Hierarchichal Representation and Computation of Approximate Solutions in Scientific Simulations

Wayne H. Enright
Department of Computer Science
University of Toronto
Toronto
Canada, M5S 3G4

Abstract

In many applications involving large scale scientific computing a massive amount of data is generated in a typical simulation. Such simulations generally require the numerical solution of systems of differential equations where the results are often generated remotely using special high-performance software and computer systems and then examined and investigated interactively using visualization tools. The visualization packages are usually run on local workstations and make use of colour, lighting, texture, sound and animation to highlight and reveal interesting characteristics or features of the approximate solution. This 'interactive' viewing of the data is the 'rendering' stage of the modeling process and it can be very selective and local in the sense that only a subset of the variables are rendered and then only in regions where something interesting is happening. The result is that, in many simulations, large amounts of data must be stored and shared in a distributed environment while only a very small fraction of this data will ever be viewed by anyone. In this paper we propose an approach, using a hierarchichal representation of the approximate solution, which will avoid the generation and storage of data that is not required.

1 Introduction

In scientific simulations results are often represented by approximate solutions to large systems of differential equations – either partial differential equations (PDEs) or ordinary differential equations (ODEs). Such simulations arise in all areas of science and engineering. With the dramatic increase in computing power and the development of more accurate and realistic mathematical models there has been an evolution in how the results of a simulation can be viewed and/or interpreted.

It is typically the case that an underlying numerical method will determine a discrete approximation to the solution on an adaptive unstructured mesh.

While tables of the numerical values associated with the meshpoints can be used to represent the discrete solution, such tables are difficult to interpret or understand. A research area has been developed to address the question of how to present and display results in the most effective way. This area, scientific visualization, has introduced color, sound and animation together with other computer graphics techniques to effectively display key properties of the results of large scale simulations. In order to use these techniques effectively the property to be rendered must be represented on a fine enough mesh to match the resolution of the hardware display device being used (typically a computer monitor). It must be emphasized that this resolution is independent of the underlying discrete mesh which is a property of the underlying numerical method and the accuracy that is specified in the simulation. The accuracy of the simulation affects the number of meshpoints and the total computer time required to generate the approximate solution on the initial (coarse) mesh. The resolution of the renderer will impose a constraint on how fine the 'refined' mesh must be in order to avoid distracting artifacts in the displayed images.

The key idea we advocate in this paper is the use of multivariate piecewise polynomial interpolants to represent the results of scientific simulations. These piecewise polynomials are defined on the coarse initial mesh, interpolating the discrete data, and they can be evaluated during the rendering process to deliver a discrete approximation to the solution on an arbitrary mesh (that is consistent with the initial mesh). It is crucial develop a language-independent representation of such piecewise polynomials so that different application environments can access and directly manipulate or extend these results.

We propose the use of a paradigm where the 'approximate' solution associated with a simulation is defined in terms of an approximation on a hierarchichal collection of associated consistent meshes. An underlying numerical method explicitly produces a discrete approximate solution on a relatively coarse initial mesh. This initial approximation, together with a generic procedure for 'extending' this approximation to any mesh that is a refinement of the initial mesh, defines the 'general' approximate solution. The key implication is that the data corresponding to a refined mesh (associated with the general approximate solution) need not be explicitly determined until it is needed since only the data associated with the coarse mesh is necessary to represent or characterize the general approximate solution.

Examples of this approach arising from models using ordinary and partial differential equations are presented to illustrate the significant reductions in both storage requirements and computer time that can be achieved. We show that speed up factors and storage reduction factors of between 10 and 1000 can be realised and that this approach can be particularly effective when working with 2D and 3D mathematical models.

1.1 Background Motivation

The hierarchichal data structure associated with our approach can adapt dynamically to reflect the changing demands of the interactive rendering process.

In particular in applications involving 'data mining' or 'feature discovery' the renderer will be 'steered' by a user to focus on particular subsets of the data on parts of the domain of interest.

An efficient implementation of this approach will involve a careful matching of the data structure, used to represent the general numerical solution to the hierarchichal hardware that is available to store the information. The approach we develop makes use of local information only (in the refinement step) and is therefore suitable for use in a parallel environment. That is, the polynomial associated with each element of the coarse mesh can be determined and rendered independently and without synchronization.

