2-1-2006

Ray-Tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering

Blake Nelson

Robert M. Kirby

Ray-Tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering

Blake Nelson and Robert M. Kirby, *Member*, *IEEE*

Abstract—The purpose of this paper is to present a ray-tracing isosurface rendering algorithm for spectral/hp (high-order finite) element methods in which the visualization error is both quantified and minimized. Determination of the ray-isosurface intersection is accomplished by classic polynomial root-finding applied to a polynomial approximation obtained by projecting the finite element solution over element-partitioned segments along the ray. Combining the smoothness properties of spectral/hp elements with classic orthogonal polynomial approximation theory, we devise an adaptive scheme which allows the polynomial approximation along a ray-segment to be arbitrarily close to the true solution. The resulting images converge toward a *pixel-exact* image at a rate far faster than sampling the spectral/hp element solution and applying classic low-order visualization techniques such as marching cubes.

Index Terms—Spectral/hp elements, high-order finite elements, ray-tracing, isosurface rendering, fluid flow visualization, error budget.

1 Introduction

The use of simulation science [1] as a means of scientific inquiry is increasing at a tremendous rate. The process of mathematically modeling physical phenomena, experimentally estimating important key modeling parameters, numerically approximating the solution of the mathematical model, and computationally solving the resulting algorithm has inundated the scientific and engineering worlds, allowing for rapid advances in our modern understanding and utilization of the world around us.

Quantification and, ideally, elimination of error is a critical component of the scientific process; it allows scientists to judiciously evaluate which component of the simulation science process (e.g., modeling, numerical approximation, parameter estimation) requires refinement in light of comparison with the real phenomenon of interest. Over the last 40 years, tremendous effort has been exerted in the pursuit of numerical methods which are both *flexible* and *accurate*, hence providing sufficient fidelity to be employed in the numerical solution of a large number of models and sufficient quantification of accuracy to allow researchers to focus their attention on model refinement and uncertainty quantification. High-order finite element methods (a variant of which is the spectral/hp element methods) using either the continuous Galerkin or discontinuous Galerkin formulation have reached a level of sophistication such that they are now commonly applied to a diverse set of real-life engineering problems in computational solid mechanics, fluid mechanics, acoustics, and electromagnetics (e.g., [2], [3], [4], [5]).

Unfortunately, there has been little emphasis on providing visualization algorithms and the corresponding software solutions tailored to high-order methods; in particular, almost no research has been done to develop visualization methods based on the a priori knowledge that the data was produced by a high-order finite element simulation (notable exceptions are Coppola et al. [6] with respect to particle tracking, Wiley et al. [7], [8], [9] with respect to ray-casting of curved-quadratic elements, and Brasher and Haimes [10] with respect to cut-planes). Current visualization methods are typically based on lower-order representations of the data (lower in the sense of what is customarily used in high-order numerical simulations)—for instance, using linear interpolation for determining the values of data in areas not at grid locations. Accurate low-order approximations of high-order data require far more degrees of freedom than the original data set and may introduce data artifacts due to mismatches in the approximation properties of the low-order representation and the original high-order discretization.

In this paper, we seek to develop a ray-tracing isosurface rendering algorithm which exploits the properties of spectral/hp element data [11], and for which the visualization error is both quantified and minimized. This is accomplished by reducing the determination of the ray-isosurface intersection of spectral/hp element data to classic polynomial root-finding applied to a polynomial approximation obtained by projecting the finite element solution over element-partitioned segments along the ray. Combining the smoothness properties of spectral/hp finite elements with classic orthogonal polynomial approximation theory, we devise an adaptive scheme which allows the polynomial approximation along a ray-segment to be arbitrary close to the true solution. The methodology presented herein has broader application than merely spectral/hp elements; this methodology can be adapted to any volumetric function which meets the smoothness properties of the spectral/hp element discretizations exploited in this work.

1.1 Objectives

The purpose of this paper is to present a spectral/hp element ray-tracing methodology for isosurface rendering which allows for quantification and minimization of visualization error. The target visualizations are those which are *pixel-exact* [12]—that is, images which contain

• *The authors are with the School of Computing and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, 84112. E-mail: {bnelson, kirby}@cs.utah.edu.*

Manuscript received 30 Nov. 2004; revised 8 June 2005; accepted 20 June 2005; published online 9 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0150-1104.

minimized visualization approximation errors and, hence, provide images useful for understanding and minimizing errors in the overall simulation science process.

After presenting a ray-tracing methodology with its theoretical underpinnings, we will present an empirical study of the error introduced by approximations made during the visualization process. The resulting images converge toward a *pixel-exact* image at a rate far faster than sampling the high-order finite element solution and applying classic low-order visualization techniques such as marching cubes [13].

Using four spectral/hp element data sets (two simple examples and two fluid flow simulations examples as representatives of "real-world" computations), we will demonstrate how tolerance parameters within the methodology allow the user to move toward a *pixel-exact* image in a tractable manner. In addition, to help the reader to appreciate the merit of exploiting the mathematical structure of spectral/hp elements, we provide comparisons of images generated with our methodology to images obtained by visualization techniques designed for low-order data.

1.2 Outline

The paper is organized as follows: In Section 2, we provide a brief summary of relevant previous work. In Section 3, we provide an overview of spectral/hp (high-order finite) elements with focus on those items of immediate relevance to this work. In Section 4, we present our ray-tracing isosurface extraction methodology with emphasis on the mathematical fundamentals. In Section 5, we present four numerical examples visualized using our methodology. In Section 6, we present an empirical study of the visualization "error budget" (places at which error can be introduced into the visualization process) associated with our methodology. We conclude in Section 7 with a summary of our findings, conclusions, and future work.

2 Previous Work

Common isosurface visualization techniques, such as Marching Cubes [13], WISE [14], [15], and the optimal isosurface extraction algorithm in [16] approximate the isosurface by generating a piecewise linear polygonal mesh. While the details vary by algorithm, the goal of each is to find a planar polygon through each cell which closely approximates the actual isosurface. The resulting mesh will provide a reasonably accurate representation of the surface contingent upon the surface being relatively smooth inside each element. If the volume contains areas of high variability, then it is likely that the mesh generated in those areas will be inaccurate.

One complication with the polygonal mesh approach is how the number of polygons necessary to represent the surface increases as the data set is subsampled and as its size increases. It is not uncommon for isosurfaces to contain hundreds of millions of polygons. While it is possible to display such volumes at interactive frame rates (as shown by Cigoni et al. [17] and Yoon et al. [18]), this is only possible once the mesh is generated. As will be discussed in Section 6, the amount of time needed to generate this mesh becomes a major bottleneck in the system, making ray-tracing an attractive alternative. Parker et al. [19] used this approach to generate interactive isosurfaces of large data sets.

While the work in [19] uses trilinear interpolation in each cell, other work has been done using the higher order information present in the element. For instance, ray-tracing has been shown to be effective in rendering spline surfaces [20]. In [21], isosurfaces are approximated with second-order splines. Williams et al. [22] and Wiley et al. [7], [8], [9] find accurate isosurfaces for second-order tetrahedra.

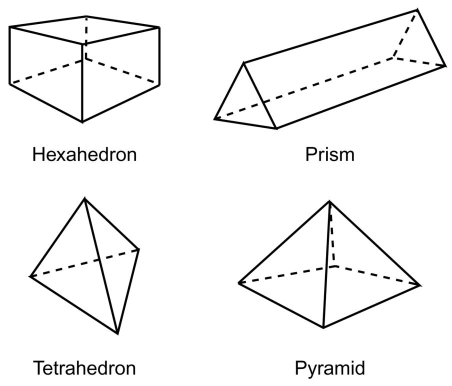


Fig. 1. Schematic of the four basic element types in three dimensions: hexahedron, prism, tetrahedron, and pyramid.

3 Spectral/hp Element Expansions

The purpose of this section is to provide a high-level overview of the key features of spectral/hp elements (the a priori knowledge mentioned in the introduction) used in this work. We refer the interested reader to the following reference works for a more in depth discussion of finite elements and, in particular, spectral/hp (high-order finite) elements [23], [24], [11], [25]. We will be drawing upon these texts for the discussion now presented. For the purposes of this paper, the terms spectral/hp elements and high-order finite elements will be used interchangeably. To solidify our discussion, we rely upon the variant of spectral/hp elements discussed in [11] for all spectral/hp element theoretical and implementation details.

