightarrow R^3$, between a standard reference element, the cube $\Omega_{st} : [-1,+1]^3$, in a *reference space* and the high-order element, Ω^e, in a *physical space* as illustrated in Figure 6.

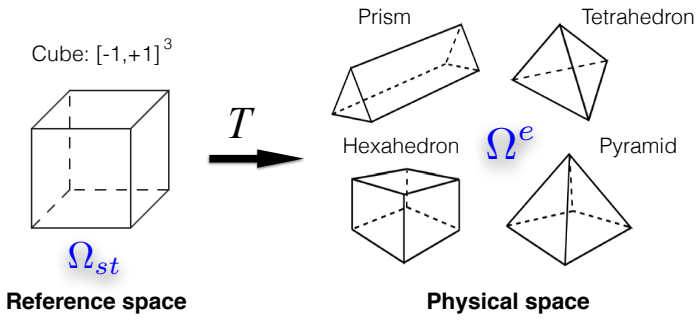

Fig. 6 Notation used for spectral/*hp* finite elements in ElVis.

The high-order representation of the solution within the finite element is a polynomial function evaluated in tensor-product fashion in the reference element. The expressions for the nodal and modal expansions used to approximate the solution within the elements are given in [1][4]. If the high-order elements employed by the flow solver have different polynomial basis functions then a data conversion process can be performed to the set of spectral/*hp* standard basis functions. This minimises the amount of coding required. Otherwise, more complex functions can be implemented directly as a runtime plugin.

[4] These are implemented in the open-source code Nektar++ available at `www.nektar.info`

ElVis uses the representation of the flow field via the spectral/*hp* basis functions and applies ray-tracing ideas to achieve a pixel-exact images in which the visualization algorithm is applied to each pixel. An overview of how ElVis achieves pixel-exact images is given in the following sections.

3.2 Isosurfaces

Perhaps one of the most often used visualizations of a data set is the isosurfaces or contour lines which depict where a given field attains a particular value. ElVis adopts an elemental approach to isosurface rendering [2, 5], whereby the field is projected onto a polynomial as a ray passes through the volume so that finding the isosurface becomes a root finding problem. This is illustrated in Figure 7.

The solution along the ray, assumed to be smooth within the element in the physical space, is approximated by a Legendre polynomial of order M. Given the entry and exit points of the ray within the element, a set of Legendre-Gauss-Lobatto quadrature points is located along the ray. The number of quadrature points is a function of the approximating polynomial order and it is chosen so that inner products of polynomials of degree M are within round-off error. The high-order finite element function is evaluated at the quadrature points along the ray and these values are used to interpolate a polynomial. The roots of this polynomial which correspond to the sought values of the isosurface are evaluated using the QR root-finding algorithm. This type of projection introduces error, but it converges to the field as we increase the order, so we can increase the order until we achieve pixel-exact images. A more detail description of the algorithm and analysis of the associated errors in given in [5].

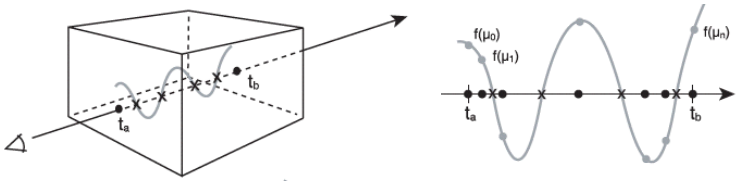

Fig. 7 Isosurface evaluation within an element in ElVis. A projected polynomial is formed along the ray, the scalar field within the element is sampled along the ray to interpolate a polynomial and its roots give the values of the isocontour sought.

One of the advantage of this method over more traditional isosurface rendering techniques such as the marching cubes algorithm is that it permits the visualization of discontinuities within the isosurfaces across element boundaries if this is present in the underlying formulation. An example of this can be seen in Figure 8 where the inset figure shows the discontinuities in the numerical solution.

