Hierarchichal Representation and Computation of Approximate Solutions in Scientific Simulations

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

enright@cs.utoronto.ca

IFIP WG2.5 Working Conference 8

Software Architecture for Scientific Computing Applications

Ottawa, October 2000

The Basic Problem

- Results from large scale scientific simulations are generally approximate solutions to large systems of Differential Equations. Usually PDEs rather than ODEs.

- Only a very small fraction of the data will ever be viewed by anyone. (Why store and transmit all the data if nobody will ever view it?) We will consider storing and transmitting only enough data so all the data can be effeciently reproduced when needed.

- Generic examples arise in climate modelling, weather forecasting, and nuclear waste management.

Inherent Difficulties

- Generating the data – New PDE algorithms (HP scientific computation).
- Storing the data – Data structures and hierarchichal storage devices.
- Transmitting the data – Bandwidth, distributed computing/collabortation.
- Viewing the Data – Contour plots (2D) and level surfaces (3D). The use of colour, sound and animation.
- Mining the data – searching for patterns, identifying new patterns. Interactive 'steering' of renderer.

$\boxed{\text{Issues to be Considered}}$

- Accuracy/Resolution.
- Cost – Bandwidth, storage requirement and computer time.

$\boxed{\text{Related Issues}}$

- Image compression.
- Interpolation of scattered unstructured data.

The basic idea of this approach is to use a 'just-in-time' strategy for refining the approximate solution. (For example when rendering or data mining.)

- Store/transmit accurate coarse data only.

- Generate fine mesh data only when investigating a particular portion of the data. (Request generated and responded to in real time.)

- Use local information only during refinement (ideal for implementation on multiprocessors).

The problem and the approximate solution will be accessed/investigated in very different contexts:

1. Generation of the approximate solution: Usually in a HP computing environment using FORTRAN or C.

2. Visualizing of the approximate solution: Often using special packages such as VDK or Tecplot.

3. Data Mining or mathematical investigation: Often in a PSE such as Maple, Matlab or Mathematica.

There is a need for a generic, language-independent data structure to represent a coarse mesh (but accurate) approximate solution which can subsequently be refined (at low cost) when necessary.

$\boxed{\text{Rendering: A Typical Application}}$

To illustrate how this approach can be effective we will consider the difficulty of realistically rendering accurate 'coarse' data.

- A one dimensional example – an ODE.

- Extend this approach to 2D and 3D – PDEs.

- A generic 3D example is the generation of temperature contours or isobar charts in weather maps. Animation can be used to show evolution over time.

$\boxed{\text{A one Dimensional Example}}$

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

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

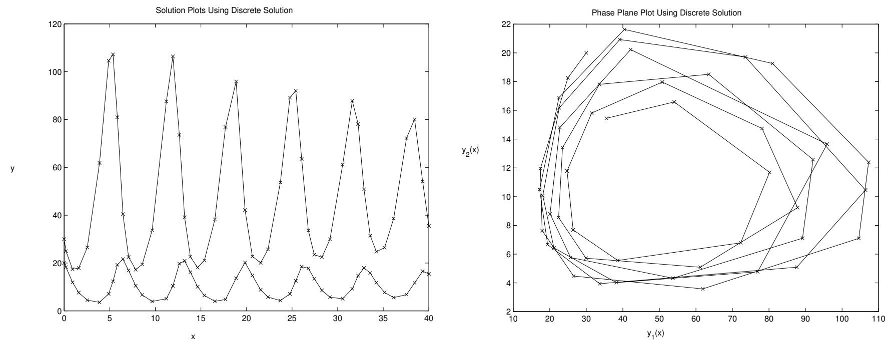
$$y_2' = -y_2 + 0.02 y_1 y_2 + 0.008x$$

with

$$y_1(0) = 30, \;\; y_2(0) = 20,$$

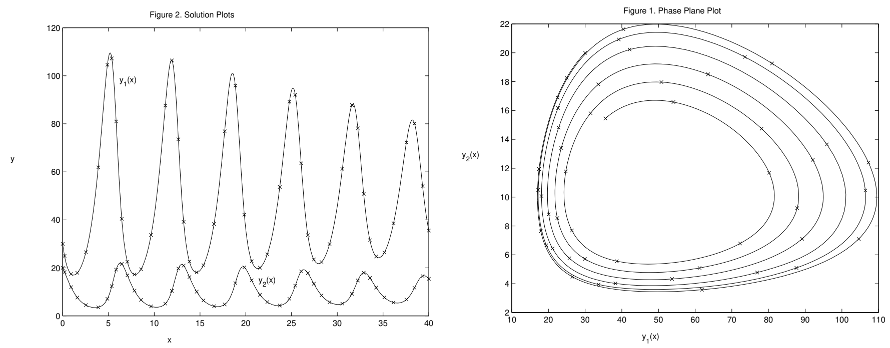
where $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x. The solution can then be visualized as a standard x/y solution plot or by a 'phase plane' plot. Consider the use of a standard ODE solver such as ode45 of MATLAB.