Although the examples we introduce and present in this paper have been implemented in MATLAB (primarily since it is an effective environment for rapid development of prototype 'proof of concept' systems) more efficient object-oriented systems are under development in C++.

1.2 Outline of Presentation

In the next section we will describe the approach for one Dimensional problems (ODEs) and show how recent research in the development of 'Continuous' numerical methods for ODEs can be interpreted as a canonical example of a general numerical solution. This general numerical solution will be represented by a univariate piecewise polynomial defined on the coarse mesh (associated with the underlying discrete approximation). The piecewise polynomial can be evaluated at any point in the interval of interest. In particular it can be used to approximate the solution on any mesh that is a refinement of the original coarse mesh.

We will then, in section 3, show how this approach generalizes to two and three Dimensional problems (PDEs), where multivariate piecewise polynomials are associated with the coarse mesh. These piecewise polynomials then represent the general solution and they can be used to approximate the solution at any point in the domain of interest (in particular, at any point that arises during the rendering process). We present detailed results for two example PDE problems which illustrate and quantify some of the advantages and limitations of this approach. These examples reveal the typical trade-off and relationships that are observed between the resolution of the final visualization and the coarseness or accuracy of the general solution.

We will conclude the paper with a discussion of current ongoing related work. We will also discuss the implications for high level problem solving environments where this approach can be used effectively provided one adopts a generic language-independent representation for the problem specification and the general approximate solution.

2 ODEs: The 'Easy' Case:

2.1 The Generic Approach in One Dimension

Consider the 1^{st}-order system of ODEs

$$y' = f(x, y).$$

A p^{th}-order, s-stage RK method determines

$$y_i = y_{i-1} + h \sum_{j=1}^{s} \omega_j k_j,$$

where

$$k_j = f(x_{i-1} + hc_j, y_{i-1} + h \sum_{r=1}^{s} a_{jr} k_r).$$

A Continuous extension (CRK) is determined by adding extra stages to obtain an order p approximation for $x \in (x_{i-1}, x_i)$

$$u_i(x) = y_{i-1} + h \sum_{j=1}^{\bar{s}} b_j(\frac{x - x_{i-1}}{h}) k_j,$$

where $b_j(\tau)$ is a polynomial of degree p.

That is, one determines $k_{s+1}, k_{s+2} \ldots k_{\bar{s}}$ and polynomials $b_j(\tau)$ to ensure:

$$u_i(x_i) = y_i,$$

$$u_i'(x_i) = f(x_i, y_i),$$

$$u_i(x) = y(x) + O(h^p), \quad x \in [x_{i-1}, x_i].$$

Collectively the $u_i(x)$ define a piecewise polynomial approximation to $y(x)$ that is $O(h^p)$ accurate for $x \in [x_0, x_F]$, on an associated mesh $x_0 < x_1 \cdots < x_N = x_F$. A refinement of this coarse mesh can be defined by introducing K equally spaced points per subinterval, $[x_{i-1}, x_i]$, and one can evaluate the piecewise polynomial on this fine mesh to obtain a discrete approximation that is more appropriate for rendering.

2.2 A One Dimensional Example

A predator-prey relationship can be modeled by the IVP:

$$y_1' = y_1 - 0.1 y_1 y_2 + 0.02x$$

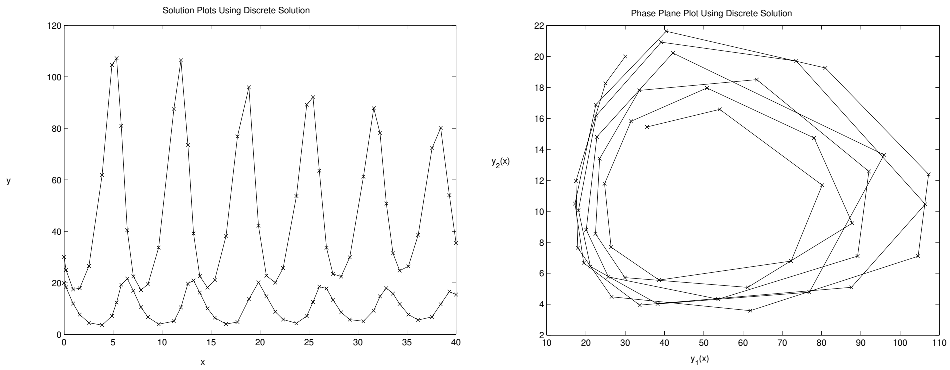
$$y_2' = -y_2 + 0.02 y_1 y_2 + 0.008x$$

with

$$y_1(0) = 30, \;\; y_2(0) = 20, \; x \in [0, 40].$$

where $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x. The solution can then be visualized using a standard y/x solution plot or using a 'phase plane' plot. Consider the use of a standard ODE solver such as ode45 of MATLAB which is based on a fourth order CRK formula.

Figure 1 shows the results when one attempts to display the solution using the discrete solution only. In this case the simulation presented no numerical difficulties and ode45 determined an accurate approximation with fewer than 50 time steps. Direct rendering of this discrete solution gives a distorted impression of the 'shape' of the solution and the behaviour of the phase-plane 'portrait' of the solution.



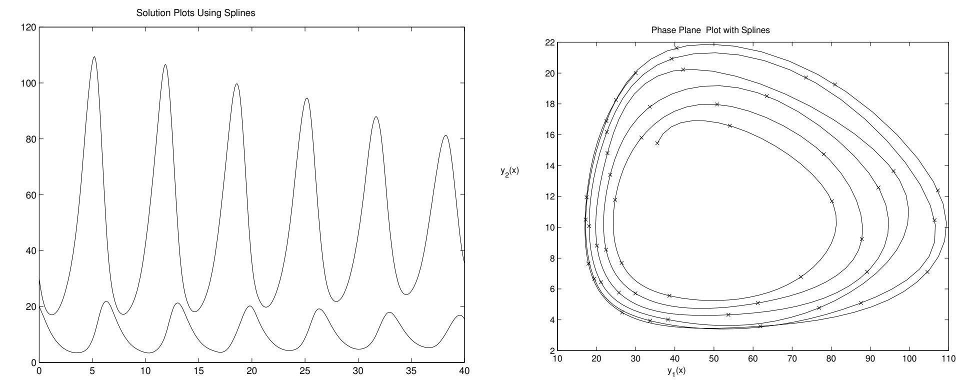
(a) Solution Plot

(b) Phase Plane Plot

Figure 1: Visualization Using Piecewise Linear Interpolation.

For one-dimensional visualization, the rendering process can sometimes use cubic splines to present a more realistic or less distracting view of the results. For example the cubic spline that interpolates the discrete values produced by ode45 can be evaluated on a fine mesh (we have used a fine mesh obtained by subdividing each of the coarse-mesh intervals associated with the discrete solution into 20 equally spaced sub-intervals). Figure 2 illustrates this technique for rendering the discrete solution generated by ode45. This is definitely an improvement but the fixed low order of this technique limits its effectiveness as a general solution. Note that while the solution plots are realistic and the phase plane portraits are less distracting they still contain unrealistic features that are not present in the true solution.

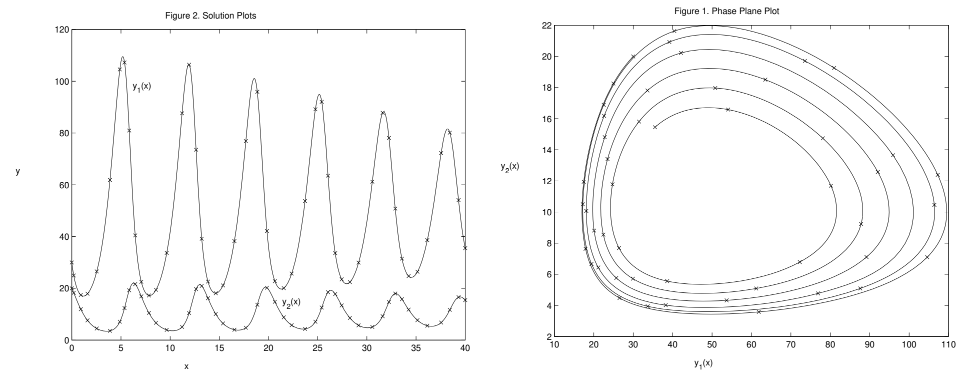
The special interpolant used to generate the visualizations presented in Figure 3 are associated with the continuous Runge-Kutta formula that is the basis of the method ode45. We have evaluated the underlying piecewise polynomial at the same fine mesh that we used with the cubic spline visualization to obtain these results. Note that, in particular, the off-mesh accuracy of this latter



(a) Solution Plot (b) Phase Plane Plot

Figure 2: Visualization Using Spline Interpolation.

piecewise polynomial leads to a much more realistic phase plane portrait.



