CompAMa
www.compama.co.usb.ve
Bull. Comput. Appl. Math., Vol.1, 2013
ISSN 2244-8659

Solving the KPI Wave Equation with a Moving Adaptive FEM Grid

Granville Sewell *

Abstract

The Kadomtsev-Petviashvili I (KPI) equation is the difficult nonlinear wave equation $U_{xt} + 6U_x^2 + 6UU_{xx} + U_{xxxx} = 3U_{yy}$. We solve this equation using PDE2D (www.pde2d.com) with initial conditions consisting of two lump solitons, which collide and reseparate. Since the solution has steep, moving, peaks, an adaptive finite element grid is used with a grading which moves with the peaks.

1 Introduction

The Kadomtsev-Petviashvili I (KPI) wave equation:

$$U_{xt} + 6U_x^2 + 6UU_{xx} + U_{xxxx} = 3U_{yy}$$

is used to model waves in thin films with high surface tension. It has been extensively studied in the mathematical community since the 1970 paper by Boris Kadomtsev and Vladimir Petviashvili [1]. [2] and [3] report that only two kinds of numerical methods have been used to solve the KPI equation: finite difference methods (which these two papers apply) and a pseudo-spectral method developed by [4].

All previous successful attempts to solve this difficult wave equation required development of numerical methods especially tailored for the equation, here we attempt to solve it using a robust, general-purpose finite element program developed by the author.

*Mathematics Dept., University of Texas El Paso, USA, e-mail: sewel@utep.edu

2 The Finite Element Method Used

PDE2D ([5],[6],[7]) is a general-purpose partial differential equation solver which solves very general systems of nonlinear, steady-state, time-dependent and eigenvalue PDEs in 1D intervals, general 2D regions (with curved boundaries), and a wide range of simple 3D regions, with general boundary conditions. It uses a collocation finite element method, with cubic elements, for 3D problems, and either a collocation or Galerkin finite element method can be used for 1D and 2D problems. If the Galerkin algorithm is used for 2D problems, as in this paper, triangular elements of up to 4th degree can be used, on a triangulation which is automatically refined and graded, either adaptively or according to user specifications.

To use PDE2D, we have to reduce this fourth order equation to a system of three first or second order equations, by introducing the variables $V \equiv U_x, W \equiv U_{xx}$:

$$\begin{aligned} 0 &= U_x - V \\ 0 &= U_{xx} - W \\ V_t &= -W_{xx} + 3U_{yy} - 6V^2 - 6UW \end{aligned}$$

[3] give a two-lump soliton analytical solution of the KPI equation, expressed as $Q(x,y,t) = 2[\Phi\Phi_{xx} - \Phi_x^2]/\Phi^2$, where $\Phi(x,y,t)$ is defined as the determinant of a certain 4 by 4 matrix. We will use this analytical solution for defining initial conditions, and for computing errors.

Initial conditions for the problems solved in this paper are

$$\begin{aligned} U(x,y,0) &= Q(x,y,0) \\ V(x,y,0) &= Q_x(x,y,0) \\ W(x,y,0) &= Q_{xx}(x,y,0) \end{aligned}$$

Two of the problems solved in [3] will be solved here:

1. An “oblique collision" problem, where two solitons of equal size collide at a 90^o angle and pass through each other.

2. A “direct collision" problem, where two solitons are initially located along the x-axis, moving to the right with different velocities. The larger soliton overtakes the smaller one, they combine and reseparate. [2] solve a very similar direct collision problem.

In both cases, as long as the two solitons are sufficiently separated initially, the initial conditions can be represented approximately by

$$Q(x,y,0) \approx 16\frac{N_1}{D_1^2} + 16\frac{N_2}{D_2^2} \tag{1}$$

where, for the oblique collision case:

$$N_j = -4(x - x_j - 2k_j(y-y_j))^2 + 16k_j^2(y-y_j)^2 + 1/k_j^2$$

$$D_j = 4(x - x_j - 2k_j(y-y_j))^2 + 16k_j^2(y-y_j)^2 + 1/k_j^2$$

with $(x_1,y_1) = (15,-15), (x_2,y_2) = (15,15), k_1 = \frac{1}{2}, k_2 = \frac{-1}{2}$

and for the direct collision case:

$$N_j = -4(x - x_j)^2 + 16k_j^2(y-y_j)^2 + 1/k_j^2$$

$$D_j = 4(x - x_j)^2 + 16k_j^2(y-y_j)^2 + 1/k_j^2$$

with $(x_1,y_1) = (15,0), (x_2,y_2) = (31,0), k_1 = \frac{\sqrt{6}}{4}, k_2 = \frac{\sqrt{6}}{8}$

