M368K Homework 13

§ 12.3 #2[1], § 12.4 #1[2]

Hershal Bhave (hb6279)

Due May 3rd, 2013

1 § 12.3

1.1 2[1]

Approximate the solution to the given wave equation (hyperbolic partial differential equation) by using the Central-Difference Method with $\Delta x = \frac{1}{8}$ and $\Delta t = \frac{1}{10}$. Compute up to $t = \frac{3}{10}$ and compare approximate and exact solutions at each node at the final time.

The actual solution is $u(x, t) = \sin t \sin 4\pi x$.

$$\begin{aligned} \frac{\delta^2 u}{\delta t^2} - \frac{1}{16\pi^2}\frac{\delta^2 u}{\delta x^2} &= 0, \quad 0 < x < 1, \quad 0 < t; \\ u(0,t) = u(1,t) &= 0, \quad 0 < t \\ u(x,0) &= \sin \pi x, \quad 0 \leqslant x \leqslant 1, \\ \frac{\delta u}{\delta t}(x,0) &= 0, \quad 0 \leqslant x \leqslant 1, \end{aligned} \tag{1}$$

From the given equation we have the following constraints:

$$\begin{aligned} h = \Delta x &= \frac{1}{8} & \alpha^2 &= \frac{1}{16\pi^2} \\ k = \Delta t &= \frac{1}{10} & \lambda = \alpha(k/h) &= \frac{1}{5\pi} \end{aligned} \tag{2}$$

Now we can construct the A matrix and initial **w** vectors using λ, $u(x, 0)$, and $\frac{\delta u}{\delta t}(x, 0)$.

$$A = \begin{pmatrix} 2(1-\lambda^2) & \lambda^2 & 0 & & \cdots & 0 \\ \lambda^2 & 2(1-\lambda^2) & \lambda^2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda^2 & 2(1-\lambda^2) & \lambda^2 \\ 0 & \cdots & & 0 & \lambda^2 & 2(1-\lambda^2) \end{pmatrix} \tag{3}$$

$$\mathbf{w}^{(0)} = f(x_i)$$
$$\mathbf{w}^{(1)} = (1-\lambda^2)f(x_i) + \frac{\lambda^2}{2} f(x_{i+1}) + \frac{\lambda^2}{2} f(x_{i-1}) + kg(x_i)$$

The Discrete Equations for this method are

$$w_{i,j+1} = 2(1-\lambda^2)wi,j + \lambda^2(w_{i+1,j} + w_{i-1,j}) - w_{i,j} - 1 \tag{4}$$

which turns out to be

$$w_{i,j+1} = 2(1 - \frac{1}{5\pi}^2)wi,j + \frac{1}{5\pi}^2 (w_{i+1,j} + w_{i-1,j}) - w_{i,j} - 1 \tag{5}$$

where the boundary conditions imply that

$$w_{0,j} = w_{m,j} = 0. \tag{6}$$

And now further approximations of $\mathbf{w}$ can be computed by

$$\mathbf{w}^{(j+1P)} = A\mathbf{w}^{(j)} - \mathbf{w}^{(j-1)} \tag{7}$$

Higher orders of $\mathbf{w}$ can be obtained by simple matrix multiplication and then subtraction, displayed in eq. (7).

Using the algorithm given in **??**, I obtained the data in tables 1 and 2

x_i	$u(x_i, 0.3)$	$w(i, 0.3)$	$\lvert u(x_i, 0.3) - w(i, 0.3)\rvert$
0	0	0	
0.125	0.29676	0.29676	1.2441×10^{-3}
0.250	0	0	
0.375	-0.29676	-0.29676	1.2441×10^{-3}
0.500	0	0	
0.625	0.29676	0.29676	1.2441×10^{-3}
0.750	0	0	
0.875	-0.29676	-0.29676	1.2441×10^{-3}
1.000	0	0	

Table 1: Data for Number 2 for t=3/10

x_i	$u(x_i, 0.5)$	$w(i, 0.5)$	$\lvert u(x_i, 0.5) - w(i, 0.5) \rvert$
0	0	0	0
0.125	0.48393	0.47943	4.5006 $\times 10^{-3}$
0.250	0.00000	0.00000	0
0.375	-0.48393	-0.47943	4.5006 $\times 10^{-3}$
0.500	0.00000	-0.00000	0
0.625	0.48393	0.47943	4.5006 $\times 10^{-3}$
0.750	0.00000	0.00000	0
0.875	-0.48393	-0.47943	4.5006 $\times 10^{-3}$
1.000	0	0	0

Table 2: Data for Number 2 for t=5/10

2 § 12.4

2.1 1^2

Use the Finite Element Method to approximate the solution to the partial differential equation. Find the finite-element basisfunctions, arrays, and approximate solution; Report the approximate solution at each node. Let $M = 2$; T_1 have vertices $(0, 0.5), (0.25, 0.75), (0, 1)$; and T_2 have vertices $(0, 0.5), (0.5, 0.5), (0.25, 0.75)$.

$$\begin{aligned} u_{xx} + 4u_{yy} &= 3 && \text{in } D, \\ u(x, 0.5) &= 2x && \text{and} \\ u(0, y) &= 0 && \text{on } \mathcal{S}_1, \\ u_x \cos\theta_1 + 4u_y \cos\theta_2 &= (y - x)\frac{\sqrt{2}}{2} && \text{on } \mathcal{S}_2. \end{aligned} \tag{8}$$

We first divide the region D into triangles T_1 and T_2 represented by fig. 1.