One important property of finite element solution methods is the declaration of the space of admissible solutions. Quite often, the first step in the finite element methodology is to define, for the domain Ω over which the partial differential equation(s) of interest are being solved, a tessellation $\mathcal{T}(\Omega)$ of Ω. Four basic element types often used in the construction of the tessellation $\mathcal{T}(\Omega)$ are the hexahedron, tetrahedron, prism, and pyramid, as presented in Fig. 1.

The result of a finite element simulation is a function $u(\vec{x}) : \mathcal{T}(\Omega) \rightarrow \mathbb{R}$ about which certain properties are known. For instance, in the case of a continuous Galerkin spectral/hp element simulation, we know that the solution is, in general, a C^0 function with C^1 discontinuities only occurring on the element boundaries. In the interior of each element, the solution is C^p with $p \geq 1$ (denoting higher levels of smoothness depending on the order of the approximating polynomial on each element and the reference to world space mappings employed) [11] and references therein). The premise of this work is that since finite elements provide functions with explicit levels of smoothness, this structure should be both *respected* and *exploited* in the visualization.

Although spectral/hp element solution techniques generally deal with "global degrees of freedom" (degrees of freedom associated with functions defined over the entire

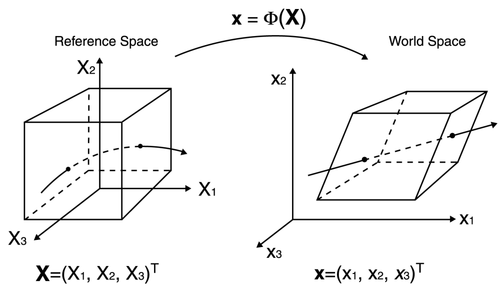


Fig. 2. Schematic of the element mapping. X denotes reference space coordinates, x denotes world space coordinates, and $x = \Phi(X)$ denotes the mapping between the two systems. Note that a line segment in world space may represent a curve in reference space (as explained in the text).

tessellation), most solution construction is done on an element-by-element basis (with global stiffness summation, for instance, being used to assemble the global system). Since our interest is in postprocessing high-order finite element results, we will focus only on the procedure for reconstructing the solution. Given a collection of global degrees of freedom, one can assess what are the local (to each individual element within the tessellation) degrees of freedom which contribute to the solution. Hence, to answer the query "What is the value of our finite element solution $u(\vec{x})$ for $\vec{x} \in \mathcal{T}(\Omega)$?", we first determine the element $\Omega_e \subseteq \mathcal{T}(\Omega)$ such that $\vec{x} \in \Omega_e$, and we then use element evaluation operations to reconstruct the solution on the element. Most finite element methods define local element operations with respect to a reference element, and use a mapping function to relate "world space" with "reference space." For instance, the reference element for a hexahedron in world space may be the cube $[-1,1]^3$ in reference space coordinates. A schematic illustrating this is given in Fig. 2.

Let us define a mapping function $x = \Phi(\mathbf{X})$ from positions in our reference element (denoted with $\mathbf{X} = (X_1, X_2, X_3)^T$) to positions in world space (denoted with $\mathbf{x} = (x_1, x_2, x_3)^T$). Spectral/$hp$ element methods seek to express solutions as a high-order polynomial function on the reference element $F(\mathbf{X}) \in \mathcal{P}^{\mathbf{N_1,N_2,N_3}}$, where N_1, N_2, N_3 denote (possibly) different polynomial orders in the three principle directions. The polynomial order per element may vary within the tessellation, hence allowing for adaptive resolution (hence, the hp in spectral/hp element methods—different element types and sizes denoted as "h" resolution and different polynomial orders per element denoted as "p" resolution). Given the solution degrees of freedom on an element, either in nodal form [25] or in modal form [11], we can find the value of the finite element solution at any point within the element $\vec{x} \in \Omega_e$ by evaluating $f(\vec{x}) = F(\Phi^{-1}(\vec{x}))$, where the function $F(\vec{X})$ is a polynomial over the reference domain determined by the local degrees of freedom of the solution, and the function $f(\vec{x})$ denotes the mapped smooth function over the world space domain. A few observations need to be mentioned at this stage, as they will become important later in the paper:

- "Curved" elements (those elements with nonplanar faces) can be accommodated with carefully constructed mapping functions. Quite often, isoparametric or superparametric representations of the curved faces are used so that the geometric approximation error is on the same order as the solution error.
- Note that the inverse of the mapping function is needed to evaluate the finite element solution in world space coordinates. In general, the inverse mapping function cannot be constructed analytically and, hence, numerical inversion must be used. The mapping from a reference tetrahedron to an all-planar-faced tetrahedron in world space is one of the few instances in which an analytic inverse is possible. In an effort for a general scheme, we will assume that analytic inversion is not possible. We assume, however, that the element construction and mappings used this work form legitimate (admissible) spectral/hp element discretizations as described in [11]. Tolerances for the numerical inversion of the mapping function will be discussed further in Section 4.1 as errors may be introduced.
- Due to the nature of the mapping function $\mathbf{x} = \Phi(\mathbf{X})$ used in spectral/hp element methods (for instance, in the case of a hexahedron, a trilinear mapping from reference space to world space is used [11]), a line segment in world space does not always correspond to a line segment in the reference space. On the contrary, it is often the case that the mappings cause line segments in world space to be curves in reference space (as illustrated in the schematic given in Fig. 2). Hence, one is often forced to decide between working with a polynomial function in reference space (and, hence, looking for zeros of a function along a not-necessarily-parameterizable curve) or a smooth function along a line segment in world space. In this work, we choose the latter approach.
- Although spectral/hp element methods as in [11] seek polynomial solutions *with respect to the reference element*, the solution with respect to world space coordinates may not be polynomial (and, in general, is not). Under the mappings used in spectral/hp finite elements, we are guaranteed that the solution is at least C^0 continuous and is smooth on the interior of each element. In this work, we assume that all discretizations investigated respect smoothness criteria on the interior of each element. This smoothness of the function within each element will be exploited in Section 4.2 when finding a projected polynomial approximation along the ray.

4 Ray-Tracing High-Order Elements

The ray-tracing process for finite element data can be summarized by the following three steps:

Step 1: Cast the ray $\vec{r} : [0, \infty) \to \mathbb{R}^3, \vec{r}(t) = \vec{o} + t\vec{v}$ (where $\vec{o}$ denotes the ray origin and $\vec{v}$ denotes the ray vector) into the volume and find the closest element (i.e., the ray-element intersection has the smallest value of the ray parameter t). Finding this element can be time consuming since simulations typically have many elements. Many strategies exist to accelerate this process, such as octrees [26], uniform spatial subdivision [27], multilevel grids [19], and hierarchical bounding volumes [28], to name a few. As mentioned in the previous section, high-order finite elements allow for the possibility of curved faces and, hence, special considerations must be taken to accurately compute the intersection point on the curved boundaries. This was discussed in [7], [8], [9] for quadratic-curved elements. For the purposes of this paper, we assume that machine precision accurate element intersection points can be obtained.

Step 2: Determine the ray-isosurface intersections inside the element. This is accomplished by classic polynomial root-finding applied to a polynomial approximation obtained by projecting the finite element solution over element-partitioned segments along the ray. This is discussed in Section 4.2. We are only interested in the closest isosurface intersection if multiple intersections exist. If there are no intersections, then repeat Step 1 to find the next closest element. The remainder of this section gives the details necessary to find these intersections accurately.

Step 3: Compute the normal vector at the intersection point. The normal is equal to the normalized gradient of the functions presented in Section 3. It is important to note that the computation of the gradient is accomplished upon the *original data*, not upon the projected polynomial. Given that gradient computations are part of the normal spectral/*hp* element solution procedure, we compute the gradient at the intersection point *as would be done as part of the original simulation*. Hence, any approximation error in the computation of the normal is of the same order as the gradient computations used throughout the simulation. Based upon the normal, we then calculate the color for the pixel based on the material properties of the surface and the lights in the scene.

