Software for Differential Equations: Accurate Approximate Solutions are not Enough

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

enright@cs.utoronto.ca

The Changing Face of Mathematical Software
Washington DC, June 3-4, 2004

Collaboraters: E. Bradbury, G. Ramos,
S. Hamdi and H. Goldani

If our software is going to be widely used, we must meet the expectations of the current user community. Such a user is typically interested in visualizing and investigating interesting properties of the approximate solutions to mathematical models that arise as systems of differential equations (either ODEs or PDEs). They are likely to be working in a PSE with a choice of methods available. The implications for developers of numerical software include:

- Adopt a standard interpretation of error control, and a standard program interface (calling sequence) for methods that perform the same or similar tasks.

- Options and additional parameters (if required) are specified in the same way for all methods. This includes stepsize or mesh constraints and accuracy specification.

- Adopt a standard representation of the approximate solution. (eg., a vector of piecewise polynomials.)

In the first part of this presentation, we will focus on ODEs where a new generation of software has been developed and is widely available. We will then describe specific developments related to PDE software and its use. In particular we will survey recent developments in the implementation of fast and reliable techniques for generating surface plots and contour curves associated with a coarse-mesh approximate solution on an unstructured adaptively-chosen mesh.

Part I – ODEs:

The Evolution of ODE software.

- Standard Discrete ODE Methods
  (with maximum stepsize, h):

$$\{x_i, y_i\}_{i=0}^{N}, \quad \max_{i=1}^{N} |y(x_i) - y_i| = O(h^p).$$

- Add continuous extension, $S(x)$,
  (for visualization):

$$S(x_i) = y_i, \quad ||S(x) - y(x)|| = O(h^p).$$

- Add direct defect error control
  (to obtain generic convergence result):

$$\begin{aligned} ||S'(x) - f(x, S(x))|| &\leq TOL, \\ ||S(x) - y(x)|| &\leq K_1 TOL \\ ||S'(x) - y'(x)|| &\leq K_2 TOL. \end{aligned}$$

Consider a typical application of the use of an IV method in a PSE.