In each case, $16\frac{N_j}{D_j^2}$ has a peak of $16k_j^2$ at (x_j,y_j), and since this term dies out at a distance, $u(x,y,0)$ will have peaks close to (x_1,y_1) and (x_2,y_2), as seen in Figures 1a and 5a. In the oblique case, the solitons have velocities $(6,6)$ and $(6,-6)$; in the direct case, $(4.5,0)$ and $(1.125,0)$.

The boundary conditions are chosen to reflect the fact that $Q(x,y,t)$ goes to zero far from the soliton peaks:

$$
\begin{aligned}
U(0,y,t) &= 0 \\
V(0,y,t) &= 0 \\
W(0,y,t) &= 0 \\
U_x(70,y,t) &= 0 \\
W_x(70,y,t) &= 0 \\
U_y(x,-30,t) &= 0 \\
U_y(x,30,t) &= 0
\end{aligned}
$$

As seen in Figures 1-9, the solutions have steep, moving peaks, so a major challenge is constructing an appropriately graded, moving grid. PDE2D does not actually allow the triangular grid to change with time. However, an adaptive, moving grid is improvised as follows: PDE2D is called multiple times, each time it solves the system from $t = t_{n-1}$ to $t = t_n$, taking several time steps, using the Crank-Nicolson method to discretize time, on a fixed grid, and the solution at t_n is dumped on a uniform (1000 by 1000) mesh. The next time PDE2D is called, it generates a new triangulation (of 4800 cubic triangular elements, with about 65,000 unknowns) adaptively, based on the final solution at t_n, with initial conditions linearly interpolated from the dumped solution at t_n. This process is done quite automatically, the call to PDE2D is simply placed inside a DO loop, with initial time t_{n-1} and final time t_n each call, and the dump/restart and adaptive triangulation options are turned on. Since an implicit method is used in time, a large nonsymmetric, sparse linear system must be solved each time step; this is done using a sparse direct solver based on the Harwell Library minimal degree routine MA37 [8].

3 Numerical Results

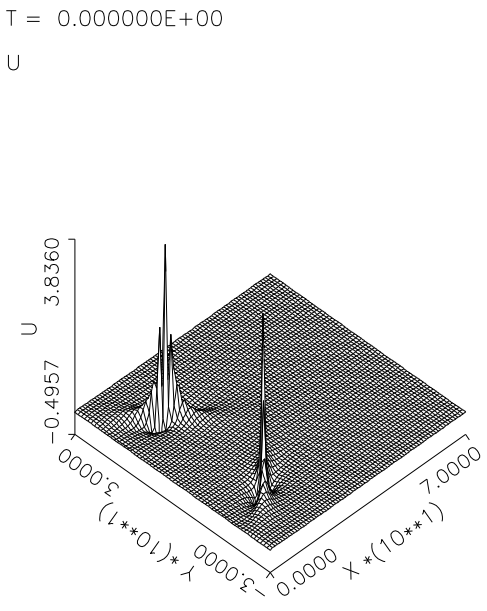
Results are shown in Figures 1-3, for the oblique collision case, where a time step of $\Delta t = 0.0125$ is used, and the grid is updated adaptively every 10 steps. The moving grid follows the peaks very nicely, and the final solution, at $t = 5$, agrees reasonably well with the analytic solution (Figures 3a,4) at that time. In all problems, it is known that both the integral of U (I1) and the integral of U^2 (I2) should be constant with time. This gives us an

easy way to estimate the numerical error. At $t = 5$, the error in I1 was 97%, and the error in I2 was 9.4%. The I1 integral is much more sensitive than the I2 integral to the smaller values near the boundary, far from the peaks, so the problem was resolved with U, V, W set to the true solution on the entire boundary, and the error in I1 decreased to 3.3%, while the error in I2 increased slightly, to 10.0%. Since U is not always positive, it may be more reasonable to divide by the integral of $|U|$, rather than the integral of U, in calculating the I1 relative error; when this is done, we get a more respectable-looking figure of 0.7% for the I1 error. Lu, Tian and Grimshaw report errors in I1 and I2 of order 0.1% and 1%, respectively, with a time step of $\Delta t = 0.0001$. Although our time step is 125 times larger, we have to solve a large linear system every time step, while they used an explicit time integration. They used a 600 by 600 uniform finite difference grid in space, which means their problem has 360,000 unknowns.