In the discussion of pixel-exactness, the correctness of the normal may have a notable perceivable influence due to its use in establishing color/shading. Given that we use the "exact" normal computed at an intersection point, the visual error introduced due to the normal computation is tacitly dependent upon the accuracy of the intersection point computation and upon the curvature of the underlying function around the intersection point (as it will determine the rate of change of the normal)—as with all isosurface approximation methods. By minimizing the error in the intersection point computation and using the "exact" normal, we expect that variations in the image due to the normal computations will be minimized. This is certainly an area of future study for this methodology—to explicitly attempt to understand the sensitivity of this methodology to small inaccuracies. In addition to further theoretical considerations, common ray-tracing antialiasing techniques are available and could be exploited [29].

In this work, we focus our attention on quantifying and minimizing the visualization error associated with Step 2. The three items within the visualization error budget upon which we will focus are the function evaluation error (mainly consisting of mapping inversion error), the function projection error, and the root finding error. We note that perceptual error could certainly be added to our visualization error budget, but is a topic beyond the scope of this work.

4.1 Function Evaluation

Recall from Section 3 that function evaluation at a point $\vec{x}$ in world space consists of three parts: 1) finding the element Ω_e for which $\vec{x} \in \Omega_e$, 2) finding the reference space coordinate $\vec{X}$ which corresponds to $\vec{x}$ through an inversion of the mapping function $\mathbf{x} = \Phi(\mathbf{X})$, and 3) evaluation of the function $F(\vec{X})$ representing the local solution.

Of the three steps outlined above, the one most susceptible to numerical errors is 2)—the numerical inversion of the mapping. The most frequently used approach is to define the Jacobian of the transformation

$$J = \frac{\partial \mathbf{x}(x_1, x_2, x_3)}{\partial \mathbf{X}(X_1, X_2, X_3)},$$

which often can be done analytically and to use the Newton-Raphson method (which uses the inverse of the Jacobian) with convergence tolerance ϵ_M (error tolerance with respect to the mapping). Hence, the first error in our visualization error budget is the numerical mapping inversion error ϵ_M.

4.2 Function Projection

As mentioned in Section 3, although the solution on the reference element is a polynomial in the reference coordinates, it is only guaranteed to be a smooth function with respect to world coordinates. This is a key feature of spectral/*hp* element discretizations which we will seek to exploit—the smoothness of the function interior to an element.

Our goal of rendering an isovalue along a ray intersecting an element equates to finding the zeros of a smooth function. Since polynomial root finding is far easier, in general, than finding the zeros of a general function, we seek a means of representing the function along the ray within an element in world space as a polynomial with quantifiable and adaptability reducible error. This can be accomplished by finding the L_2 projection of the function onto a Legendre series.

In what follows, we will review a few major concepts and results from classical orthogonal polynomial theory [30], [31], [32]. Let us focus our attention on polynomial expansions of the form

$$u(x) = \sum_{n=0}^{\infty} \hat{u}_n P_n(x), x \in [-1, 1]. \tag{1}$$

Here, $P_n(x)$ represent the nth order Legendre polynomial, which is the polynomial solution to the Sturm-Liouville problem

$$\mathcal{L}P_n(x) = \frac{d}{dx}\left((1 - x^2)\frac{d}{dx}P_n(x)\right) = -\lambda_n P_n(x), \tag{2}$$

where

$$\lambda_n = n(n + 1).$$

The Legendre polynomials are normalized such that

$$P_n(\pm 1) = (\pm 1)^n.$$

Let us introduce the inner-product and associated norm

$$(u, v) = \int_{-1}^{1} u(x)v(x)dx, \ \|u\| = \sqrt{(u, u)}.$$

We use this to define the usual spaces

$$L^2[-1, 1] = \{u \,|\, \|u\| < \infty\},$$

and the associated higher Sobolev norms

$$H^p[-1,1] = \left\{ u \in L^2 \| \|u\|^2_{H^p[-1,1]} = \sum_{i=0}^{p} \|u^{(i)}\|^2 < \infty \right\}.$$

Note that higher Sobolev norms represent higher levels of smoothness of the solution.

As the Legendre polynomials satisfy (2) [31], we know that

$$(P_n, P_m) = \delta_{nm}\gamma_n, \gamma_n = \frac{2}{2n+1}, \tag{3}$$

where δ_{nm} denotes the Kronecker delta. Thus, for all $u(x) \in L^2$, we recover the expansion coefficients, $\hat{u}_n$, in (1) of the form

$$\hat{u}_n = \frac{1}{\gamma_n}(u, P_n). \tag{4}$$

Hence, just like Fourier series, we can exploit the orthogonality of the Legendre polynomials to allow us to successively compute more terms within our Legendre expansion without having to recompute all lower-order terms. This is one of the major advantages of using the Legendre polynomial basis over other polynomial bases.

Rather than evaluating the above integral analytically, one can use Gauss quadrature rules such as the Gauss-Lobatto quadrature

$$\tilde{u}_n = \frac{1}{\tilde{\gamma}_n} \sum_{i=0}^{N} u(x_i) P_n(x_i) w_i, \tag{5}$$

where (x_i, w_i) represent the Legendre-Gauss-Lobatto quadrature nodes and weights, respectively (see [31]). The quadrature is exact if $u(x)$ is a polynomial of degree $2N-1$ at most. Note that Legendre-Gauss or Legendre-Gauss-Radau quadrature would also be equally valid; the only change would be the number of quadrature points needed to guarantee the exactness of the computation for a given polynomial order.

For the general case, $\hat{u}_n \neq \tilde{u}_n$; this difference in the coefficients is recognized as the "aliasing error." This aliasing error is due to the inexactness of the quadrature rule when integrating a polynomial times an arbitrary smooth function. In requiring that the quadrature rule be exact for the inner product of the polynomials of interest, we guarantee (under the assumption of smooth functions) that the aliasing error is on the same order as, if not less than, the approximation error. Thus, the error can be viewed as folding into the error estimates presented shortly (as discussed in [31]). In the present context, we shall not make an effort to distinguish between the two sets of expansion coefficients.

For later use, let us define the discrete inner product and associated $L_2[-1,1]$-equivalent discrete norm as

$$[u,v]_Q = \sum_{i=0}^{Q} u(x_i) v(x_i) w_i, \ \|u\|^2_Q = [u,u]_Q, \tag{6}$$

where (x_i, w_i) represent the Legendre-Gauss-Lobatto quadrature nodes and weights. As discussed in [31], the number of nodes/weights Q can be chosen such that integration of polynomials is done exactly to machine precision.

In computational methods, one is concerned with the truncated expansion

$$u_N(x) = \sum_{n=0}^{N} \hat{u}_n P_n(x), \ x \in [-1,1] \tag{7}$$

and how it behaves as N increases. In other words, one wishes to understand how $u - u_N$ decays when increasing N.

Insight into this can be gained by recalling Bessel's inequality,

$$\|u\|^2 = \sum_{n=0}^{\infty} \gamma_n (\hat{u}_n)^2,$$

implying that

$$\|u - u_N\|^2 = \sum_{n=N+1}^{\infty} \gamma_n (\hat{u}_n)^2,$$

i.e., the accuracy depends solely on the decay of the expansion coefficients, $\hat{u}_n$, and the behavior of γ_n, given in (3). Repeated integration by parts of (4) yields

$$\hat{u}_n = (-1)^q \frac{1}{\lambda_n^q} \frac{1}{\gamma_n} (\mathcal{L}^q u, P_n).$$

Recall (from [31], for instance) that $\mathcal{L}$ essentially is a second order operator, i.e., if $u(x) \in H^{2q}[-1,1]$, we can expect that $\mathcal{L}^q u \in L^2[-1,1]$, which again implies that

$$\frac{1}{\gamma_n}(\mathcal{L}^q u, P_n) \simeq \mathcal{O}(1).$$

Combining these results yields

$$\|u - u_N\| \leq C N^{-2q} \|u\|_{H^{2q}[-1,1]}.$$

Likewise, we get the pointwise estimate [30]

$$\|u - u_N\|_{L^\infty[-1,1]} \leq C N^{-2q+1} \|u\|_{H^{2q}[-1,1]}, \tag{8}$$

provided $2q > 1$. Further discussion of these estimates can be found in [31], [32]. The important point to note with respect to this work is that, given a sufficiently smooth function $u(x)$ (at least C^1—i.e., $q = 1$) which we are interested in projecting, the Legendre expansion is guaranteed to converge pointwise. No further continuity beyond C^1 is required to guarantee pointwise convergence. The *rate of convergence*, however, is determined by the levels of smoothness of the function. The smoother the function, the faster the Legendre series converges (see [31], [32], and references therein).