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

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

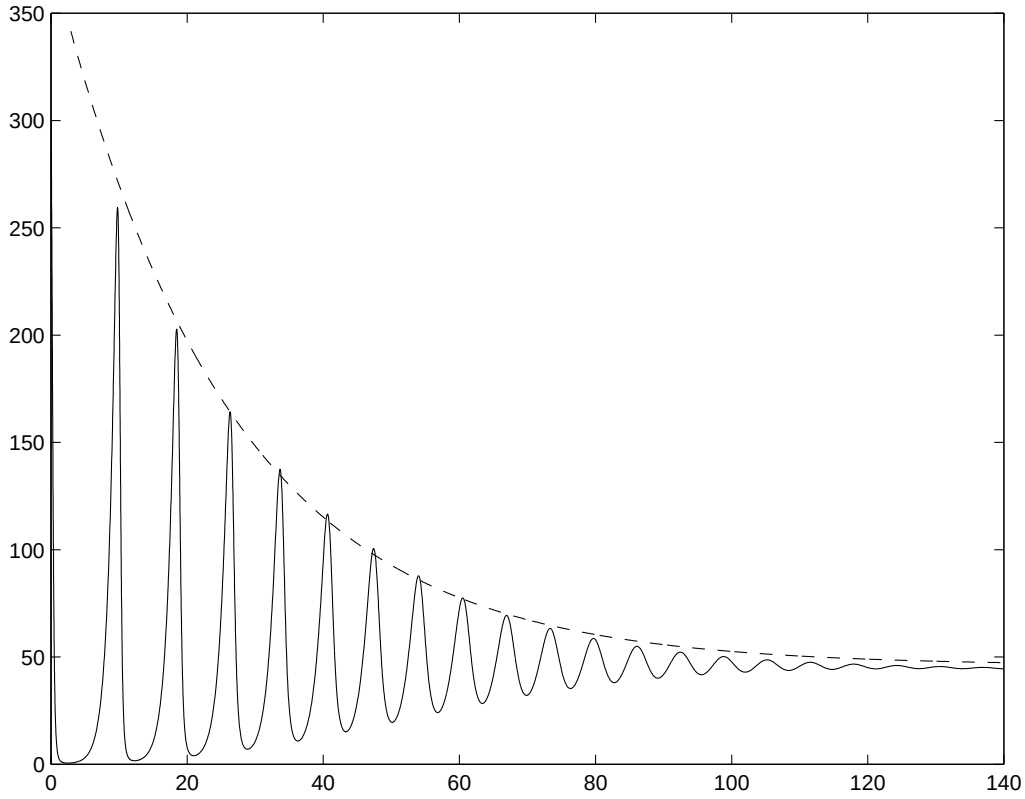
$$y_2' = -y_2 + 0.02 y_1 y_2 + 0.008x$$

with

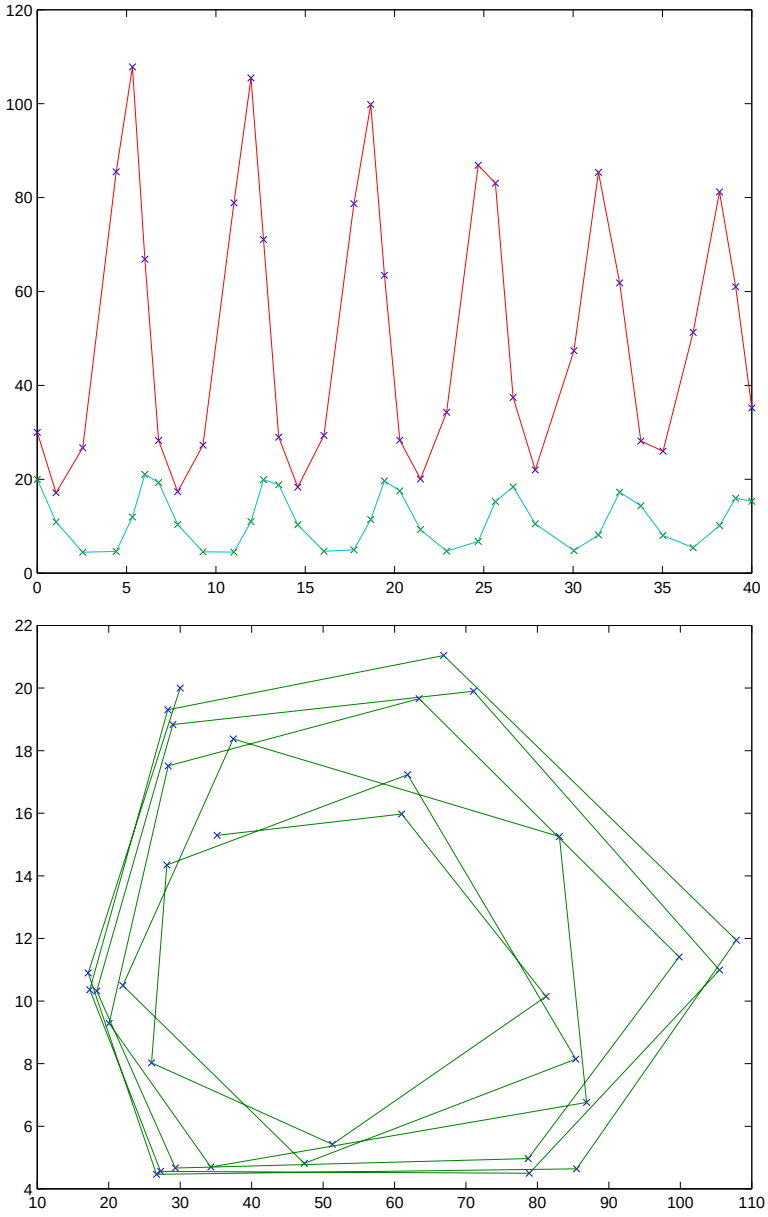
$$y_1(0) = 30, \;\; y_2(0) = 20,$$

where $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x.

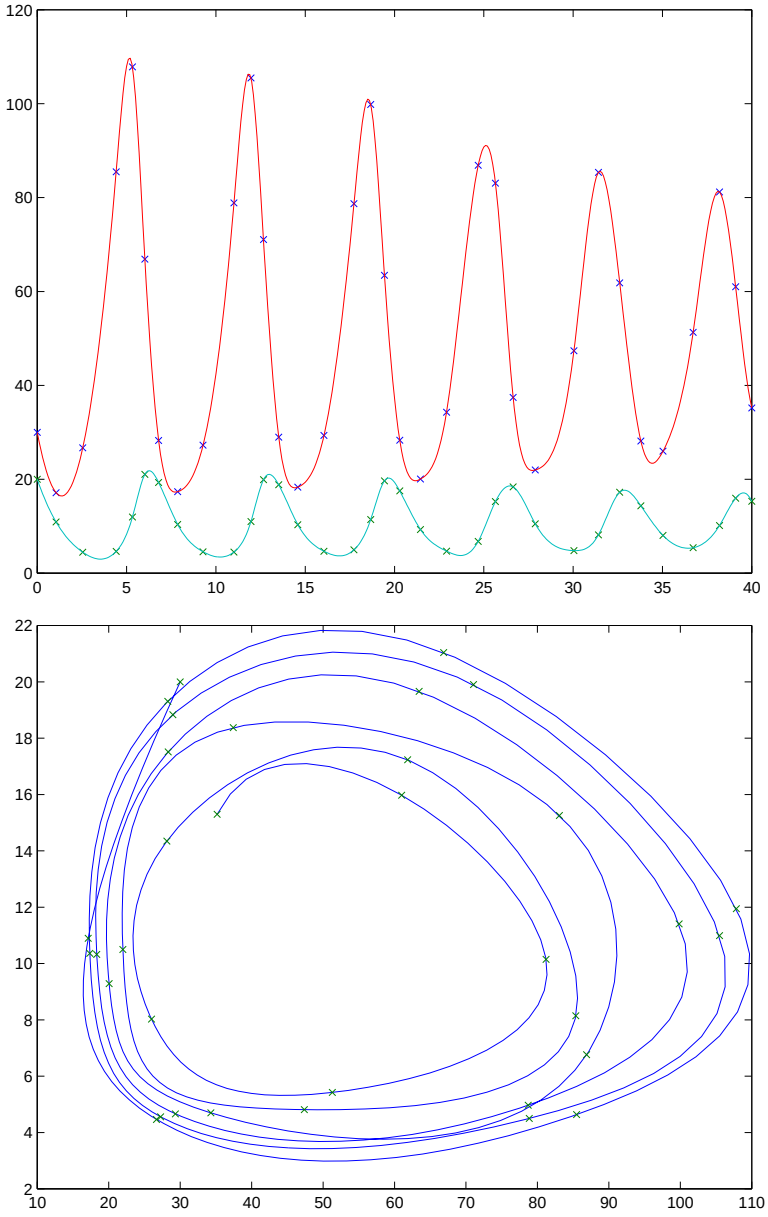
We know that solutions to this problem exhibit oscillatory behaviour as x increases. A biologist may be interested in whether the solutions to this equation are 'almost periodic' (in the sense that the difference between successive maximum is constant) and whether the local maximums approach a steady state exponentially.