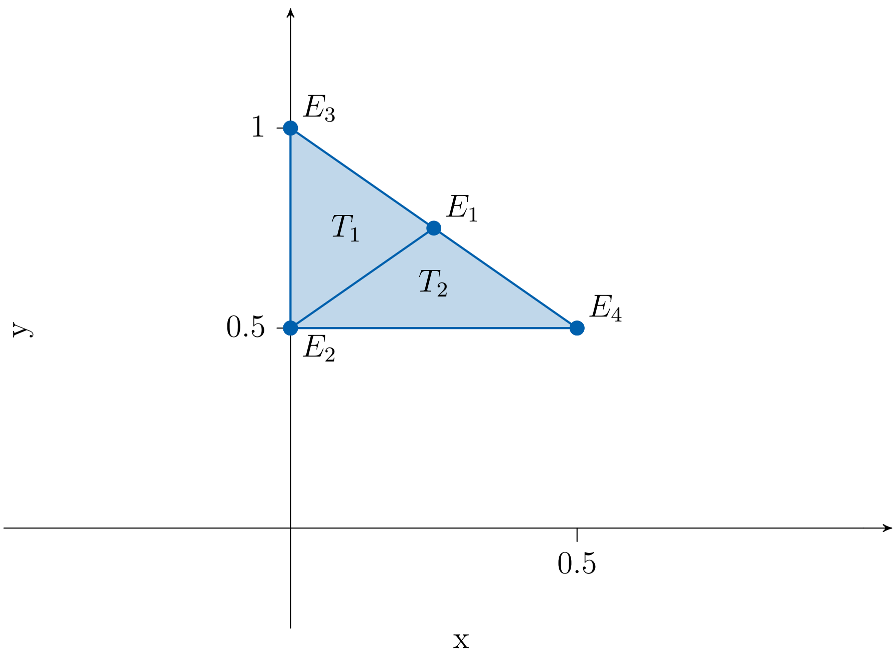


Figure 1: Triangles for Number 1

In this problem, we have

$$\begin{aligned} V_1^{(1)} &= (0.25, 0.75) & V_1^{(2)} &= (0.25, 0.75) \\ V_2^{(1)} &= (0, 0.5) & V_2^{(2)} &= (0, 0.5) \\ V_3^{(1)} &= (0, 1) & V_3^{(2)} &= (0.5, 0.5) \end{aligned} \tag{9}$$

which we can write for simplicity as

$$\begin{aligned} E_1 &= V_1^{(1)} = V_1^{(2)} &&= (0.25, 0.75) \\ E_2 &= V_2^{(1)} = V_2^{(2)} &&= (0, 0.5) \\ E_3 &= V_3^{(1)} &&= (0, 1) \\ E_4 &= V_3^{(2)} &&= (0.5, 0.5). \end{aligned} \tag{10}$$

We must now find the elements of the matrix $A = (\alpha_{i,j})$, for $i = 1, \dots, n$ and $j = 1, \dots, m$, is in the form

$$\alpha_{i,j} = \int\int_D \left[p \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + q \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} - r\phi_i\phi_i \right] \mathrm{d}x\, \mathrm{d}y + \int_{\mathcal{S}_2} g_1 \phi_i \phi_j \, \mathrm{d}\mathcal{S}_2 \tag{11}$$

and β_i, for $i = 1, \dots, n$, in the form

$$\beta_i = -\int\int_{\mathcal{D}} f\phi_i \, \mathrm{d}x\, \mathrm{d}y + \int_{\mathcal{S}_2} g_2 \phi_i \, \mathrm{d}\mathcal{S} - \sum_{k=n+1}^{m} \alpha_{ik}\gamma_k \tag{12}$$

where $\phi(x, y) = a + bx + cy$ for triangles. For triangle T_1 we have

$$
\begin{aligned}
\phi_1(x,y) &= N_1^{(1)}(x,y) = a_1^{(1)} + b_1^{(1)}x + c_1^{(1)}y, \\
\phi_2(x,y) &= N_2^{(1)}(x,y) = a_2^{(1)} + b_2^{(1)}x + c_2^{(1)}y, \\
\phi_3(x,y) &= N_3^{(1)}(x,y) = a_3^{(1)} + b_3^{(1)}x + c_3^{(1)}y, \\
\phi_4(x,y) &= 0
\end{aligned}
\tag{13}
$$