In this work, the solution we are attempting to fit is $f(\vec{x})$ along the ray (represented by the $u(x)$ in the discussion above), approximated by a polynomial $p(t)$ (denoted by $u_N(x)$ in the discussion above). Clearly, the smoothness of the solution $f(\vec{x})$ as indicated by the decay of the expansion coefficients is the main source of accuracy in this methodology. Since we know that the solution on an element is smooth with respect to the world space coordinates [11], we would expect that projection of the function onto Legendre polynomials would yield highly accurate results. Note that the smoothness requirement for pointwise convergence is

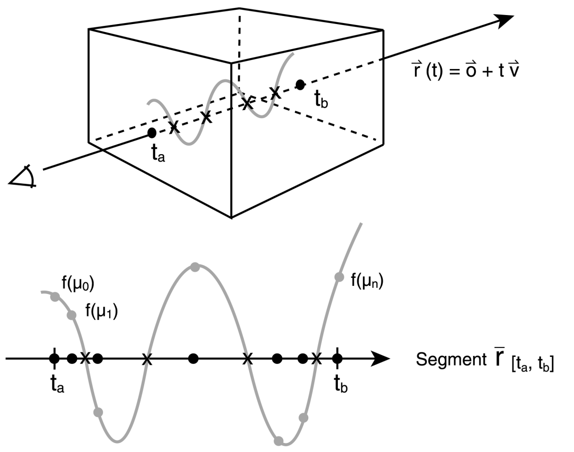


Fig. 3. Schematic showing how a projected polynomial is formed along the ray. Once the entry and exit points within an element are known, the scalar field within the element $f(\vec{x})$ can be sampled along the ray using a Legendre-Gauss-Lobatto point distribution. A "best-fit" polynomial (in L_2) can be formed, upon which root finding can be accomplished.

the reason we only accomplish the projection within elements, and not across elements (recall that, at element interfaces, we merely have C^0 continuity). The error estimates above give us an indication of how rapidly the approximating polynomial will converge to the true function in both the L_2 (integral) and L_∞ (pointwise) norms.

Given the discussion above, our procedure, as denoted in Fig. 3, is the following:

- Specify the order M of the approximating polynomial to be used along the ray ($p(t) \in \mathcal{P}^M$).
- Given entry and exit points $[t_a, t_b]$ along the ray (with respect to the element of interest), determine the necessary Legendre-Gauss-Lobatto points with respect to world space (which is accomplished through an affine transformation between $[-1, 1]$ and $[t_a, t_b]$). The number of quadrature points is a function of the approximating polynomial order; the number is chosen so that inner products of polynomials of degree M are exact to machine zero. As previously commented, by choosing the quadrature points in this way, we are guaranteed that all numerical integration errors are on the same order as the projected polynomial and that the quadrature order converges at a rate at least as fast as the projection error (see [31]).
- Evaluate the high-order finite element function $f(\vec{x})$ at the Legendre-Gauss-Lobatto points along the ray.
- Use Gauss-Lobatto quadrature (as given in (6)) to compute the projected polynomial $p(t)$ of order M.
- Given the orthogonal nature of the Legendre basis functions, adaptive error control can be accomplished. If the polynomial of order M does not sufficiently represent the function $f(\vec{x})$ along the ray, additional terms in the expansion can be computed without recomputing the initial terms (exploiting the orthogonality property just as you are able to do so with Fourier series expansions).

We can compute the (discrete) L_2 error ϵ_P with respect to the projected polynomial to determine if the order of the projection is sufficient by using the quadrature rules described within the projection process (here, we tacitly rely upon the convergence properties of the quadrature as discussed in [31] to justify ignoring the quadrature error introduced in this process). This error can be viewed as measuring the difference between a full expansion approximation given by (1) and the truncated expansion of (7). If the error is higher than a user-specified tolerance, more polynomial terms can be added. Note that, although we are using the integral error measure, we are tacitly exploiting the relationship with the pointwise error (as discussed in [31]). A pointwise error estimator could be explicitly adopted (evaluating along the segment at a finite number of points) if required. Hence, the second error in our visualization error budget is the numerical projection error ϵ_P.

4.3 Root Finding

Given the projection to a polynomial described above, the ray root finding problem can now be formulated mathematically as the following problem: Find the smallest value $t \in [t_a, t_b]$ such that $p(t) - C = 0$, where C is the isovalue of interest.

There are several means of accomplishing root finding of polynomials, most of which are iterative techniques. The most common technique is a Newton-Raphson iteration combined with polynomial deflation [1]. A modified version of this methodology was successfully applied in [33] for ray-tracing algebraic surfaces. We present an alternative numerical means of finding the roots of a polynomial. The availability of several different ways of solving this problem allows the scientist to chose the one that is most efficient and least susceptible to numerical errors (or instabilities).

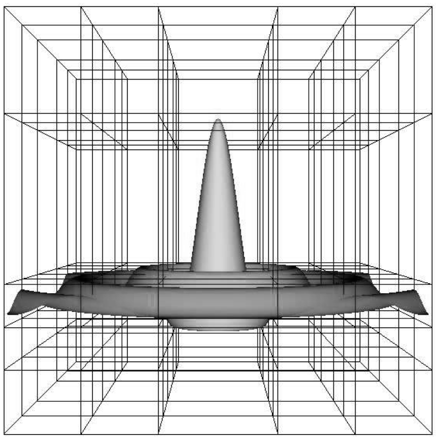
Since we want to guarantee that we have found the smallest root in the interval $[t_a, t_b]$, we desire to find all possible roots from which we can then easily ascertain the smallest one which lies within our interval of interest. Since we desire an algorithm which extends to arbitrary order, we utilize the mathematical idea of a *companion matrix* of a given polynomial. The linear algebra concept of a companion matrix provides us with the following result: Given a polynomial of degree M, there exists a matrix $\mathbf{A}$ determined by the coefficients a_j (called the companion matrix of the polynomial $p(t)$) such that the eigenvalues of $\mathbf{A}$ provide the roots of the polynomial $p(t)$. Hence, instead of a root-finding problem *per se*, we must solve an eigenvalue problem. This concept naturally follows from the linear algebra concept of the characteristic polynomial of a matrix [34]; there is a rich history between root finding and eigenvalue solutions which we seek to exploit.

To help provide guidance as to how the companion matrix of a polynomial is formed, we will demonstrate with respect to a sixth-order polynomial example. For example, consider the following polynomial:

$$p(z) = a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0,$$

where $a_j \in \mathbb{R}, j = 0 \ldots 6$, and where we assume that $a_6 \neq 0$.

The companion matrix for this polynomial is as follows [35]:


(a)

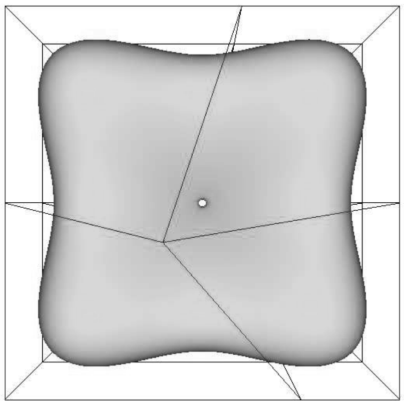

(b)

Fig. 4. (a) Isosurface corresponding to an isovalue of $C = 0.0$ for Example 1. (Image generation data: resolution = 512×512, mapping tolerance = 10^{-12}, projected polynomial order = 25). (b) Isosurface corresponding to an isovalue of $C = -0.00716511$ for Example 2. (Image generation data: resolution = 512×512, mapping tolerance = 10^{-12}, projected polynomial order = 25). The respective spectral/hp element grids are shown in each image.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -a_0/a_6 \\ 1 & 0 & 0 & 0 & 0 & -a_1/a_6 \\ 0 & 1 & 0 & 0 & 0 & -a_2/a_6 \\ 0 & 0 & 1 & 0 & 0 & -a_3/a_6 \\ 0 & 0 & 0 & 1 & 0 & -a_4/a_6 \\ 0 & 0 & 0 & 0 & 1 & -a_5/a_6 \end{pmatrix}.$$

This matrix is of upper Hessenberg form and, hence, a prime candidate for typical eigenvalue-finding techniques. Easily generated companion matrices exist for the monomial basis [35] and the Bernstein basis [36], [37]. The companion matrix associated with the Bernstein basis has been shown to be less prone to numerical precision errors. Given our polynomial $p(t)$ from Section 4.2 as a Legendre sequence, we can do a basis rotation to either the monomial or the Bernstein basis. Once this has been accomplished, a companion matrix can be formed and the corresponding eigenvalue problem solved. For all root finding results presented in this paper, a balanced QR iterative eigenvalue finding algorithm was used [1].

