Fast Visualization/Animation of Approximate Solutions of PDEs

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

enright@cs.utoronto.ca

Workshop on the Mathematics of Computer Animation

Toronto, November 2002

Report of some results of an ongoing research project.

Collaborators:

Gonzalo Ramos, Emma Bradbury, Mohammad Kaazempur-Mofrad, Samir Hamdi, Paul Muir

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 that the relevant data can be efficiently regenerated when needed.

- Generic examples arise in climate modeling, 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/collaboration.
- 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.

Issues to be Considered

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

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. (NEOS, ELLPACK)

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. (A piecewise polynomial.)

Visualization: A Typical Application

To illustrate how this approach can be effective we will consider the difficulty of realistically displaying an accurate approximate solution generated on a 'coarse' mesh.

- 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.
- We will present a fast contouring algorithm that can be computed directly from an associated piecewise polynomial.

A 1-D example – an ODE Simulation:

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

$$y_1^{'} = y_1 - 0.1 y_1 y_2 + 0.02x$$

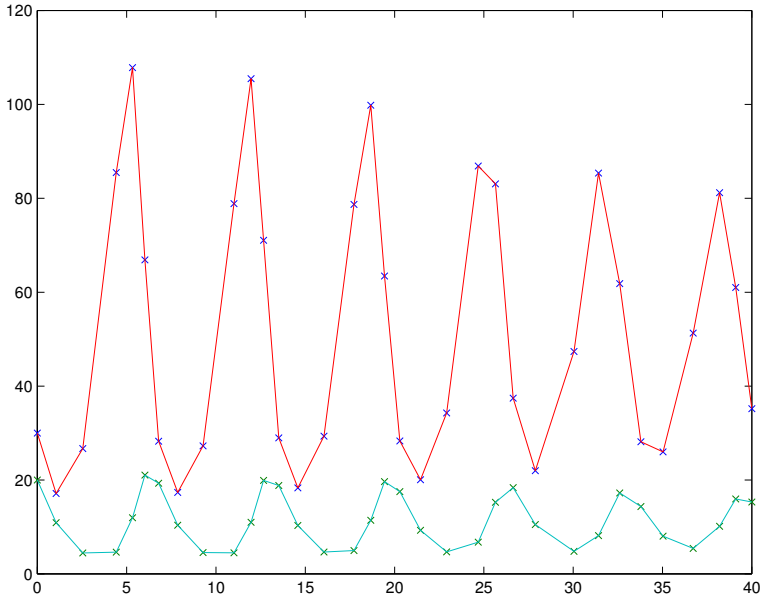
$$y_2^{'} = -y_2 + 0.02 y_1 y_2 + 0.008x$$