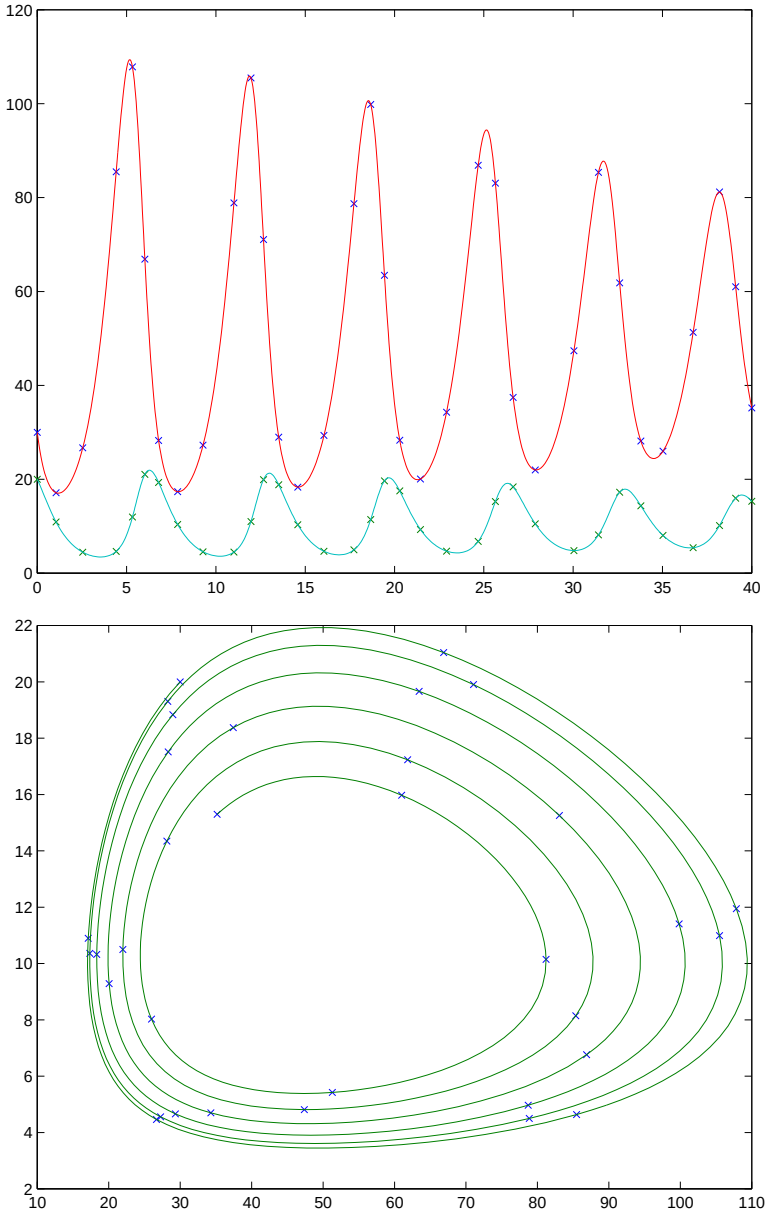
A) Visualizing Using a Discrete Solution:



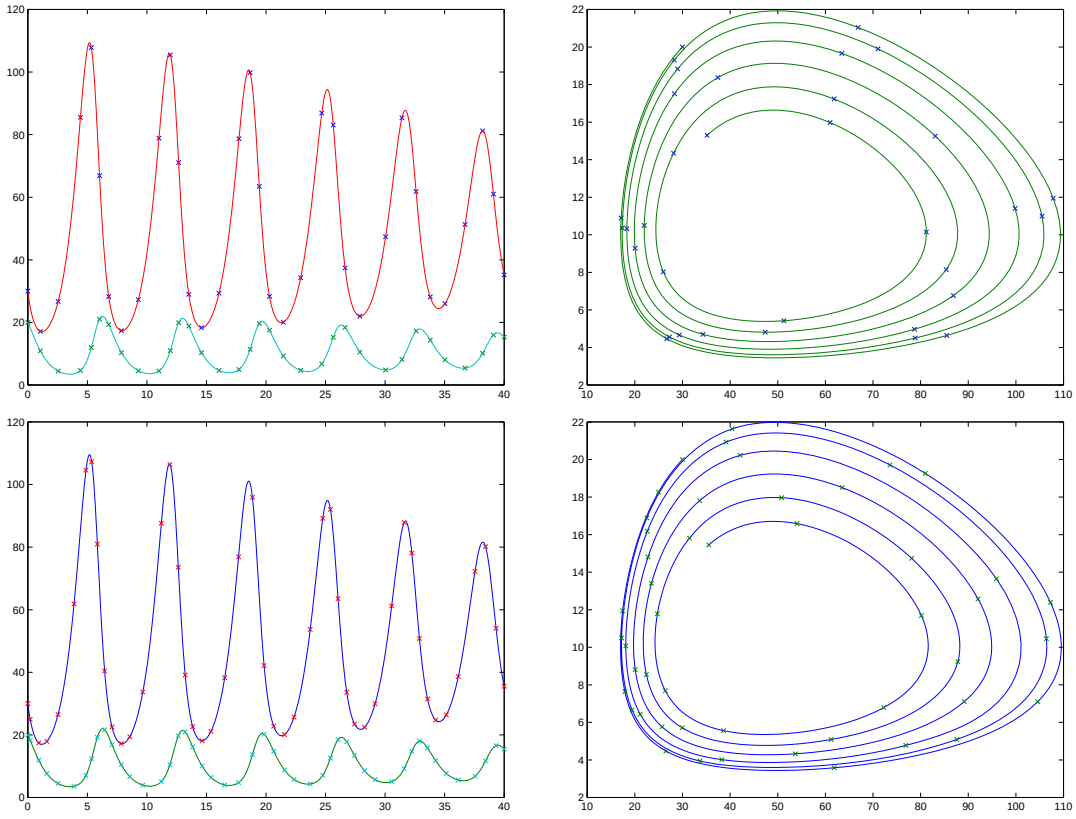
B) Visualizing Using Splines:



C) Visualizing Using a CRK formula, S(x):



Consider approximate solutions to this problem with an 8th order CRK method and the built-in MATLAB method, ode45. Visualizing using the piecewise polynomial representation of the corresponding approximate solutions we have:



An Essential Requirement:

- When applied to the IVP:

$$y' = f(x, y),\ y(a) = y_0,\ \text{on}\ \ [a, b],$$

with a specified accuracy, TOL, the method generates a piecewise polynomial, $S(x)$, defined for $x \in\ [a, b]$ satisfying,

$$\|S(x) - y(x)\| \leq K_M TOL.$$

Note that K_M can depend on the method and the problem and is the 'numerical condition number' associated with method M.

- With the right choice of error and stepsize control (For example, direct defect control) we can ensure that K_M will be almost independent of the method. The associated defect is defined to be
$\delta(x) \equiv S'(x) - f(x, S(x))$.

Let $z_i(x)$ be the solution of the local IVP on step i. We identify 3 types of continuous extensions of a p^{th}- order discrete RK formula, for $x \in [x_{i-1}, x_i]$:

$S(x) = z_i(x) + O(h^p)$: with,

$$\delta(x) = d(f)h^{p-1} + O(h^p),$$

where $d(f)$ depends on the problem and the method.

$\bar{S}(x) = z_i(x) + O(h^{p+1})$: with,

$$\bar{\delta}(x) = \bar{d}(f)h^p + O(h^{p+1}),$$

where $\bar{d}(f)$ depends on the problem and the method.