Rendering of an Approximate ODE solution:



(a) Solution Plot (b) Phase Plane Plot

Figure 1: Visualization Using Piecewise Linear Interpolation.



(a) Solution Plot (b) Phase Plane Plot

Figure 2: Visualization Using Special Interpolation.

$\boxed{\text{The Generic Approach in one Dimension}}$

Consider the 1^{st}-order system

$$y' = f(t, y).$$

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

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

where

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

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

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

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:

$$u_i(t_i) = y_i,$$

$$u_i^{'}(t_i) = f(t_i, y_i),$$

$$u_i(t) = y(t) + O(h^p), \quad t \in [t_{i-1}, t_i].$$

Collectively the $u_i(t)$ define a piecewise polynomial approximation to $y(t)$ that is $O(h^p)$ accurate for $t > t_0$.

$\boxed{\text{Choices for Data Structure}}$

There are alternatives for representing this 'general approximate solution' (the piecewise polynomial):

1. Save the discrete solution values only. (Requires 1 vector per step and access to ODE routine wherever the fine-mesh solution might be accessed.)

2. Save the coefficients of $u_i(x)$ explicitly. (Requires p vectors per step).

$\boxed{\text{Extension to 2D and 3D}}$

–For details see Enright(2000),'Accurate approximate solution of PDEs at off-mesh points', ACM TOMS, to appear.

Consider the following 2D-PDE from the ELLPACK collection

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

on the domain

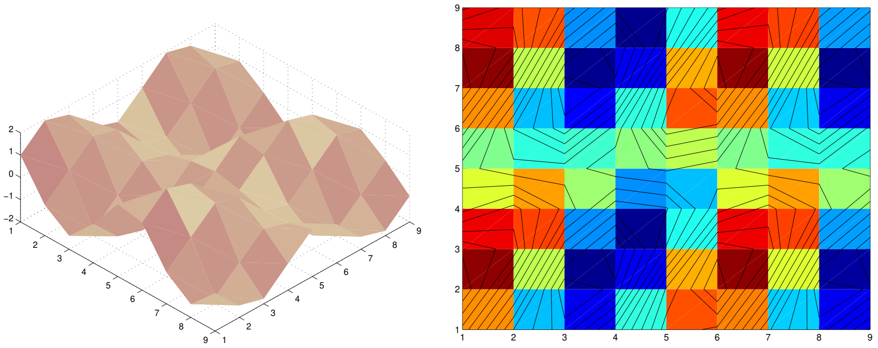
$$0 \leq x \leq 1,\ 0 \leq y \leq 1,$$

with boundary conditions

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

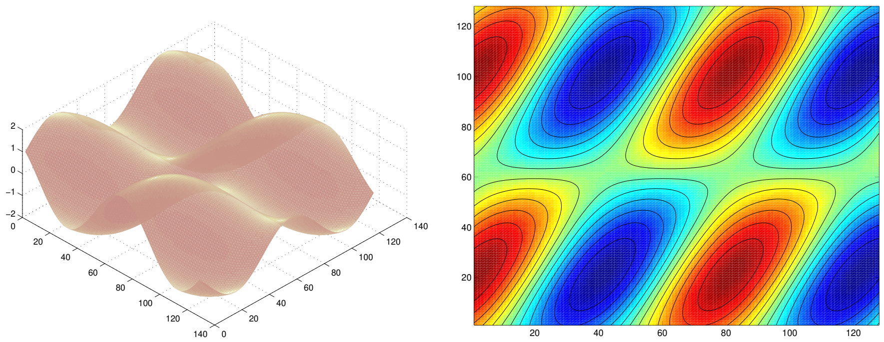
where $B = 10$.

Rendering the solution to a 2D-PDE:



(a) Surface Plot (b) Contour Plot

Figure 1: Visualization Using Piecewise Linear Interpolation.