which implies that the partials for ϕ for T_1 are

$$
\begin{aligned}
&\frac{\partial \phi_1}{\partial x} = b_1^{(1)}, \quad \frac{\partial \phi_1}{\partial y} = c_1^{(1)}, \qquad \frac{\partial \phi_2}{\partial x} = b_2^{(1)}, \quad \frac{\partial \phi_2}{\partial y} = c_2^{(1)}, \\
&\frac{\partial \phi_3}{\partial x} = b_3^{(1)}, \quad \frac{\partial \phi_3}{\partial y} = c_3^{(1)}, \qquad \frac{\partial \phi_4}{\partial x} = 0, \quad \frac{\partial \phi_4}{\partial y} = 0.
\end{aligned}
\tag{14}
$$

and for triangle T_2 we have

$$
\begin{aligned}
\phi_1(x,y) &= N_1^{(2)}(x,y) = a_1^{(2)} + b_1^{(2)}x + c_1^{(2)}y, \\
\phi_2(x,y) &= N_2^{(2)}(x,y) = a_2^{(2)} + b_2^{(2)}x + c_2^{(2)}y, \\
\phi_3(x,y) &= 0 \\
\phi_4(x,y) &= N_3^{(2)}(x,y) = a_3^{(2)} + b_3^{(2)}x + c_3^{(2)}y,
\end{aligned}
\tag{15}
$$

which implies that the partials for ϕ for T_2 are

$$
\begin{aligned}
&\frac{\partial \phi_1}{\partial x} = b_1^{(2)}, \qquad \frac{\partial \phi_1}{\partial y} = c_1^{(2)}, \qquad \frac{\partial \phi_2}{\partial x} = b_2^{(2)}, \qquad \frac{\partial \phi_2}{\partial y} = c_2^{(2)}, \\
&\frac{\partial \phi_3}{\partial x} = 0, \qquad \frac{\partial \phi_3}{\partial y} = 0, \qquad \frac{\partial \phi_4}{\partial x} = b_3^{(2)}, \qquad \frac{\partial \phi_4}{\partial y} = c_3^{(2)}.
\end{aligned}
\tag{16}
$$

Each $N_j^{(i)}$ equation corresponds to the vertex $V_j^{(i)}$ and produces systems in the form

$$
\begin{bmatrix} 1 & x_1^{(i)} & y_1^{(i)} \\ 1 & x_2^{(i)} & y_2^{(i)} \\ 1 & x_3^{(i)} & y_3^{(i)} \end{bmatrix}
\begin{bmatrix} a_j^{(i)} \\ b_j^{(i)} \\ c_j^{(i)} \end{bmatrix}
=
\begin{bmatrix} j \stackrel{?}{=} k \\ j \stackrel{?}{=} k \\ j \stackrel{?}{=} k \end{bmatrix}
\tag{17}
$$

and solving each of the systems gives

$$
\begin{aligned}
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0 & 1\end{bmatrix}\begin{bmatrix}a_1^{(1)}\\ b_1^{(1)}\\ c_1^{(1)}\end{bmatrix}=\begin{bmatrix}1\\0\\0\end{bmatrix}; &\qquad \begin{bmatrix}a_1^{(1)}\\ b_1^{(1)}\\ c_1^{(1)}\end{bmatrix}=\begin{bmatrix}0\\4\\0\end{bmatrix}\\
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0 & 1\end{bmatrix}\begin{bmatrix}a_2^{(1)}\\ b_2^{(1)}\\ c_2^{(1)}\end{bmatrix}=\begin{bmatrix}0\\1\\0\end{bmatrix}; &\qquad \begin{bmatrix}a_2^{(1)}\\ b_2^{(1)}\\ c_2^{(1)}\end{bmatrix}=\begin{bmatrix}2\\-2\\-2\end{bmatrix}\\
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0 & 1\end{bmatrix}\begin{bmatrix}a_3^{(1)}\\ b_3^{(1)}\\ c_3^{(1)}\end{bmatrix}=\begin{bmatrix}0\\0\\1\end{bmatrix}; &\qquad \begin{bmatrix}a_3^{(1)}\\ b_3^{(1)}\\ c_3^{(1)}\end{bmatrix}=\begin{bmatrix}-1\\-2\\2\end{bmatrix}\\
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0.5 & 0.5\end{bmatrix}\begin{bmatrix}a_1^{(2)}\\ b_1^{(2)}\\ c_1^{(2)}\end{bmatrix}=\begin{bmatrix}1\\0\\0\end{bmatrix}; &\qquad \begin{bmatrix}a_1^{(2)}\\ b_1^{(2)}\\ c_1^{(2)}\end{bmatrix}=\begin{bmatrix}-2\\0\\4\end{bmatrix}\\
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0.5 & 0.5\end{bmatrix}\begin{bmatrix}a_2^{(2)}\\ b_2^{(2)}\\ c_2^{(2)}\end{bmatrix}=\begin{bmatrix}0\\1\\0\end{bmatrix}; &\qquad \begin{bmatrix}a_2^{(2)}\\ b_2^{(2)}\\ c_2^{(2)}\end{bmatrix}=\begin{bmatrix}2\\-2\\-2\end{bmatrix}\\
&\begin{bmatrix}1 & 0.25 & 0.75\\ 1 & 0 & 0.5\\ 1 & 0.5 & 0.5\end{bmatrix}\begin{bmatrix}a_3^{(2)}\\ b_3^{(2)}\\ c_3^{(2)}\end{bmatrix}=\begin{bmatrix}0\\0\\1\end{bmatrix} &\qquad \begin{bmatrix}a_3^{(2)}\\ b_3^{(2)}\\ c_3^{(2)}\end{bmatrix}=\begin{bmatrix}1\\2\\-2\end{bmatrix}.
\end{aligned}
\tag{18}
$$