$\tilde{S}(x) = z_i(x) + O(h^{p+1})$: with,

$$\tilde{\delta}(x) = \tilde{d}h^p + O(h^{p+1}),$$

where $\tilde{d}$ depends only on the problem.

Error and Stepsize Control

With these continuous extensions one can monitor the magnitude of the defect associated with each step and accept the step only if an estimate of this quantity is less than the error tolerance, TOL.

'Direct' defect control refers to error control strategies that attempt to correctly estimate the leading term in the expansion of the defect. In particular methods that use direct defect control cannot employ local extrapolation, either to estimate the magnitude of the defect or improve the accuracy of the accepted solution.

Note that with direct defect control, one can derive estimates of $\delta(x)$, $\overline{\delta}(x)$ that are reliable on each step 'with high probability' and estimates of $\tilde{\delta}(x)$ that are reliable (asymptotically justified) for all steps.

With direct defect control one can prove the desired convergence result:

$$\|S(x) - y(x)\| \leq K_M TOL,$$

where K_M depends primarily on the problem.

We have developed a family of IVP, BVP and DDE methods based on this approach. These methods provide, as an option to the user, the choice of either a continuous extension of the type $\bar{S}(x)$ or the more reliable but more expensive $\tilde{S}(x)$. DDVERK is the DDE method of this family and is available through NETLIB.

The following table identifies the number of stages required for the different types of continuous extensions. In this table, s is the number of stages necessary to determine $S(x)$; $\bar{s}$ is the number of stages necessary to determine $\bar{S}(x)$; and $\tilde{s}$ is the number of stages necessary to determine $\tilde{S}(x)$.

Note that this table applies to all ODE methods as it identifies the 'cost' of forming the underlying interpolant and the associated direct defect.

Formula	p	s	$\bar{s}$	$\tilde{s}$
CRK4	4	4	6	7
CRK5	5	7	9	11
CVSS6B	6	9	11	14
CVSS7	7	11	15	20
CVSS8	8	15	21	28
ode45	5	7	9	11

Table 1: Cost per step of some CRK Methods

Three Versions of ode45:

We have modified the MATLAB method ode45 so the user can select one of three error control strategies:

eropt = I: Indirect local control– using $S(x)$. This gives the identical results that the built-in routine provides.

eropt = II: Direct defect control – using $\bar{S}(x)$. The error estimate is based on a sampled evaluation of the defect and has a high probability of being reliable.

eropt = III: Strict direct defect control – using $\tilde{S}(x)$. The leading term in the expansion of this defect is reliably estimated.

There is clearly a cost/reliability trade off to be considered when selecting the error control option for a particular application.

We report the following statistics for the Predator - Prey investigation for the 3 versions of ode45:

steps The number of time steps.

fcn: The number of derivative evaluations.

ger: The maximum magnitude of the error in the solution, measured in units of TOL.

ymerr: The maximum magnitude of the error in the identified local maximums (of the prey population), measured in units of TOL.

experr: The error in the reported value of the 'best' exponential fit to the decay exhibited by the mathematical model, measured in units of TOL.

R(res): Best least square fit and residual for determining whether the prey population is almost periodic.

ero	TOL	10^{-2}	10^{-4}	10^{-6}
I	steps	71	148	367
	fcn	511	961	2239
	ger	30.	8.3	3.9
	ymerr	12.	1.1	2.2
	experr	24.8	2.9	5.6
	R(res)	6.43 (.4)	6.37 (.05)	6.37 (.05)
II	steps	92	184	397
	fcn	921	1769	3385
	ger	4.1	2.3	4.3
	ymerr	.70	1.1	3.5
	experr	2.2	1.7	5.4
	R(res)	6.36 (.07)	6.37 (.05)	6.37 (.05)
III	steps	92	185	408
	fcn	1171	2131	4441
	ger	1.5	1.7	2.6
	ymerr	.78	.82	2.2
	experr	2.7	1.7	3.7
	R(res)	6.36 (.06)	6.37 (.05)	6.37 (.05)

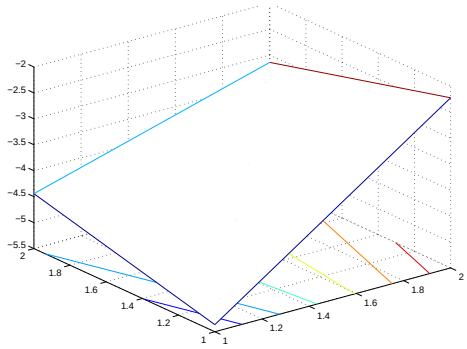
Table 2: Results for the 3 versions of ode45 on the Predator-Prey investigation

Extension to PDEs:
We have extended this approach for visualization and the investigation of interesting properties of the numerical solution to PDEs. In this case, the piecewise interpolant of the discrete numerical solution, is a multivariate polynomial, $S(x, y)$. (Note that we will assume the PDE is 2D, although the results extend to higher dimensional problems in an obvious way.)

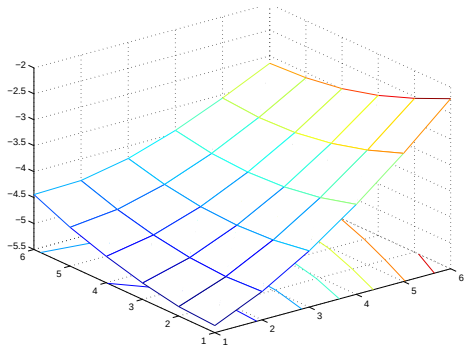
The approach applies to an unstructured discrete mesh which is assumed to have been adaptively chosen by the underlying numerical method. We also assume that the method has generated accurate approximations at the associated mesh points. We introduce interpolation schemes to 'extend' this discrete approximation to a continuous approximation.

For each mesh element, e, we introduce a 'local' bi-variate interpolant $u_{d,e}(x,y)$ of degree d that interpolates the 'local' mesh data and attempts to satisfy the underlying PDE.