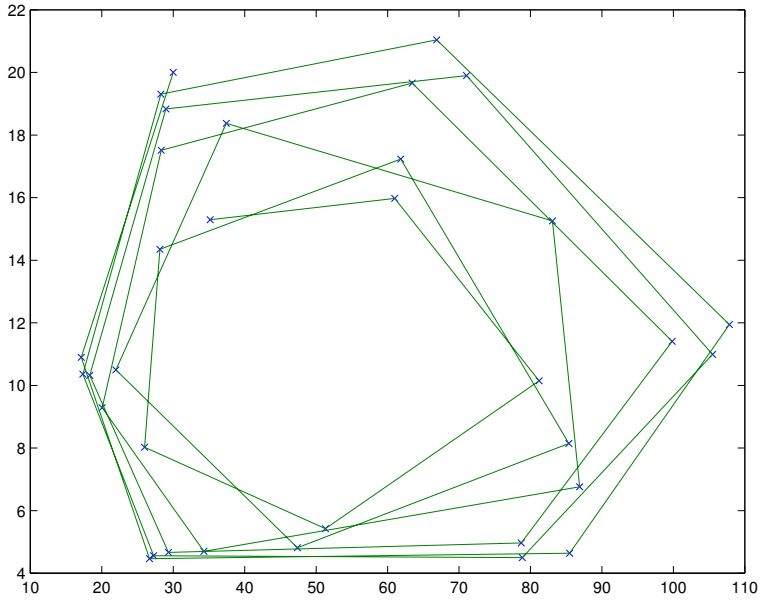
with

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

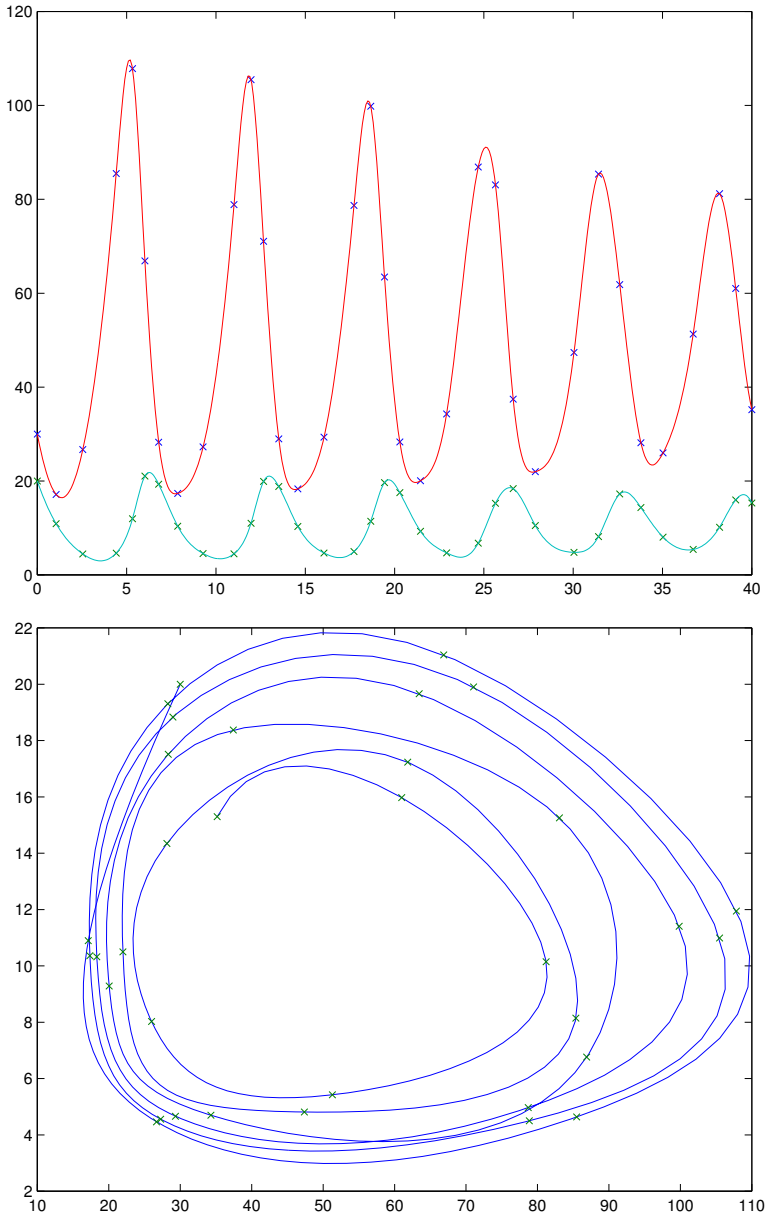
where $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x. The solution can then be visualized as a standard x/y solution plot or by a 'phase plane' plot. Consider the use of a standard ODE software package such as an 8th order CRK method to investigate properties of solutions to this problem.

A) Visualization Using Discrete Solution Only:

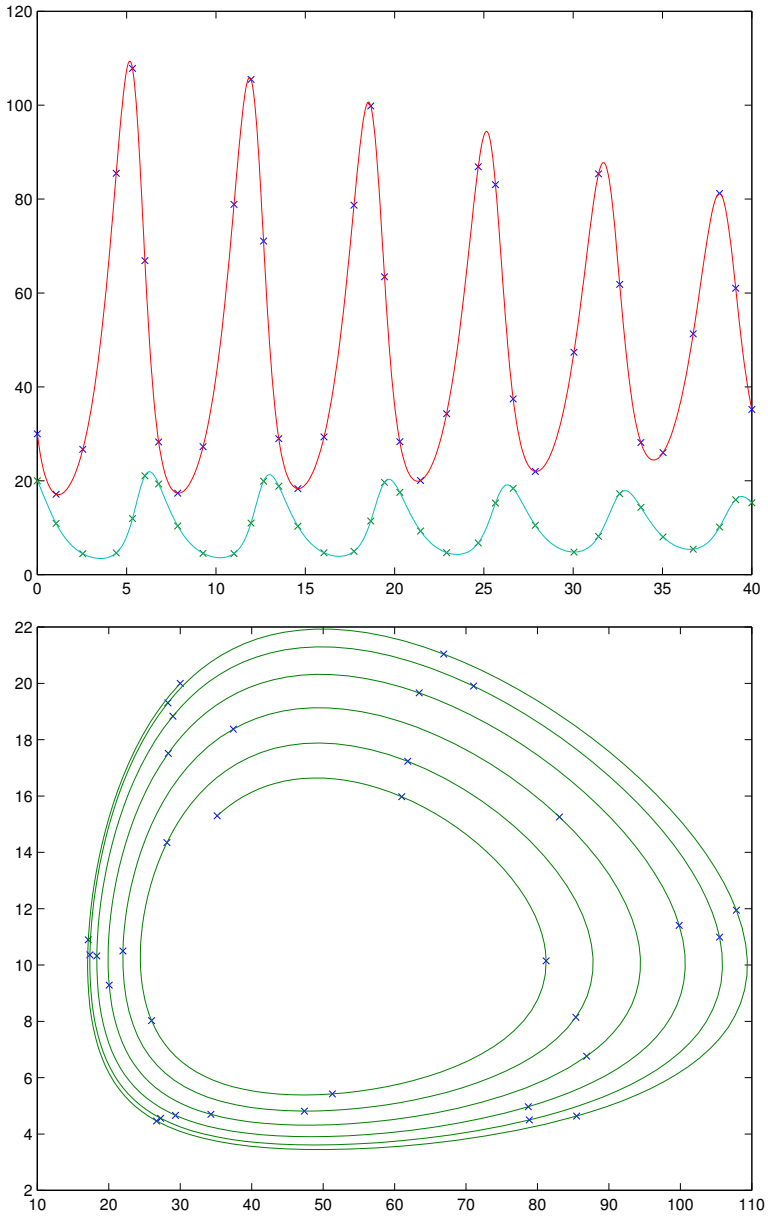




B) Visualization Using Splines:



C) Visualization Using a DEINT (Differential Equation Interpolant):



The Generic Approach in one Dimension

Consider the 1^{st}-order system

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

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:

$$u_i(x_i) = y_i,$$

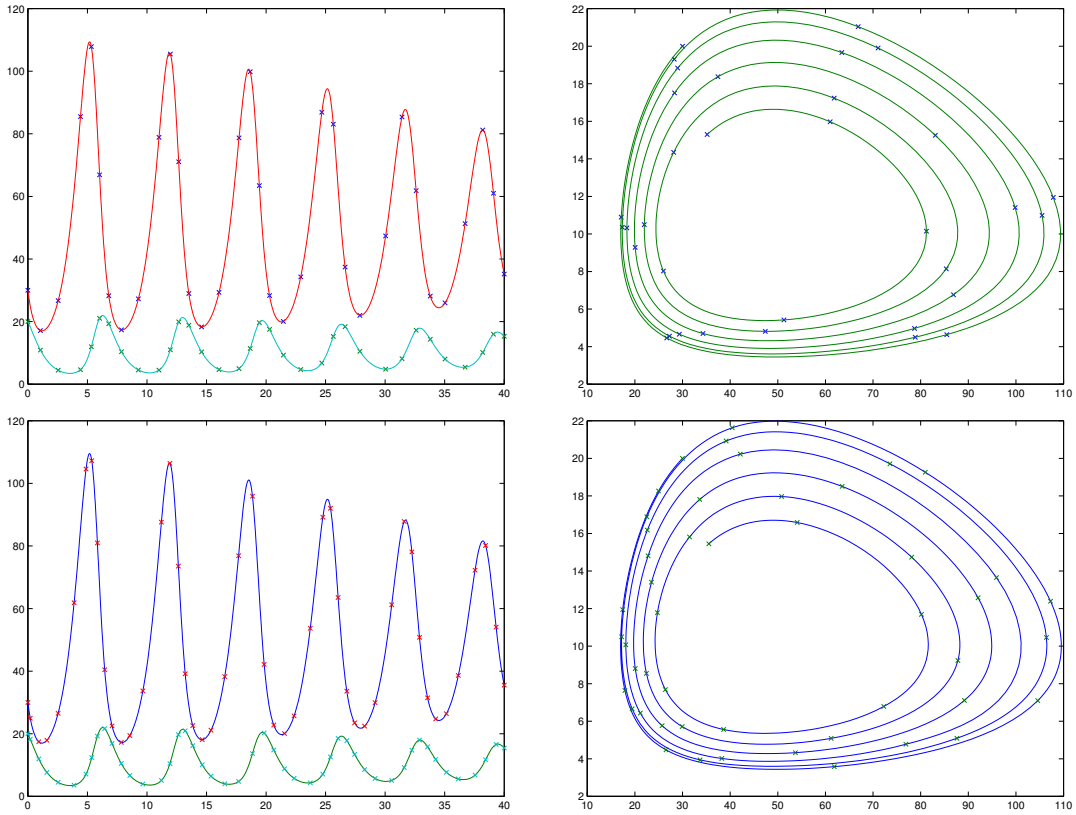
$$u_i^{'}(x_i) = f(x_i, y_i),$$

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

Collectively the $u_i(x)$ define a piecewise polynomial approximation to $y(x)$ that is $O(h^p)$ accurate for $x > x_0$. We call this pp the Differential Equation Interpolant (DEINT).