(a) Surface Plot (b) Contour Plot

Figure 2: Visualization Using Piecewise Polynomial on 8×8 mesh with 16×16 refinement.

Two 3D-PDE Examples:

a)2D Wave Equation: (Vibrating Membrane)

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

domain: $0 \leq t \leq 2,\ 0 \leq x \leq 2,\ 0 \leq y \leq 2,$ bound cond: $u(t, x, y) = 0$, and init cond:

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

b)2D Heat Equation: (Heated Plate)

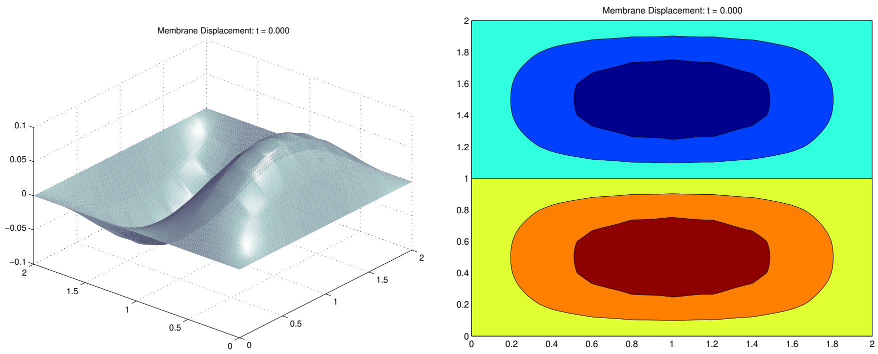
$$u_t - .0001(u_{xx} + u_{yy}) = 0,$$

domain: $0 \leq t \leq 5000,\ 0 \leq x \leq 4,\ 0 \leq y \leq 4,$ bound cond :

$$u(t, x, y) = exp(y) cos(x) - exp(x) cos(y),$$

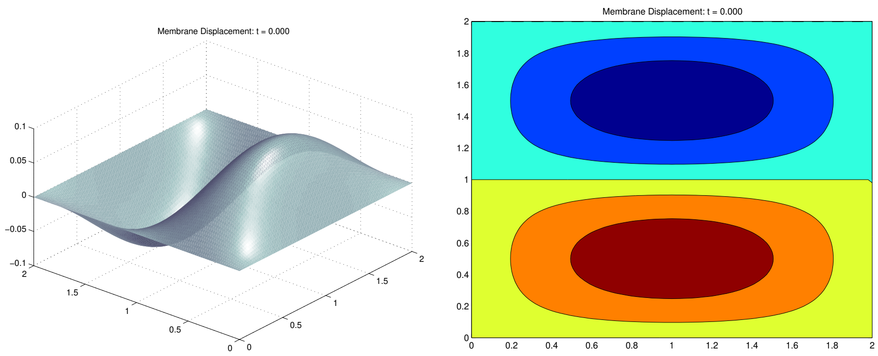
and init cond: $u(0, x, y) = 0.$

Snapshot at $t = 0$:



(a) Surface Plot (b) Contour Plot

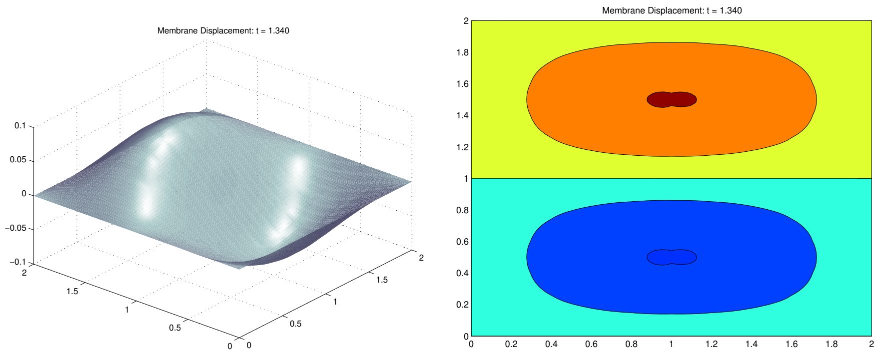
Figure 1: Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 0$.