The underlying method has a piecewise linear extension associated with e,



With $S_d(x,y)$ defined by the collection of $u_{d,e}(x,y)$, we obtain on a 'refined' fine mesh,



A Two Dimensional Example

The PDE is from the ELLPACK collection

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

on the domain

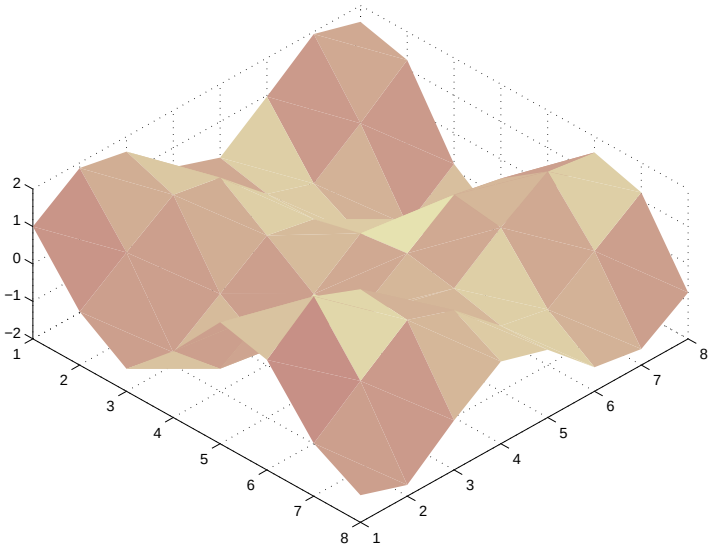
$$0 \leq x \leq 1,\ 0 \leq y \leq 1,$$

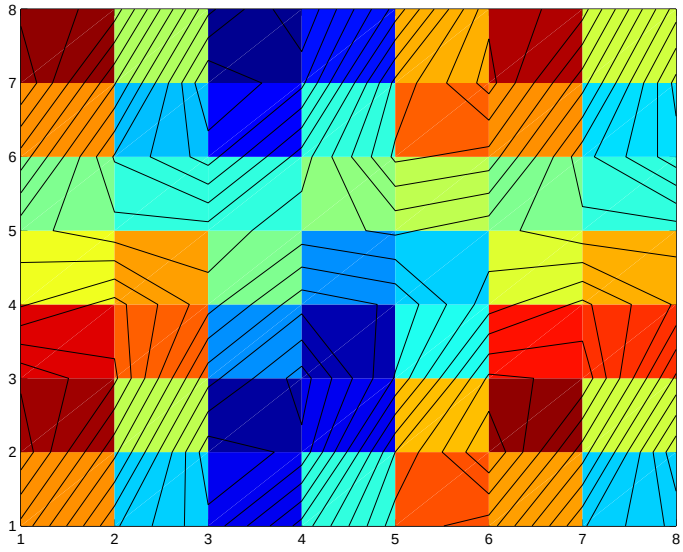
with boundary conditions

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

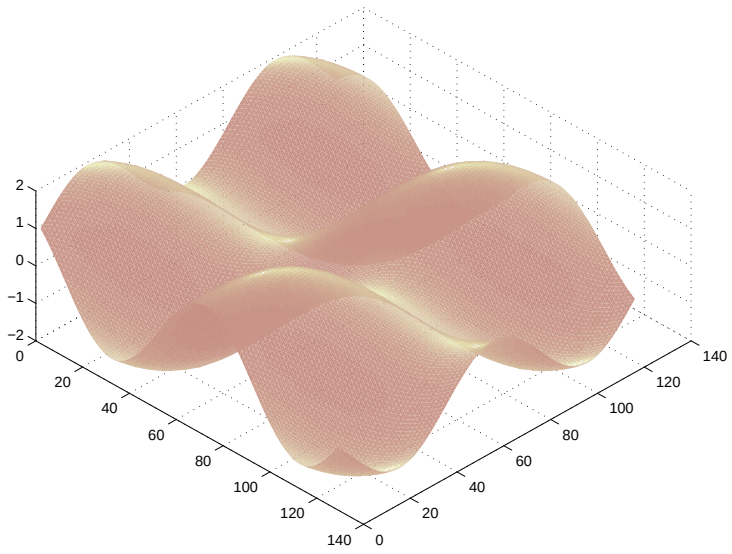
where $B = \pi$ or $B = 10$.

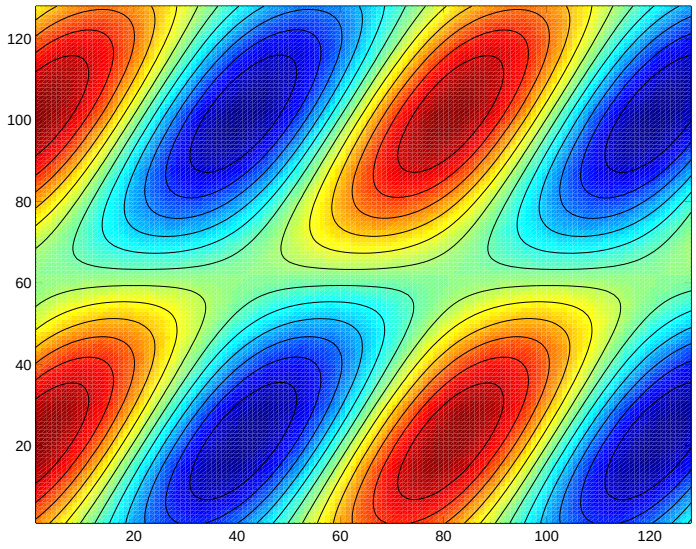
For this PDE defined by $B = 10$, visualizing with 8×8 coarse mesh (Discrete solution only):





Visualizing with 8×8 coarse mesh and 16×16 fine mesh using $S_3(x, y)$:





Approach has been modified and extended in obvious ways:

- 'Non-local' solution values can be used if derivative data not available at meshpoints.
- Has been applied to unstructured triangular meshes.
- Has been applied to 3D problems.
- Approach can be applied to first-order and higher-order systems.
- Approach can also be applied to mixed-order systems.

The globally accurate piecewise polynomial $S_d(x, y)$ can be effectively used in many situations:

- To generate contour plots or level surfaces.
- To locate discontinuities or identify 'fronts'.
- To perform error estimation and/or mesh refinement.
- To improve heuristics/strategies in the PDE solver.

Efficient Generation of Contour Curves.

In most PSEs, contouring a surface in 2D involves rendering the surface over a fine uniform mesh (associated with the resolution of the display device) and then using piecewise linear interpolation to contour this rendered surface.