Given the iterative root finding procedure above (or any other iterative root finding procedure), we can access how accurately we have determined the roots by evaluating the polynomial at the computed roots. Hence, the third error in our visualization error budget is the numerical root finding error ϵ_R.

5 Overview of Numerical Examples

To demonstrate the utility of the ray-tracing procedure outlined in Section 4 and to provide data for the empirical examination performed in Section 6, we will examine four numerical examples. Two examples were generated through Galerkin (finite element) projections of smooth functions onto spectral/hp discretizations and two examples consist of computational solutions of the three-dimensional incompressible Navier-Stokes equations. All spectral/hp element results were obtained using the simulation code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ [11], [38]. Error budget information will be presented in Section 6.

5.1 Example 1: Projected Sinc Function

For our first example, we project (in the Galerkin finite element sense) the following function onto a spectral/hp element discretization:

$$u(x, y, z) = 2.0(y - y_c) - sinc(\sqrt{(x - x_c)^2 + (z - z_c)^2}),$$

where $(x_c, y_c, z_c)^T = (0.5, 0.3, 0.5)^T$ and where the domain is $[0, 1]^3$. A mesh consisting of 125 nonevenly spaced hexahedra were used, each element supporting eighth order polynomials per direction. In Fig. 4a, we present an isosurface rendering using our ray-tracing methodology for an isovalue of $C = 0.0$; the spectral/hp element grid is also shown.

5.2 Example 2: Projected Quartic Function

For our second example, we project (in the Galerkin finite element sense) the following function onto a spectral/hp element discretization:

$$\begin{aligned} u(x, y, z) =& (0.5 - x)^2 \Big((0.5 - x)^2 - K\Big) + \\ & (0.5 - y)^2 \Big((0.5 - y)^2 - K\Big) + \\ & (0.5 - z)^2 \Big((0.5 - z)^2 - K\Big), \end{aligned}$$

where $K = 0.169$ and where the domain is $[0, 1]^3$. A mesh consisting of eight nonevenly spaced, nonrectangular hexahedra were used, each element supporting eighth order polynomials per direction. In Fig. 4b, we present an isosurface rendering using our ray-tracing methodology for an isovalue of $C = -0.00716511$; the spectral/hp element grid is also shown.

5.3 Example 3: Flow Past a Block/Plate Configuration

For our third example, we present flow past a block with an array of splitter plates placed downstream of the block. A schematic of the flow scenario under consideration is presented in Fig. 5a.

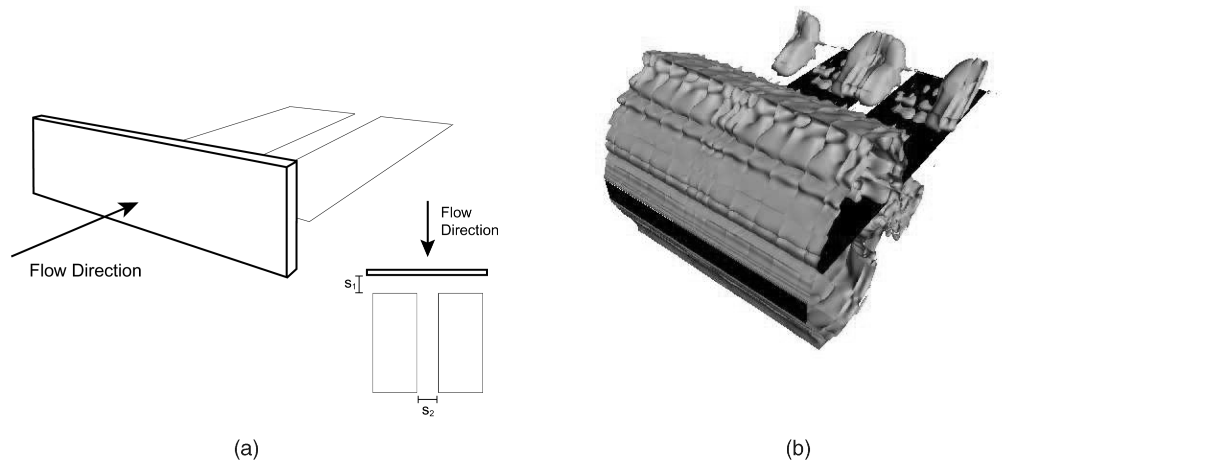


Fig. 5. Schematic showing the (a) basic block/splitter plate configuration under consideration and (b) third-order ray-traced solution for isosurface of pressure at $C = -0.1$. (Image generation data: resolution = 512×512, mapping tolerance = 10^{-11}, projected polynomial order = 12).

As the fluid impinges upon the block, it is diverted around the structure, generating vorticity along the surface. For the purposes of this paper, we will focus our attention on a configuration consisting of a nondimensional plate spacing of one unit (nondimensionalized with respect to the block height). The three-dimensional computational mesh consists of $3,360$ hexahedra and $7,644$ prisms with the computational fluid mechanics problem being solved with third-order polynomials per element. All computations were accomplished at a Reynolds number of $Re = 200$. In Fig. 5b, we present a pressure isosurface rendering using our ray-tracing methodology for an isovalue of $C = -0.1$.

5.4 Example 4: Flow Past a Rotating Canister

For our fourth example, we examine the wake of a rotating canister traveling through a fluid. A schematic of the flow scenario under consideration is presented in Fig. 6a.

The three-dimensional computational mesh consists of 5040 hexahedra and 696 prisms, with the computational fluid mechanics problem being solved with third-order polynomials per element. All flow computations presented herein were done at a Reynolds number of $Re = 1,000$ and with an angular velocity of $\Omega = 0.2$. In Fig. 6b, we present a pressure isosurface rendering using our ray-tracing methodology for an isovalue of $C = 0.0$.

6 Empirical Study of the Visualization "Error Budget"

In Section 4, we isolated three major contributors to the visualization error budget of spectral/hp element solutions: the mapping inversion error (ϵ_M), function projection error (ϵ_P), and root finding error (ϵ_R). To elucidate the influences of these errors, we performed empirical tests on the four data sets presented above; instructive results are presented.

In this section, we will discuss two of the errors previously mentioned: the mapping inversion error and the projection error. The root finding (with respect to polynomials) procedure is a well-understood process; most practitioners set the root finding tolerance to single or double precision machine zero (depending on the application) as a matter of habit. In this work, all polynomial root finding will be accomplished to a precision of at least 10^{-14}.

After presenting results for the mapping inversion error and the function projection error, we will provide image comparison results between ray-traced images with different tolerances and between ray-traced and marching cube isosurfaces.