(a) Solution Plot (b) Phase Plane Plot

Figure 3: Visualization Using Special Interpolation.

3 Application to PDEs

3.1 The generic Approach in Two Dimensions and Higher:

Assume the PDE is a 2D, 2^{nd}-order problem:

$$Lu = g(x, y, u, u_x, u_y), \tag{1}$$

where

$$L = a_1(x,y)u_{xx} + a_2(x,y)u_{yy} + a_3(x,y)u_{xy}.$$

Assume also that an underlying method has generated a discrete approximation to u, u_x and u_y on a mesh consisting of rectangular elements. The approximations associated with each meshpoint, $(\bar{x}, \bar{y})$, satisfy

$$\begin{aligned} u(\bar{x},\bar{y}) - \bar{u} &= O(h^{p_1}), \\ u_x(\bar{x},\bar{y}) - \bar{u}_x &= O(h^{p_2}), \\ u_y(\bar{x},\bar{y}) - \bar{u}_y &= O(h^{p_2}), \end{aligned} \tag{2}$$

where h is the maximum mesh spacing in either x or y. Note that with this assumption we have $O(h^2)$ accuracy at off-mesh points if we use piecewise linear interpolation of the mesh data. For each element, e, We determine a bivariate polynomial, $\bar{p}_{d,e}(x,y)$, of degree d that

- interpolates the mesh data (12 constraints)
- almost satisfies (1) at m collocation points
- provides optimum accuracy at off-mesh points

The best we can expect is for $(x, y) \in e$

$$\begin{aligned} u(x,y) - \bar{p}_{d,e}(x,y) &= O(h^p), \\ u_x(x,y) - \frac{\partial \bar{p}_{d,e}(x,y)}{\partial x} &= O(h^{p-1}), \\ u_y(x,y) - \frac{\partial \bar{p}_{d,e}(x,y)}{\partial y} &= O(h^{p-1}), \end{aligned} \tag{3}$$

where $p = min(p_1, p_2 + 1, d + 1)$.

For a detailed discussion and justification of this approach see [1].

3.2 A Two Dimensional Example:

Consider the following PDE from the ELLPACK collection [3]:

$$u_{xx} + u_{yy} = \cos(\pi y)u - (1 + \sin(\pi x))u_x + f(x, y)$$

on the domain

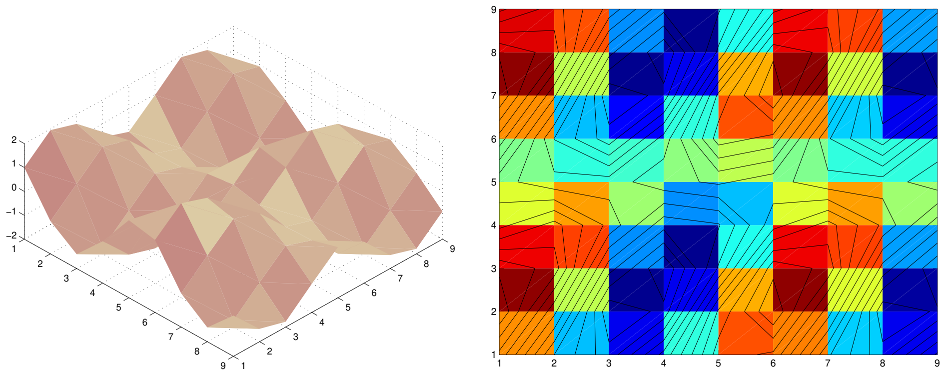
$$0 \le x \le 1, \; 0 \le y \le 1,$$

with boundary conditions

$$u(x, y) = cos(By) + sin\, B(x - y),$$

where $B = \pi$ or $B = 10$ and $f(x, y)$ is defined to ensure that the true solution agrees with that specified on the boundary for the whole domain.