With our approach $S_d(x, y)$ can be used to generate the values over the fine mesh and this can result in efficient algorithms for displaying the surface and the associated contours. In the case that only contours are desired we can use $S_d(x, y)$ to directly approximate the contours without the expense of approximating the surface.

Fast Algorithms for Contouring:

We have developed three algorithms of this type with different time-complexities and error behaviour. We will describe one of the most promising here that applies to the situation where $S_d(x, y)$ is defined over an unstructured mesh with triangular elements. The algorithm is designed to find the contour curve corresponding to a value of C. (That is, find the curve defined by $S_d(x, y) = C$.)

The ODEA algorithm is based on introducing 'arclength', s, to parametrize the problem, and defining a simple ODE whose solution is the vector of coordinates, $(x(s), y(s))$ of the contour curve.

Since $S_d(x, y)$is a bi-variate polynomial, its derivatives $S_x(x, y)$ and $S_y(x, y)$ are easily computed and by differentiating

$$S_d(x(s), y(s)) = C,$$

we see that the contour curve satisfies the ODE,

$$S_x \cdot x_s + S_y \cdot y_s = 0$$

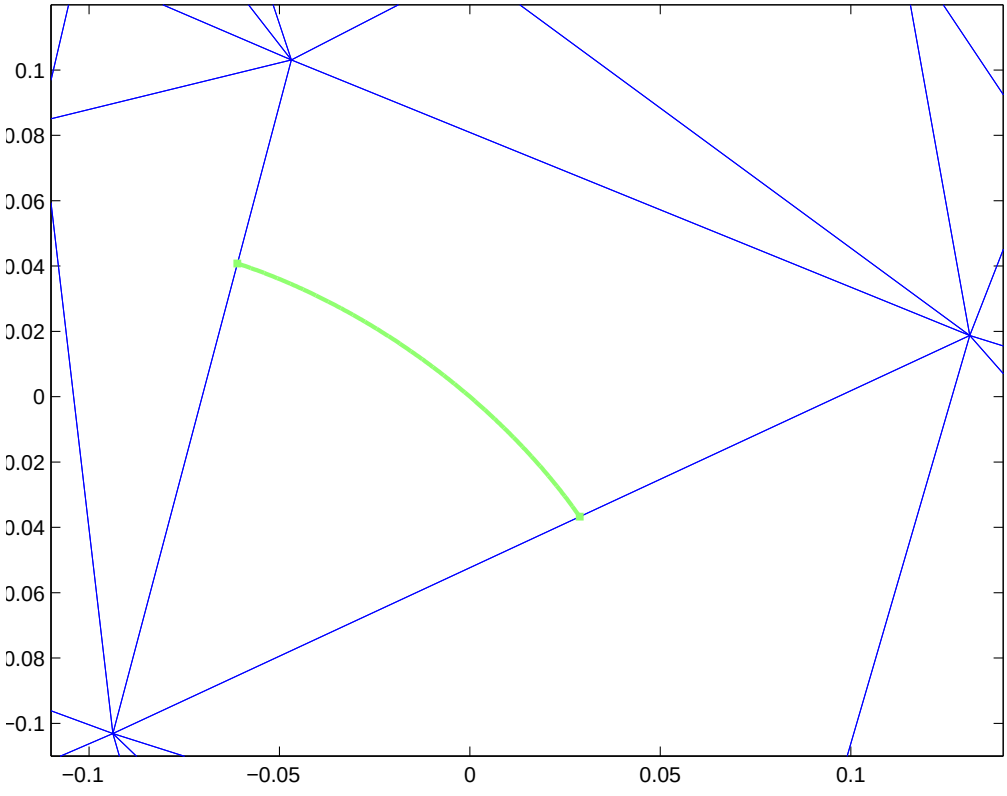
subject to the 'arclength' constraint,

$$x_s^2 + y_s^2 = 1$$

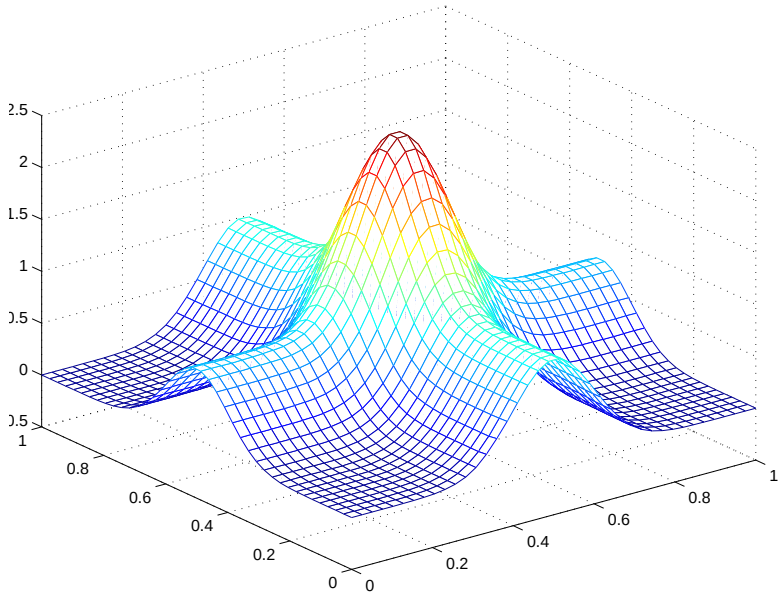
The resulting ODE can be approximated over each element using cubic Hermite interpolation in an efficient way. The resulting algorithm becomes most efficient if high resolution is required and the discrete mesh is coarse.