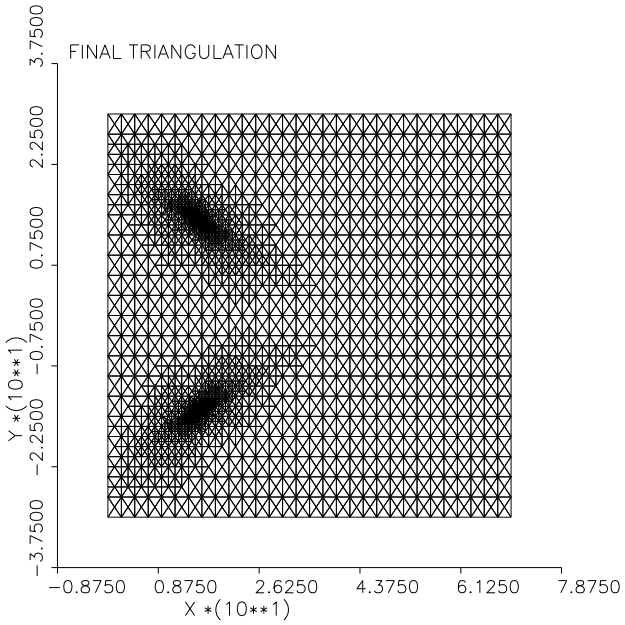
For the direct collision case, results are shown in Figures 5-8. Again the moving grid follows the peaks nicely, and the PDE2D solution agrees well with the analytic solution until about $t = 5.0$, when the taller, faster, peak catches the smaller one (Figures 6a,9a). After that, the peaks computed by PDE2D separate more slowly than they should: the PDE2D solution at $t = 10$ looks much like the true solution at $t = 8$ (Figures 8a,9b)! [3] report similar results on this problem, but they attribute the slow evolution in time of their finite difference solution to the fact that they are using the approximate initial condition 1 rather than $Q(x, y, 0)$ at $t = 0$. But we are using $Q(x, y, 0)$ as our initial condition, so our slower evolution cannot be explained similarly. In fact, when we used the approximate initial condition 1, our solution developed even more slowly.

For the direct collision problem, a time step of $\Delta t = 0.025$ was used, and again the grid was updated adaptively every 10 steps. The I2 integral differs from the true value at $t = 10$ by about 5.9%. Lu, Tian and Grimshaw report an error in I2 of only 0.45% at $t = 10$ for this problem, using a time step of $\Delta t = 0.0001$.

Finally, we re-solved both problems with the same number of elements and same time step sizes, but this time using a constant, uniform triangulation. The resulting solutions, shown in Figures 10a-b, are very bad, and clearly illustrate the importance of the moving, adaptive grid. The error in the integral of U^2 for the oblique collision problem at $t = 2.5$ is now 500%, and 4000% at $t = 5$ for the direct collision problem! Notice that the direct collision solution is not only quite noisy, but the peaks are very far from

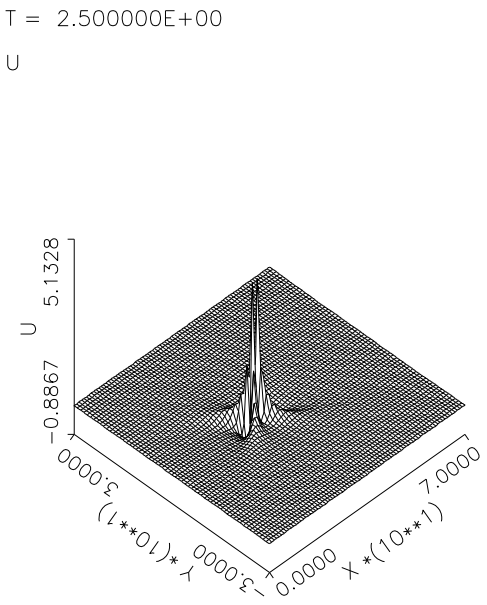


(a) U(x,y,0)