Different Numerical Methods will give indistinguishable Visualizations.

Consider approximate solutions to the predator/prey problem with the 8th order CRK method and the built-in MATLAB method, ode45. Visualizing using the associated DEINTs we have:



Extension to 2D and 3D

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

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

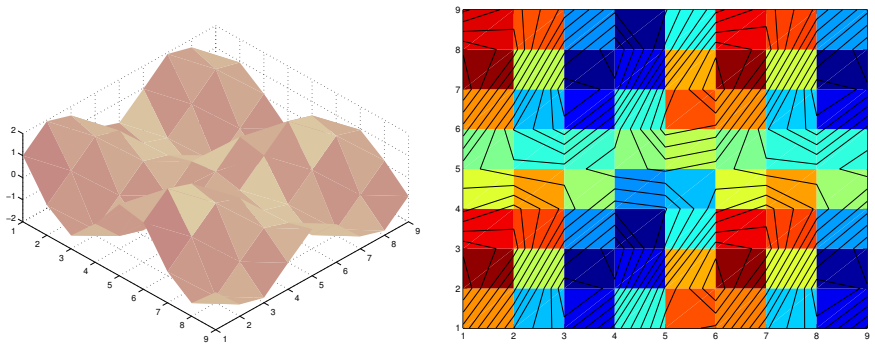
$$0 \leq x \leq 1,\ 0 \leq y \leq 1,$$

with boundary conditions

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

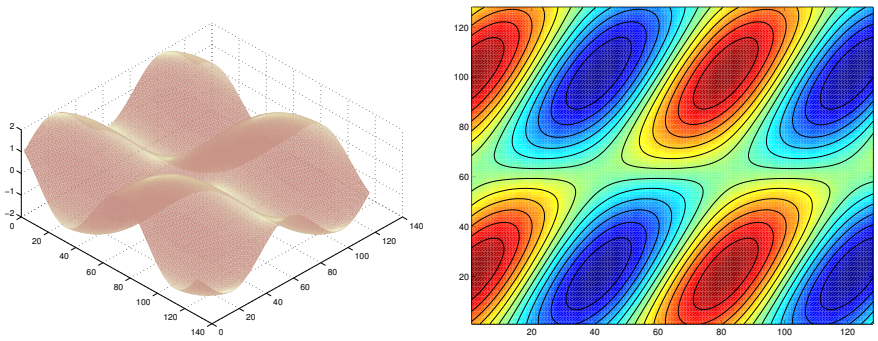
where $B = 10$.

Visualizing the solution with a DEINT:



(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.

A 3D PDE Example:

The Wave Equation: (Vibrating membrane)

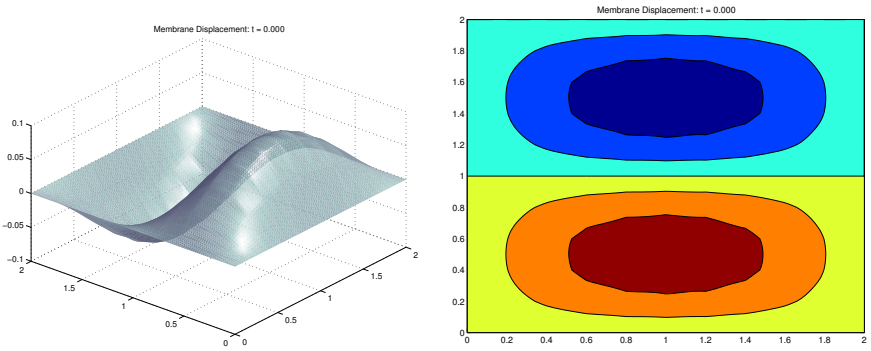
$$u_{tt} - .25(u_{xx} + u_{yy}) = 0,$$

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:

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

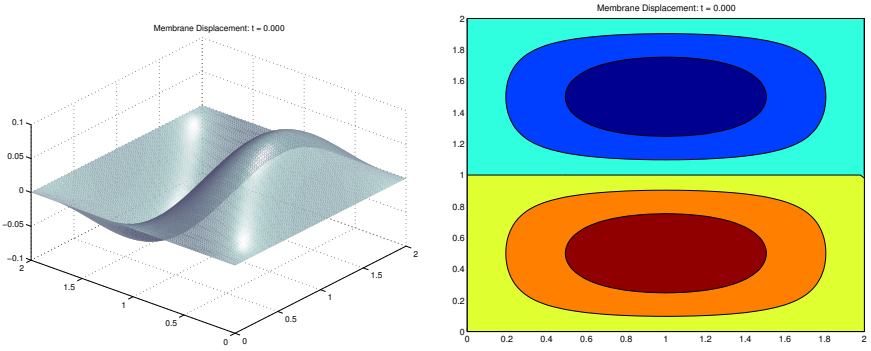
The approximate solution of this problem can be effectively displayed using a movie (animation) to show the evolution of the membrane over time. For reasonable visualization we require at least a $128 \times 128 \times 128$ resolution.

Snapshot at $t = 0$ Wave eqn:



(a) Surface Plot (b) Contour Plot

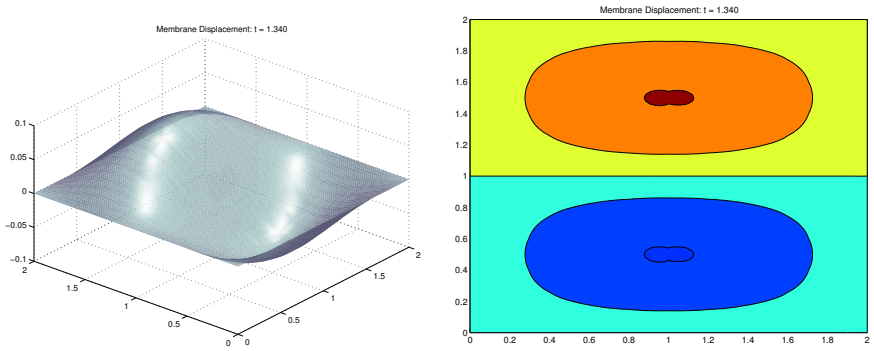
Figure 3 Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 0$.



(a) Surface Plot (b) Contour Plot

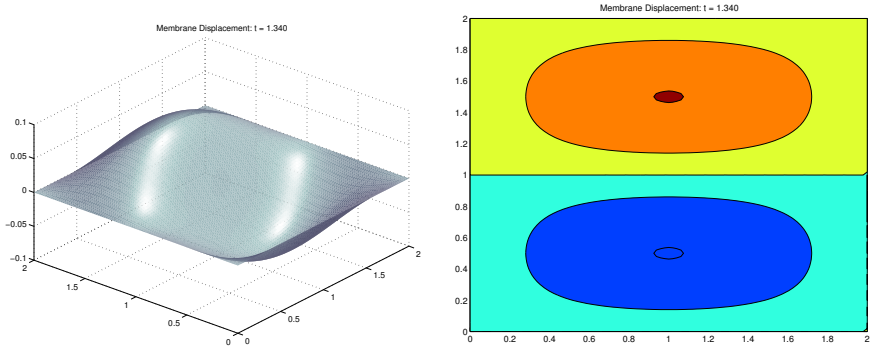
Figure 4 Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 0$.

Snapshot at $t = 1.34$ Wave eqn:



(a) Surface Plot (b) Contour Plot

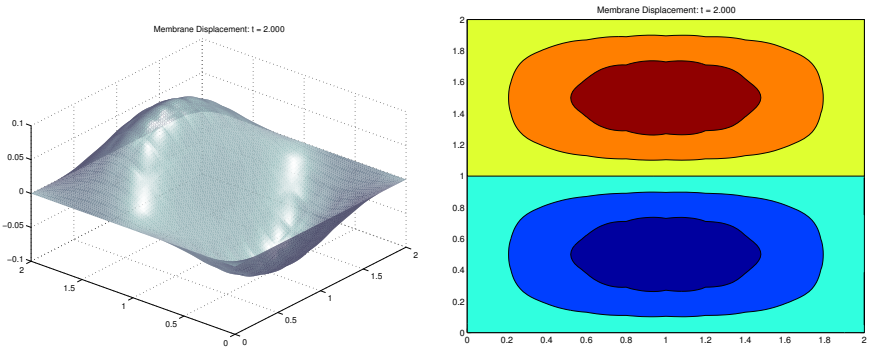
Figure 5 Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 1.34$.