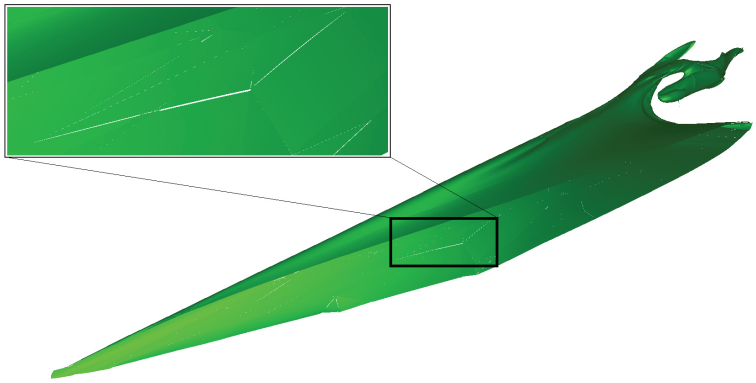

Fig. 8 Isosurfaces of the Mach number of a delta wing simulation. Cracks are visible within the surface of this isocontour due to the use of a discontinuous Galerkin formulation which are accurately rendered by ElVis.

ElVis also supports the rendering of contour lines, as shown in figure 9. We again see that linear elements, even when heavily subdivided, do not yield the same quality of visualization, and indeed do not accurately depict the true regularity of the solution field.

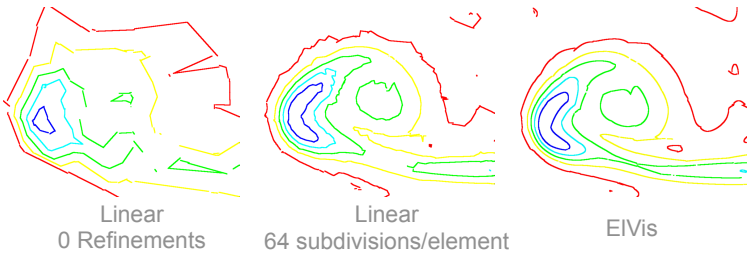

Fig. 9 Contour plots demonstrating the apparent discontinuity and lack of regularity that linearly interpolated visualization shows in the solution field.

3.3 Surface and cut-surface visualizations

Possibly the most widely used visualization tool is that of a surface rendering, where the solution field is restricted to a particular surface or cut-surface and a colour map is applied to show variation in the solution. Surfaces are simple to visualise and offer a clear indication of how the solution field is behaving at specific points in the

domain. ElVis supports the use of multiple cut planes, as well as the visualization of solution fields on curvilinear mesh surfaces (as depicted later).

The algorithm that renders each surface is again based on ray-tracing. The first step is to calculate the intersection of the ray with the cut-surface which might be defined as an implicit or a parametric surface. Apart from simple cases such as a plane, finding the location of the intersection point is a non-linear problem that must be solved via iteration. Once an intersection point is located, the element containing the point is found and the solution field evaluated using the element basis functions at the corresponding coordinates in the reference space. It is worth noting that finding the coordinates in the reference space for curved elements is a non-linear problem that requires an iterative method. A more detailed description of the algorithms involved in this process is given in reference [4].

Figure 10 shows the effect that high-order rendering using ElVis has on the visualizations for a delta wing mesh. We see that the linearly interpolated solutions in Figures 10(a) and 10(b) present an unclear rendering of the flow characteristics, and in particular the boundary layer is nearly impossible to detect, which may lead one to believe that the simulation is under-resolved. In Figure 10(c) however, ElVis gives a clearer picture of the flow behaviour, and the boundary layer is clearly identifiable.

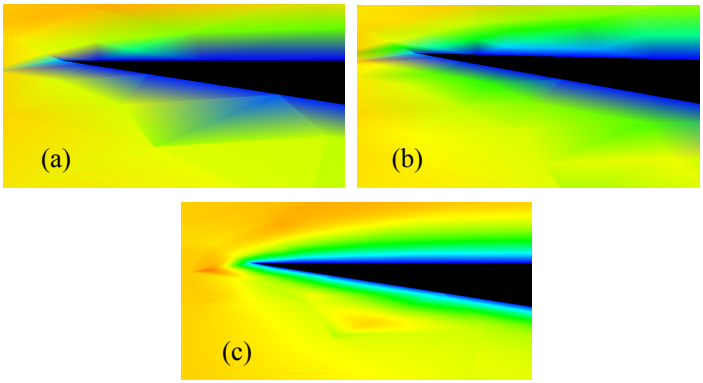

Fig. 10 Linear versus high-order visualization of planar cuts: (a) Linear visualization without subdivision; (b) linear visualization with subdivision; (c) high-order visualization.