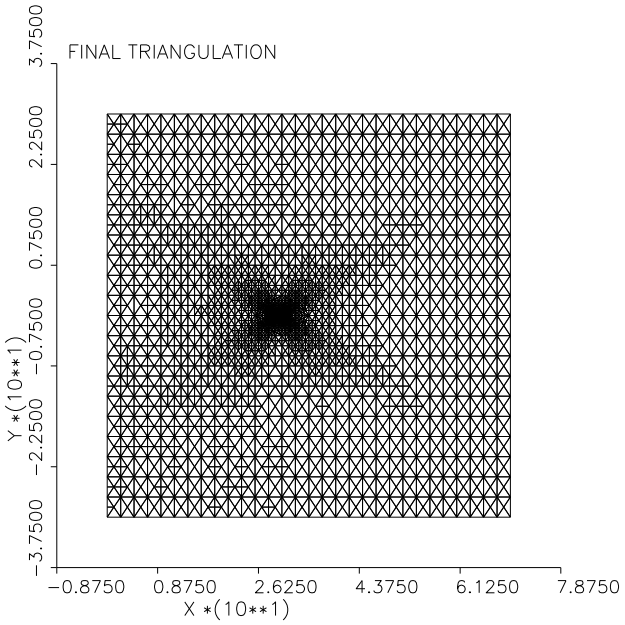


(b) Triangulation

Fig. 1: Oblique collision, $t = 0$

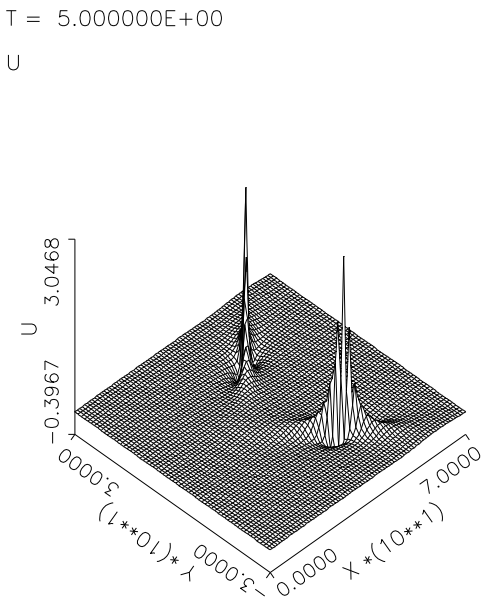


(a) U(x,y,2.5)

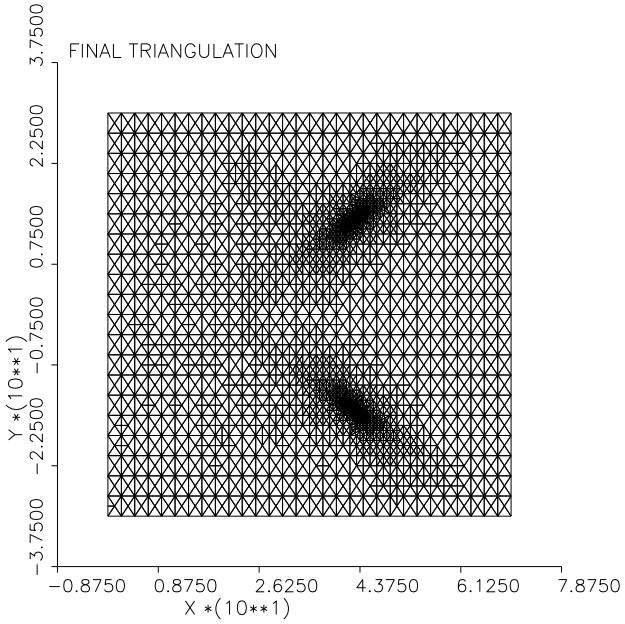


(b) Triangulation

Fig. 2: Oblique collision, $t = 2.5$



(a) U(x,y,5.0)