(a) Surface Plot (b) Contour Plot

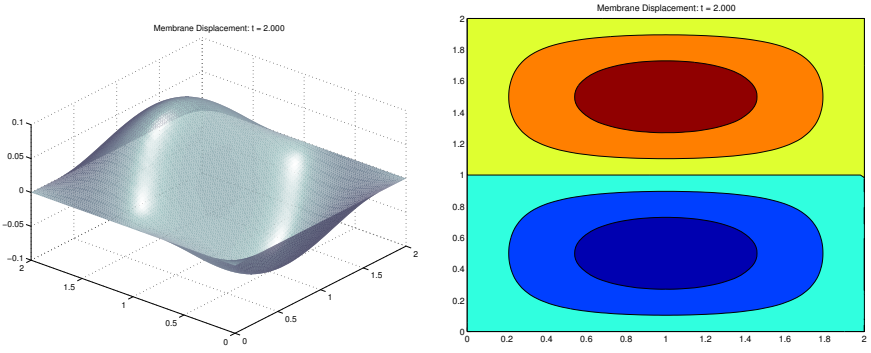
Figure 6 Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 1.34$.

Snapshot at $t = 2.0$ Wave eqn:



(a) Surface Plot (b) Contour Plot

Figure 7 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 8 Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 2.0$.

For 3D problems the DEINT associated with a quasi-linear PDE is defined by solving a sequence of linear system of equations. For each coarse mesh element, e, We determine a trivariate polynomial, $\bar{p}_{d,e}(t,x,y)$, of degree d that

- interpolates the mesh data (K constraints)
- almost satisfies the differential equation at m collocation points where $m = (d+1)^3 - K$.

Note that the 'cost' of determining the coefficients of $\bar{p}_{d,e}(t,x,y)$ is a few evaluations of the PDE and the solution of one linear system of dimension $(d+1)^3$.

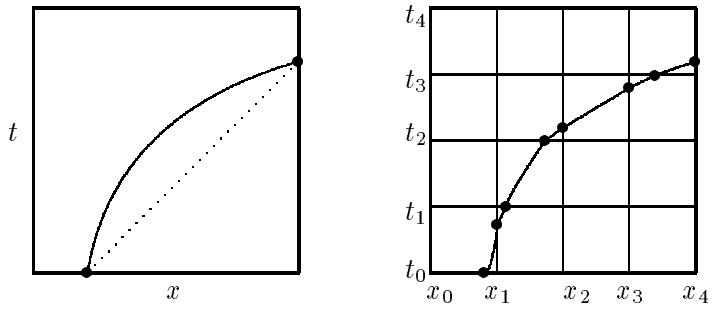
Note:

1. If the resolution of the display device requires a 'fine mesh' of NFM points to display a non-distracting visualization of a 1D curve then we should expect that NFM^2 points would be required to diplay a surface.

2. Contour curves should require less work to generate than surface plots. (O(NFM) vs O(NFM^2).)

3. This could be particularly significant if we are using animation to visualize the evolution over time of a PDE solution.

The basic MATLAB contouring algorithm:



(a) Contour line associated with the DEI and a straight line approximation on a coarse mesh element

(b) MATLAB approximation (refined in x and t directions with $NFM = 4$)

Figure 7 MATLAB uses linear interpolation and requires a regularly spaced mesh

– 'Refine' in x,t by evaluating $p_d(t_i, x_j)$ at NFM^2 points.

– 'Contour' the PL interpolant associated with this fine mesh.

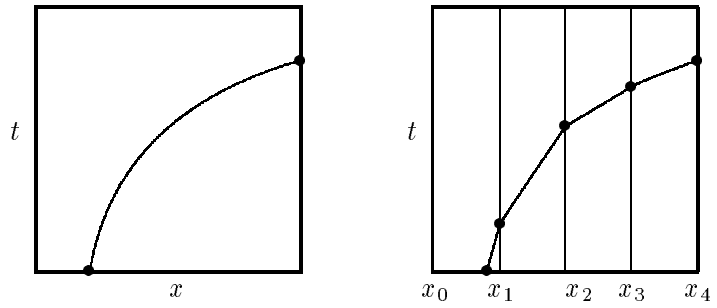
– The cost is $O(NFM^2)$ for each element. This is usually considerably cheaper than 'solving' the PDE on the fine mesh. (But more expensive than necessary when one is approximating a 1D-object.)