(a) Surface Plot (b) Contour Plot

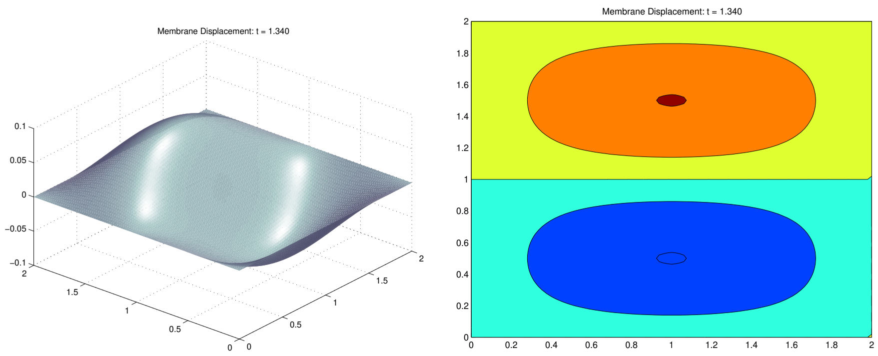
Figure 2: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 0$.

Snapshot at $t = 1.34$:



(a) Surface Plot (b) Contour Plot

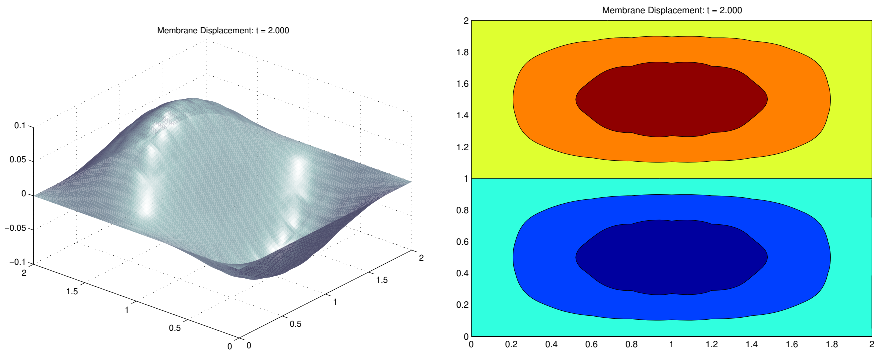
Figure 3: Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 1.34$.



(a) Surface Plot (b) Contour Plot

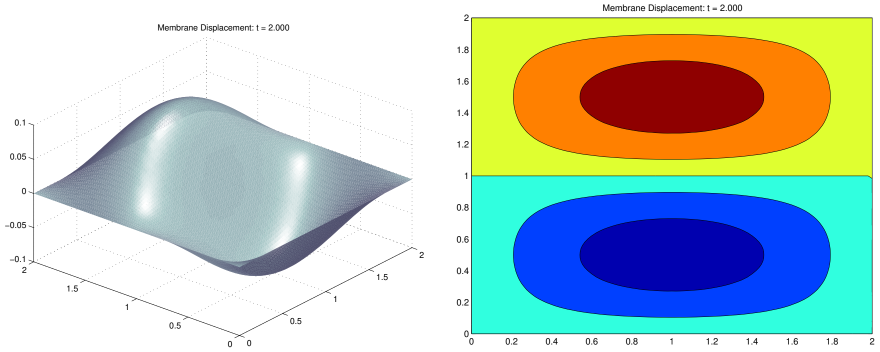
Figure 4: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 1.34$.

Snapshot at $t = 2.0$:



(a) Surface Plot (b) Contour Plot

Figure 5: 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 6: Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 2.0$.

IV) Generic Approach for PDEs

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

$$Lu = g(x, y, u, u_x, u_y), \qquad (1)$$

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 grid consisting of rectangular elements. The approximations associated with each meshpoint, $(\bar{x}, \bar{y})$, satisfy

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

where h is the maximum mesh spacing in either x or y.

We will also assume that we have $O(h^2)$ accuracy at off-mesh points.

For each element, e, We determine a bivariate polynomial, $\bar{p}_{d,e}(x, y)$, of degree d that

- interpolates the mesh data (9 or 12 constraints)

- almost satisfies the PDE at $(d + 1)^2 - 9$ or $(d + 1)^2 - 12$ points

The 'cost' of determining the coefficients of $\bar{p}_{d,e}(x, y)$ is a few evaluations of the PDE and the solution of one linear system of dimension $(d + 1)^2$.

Approach can be modified and extended in several ways:

- 'Nearby' solution values can be used if derivative data not available at meshpoints.

- Can be applied to unstructured triangular meshes.

- 3D problems.

- First-order, higher-order and mixed-order systems.