(b) Triangulation

Fig. 3: Oblique collision, $t = 5.0$

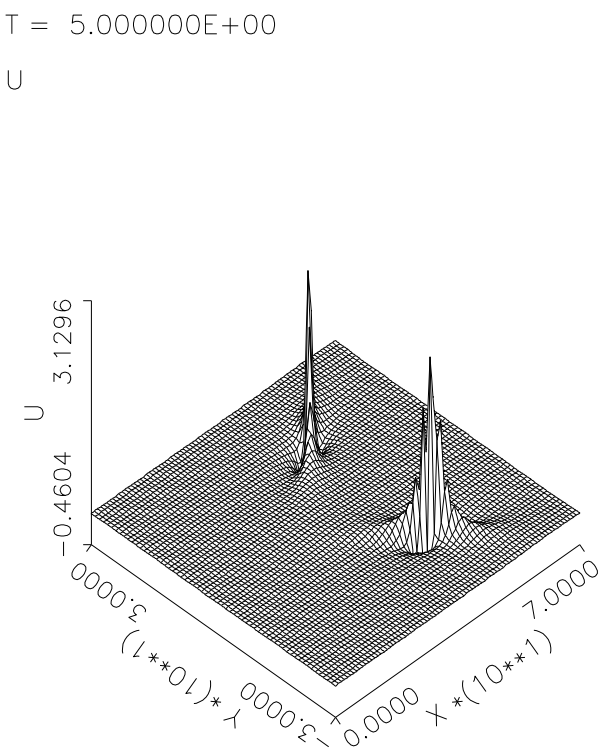
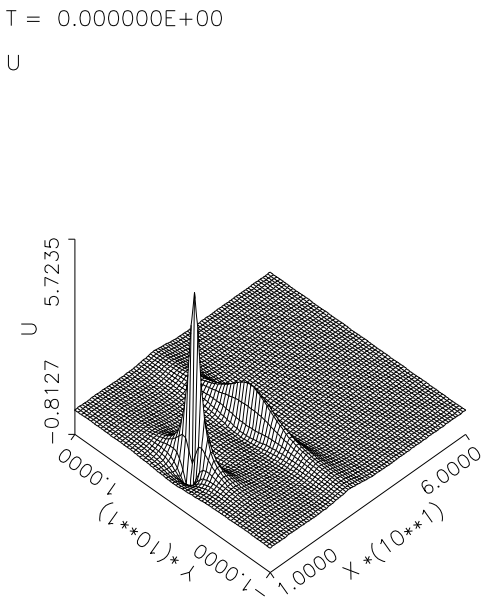
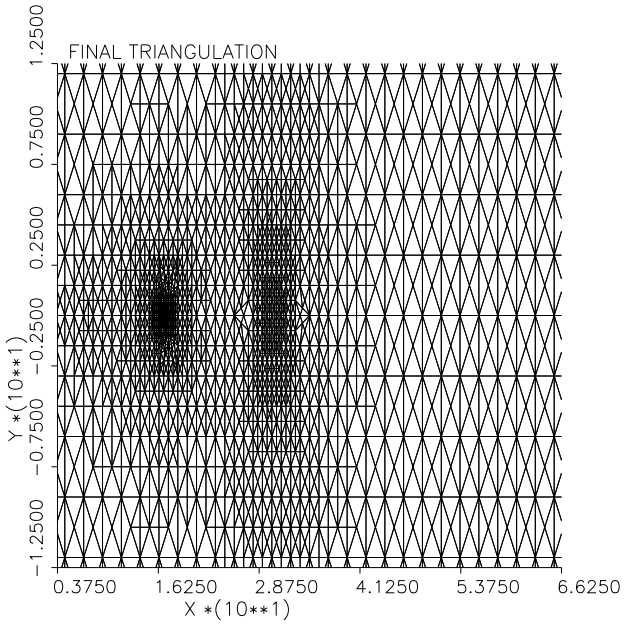