For this PDE defined by $B = 10$, visualizing with an 8×8 coarse mesh (Discrete solution only), and no refinement for rendering we obtain the results presented in Figure 4. Note that the off-mesh values required in this rendering are generated using bivariate piecewise linear interpolation. Clearly these results are distracting and not satisfactory.



(a) Surface Plot

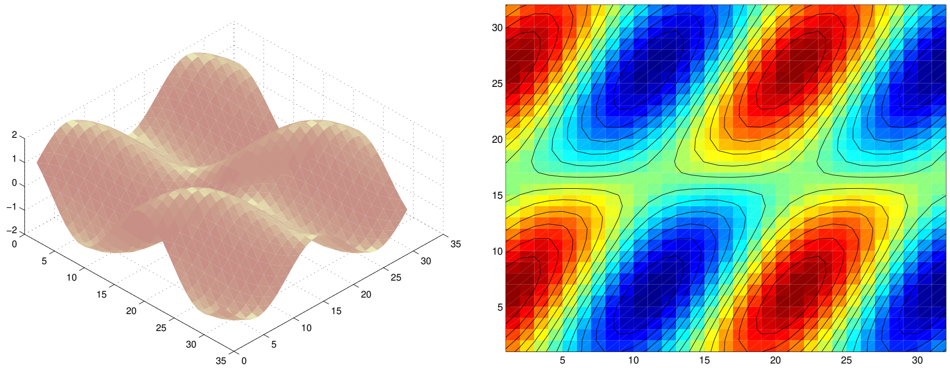
(b) Contour Plot

Figure 4: Visualization Using Piecewise Linear Interpolation.

When our approach is applied to improve the resolution of this discrete solution we can use a piecewise cubic evaluated at the refined mesh defined by subdividing each of the 64 elements of the discrete solution with a 4×4 uniform partitioning. The resulting visualization is presented in Figure 5 where the improved resolution has resulted in a more realistic and less distracting display of the solution.

Even better resolution is possible by evaluating the same piecewise cubic on a finer mesh prior to rendering. When a 16×16 partitioning of each of the 64 elements is used one obtains the results presented in Figure 6 where the lighting effects and contour lines are now much more natural.

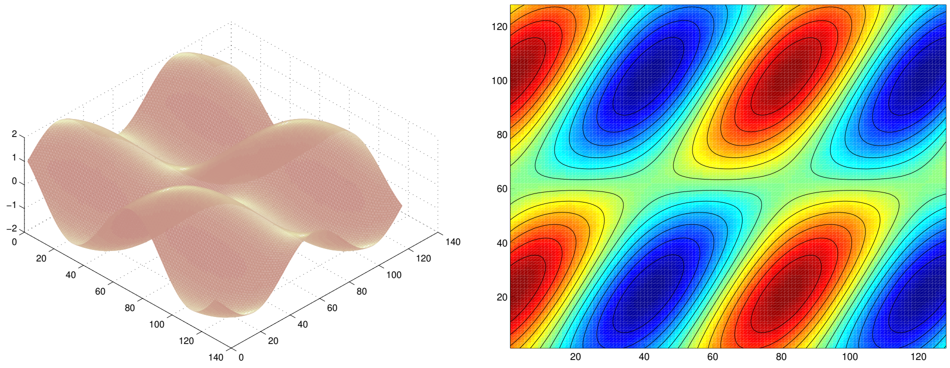
To illustrate the effects of using too coarse a mesh for the discrete solution we show in Figure 7 the results corresponding to an accurate discrete solution defined on a 4×4 coarse mesh with the associated piecewise cubic evaluated on a fine mesh obtained by a uniform 32×32 partitioning of each of the 16 elements. In this case the 'ripples' in the surface plots and the distortions evident in the contour lines reflect the influence of the underlying coarse mesh.



(a) Surface Plot

(b) Contour Plot

Figure 5: Visualization Using Piecewise Polynomial on 8×8 mesh with 4×4 refinement.