6.1 Mapping Inversion Error

Recall from Section 4.1 that, to evaluate a point within a spectral/hp element, we must accomplish a numerical

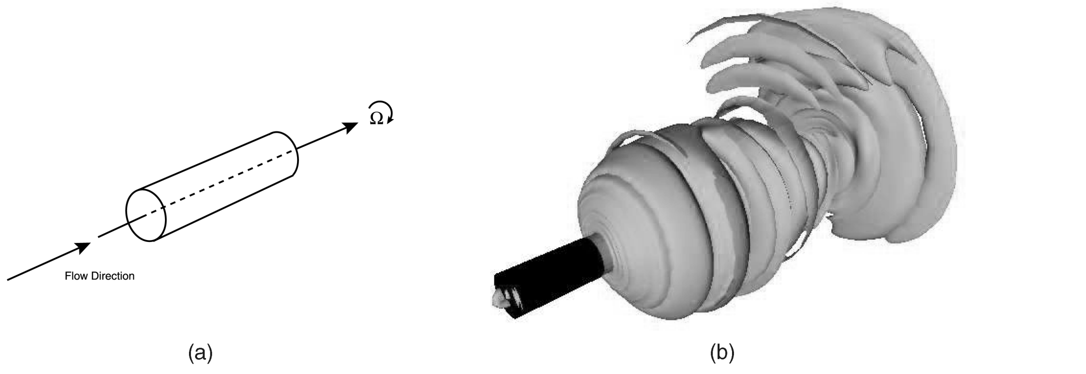


Fig. 6. Schematic showing the (a) basic rotating canister configuration under consideration and (b) third-order ray-traced solution for isosurface of pressure at $C = 0.0$. (Image generation data: resolution = 512×512, mapping tolerance = 10^{-8}, projected polynomial order = 13).

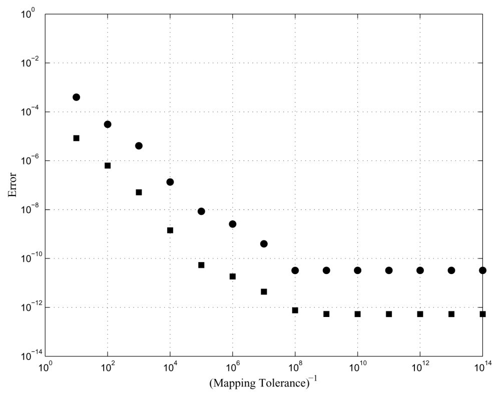


Fig. 7. Mapping inversion error as a function of the mapping inversion tolerance for the canister example. Circles denote the maximum absolute error and squares denote the root-mean-square error. Note that the abscissa is presented as the reciprocal of the mapping tolerance for easy left-to-right reading of the graph.

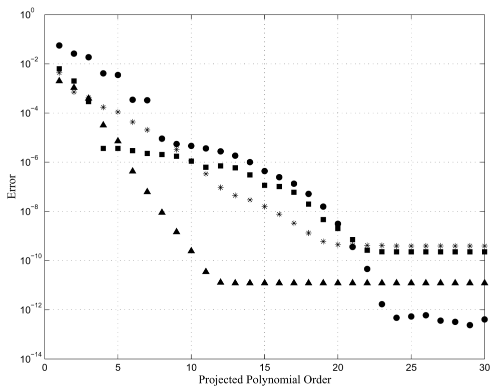


Fig. 8. Root-mean-square error between approximated isosurface value and true isovalue as a function of polynomial order for Example 1 (circles), Example 2 (squares), Example 3 (asterisks), and Example 4 (triangles).

inversion of the mapping between a reference element and a world space element. To see the influence of the mapping inversion tolerance, we performed the following test. For a 512×512 pixel image (i.e., $262,144$ rays), we ray-traced the data sets corresponding to Examples 1-4 (corresponding to Figs. 4, 5, and 6, respectively) at the isovalues and viewing parameters presented in the respective captions. For the positions on the rendered isosurface, we evaluated the original high-order finite element function to determine how close to the true isovalue the rendered isosurface was. In Fig. 7, we present the maximum absolute difference and the root-mean-square (rms) difference for the canister example (Example 4) as an indicative result.

Observe that, after a mapping tolerance of 10^{-8}, the maximum error and root-mean-square errors are both below 10^{-10} and level off. As expected, the trends of the maximum error and the root-mean-square error follow each other; henceforth, we will only show root-mean-square error. Since, in this example, we needed to specify a projected polynomial order, the leveling off is consistent with the fact that the mapping error is no longer the dominant error past a certain tolerance. In the next section, we will examine the influence of the function projection error. Based upon these results, all images produced in this paper used a mapping inversion tolerance of 10^{-8} or less.

6.2 Function Projection Error

Recall from Section 4.2 that the general zero-finding problem was simplified to a polynomial root-finding problem by finding a "best fit" projected polynomial. The projection process introduces an error ϵ_P into the error budget. To see the influence of the projected polynomial order, we performed a test similar to the one above, but varied the projected polynomial order. For a 512×512 pixel image, we ray-traced the data sets corresponding to Examples 1-4 (corresponding to Figs. 4, 5, and 6, respectively) at the isovalues and viewing parameters presented in the respective captions. For the positions on the rendered isosurface, we evaluated the original high-order finite element function to determine how close to the true isovalue the rendered isosurface was. In Fig. 8, we present the root-mean-square (rms) difference for the four examples.

Notice that, although the rate of convergence is similar in trend, there are differences between the different examples. This is consistent with the fact that the projection method theory does not provide a specific polynomial order to use for all cases, but provides us with guidelines for quantifying and minimizing the error. Successive (adaptive) increase in the polynomial order used allows for an root-mean-square error below single precision in all cases.

In the ideal case, all four lines would converge to the same error; the discrepancies observed are consistent with the mapping tolerance used and with the (known) rounding error susceptibilities of the root finding employed (see Section 4.3 and references mentioned therein). However, the important point to observe is that all errors converge below single-precision machine-zero in a way consistent with the theory.

One natural question which arises is: Is it worth it? By this we mean, does the increase in the accuracy of the projection (or of the mapping inversion) truly create a different image? This is addressed in the next section.

6.3 Image Comparison

To understand to what level the error tolerances mentioned above matter to the actual image, we present two discussions. In the first discussion, we compare the images produced using two different projected polynomial orders. In the second discussion, we present isosurfaces obtained via marching cubes versus our isosurface rendered technique.

6.3.1 Tolerance Sensitivity

In Fig. 9, we present two different images generated by our ray-tracing technique. On the left, we present the image obtained by using second-order projected polynomials and, on the right, we present the image generated using 13th order projected polynomials.

With second-order projected polynomials, the difference between the images is rather large. Increasing the projected polynomial order to eighth order projected polynomials provides images that were *pixel-identical* up to machine precision. This was verified by examining the three individual color values assigned to each pixel and generating the absolute maximum difference and root-mean-square error. Using high-order projected polynomials past eighth order provides no perceivable difference in the images.

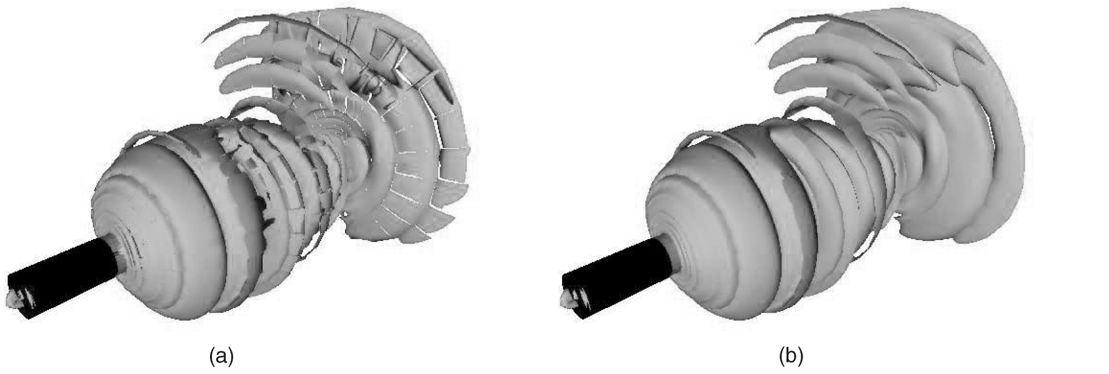


Fig. 9. Ray-traced solution of the canister using (a) second-order projected polynomials and (b) 13th order projected polynomials for isosurface of pressure at $C = 0.0$. The left image shows the ramification of reducing the projected polynomial order. The right image shows the "best" case as presented in the paper. Projected polynomials past eighth look no different than the right image. (Image generation data: resolution = 512×512, mapping tolerance = 10^{-8}, projected polynomial order = 2 (a) and 13 (b)).