Fig. 4: Oblique collision exact solution, $t = 5.0$

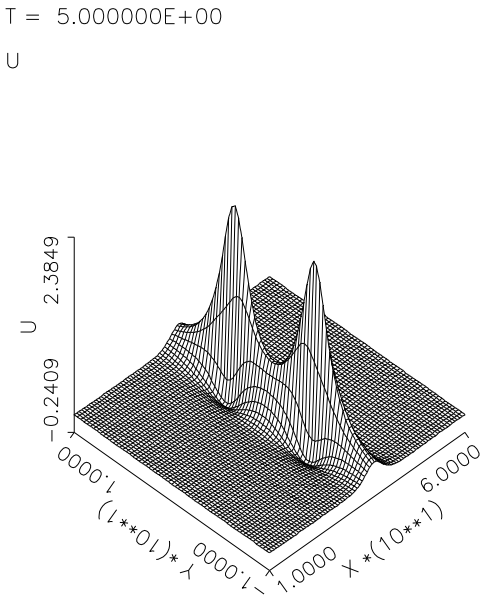


(a) U(x,y,0)

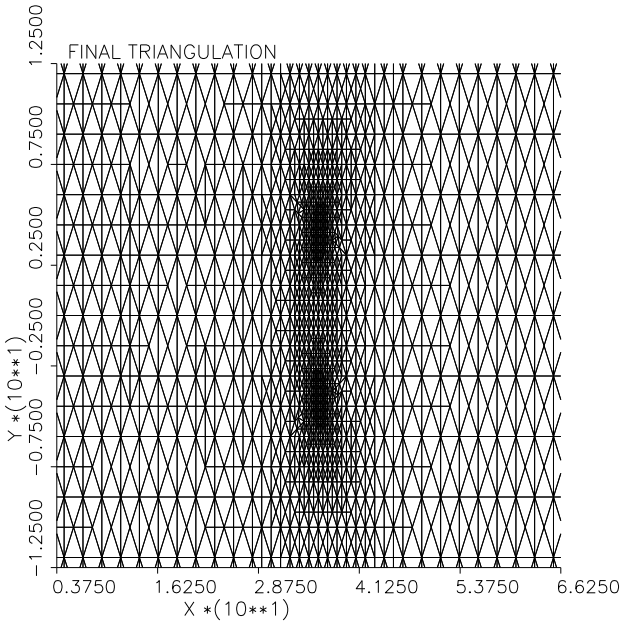


(b) Triangulation

Fig. 5: Direct collision, $t = 0$

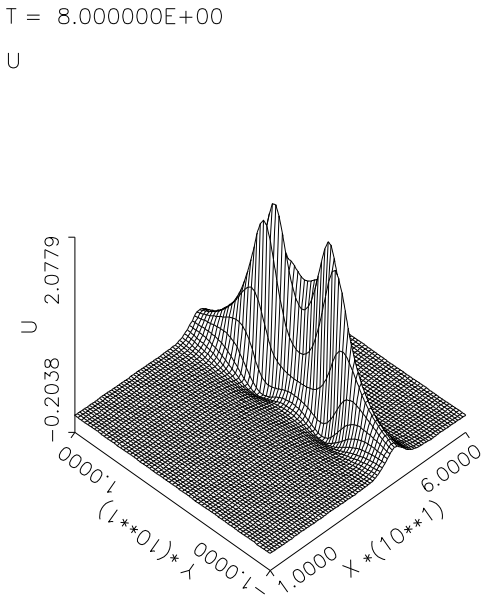


(a) U(x,y,5.0)