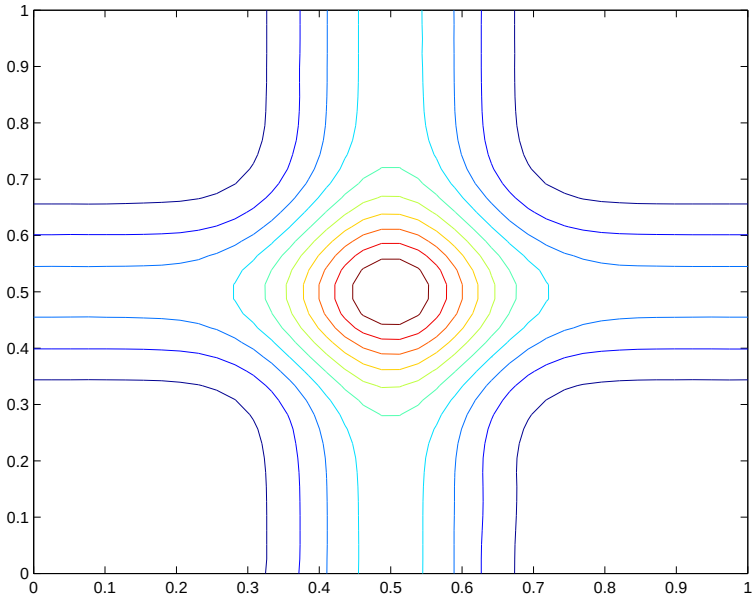
Motivation for ODEA:

For each element, whose boundary intercepts the contour, we approximate the solution of one or more ODEs. The most likely situation involving an intersection of a contour curve and a triangular element:

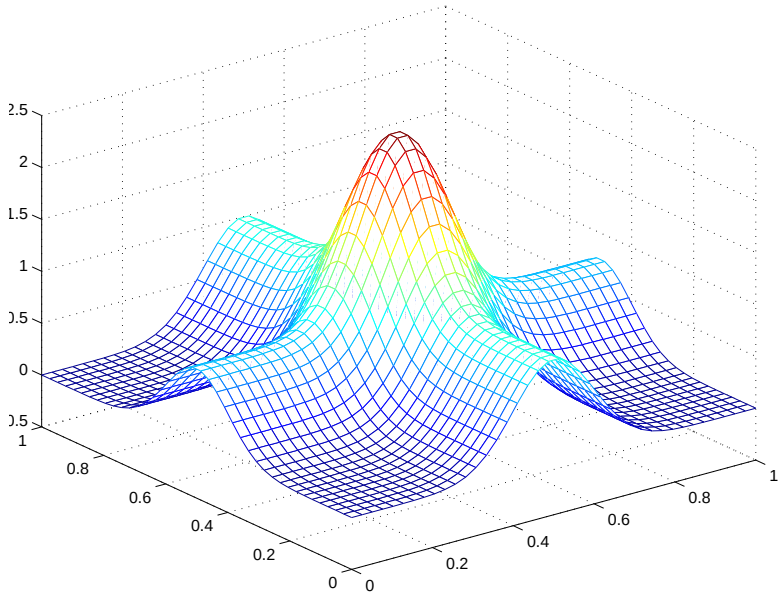


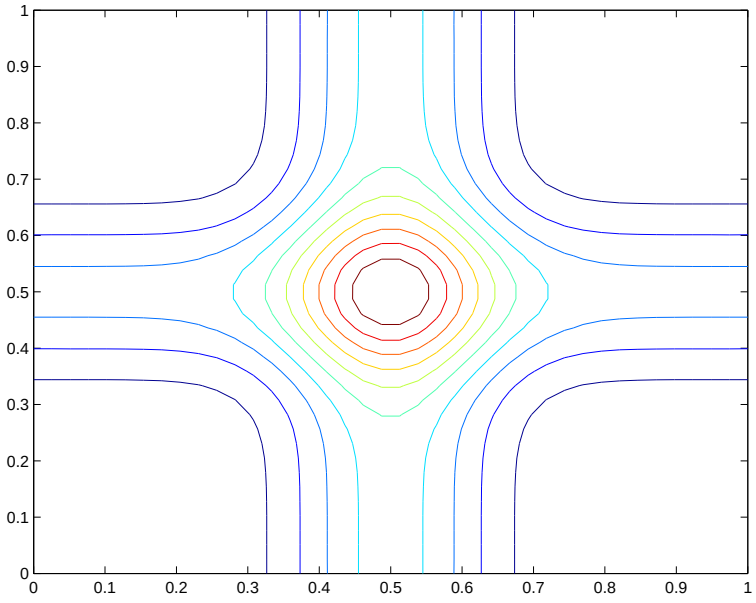
The Surface plots and Contour curves for a simple hyperbolic PDE defined over an unstructured triangular mesh with 500 mesh points using a piecewise bi-cubic, $S_3(x)$, (PCI)