Figure 8 Generic Fast Contouring Algorithm

```
for each element e
  lmin(e) ← minimum value of element
  lMax(e) ← maximum value of element
end for
gmin ← min(lmin)
gMax ← max(lMax)
if contour vector not provided then
  generate evenly spaced contour levels between gmin and gMax
end if
for each element e
  for each contour level v
    if lmin(e) ≤ v ≤ lMax(e) then
      for each edge s
        if v intersects s then
          intersections(e) ← (x, t)_intersect
        end if
      end for
      count ← size(intersections(e))
      compute contour using FCINT, FCODE or FCODEA
    end if
  end for
end for
```

Note that the most common case will be $count = 2$ and we will focus on this in our analysis and algorithm discussion. Everything generalizes in an straightforward way to other values of $count$.

The 'Intercept Method', FCINT:



(a) True contour line associated with the DEI

(b) FCINT approximation (refined in x only)

Figure 9 FCINT refines in only one direction.

– Based on the observation that for fixed x , $p_d(t, x)$ reduces to a univariate polynomial of degree d in t, $q(t)$ and along such a line, we can solve for the roots of

$$q(t) - v = 0.$$

The coefficients defining $q(t)$ will depend on x.

– The cost is O(NFM) for each element but the constant can be large since it involves forming and determining the roots of a polynomial. These tasks are not easily vectorizable.

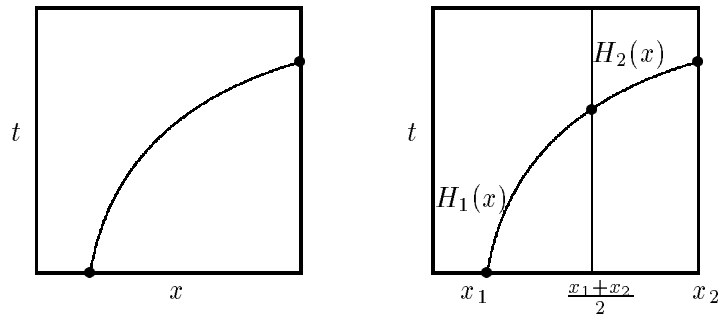
– We can choose to 'Refine' in t rather than x. That is, we can parametrize the contour curve by $(x(t), t)$ rather than by $(x, t(x))$.

The 'Simple ODE' method, FCODE:

We attempt to directly approximate the curve $(x, t(x))$ such that $p_d(t(x), x) = v$ over a prescribed range of x. Differentiating this relation wrt x and rearranging terms we observe that $t(x)$ satisfies the ODE,

$$t_x(x) = -\frac{p_x(t(x), x)}{p_t(t(x), x)} \equiv F(t(x), x).$$

For any fixed x value, $t(x)$ can easily be computed (if it exists) by solving a univariate polynomial equation (as in FCINT). We can use this observation at $x = x_1$ to determine an 'initial value' for the ODE. We then approximate the contour by approximating the solution of the ODE IVP problem for $x \in [x_1, x_2]$.



(a) True contour line associated with the DEI

(b) FCODE approximation for two steps of an ODE solver

Figure 10 FCODE computes t and t_x at the midpoint and end points, and uses Hermite Interpolants $H_1(x)$ and $H_2(x)$ to approximate $t(x)$.

- When $d \leq 3$ an appropriate 1D ODE DEINT is the piecewise Hermite that interpolates $t(x)$ and $t_x(x)$ on the discrete mesh used to solve the ODE on the interval $[x_1, x_2]$.

- If the ODE is close to singular (ie. $p_t(t(x), x)$ near 0), we can use the parametrization $(x(t), t)$ and solve an ODE IVP for x(t).

- The cost of FCODE is $O(NFM)$ per element with a small constant (vectorizable).

– If neither parametrization is well behaved for the whole interval of interest one can use pseudo-arclength, (FCODEA). This will result in a more expensive but more robust algorithm to approximate the contour curve, $(x(s), t(s))$ by solving an associated system of IVP ODEs.

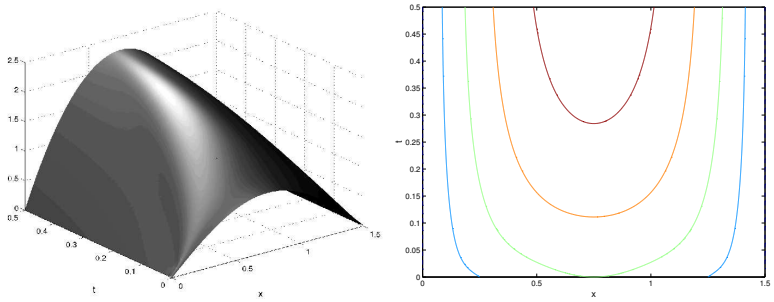
  The cost will be roughly double the cost of FCODE. In all our testing we did not encounter a situation where it was necessary to use this alternative parametrization.