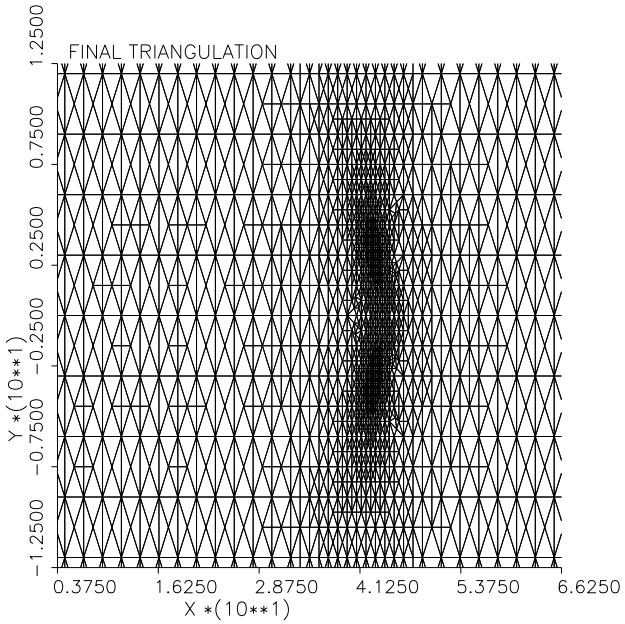


(b) Triangulation

Fig. 6: Direct collision, $t = 5.0$

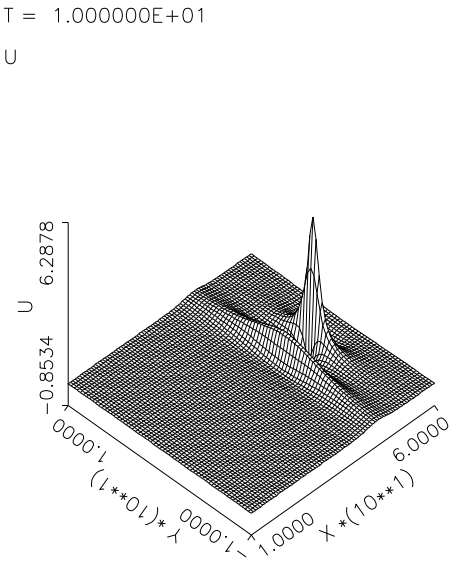


(a) U(x,y,8.0)

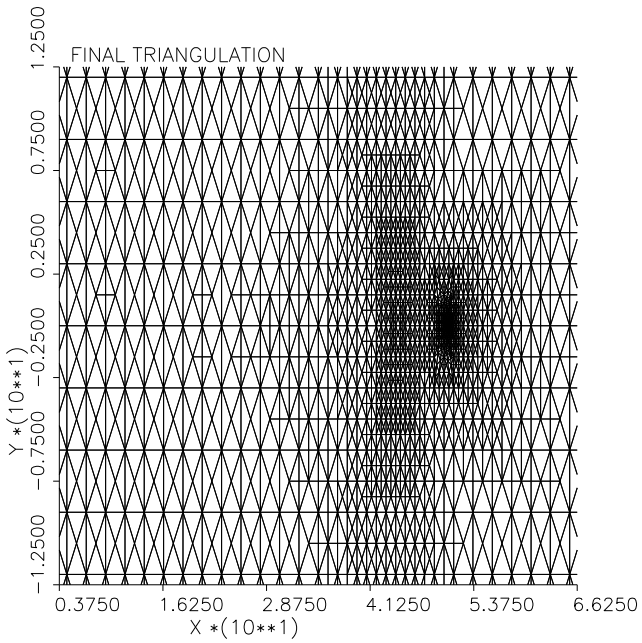


(b) Triangulation

Fig. 7: Direct collision, $t = 8.0$

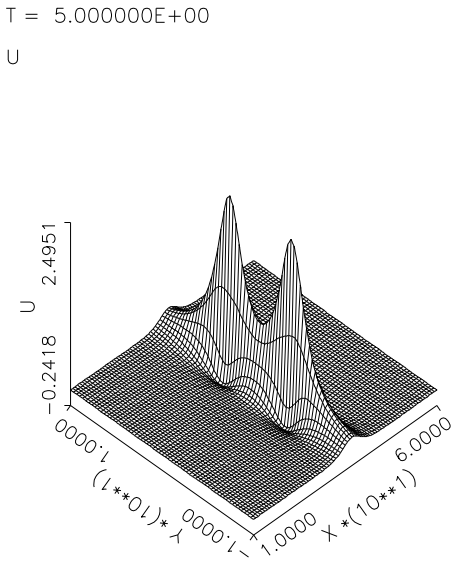


(a) U(x,y,10.0)