6.3.2 Marching Cubes Comparison

Since many people use marching cube type solutions for isosurface rendered, we now provide in Fig. 10 an example showing the differences between a marching cubes isosurface and a high-order ray-traced isosurface for the canister data.

The marching cubes image was generated by sampling the finite element volume on a rectilinear grid of spacing h, using a marching cubes algorithm to provide a tessellated isosurface, and rendering the triangular isosurface using ray-tracing (since the marching cubes result is a triangular mesh, the ray-tracing can be done exactly as done in [19]). For the marching cubes image presented, a grid spacing of $h = 0.015$ (yielding $4,705,274$ voxels) was used. For the high-order ray-traced image, mapping inversion error of 10^{-8} and 11th order projected polynomials were used. These parameters were chosen such that the spectral/hp element evaluation time and rendering time were nearly identical to generate the two images. The root-mean-square error for the marching cubes image is 0.0158; the root-mean-square error for the ray-traced image is 3.5×10^{-11}. The images look very similar, however, the root-mean-square error difference between the images is significant. We should also point out that the file size for the marching cubes representation is over an order of magnitude larger than the high-order representation.

As we did previously, we can examine the convergence rate of the marching cubes result by evaluating the finite element solution at the nodes of the triangular mesh defining the isosurface so as to see how close the isovalue we are. In Fig. 11, we present the root-mean-square error for the marching cubes solution at different mesh spacings for Examples 1 and 2 (the two examples where the spectral/hp elements are eighth order).

The marching cubes result converges in a manner bounded by second-order convergence as would be expected of linear interpolation of the high-order results. We observe that the marching cubes image will converge given sufficient resolution and, hence, acknowledge that the strategy of highly sampling of the high-order finite element solution and using marching cubes is a valid strategy. However, as the polynomial order of the spectral/hp element solutions increase, the visualization error to speed trade-off will only become more pronounced. In Fig. 12, we plot the root-mean-square error (error computed as the difference between rendered and true isovalue over all rays) versus the time to evaluate and render a marching cubes and ray-traced solution for Example 1 (top) and Example 2 (bottom) for the isovalue and viewing para-

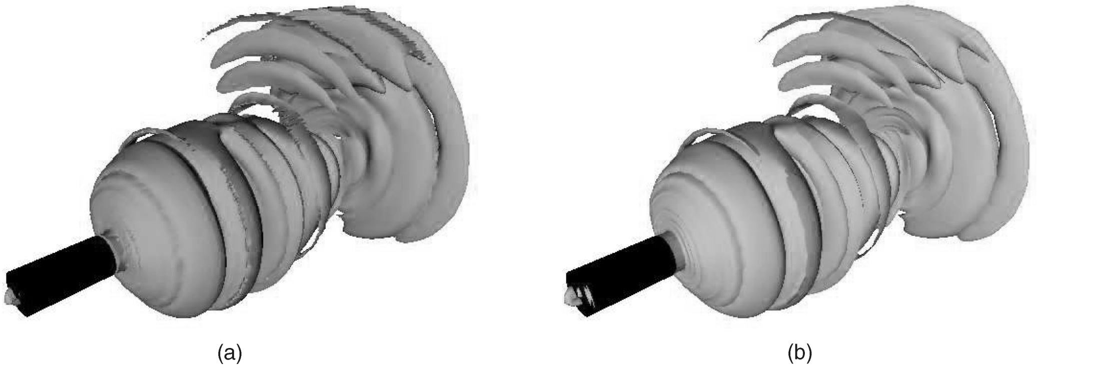


Fig. 10. (a) Marching cubes image with $h = 0.015$ corresponding to $4,705,274$ voxels and (b) ray-traced solution using 11th order projected polynomials for isosurface of pressure at $C = 0.0$ chosen such that the spectral/hp element data evaluation and rendering time is nearly identical (on the order of 200 seconds). The root-mean-square error for the marching cubes image is 0.0158; the root-mean-square error for the ray-traced image is 3.5×10^{-11}.

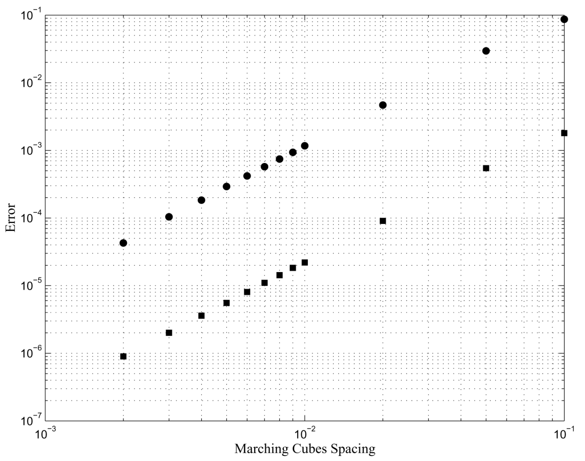


Fig. 11. Root-mean-square error in the marching cubes solution as a function of the (voxel) grid spacing for Example 1 (circles) and Example 2 (squares). Details provided in the text.

meters in Fig. 4 (end-to-end time). Time in this plot is proportional to resolution, in the case of marching cubes being proportional to number of sampling voxels and in the case of ray tracing being proportional to the projected polynomial order. In the context of the current investigation, the performance win normally associated with marching cubes (some of which is gained by data preprocessing) must be balanced with the number of sampling voxels required and the number of triangles needed to generate a *pixel-exact* isosurface. We emphasize that the dominant cost in this MC comparison is the voxelization of the high-order data. If only triangularization and rendering are considered (for example, when dealing with nontime-dependent data for which preprocessing can be done to accomplish the voxelization), the triangle solution is more anemable to current rendering acceleration stategies (e.g., use of GPUs). However, for most spectral/hp element data sets used in real-world engineering and science applications, it is impractical and, in some cases, infeasible to sample and store a voxelized volume sufficient to obtain the same level of accuracy as the proposed method.

7 Summary and Conclusion

If scientific visualization is to be used as part of simulation science, error budget analysis of the techniques employed must fall under the same critical eye as other components of the simulation science process (e.g., modeling error, numerical simulation error). Visualizations of high-order finite element results which do not respect the a priori knowledge of how the data were produced and which do not provide a quantification of the visual error produced undermine the scientific process. The purpose of this paper was to present a ray-tracing isosurface rendering algorithm for spectral/hp (high-order finite) element methods in which the visualization error budget is both quantified and minimized. A methodology was prescribed in Section 4, *pixel-exact* images of high-order finite element data were provided in Section 5, and a discussion of the corresponding error budget analysis was given in Section 6.

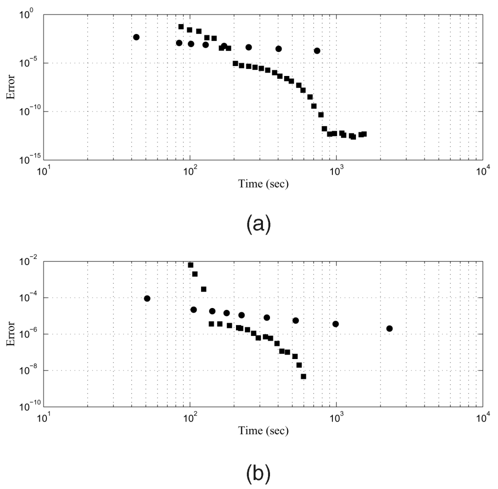


Fig. 12. Root-mean-square error in marching cubes solution (circles) and ray-traced solution (squares) versus spectral/hp element evaluation and rendering time for (a) Example 1 and (b) Example 2. Details provided in the text.

As with most numerical methodologies, the future work for this project is to investigate more computationally efficient means of accomplishing the ray-tracing methodology without sacrificing the quantification of the method. All images presented in this work took on the order of several minutes to generate (using an unoptimized, single-processor ray-tracer). Parallelization as in [19] and software optimization will certainly aid in accelerating the visualization process. We believe that balancing efficiency, robustness, and accuracy will lead to new visualization methodologies which are high-order finite element aware.