Numerical Results:

Consider the modeling of a conducting rod in a nuclear reactor

$$\frac{\partial^2 u}{\partial x} + \frac{\kappa r}{\rho C} = \kappa \frac{\partial u}{\partial t}, \tag{1}$$

on the domain $0 \leq x \leq 1, 0 \leq t \leq .5$ with boundary conditions $u(0,t) = u(l,t) = 0$, for $0 \leq x \leq l$, and $u(x,0) = \sin \frac{\pi x}{l}$, for $t > 0$. We have used $l = 1.5$, $\kappa = 1.04$, $\rho = 10.6$ $C = 0.056$ and $r(x,t,u) = 5.0$.

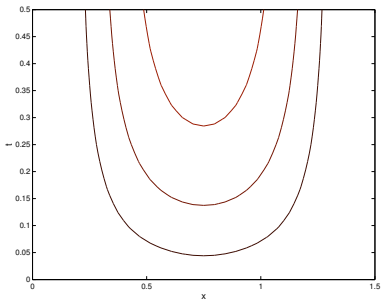


(a) Surface representation with lighting effects.

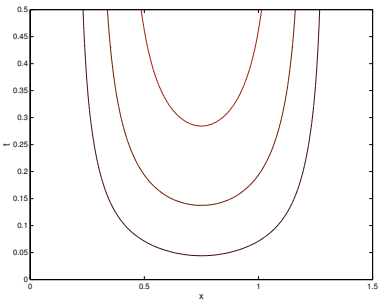
(b) Default MATLAB contour plot.

Figure 14 Solution to the Nuclear test problem

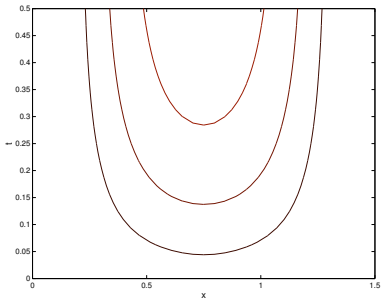
Contour curves for the different Algorithms:



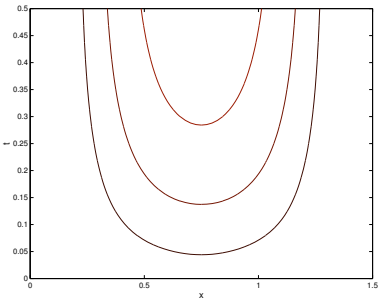
(a) MATLAB/DEI ($fm \cdot cm = 2^5$)



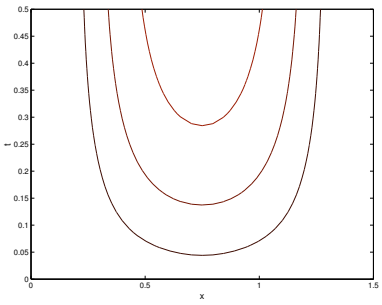
(b) MATLAB/DEI ($fm \cdot cm = 2^6$)



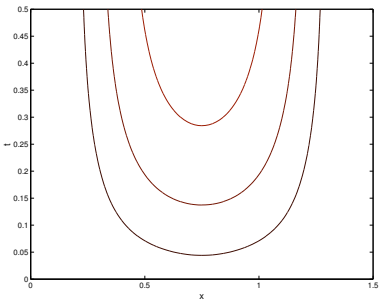
(c) FCINT ($fm \cdot cm = 2^5$)



(d) FCINT ($fm \cdot cm = 2^6$)



(e) FCODE ($fm \cdot cm = 2^5$)



(f) FCODE ($fm \cdot cm = 2^6$)

Figure 16 Contour for the Nuclear test problem based on a 2^4x2^4 coarse mesh .

Results and Conclusions:

- Direct computation of contour lines can be efficient and accurate. It is not necessary to approximate the solution surface.

- The techniques we have developed apply to unstructured rectangular and triangular meshes in 2 and 3 dimensions.

- The fast algorithms we have developed for computing contour curves can be extended to apply to more general contouring problems (such as those discussed by Grandine in a recent SIAM Review paper).

Approach can be modified and extended in several ways:

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

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

- Total degree d interpolation (rather than tensor product interpolation). This gives fewer unknowns, same order but larger error coefficients.

- 4D problems (t and 3 space dimensions).

- Extension to special classes of nonlinear PDEs such as KdV and other Boussinesq-like PDEs.