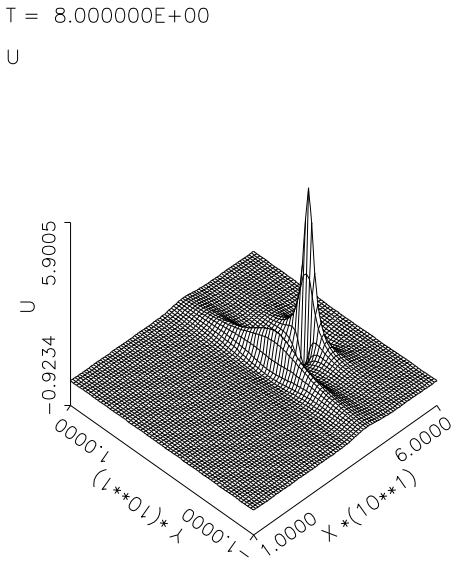


(b) Triangulation

Fig. 8: Direct collision, $t = 10.0$

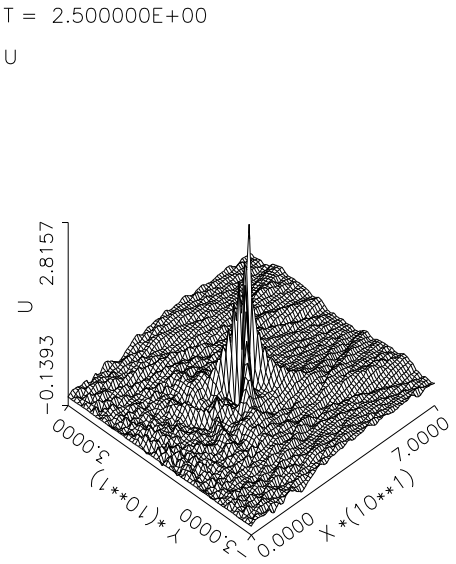


(a) Q(x,y,5.0)

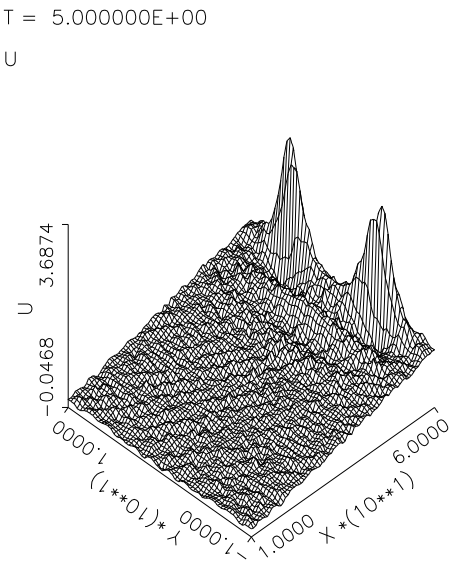


(b) Q(x,y,8.0)

Fig. 9: Direct collision exact solution, $t = 5.0$ and $t = 8.0$



(a) Oblique collision, $t = 2.5$



(b) Direct collision, $t = 5.0$

Fig. 10: Results with uniform grid

where they should be (compare Figures 9a and 10b). The fact that a uniform triangulation of 4800 cubic elements produces such a poor solution illustrates how difficult this nonlinear problem is.

4 Conclusions

Although our PDE2D results appear to be substantially less accurate than those in Lu, Tian and Grimshaw, to judge by the errors in the integrals of U and U^2, they, and to our knowledge every other successful attempt to solve this notoriously difficult PDE, used a numerical method carefully tailored to the KPI equation. We have shown that it is possible to get reasonable accuracy using a general-purpose finite element program, provided an adaptive, moving grid is used which follows the peaks.

References

[1] B. Kadomtsev and V. Petviashvili. On the stability of solitary waves in weakly dispersive media. *Sov. Phys. Dok*, 15:539–541, 1970.

[2] B. Feng and T. Mitsui. A finite difference method for the Korteweg-de Vries and the kadomtsev-Petviashvili equations. *Journal of Computational and Applied Mathematics*, 90:95–116, 1998.

[3] Z. Lu, E. Tian, and R. Grimshaw. Interaction of two lump solitons described by the kadomtsev-petviashvili i equation. *Wave Motion*, 40:123–135, 2004.

[4] B. Fornberg and G.B. Whitham. A numerical and theoretical study of certain nonlinear wave phenomena. *Phil. Trans. Royal Society London*, 289:373–404, 1978.

[5] G. Sewell. *The Numerical Solution of Ordinary and Partial Differential Equations, second edition.* John Wiley & Sons, 2005.

[6] G. Sewell. Solving pdes in non-rectangular 3d regions using a collocation finite element method. *Advances in Engineering Software*, 5:748–753, 2010.

[7] G. Sewell. (free) software for general partial differential equation problems in non-rectangular 2d and 3d regions. *Bulletin of Computational Applied Mathematics*, 1:27–30, 2010.

[8] I. Duff and J. Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal of Scientific and Statistical Computing*, 5:633–641, 1984.