Acknowledgments

The authors thank Dr. Steve Parker and Dr. Pete Shirley of the School of Computing, University of Utah, Dr. Spencer Sherwin of the Imperial College London (United Kingdom), and Dr. Jan Hesthaven of Brown University for helpful discussions, remarks, and support. The first author gratefully acknowledges the assistance of Mr. James Bigler in understanding the Utah Real-Time Ray Tracer (RTRT). The second author gratefully acknowledges the computational support and resources provided by the Scientific Computing and Imaging Institute at the University of Utah and, specifically, the NCRR cluster resources provided by the NCRR Center for Bioelectric Field Modeling, Simulation, and Visualization (www.sci.utah.edu/ncrr), NIH NCRR Grant No. 5P41RR012553-02, upon which simulations for this paper were accomplished.

References

[1] G.E. Karniadakis and R.M. Kirby, *Parallel Scientific Computing in C++ and MPI.* Cambridge Univ. Press, 2003.
[2] I. Babuska, B. Szabo, and I. Katz, "The P-Version of the Finite Element Method," *SIAM J. Numerical Analysis,* vol. 18, p. 515, 1981.
[3] I. Babuska and M. Dorr, "Error Estimates for the Combined H- and P-Version of the Finite Element Method," *J. Numerical Math.,* vol. 37, p. 257, 1981.
[4] A. Patera, "A Spectral Method for Fluid Dynamics: Laminar Flow in a Channel Expansion," *J. Computational Physics,* vol. 54, p. 468, 1984.

[5] G. Karniadakis, E. Bullister, and A. Patera, "A Spectral Element Method for Solution of Two- and Three-Dimensional Time Dependent Navier-Stokes Equations," *Finite Element Methods for Nonlinear Problems*, Springer-Verlag, p. 803, 1985.
[6] G. Coppola, S. Sherwin, and J. Peiro, "Non-Linear Particle Tracking for High-Order Elements," *J. Computational Physics,* vol. 172, pp. 356-380, 2001.
[7] D. Wiley, H. Childs, B. Gregorski, B. Hamann, and K. Joy, "Contouring Curved Quadratic Elements," *Data Visualization Conf., Proc. VisSym 2003,* pp. 167-176, 2003.
[8] D. Wiley, H. Childs, B. Hamann, and K. Joy, "Ray Casting Curved-Quadratic Elements," *Data Visualization Conf., Proc. VisSym 2004,* pp. 201-209, 2004.
[9] D. Wiley, "Approximation and Visualization of Scientific Data Using Higher-Order Elements," PhD dissertation, Univ. of California, Davis, 2003.
[10] M. Brasher and R. Haimes, "Rendering Planar Cuts through Quadratic and Cubic Finite Elements," *Proc. IEEE Visualization Conf.,* 2004.
[11] G.E. Karniadakis and S.J. Sherwin, *Spectral/hp Element Methods for CFD.* Oxford Univ. Press, 1999.
[12] Y. Zhou, M. Garland, and R. Haber, "Pixel-Exact Rendering of Spacetime Finite Element Solutions," *Proc. IEEE Visualization Conf.,* 2004.
[13] W. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics,* vol. 21, no. 4, pp. 163-169, July 1987.
[14] Y. Livnat and C. Hansen, "View Dependent Isosurface Extraction," *Proc. IEEE Visualization Conf. '98,* pp. 175-180, Oct. 1998, http://www.sci.utah.edu/publications/yarden98/wise.ps.gz.
[15] "Dynamic View Dependent Isosurface Extraction," Technical Report UUSCI-2003-004, Univ. of Utah, SCI Inst., 2003, http://www.sci.utah.edu/publications/yarden03/UUSCI-2003-004.pdf.
[16] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno, "Optimal Isosurface Extraction from Irregular Volume Data," *Proc. 1996 Symp. Volume Visualization,* pp. 31-38, 1996.
[17] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Adaptive TetraPuzzles–Efficient Out-of-Core Construction and Visualization of Gigantic Polygonal Models," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 796-803, Aug. 2004, http://www.crs4.it/vic/cgi-bin/bib-page.cgi?id='Cignoni:2004:ATE'.
[18] S.-E. Yoon, B. Salomon, R. Gayle, and D. Manocha, "Quick-VDR: Interactive View-Dependent Rendering of Massive Models," *Proc. IEEE Visualization Conf.,* pp. 131-138, 2004.
[19] S. Parker, M. Parker, Y. Livnat, P. Sloan, C. Hansen, and P. Shirley, "Interactive Ray Tracing for Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 3, pp. 238-250, July-Sept. 1999.
[20] W. Martin, E. Cohen, R. Fish, and P. Shirley, "Practical Ray Tracing of Trimmed NURBS Surfaces," *J. Graphics Tools,* vol. 5, no. 1, pp. 27-52, 2000.
[21] C. Rossel, F. Zeilfeldur, G. Nurnberger, and H.-P. Seidel, "Reconstruction of Volume Data with Quadratic Super Splines," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 4, pp. 397-409, July/Aug. 2004.
[22] P.L. Williams, N.L. Max, and C.M. Stein, "A High Accuracy Volume Renderer for Unstructured Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 4, no. 1, pp. 37-54, Jan.-Mar. 1998.
[23] T.J.R. Hughes, *The Finite Element Method.* Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1987.
[24] B. Szabó and I. Babuska, *Finite Element Analysis.* New York: John Wiley & Sons, 1991.
[25] M. Deville, E. Mund, and P. Fischer, *High Order Methods for Incompressible Fluid Flow.* Cambridge Univ. Press, 2002.
[26] A.S. Glassner, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications,* vol. 4, no. 10, pp. 15-22, 1984.
[27] A. Fujimoto, T. Tanaka, and K. Iwata, "Arts: Accelerated Ray-Tracing System," *IEEE Computer Graphics and Applications,* vol. 6, no. 4, pp. 16-26, 1986.
[28] S. Rubin and T. Whitted, "A Three-Dimensional Representation for Fast Rendering of Complex Scenes," *Computer Graphics,* vol. 14, no. 3, pp. 110-116, 1980.
[29] A.S. Glassner, *An Introduction to Ray Tracing.* San Francisco, Calif.: Morgan Kauffmann Publishers, Inc., 2000.
[30] C. Canuto and A. Quarteroni, "Approximation Results for Orthogonal Polynomials in Sobolev Spaces," *Math. of Computation,* vol. 38, no. 157, pp. 67-86, 1982.
[31] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang, *Spectral Methods in Fluid Mechanics.* Springer-Verlag, 1987.
[32] D. Funaro, *Polynomial Approximations of Differential Equations: Lecture Notes in Physics,* vol. 8, Springer-Verlag, 1992.
[33] P. Hanrahan, "Ray Tracing Algebraic Surfaces," *Proc. SIGGRAPH Computer Graphics,* vol. 17, no. 3, pp. 83-90, 1983.
[34] K. Hoffman and R. Kunze, *Linear Algebra.* Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1961.
[35] L.N. Trefethen and I.D. Bau, *Numerical Linear Algebra.* Philadelphia, Penn.: SIAM, 1997.
[36] J.R. Winkler, "A Companion Matrix Resultant for Bernstein Polynomials," *Linear Algebra and Its Applications,* vol. 362, pp. 153-175, 2003.
[37] "The Transformation of the Companion Matrix Resultant between the Power and Bernstein Polynomial Bases," *Applied Numerical Math.,* vol. 48, pp. 113-126, 2004.
[38] R.M. Kirby, T.C. Warburton, S.J. Sherwin, A. Beskok, and G.E. Karniadakis, "The $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ Code: Dynamic Simulations without Remeshing," *Proc. Second Int'l Symp. Computational Technologies for Fluid/Thermal/Chemical Systems with Industrial Applications,* ASME PVP Division, vol. 397, 1999.

Blake Nelson received the BS degree in computer science from the University of Utah in 2001. He is a MS candidate in computer science at the University of Utah. His research interests include graphics, scientific visualization, and software architecture.

Robert M. Kirby received ScM degrees in computer science and in applied mathematics and the PhD degree in applied mathematics from Brown University. He is an assistant professor of computer science at the University of Utah's School of Computing and is a member of the Scientific Computing and Imaging Institute at Utah. His research interests lie in scientific computing and visualization. He is a member of the IEEE.