(a) Surface Plot

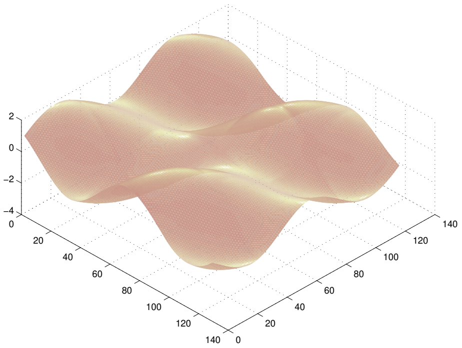
(b) Contour Plot

Figure 6: Visualization Using Piecewise Polynomial on 8×8 mesh with 16×16 refinement.

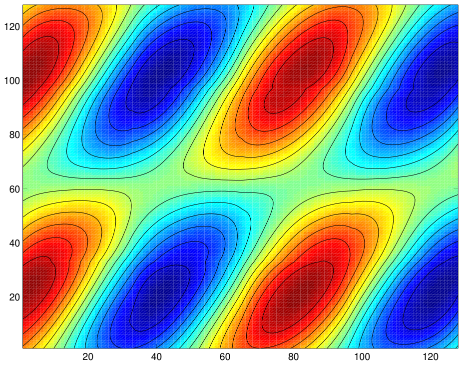
3.3 A Three Dimensional Example:

Consider the wave equation with two spatial dimensions describing a clamped vibrating membrane:

$$u_{tt} - .25(u_{xx} + u_{yy}) = 0,$$



(a) Surface Plot



(b) Contour Plot

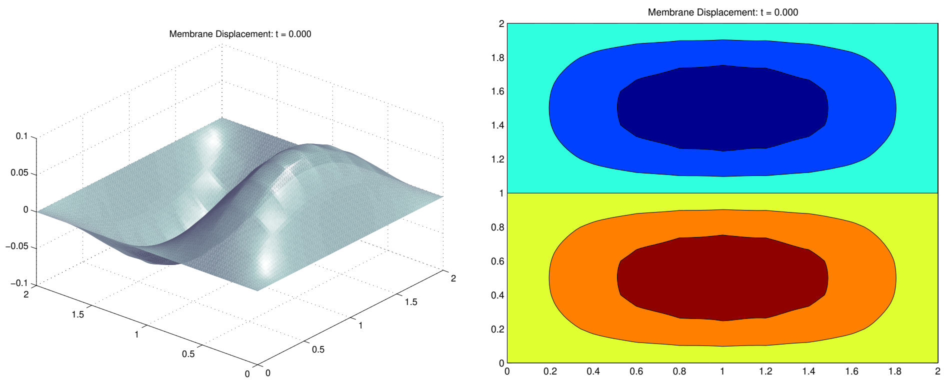
Figure 7: Visualization Using Piecewise Polynomial with 4×4 mesh and 32×32 Refinement.

on the domain: $0 \leq t \leq 2$, $0 \leq x \leq 2$, $0 \leq y \leq 2$, with boundary conditions: $u(t, x, y) = 0$, and initial conditions:

$$u(0, x, y) = 0.1 sin(\pi x) sin(\pi y/2), \; u_t(0, x, y) = 0.$$

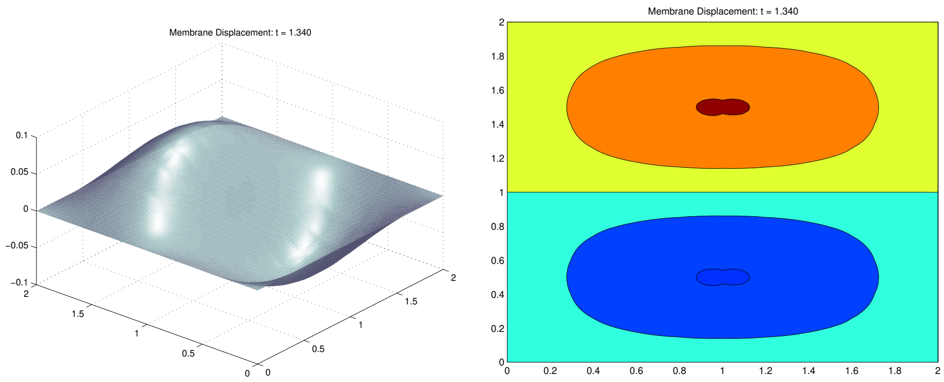
To display the solution to this problem one can use 'animation' and generate a movie representation of the solution by determining 'snapshots' of the solution for various values of $t >= 0$. For example, 100 values of t in the range $0 \leq t \leq 2$ gives a smooth animation that effectively diplays the motion of the membrane as it passes through one oscillation. This realistic visualization would then require the generation of 100 snapshots. We will use snapshots at $t = 0$, $t = 1.34$ and $t = 2.0$ to illustrate the typical resolution of this way of diplaying the results. For example if we use our approach with a $10 \times 10 \times 10$ coarse mesh and a uniform $10 \times 10 \times 10$ partitioning of each element to define the refined mesh we can use a tricubic piecewise polynomial to obtain the results presented in Figures 8, 9 and 10. Note that the complete animation (100 snapshots – each comprised of a surface or contour plot involving the solution approximated at 10000 points) is represented by the discrete values associated with the coarse mesh(consisting of only 1000 elements). This corresponds to a 'compression' factor of 1000, with respect to storage.