We can consider E_1, E_2, and E_4 as nodes on $\mathcal{S}_1$ where the boundary conditions

$$
\begin{aligned}
g_1(x,y) &= u(x,0.5) = 2x, \text{ and}\\
g_2(x,y) &= u(0,y) = 0
\end{aligned}
$$

are imposed. This implies

$$\gamma_2 = 0, \gamma_3 = 0, \text{ and } \gamma_4 = 1. \tag{19}$$

Since we have already determined γ_2, γ_3, and γ_4, we only need to find γ_1. To do this we need to consider $\alpha_{1,1}$ for both triangles. So now the big integral for α in eq. (11) becomes

$$\begin{aligned}
\alpha_{1,1} &= b_1^{(1)}b_1^{(1)}\int\int_{T_1} p\,\mathrm{d}x\,\mathrm{d}y + c_1^{(1)}c_1^{(1)}\int\int_{T_1} q\,\mathrm{d}x\,\mathrm{d}y \\
&\quad - \int\int_{T_1} r(a_1^{(1)} + b_1^{(1)}x + c_1^{(1)}y)(a_1^{(1)} + b_2^{(1)}x + c_2^{(1)}y)\mathrm{d}x\,\mathrm{d}y \\
&\quad + b_1^{(2)}b_1^{(2)}\int\int_{T_2} p\,\mathrm{d}x\,\mathrm{d}y + c_1^{(2)}c_1^{(2)}\int\int_{T_2} q\,\mathrm{d}x\,\mathrm{d}y \\
&\quad - \int\int_{T_2} r(a_1^{(2)} + b_1^{(2)}x + c_1^{(2)}y)(a_1^{(2)} + b_1^{(2)}x + c_1^{(2)}y)\mathrm{d}x\,\mathrm{d}y \\
&= 16\int\int_{T_1} \mathrm{d}x\,\mathrm{d}y + 64\int\int_{T_2} \mathrm{d}x\,\mathrm{d}y \\
\alpha_{1,1} &= 10
\end{aligned} \tag{20}$$

and now the β equation from eq. (12) becomes

$$\begin{aligned}
\beta_1 = &- \int\int_{T_1} 3\phi_1\,\mathrm{d}x\,\mathrm{d}y - \int\int_{T_2} 3\phi_1\,\mathrm{d}x\,\mathrm{d}y \\
&+ \ \mathrm{proj}_{T_1}\int_{\mathcal{S}_2} (y-x)\frac{\sqrt{2}}{2}\phi_1^{(1)}\,\mathrm{d}\mathcal{S} + \ \mathrm{proj}_{T_2}\int_{\mathcal{S}_2} (y-x)\frac{\sqrt{2}}{2}\phi_1^{(2)}\,\mathrm{d}\mathcal{S} \\
&- \sum_{k=2}^{4}\alpha_{1,k}\gamma_k
\end{aligned} \tag{21}$$

We must parametrize the proj_{T_i} entries in order to evaluate the line integral. We do this by

$$\begin{aligned}
\mathrm{proj}_{T_1}\mathcal{S}_2 &:=< 0.5 - 0.25t,\ 0.5 + 0.25t >, \qquad t \in (0,1) \\
\mathrm{proj}_{T_2}\mathcal{S}_2 &:=< 0.25 - 0.25t,\ 0.75 + 0.25t >, \quad t \in (0,1)
\end{aligned} \tag{22}$$

which makes

$$\begin{aligned}
\beta_1 &= 0.375 + \int_0^1 \left[((0.5 - 0.25t) - (0.5 - 0.25t))\frac{\sqrt{2}}{2}(4(0.5 - 0.25t))\right]\mathrm{d}t \\
&\quad + \int_0^1 \left[((0.75 - 0.25t) - (0.25 + 0.25t))\frac{\sqrt{2}}{2}(-2 + 4(0.75 + 0.25t))\right]\mathrm{d}t \\
&\quad - 2.5 \\
&= 0.375 + 3.53553 - 2.5 \\
\beta_1 &= 1.4105
\end{aligned} \tag{23}$$