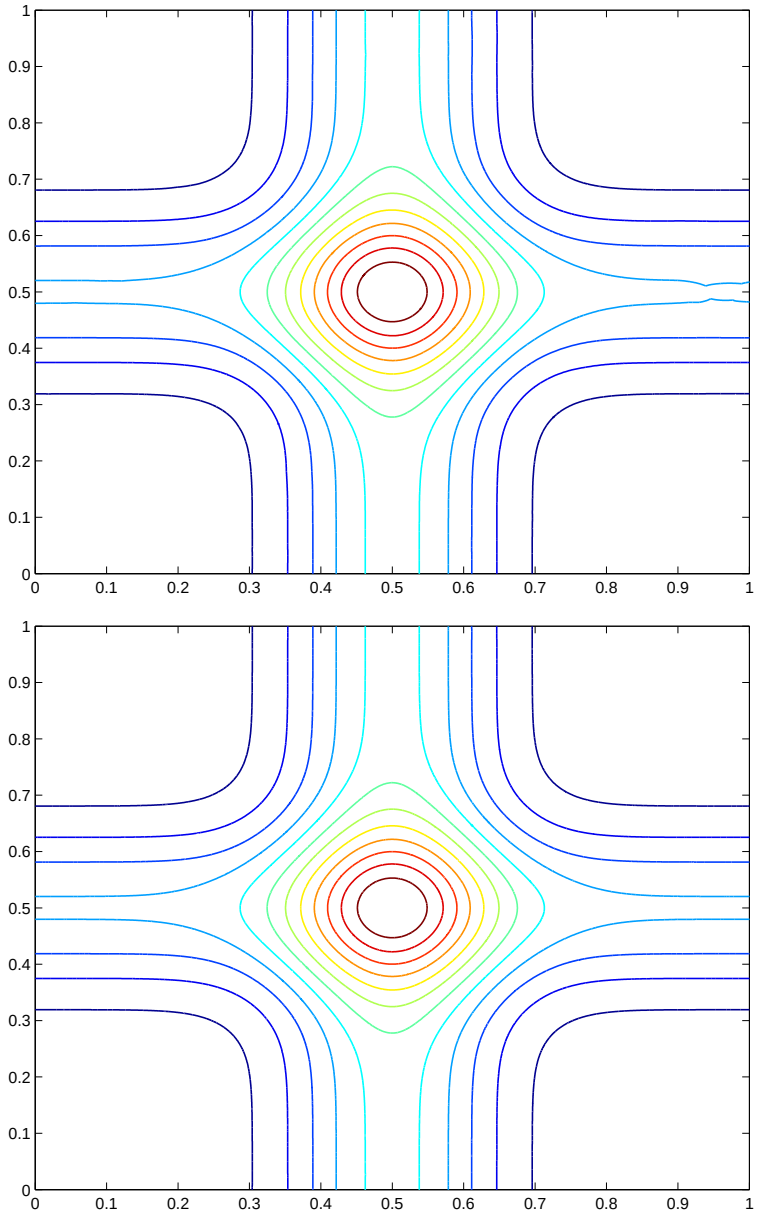


The Surface plots and Contour curves for a simple hyperbolic PDE defined over an unstructured triangular mesh with 2000 mesh points using PCI.





The contour curves for the simple hyperbolic PDE computed using ODEA and PCI with an unstructured triangular mesh with 900 and 2500 mesh points.



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:

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

b)2D Heat Equation: (Heated Plate)

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

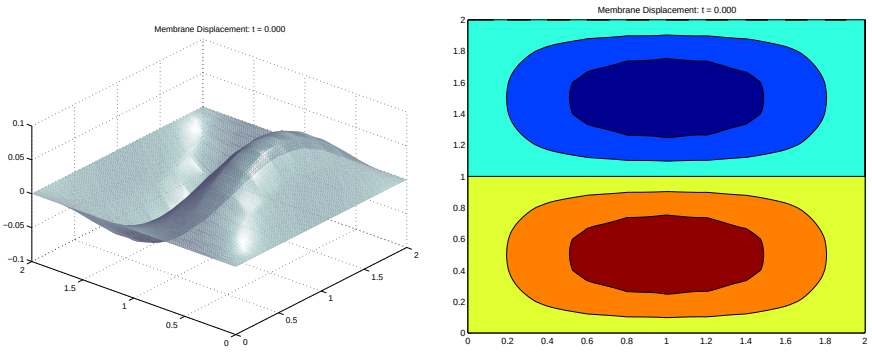
domain: $0 \leq t \leq 5000,\ 0 \leq x \leq 4,\ 0 \leq y \leq 4,$
bound cond :

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

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

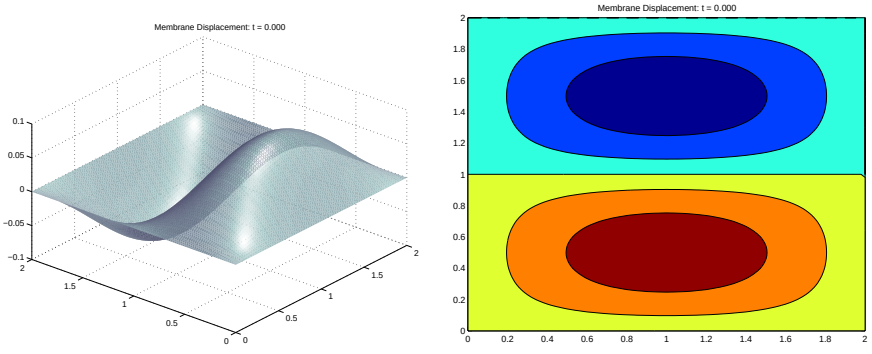
In both cases animation requires a $128 \times 128 \times 128$ fine mesh.

Snapshot at $t = 0$ Wave eqn:



(a) Surface Plot (b) Contour Plot

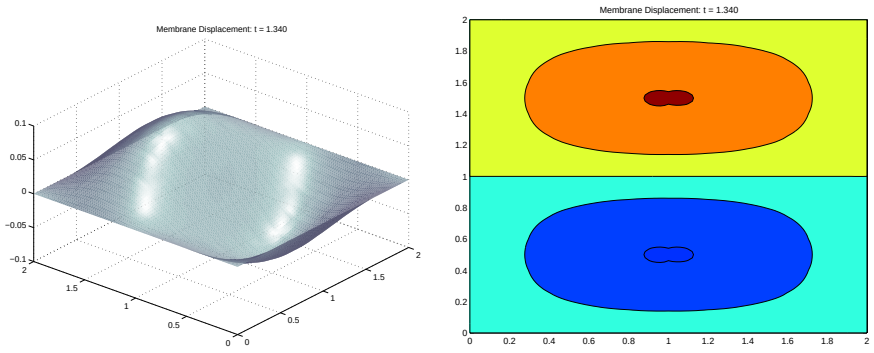
Figure 3 Visualization Using Piecewise Polynomial on $10 \times 10 \times 10$ mesh with $10 \times 10 \times 10$ refinement at $t = 0$.



(a) Surface Plot (b) Contour Plot

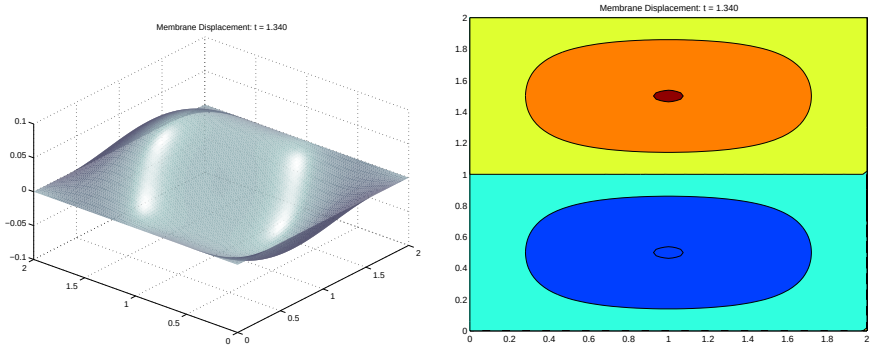
Figure 4 Visualization Using Piecewise Polynomial on $20 \times 20 \times 20$ mesh with $5 \times 5 \times 5$ refinement at $t = 0$.

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 $t = 1.34$.