At this resolution some artifacts of the coarse mesh are visible and may be distracting. To obtain better visualization one could increase the degree of the piecewise polynomial or use a different initial coarse mesh. For example Figures 11, 12 and 13 show the results corresponding to the use of tricubics on this problem with a $20 \times 20 \times 20$ coarse mesh and a uniform $5 \times 5 \times 5$ partitioning of each element to define the refined mesh.



(a) Surface Plot (b) Contour Plot

Figure 8: Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 0$.



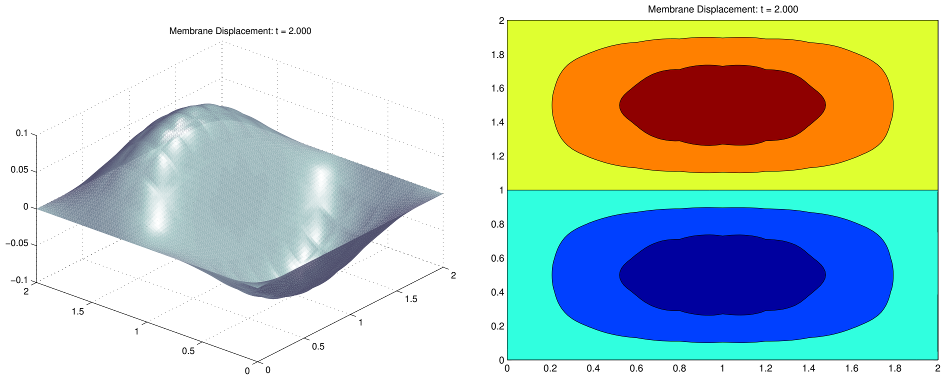
(a) Surface Plot (b) Contour Plot

Figure 9: Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 1.34$.

4 Extensions and Implications

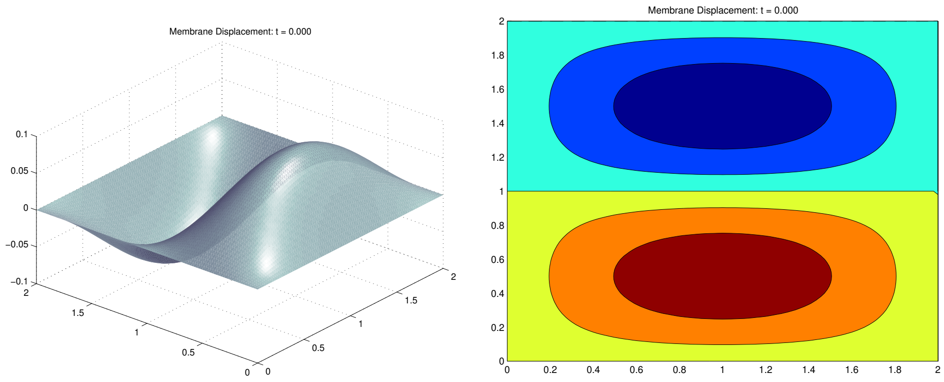
4.1 Extensions and related applications

Although, in our examples involving PDEs, we have focused on tensor product meshes with multivariate cubic interpolation the approach can be applied to unstructured triangular (or tetrahedral) meshes and higher degree multivariate



(a) Surface Plot (b) Contour Plot

Figure 10: Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 2.0$.