Now finally we can obtain γ_4

$$(\alpha_{1,1})\gamma_4 = \beta_1 \Rightarrow \gamma_4 = 0.14105 \tag{24}$$

And finally have an approximation to the solution for eq. (8):

$$\boxed{\begin{aligned}
&T_1 : \phi(x,y) = 0.14105(4x), \\
&T_2 : \phi(x,y) = 0.14105(-2 + 4y) + (1 + 2x - 2y)
\end{aligned}} \tag{25}$$

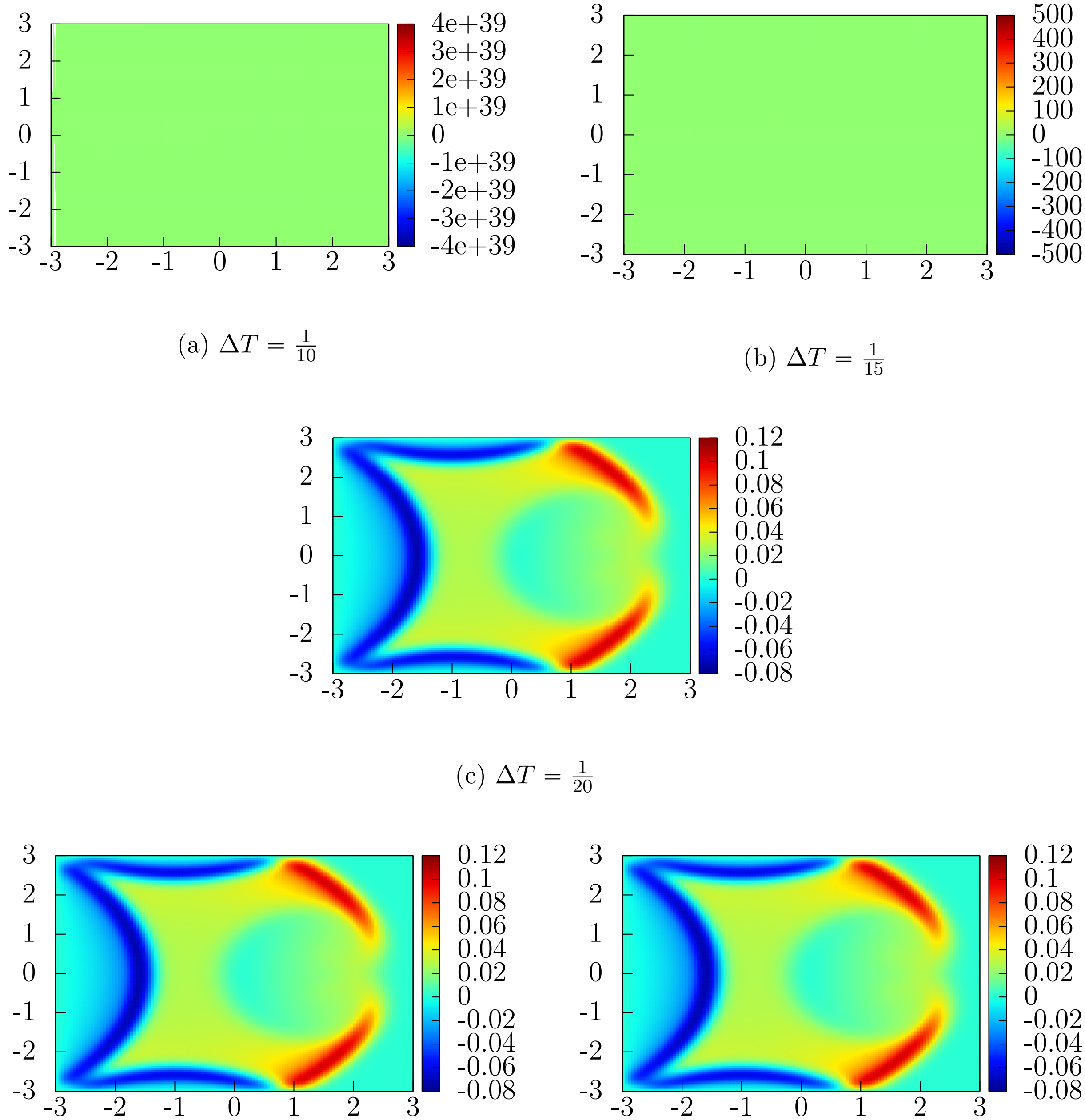
3 Programming Minilab

3.1 Part b

i	Δt_i	$u_i(0,2,7)$
1	$\frac{1}{10}$	-7.02105 $\times 10^{23}$
2	$\frac{1}{15}$	0.00586
3	$\frac{1}{20}$	0.03561
4	$\frac{1}{25}$	0.03558
5	$\frac{1}{30}$	0.03557

Table 3: Data for Minilab Part a

The stability threshold was crossed after Δt crossed $\frac{1}{15}$, as the solution seems to stabilize after that spacing. As can be clearly seen from the plots, there doesn't seem to be any usable data until ΔT hits $\frac{1}{20}$, evidenced by **??**.