3.4 Volume rendering

It is often the case that the visualization of noisy data, as arises in for example turbulence simulations, can be difficult. When this occurs, it can be useful to instead consider a volume rendering which can visualise the isocontours in a way that is easier to interpret. Volume rendering in ElVis is based on the emission-absorption

optical model [6] in which the irradiance is evaluated along rays through the integral

$$\int_a^b \kappa(t)\,\tau(t) e^{-I(t)}\,dt\,; \quad I(t) = \int_a^t \tau(u)du \tag{1}$$

where $[a,b]$ is denotes a parametric interval along the ray and $\kappa(t)$ and $\tau(t)$ are the colour and density transfer functions, respectively.

The evaluation of the integrals in equation (1) is performed element by element. This requires a traversal of the mesh and the calculation of *all* the intersections of the ray with the boundary of the elements to determine the ray segments contained within the element. This operation is complicated by the presence of curved element since a ray might intersect a curved element face several times. For a ray segment, the colour and density transfer function falls into one of three categories: empty, smooth and piecewise smooth. The evaluation of the volume rendering integral (1) is handled on a case-by-case basis and appropriate quadrature rules are use for each of these categories. One downside of integrating functions that are piece-wise smooth is that the best order of convergence that can be achieved is second order.

An example of a typical volume rendering produced by ElVis can be seen in figure 11. This figure also illustrates the importance of selecting an appropriate quadrature for the evaluation of both integrals in equation (1).

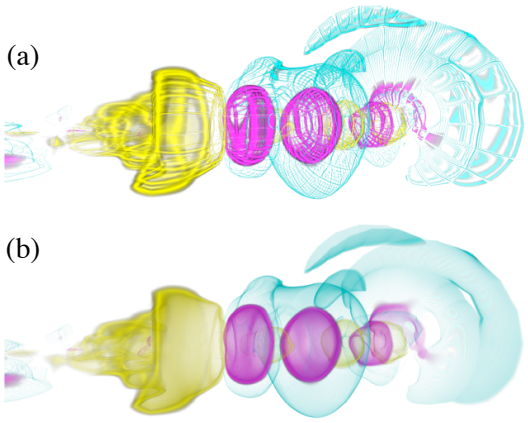

Fig. 11 Effect of quadrature on the accuracy of volume rendering: (a) Riemann integration and (b) high-order integration, both with 60 million samples.

A more detail description of the algorithms and their performance is given in reference [6].

3.5 Mesh visualization

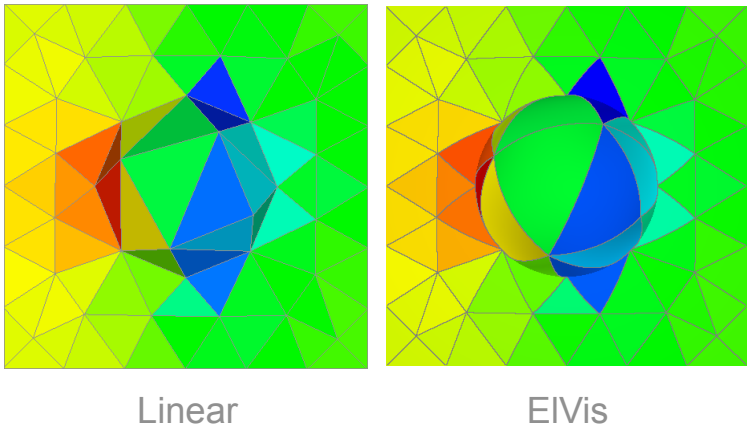

Fig. 12 A visualization of a curvilinear hemisphere using linearly interpolated elements (left) and ElVis (right).

The generation of curvilinear meshes for high-order simulations is a challenging and difficult issue. However, this problem is made more difficult by the fact that it is often impossible, extremely difficult, or computationally prohibitive to accurately visualise the surface of a curvilinear mesh. Whilst linearly interpolated visualizations may give an overall idea of the mesh surface, small oscillations may remain hidden, as the example of Section 2.2 (and figure 4) demonstrate.