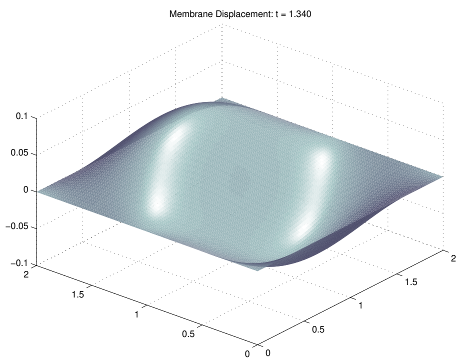


(a) Surface Plot (b) Contour Plot

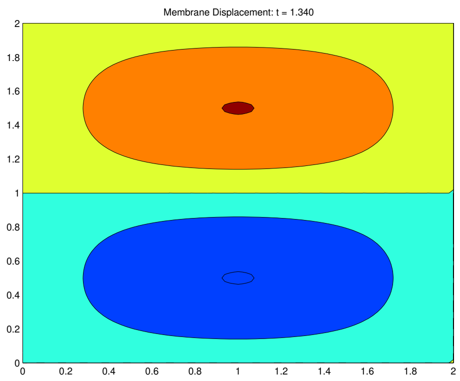
Figure 11: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 0$.

interpolation. (See [1] for a detailed discussion of the general applicability of the approach.) The underlying piecewise polynomial can also be directly exploited for error estimation or discontinuity detection and these extensions are being investigated at this time.

In the rendering application we have considered in this paper as well as in other application discussed elsewhere it is critical that the data structure used to store the general approximate solution be carefully chosen and available for use

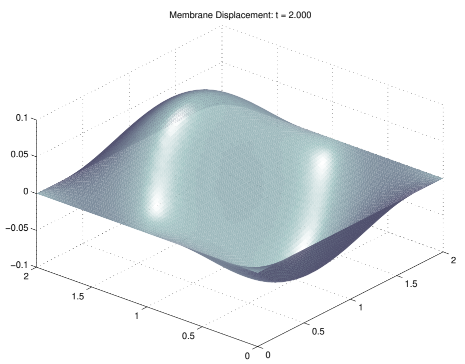


(a) Surface Plot

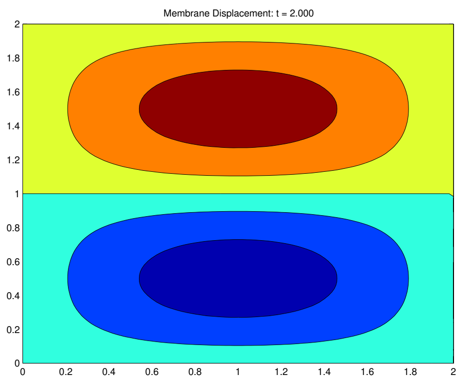


(b) Contour Plot

Figure 12: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 1.34$.



(a) Surface Plot



(b) Contour Plot

Figure 13: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 2.0$.

in a multi-language environment. Typical operations or transformations that can arise and which should be efficiently implemented include the multipoint evaluation of the general approximate solution, the location of extreme values of the general approximate solution and the checking of whether two different approximate solutions are 'close' to each other. The computer language most appropriate for implementing such operations will vary but a well designed data structure can be effectively shared by all such applications.

In order to carry out the numerical experiments reported in this paper we have adopted a recursive data structure using a rooted tree representation of the general approximate solution and implemented it in such a way that it could be accessed and manipulated using Fortran, Matlab or C.

Once one adopts such a generalized data structure other questions arise which require further investigation. For example we have assumed that the renderer must know the solution on a uniform, fine, rectangular mesh before it can display the solution. If one is interested for example in displaying contour lines one may be able to do this directly from the piecewise polynomial, without having to first perform a multipoint evaluation on a very fine uniform mesh (see [2] for an example of how contour lines can be directly computed).

References

[1] W.H. Enright, Accurate approximate solution of PDEs at off-mesh points, ACM Trans. Math. Soft., to appear, 2000.

[2] T.A. Grandine, Applications of Contouring, SIAM Review, 42, 2, 2000, pp. 297-316.

[3] J.R. Rice and R.F. Boisvert, *Solving Elliptic Problems using ELLPACK*, SCM series, 2, Springer-Verlag, New York, 1985.