(a) $\Delta T = \frac{1}{10}$

(b) $\Delta T = \frac{1}{15}$

(c) $\Delta T = \frac{1}{20}$

(d) $\Delta T = \frac{1}{25}$

(e) $\Delta T = \frac{1}{30}$

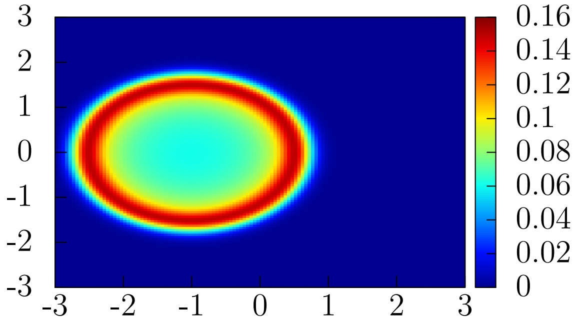
Figure 2: Minilab Part B results

3.2 Part c

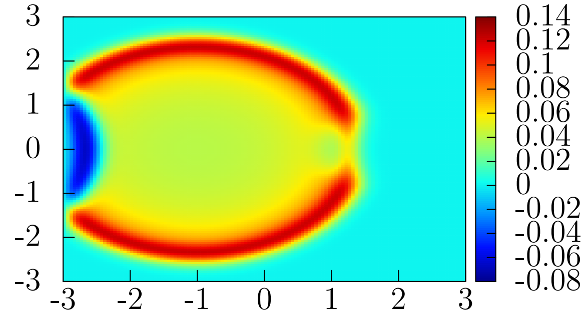
i	t_i	$u_i(0,2,7)$	O_1	O_2	I_{O_1}	I_{O_2}	$\frac{I_{O_2}}{I_{O_1}}$
1	3.5	0.00004	0.15017	0.00000	1.5017	0.0	0.0
2	5	0.11079	0.05088	0.00001	1.5017	0.00001	6.65911×10^{-5}
3	7	0.03557	0.03057	0.02755	1.5017	0.02755	0.18345

Table 4: Data at $u_i(0,2,7)$ for each T

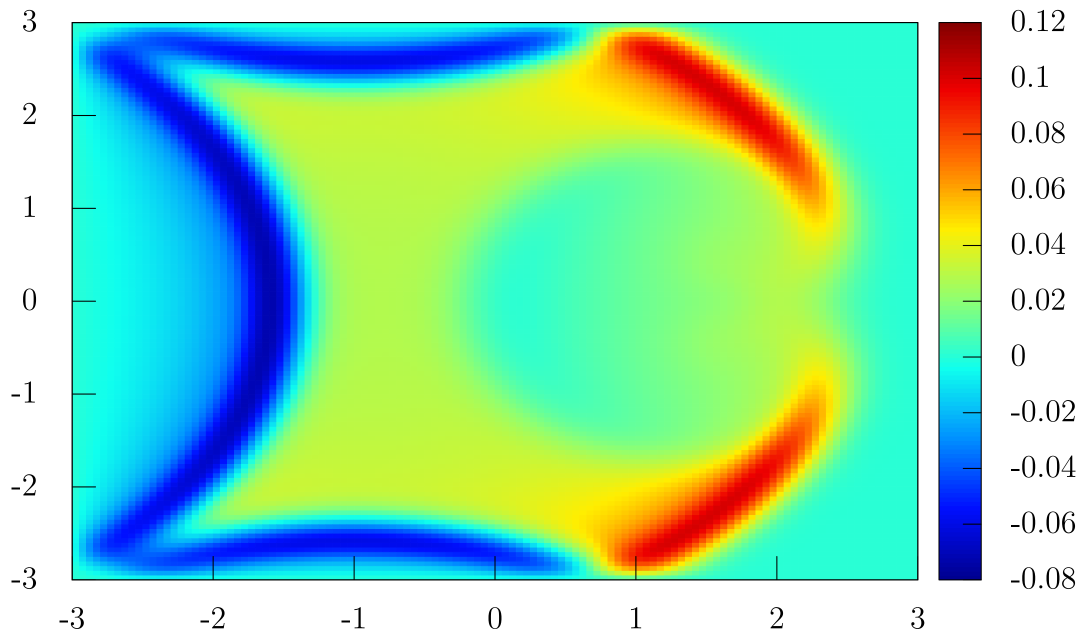
The data is listed in **??**. The figures for each time step are shown in **??**. The interval $[0,7]$ would probably not be appropriate if the objective was to measure the outgoing waves at each sensor, since the sensors would start to pick up the echoes around $T = 7$ (as can be seen in **??**).