One of the core capabilities of ElVis is to assist in the visualization of curvilinear meshes. Figure 12 shows a curvilinear hemisphere mesh in combination with a cut-plane, which also demonstrates the use of several simultaneous visualization techniques. We clearly see that the introduction of curvature yields two very different images. Additionally, we note that even when subdivision or refinement is used to more clearly visualise curvature using linear visualization software, this usually causes confusion in determining where the high-order elemental boundaries lie, since additional faces are drawn wherever refinement occurs. ElVis therefore gives an accurate representation of the features of the numerical discretisation.

We conclude with a final observation regarding mesh generation and how visualization tools such as ElVis provide a valuable debugging tool. Curvature in an element Ω^e is usually imposed through the use of a mapping T from a reference element Ω_{st}, which is necessarily required to be invertible and preserve orientation so that the determinant of the Jacobian of T is positive at all points within Ω_{st}. However, it is often the case that when curvature is imposed in the mesh generation procedure, self-intersection can occur, causing the resulting mesh to be invalid and be unsuitable for simulations. For linear finite elements, detecting invalid ele-

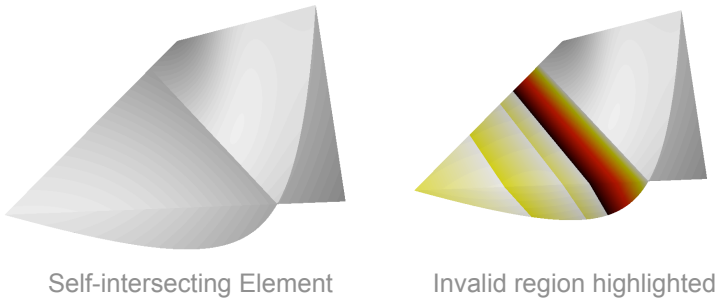

Fig. 13 visualization of a self-intersecting tetrahedron, where on the right ElVis is used to highlight areas where the element is invalid.

ments is an easy task – however at higher polynomial orders, the task turns into a multi-dimensional root finding problem which is extremely costly.

An alternative approach is to select a distribution of nodal points at which the Jacobian determinant is evaluated and is confirmed to be positive. However, this does not give a guarantee that the element is valid. High-order visualization tools such as ElVis are therefore a powerful tool if it is suspected that the mesh may contain negative Jacobians. Figure 13 shows how the Jacobian function can be visualised to highlight areas in elements which are invalid.

4 Conclusions

We have given an overview of the various capabilities and features of the Element Visualizer ElVis, an integrated visualization system designed specifically for high-order finite element solutions. ElVis has an extensible architecture that supports data originating from arbitrary simulation systems and its visualization algorithms are decoupled from the data representation. This provides accurate visualization and avoids introducing error into the final image by operating on the high-order data directly. ElVis also uses the parallel processing capabilities of recent Graphics Processing Units (GPU) to provide interactive visualizations on desktop computers.

We have shown that high-order visualization is not only desirable, but necessary in order to truly appreciate the solution data that high-order methods produce, and indeed to visualise the geometry on which such methods are used. Furthermore, these techniques are crucial not only to visualise solution data but can play a crucial role in diagnosing problems and also in the mesh generation process. It is clear that if high-order methods are to be more widely adopted amongst industry, the development of tools such as ElVis is essential.

References

1. G.E. Karniadakis and S.J. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2005. Second edition.
2. M. Meyer, B.W. Nelson, R.M. Kirby, and R. Whitaker. Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1015–1026, 2007.
3. B.W. Nelson. *Accurate and interactive visualization of high-order finite element fields*. PhD thesis, School of Computing, The University of Utah, Salt Lake City, August 2012.
4. B.W. Nelson, R. Haimes, and R.M. Kirby. GPU-based interactive cut-surface extraction from high-order finite element fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1803–1811, 2011.
5. B.W. Nelson and R.M. Kirby. Ray-tracing polymorphic multi-domain spectral/hp elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):114–125, 2006.
6. B.W. Nelson, R.M. Kirby, and R. Haimes. GPU-based volume visualization from high-order finite element fields. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):70–83, Jan 2014.
7. B.W. Nelson, E. Liu, R. M. Kirby, and R. Haimes. ElVis: A system for the accurate and interactive visualization of high-order finite element solutions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2325–2334, 2012.