(a) T = 3.5



(b) T = 5



(c) T = 7

Figure 3: Minilab Part C results

4 Code

```
#!/usr/bin/octave
# Created by Hershal Bhave on 05/02/13
# For M368K HW13,  12.3 Number 2
# Written in GNU Octave
#
# Description: Computes the approximate solution to the Wave Equation
# (Hyperbolic PDE), where f=u(x,0), g=(du/dt)(x,0), alpha is alpha
# in du/dt-alpha^2*(d^2u/dt^2), h is deltax, k is deltat, m is the
# length of x, and n is the length of t

function [w] = wavediff(f,g,alpha,h,k,m,n)

  i = 1:m-1;
  x = (i*h)';
  ## wold = f(x);
  ## w = wold + k*g(x);
  lambda = alpha*(k/h);

  w(:,1) = f(x);
  w(:,2) = (1-lambda^2)*f(i*h)+(lambda^2/2)*(f((i+1)*h) + \
              f((i-1)*h)) + k*g(i*h);

  ## Not sure why this doesn't work
  A(1,1) = 2*(1-lambda^2);
  A(1,2) = lambda^2;

  for i=2:m-2
    A(i,i-1) = lambda^2;
    A(i,i) = 2*(1-lambda^2);
    A(i,i+1) = lambda^2;
  endfor

  A(m-1,m-2) = lambda^2;
  A(m-1,m-1) = 2*(1-lambda^2);

  for i=1:n-1
    w(:,i+2) = A*w(:,i+1) - w(:,i);;
  endfor

endfunction
```

Listing 1: `wavediff.m`

```
/*************************************************************
Program 13. Uses the central-difference method to find an
approximate solution of a hyperbolic IBVP in a rectangular
domain of the form

 utt + s ut = P uxx + Q uyy + p ux + q uy + r u + eta,
                              a<=x<=b, c<=y<=d, 0<=t<=T

 u(a,y,t) = ga(y,t), u(b,y,t) = gb(y,t), c<=y<=d, 0<=t<=T
 u(x,c,t) = gc(x,t), u(x,d,t) = gd(x,t), a<=x<=b, 0<=t<=T

 u(x,y,0) = f(x,y), a<=x<=b, c<=y<=d
 ut(x,y,0) = gamma(x,y), a<=x<=b, c<=y<=d


Inputs:
  PDEeval Function to evaluate P,Q,p,q,r,s,eta
  BCeval Function to evaluate ga,gb,gc,gd
  ICeval Function to evaluate f,gamma
  a,b,c,d Space domain parameters
  N,M Number of interior x,y pts (N+2,M+2 total pts)
  x Grid point vector: x(i)=a+i*dx, i=0...N+1
  y Grid point vector: y(j)=c+j*dy, j=0...M+1
  T Time interval parameter (final time)
  L Number of time steps

Outputs:
  x Grid point vector: x(i)=a+i*dx, i=0...N+1
  y Grid point vector: y(j)=c+j*dy, j=0...M+1
  u Approx soln at time t=T: u(i,j) is soln
              at x(i),y(j), i=0...N+1, j=0...M+1


Note 1: The function file ctrdiff2D.cpp is incomplete; you'll
need to finish coding the method as indicated in that file.

Note 2: For any given problem, the functions PDEeval, BCeval
and ICeval must be changed.

Note 3: For any given problem, the grid parameters a,b,c,d,T
and N,M,L must be specified.

Note 4: To compile this program use the command (all on one
line)

  c++ -o program13 matrix.cpp ctrdiff2D.cpp program13.cpp

Note 5: The program output is written to a file.
*************************************************************/
#include <iostream>
#include <iomanip>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include "matrix.h"
using namespace std;


/*** Define output file ***/
const char myfile[20]="program13.out" ;
ofstream prt(myfile) ;


/*** Declare external function ***/
```

```
int ctrdiff2D(int, int, double, double, double, double,
             vector&, vector&, double&, double&, matrix&, matrix&) ;

/*** Define P(x,y,t), Q(x,y,t),
           p(x,y,t), q(x,y,t), r(x,y,t), s(x,y,t), eta(x,y,t) ***/
void PDEeval(const double& x, const double& y, double& t,
      double& P, double& Q, double& p, double& q,
      double& r, double& s, double& eta){

  // P = 0.1*(1 + exp(-x*y*t/10)) ;
  // Q = 0.1 ;
  // p = 0.02*x*y ;
  // q = 0.04*x ;
  // r = 0.01 ;
  // s = 0.05 ;
  // eta = 0.01*t - 0.02*x*sin(y) ;

  // For Programming mini[]lab
  double pi=4.0*atan(1.0);
  double xs = 1.5;
  double ys = 0.0;
  double xx = -1;
  double yy = 0;
  double tt = 0.5;
  double w = sqrt(pow(x-xs,2) + 0.25*pow(y-ys,2));
  eta = 10.0*(exp(-30*pow(x-xx,2)
      - 30*pow(y-yy,2)
      - 30*pow(t-tt,2)));
  if(w>=0 && w<=0.5) {
    s = 10*pow(cos(pi*w),2);
  } else {
    s = 0;
  }
  P = Q = 0.3;
  p = q = r = 0;

}

/*** Define ga(y,t), gb(y,t), gc(x,t), gd(x,t) ***/
void BCeval(const double& x, const double& y, double& t,
          double& ga, double& gb, double& gc, double& gd){

  // ga = y ;
  // gb = sqrt(y) ;
  // gc = 0 ;
  // gd = x ;

  ga = gb = gc = gd = 0;

}

/*** Define f(x,y), gamma(x,y) ***/
void ICeval(const double& x, const double& y,
     double& f, double& gamma) {

  // f = 1 ;
  // gamma = x*y*(1-x)*(1-y) ;
  f = gamma = 0;
}

int main() {
  /*** Define problem parameters ***/
  // int N=9, M=9, L=50, success_flag=0 ;
  // matrix u(N+2,M+2), uold(N+2,M+2) ;
  // vector x(N+2), y(M+2) ;
```

```
// double a=0, b=1, c=0, d=1 ;
// double dx=(b-a)/(N+1), dy=(d-c)/(M+1) ;
// double T=5, dt=T/L ;

// Minilab Part B
// // For \Delta t = \frac{1}{10}
// int L = 70;
// double T = 7;

// // For \Delta t = \frac{1}{15}
// int L = 105;
// double T = 7;

// // For \Delta t = \frac{1}{20}
// int L = 140;
// double T = 7;

// // For \Delta t = \frac{1}{25}
// int L = 175;
// double T = 7;

// // For \Delta t = \frac{1}{30}
// int L = 210;
// double T = 7;

// Minilab Part C
// // For [0,T] = [0, 3.5]
// double T = 3.5;
// int L = 105;

// // For [0,T] = [0, 5]
// double T = 5;
// int L = 150;

// For [0,T] = [0, 7]
double T = 7;
int L = 210;

int N=119 , M=119 , success_flag=0;
matrix u(N+2,M+2), uold(N+2,M+2) ;
vector x(N+2), y(M+2) ;
double a=-3, b=3, c=-3, d=3;
double dx=(b-a)/(N+1), dy=(d-c)/(M+1) ;
double dt=T/L ;

/*** Construct xy-grid ***/
for(int i=0; i<=N+1; i++){
  x(i) = a + i*dx ;
}
for(int j=0; j<=M+1; j++){
  y(j) = c + j*dy ;
}

/*** Load initial values for u ***/
double t=0 ;
double gLeft, gRight, gBottom, gTop, f, gamma ;
for(int i=0; i<=N+1; i++){ //actual i-index on grid
  for(int j=0; j<=M+1; j++){ //actual j-index on grid
    BCeval(x(i),y(j),t,gLeft,gRight,gBottom,gTop) ;
    ICeval(x(i),y(j),f,gamma) ;
    if( j==M+1 ){ u(i,j) = gTop ; }
    if( i==N+1 ){ u(i,j) = gRight ; }
    if( i==0 ){ u(i,j) = gLeft ; }
    if( j==0 ){ u(i,j) = gBottom ; }
    if((i>0)&&(i<N+1)&&(j>0)&&(j<M+1)){ u(i,j) = f ; }
```

```
    }
  }
  uold = u ; //initialize uold

  /*** Call ctr-diff method at each time step (overwrites u,uold) ***/
  for(int n=0; n<L; n++){
    success_flag = ctrdiff2D(N,M,a,b,c,d,x,y,t,dt,uold,u) ;
    t = t + dt ;
  }

  /*** Print results at final time to output file ***/
  prt.setf(ios::fixed) ;
  prt << setprecision(5) ;
  cout << "Ctr-Diff-2D: output written to " << myfile << endl ;
  prt << "Ctr-Diff-2D results" << endl ;
  prt << "Number of interior x-grid pts: N = " << N << endl ;
  prt << "Number of interior y-grid pts: M = " << M << endl ;
  prt << "Number of time steps: L = " << L << endl ;
  prt << "Final time of simulation: t = " << t << endl ;
  prt << "Approximate solution at time t: x_i, y_j, u_ij" << endl ;
  for(int i=0; i<=N+1; i++){
    for(int j=0; j<=M+1; j++){
      prt << setw(8) << x(i) ;
      prt << " " ;
      prt << setw(8) << y(j) ;
      prt << " " ;
      prt << setw(8) << u(i,j) ;
      prt << endl;
    }
  }

  return 0 ; //terminate main program
}
```

Listing 2: `program13.cpp`

```
/*************************************************************
Function to implement the central-difference method to find
an approximate solution of a hyperbolic IBVP in a rectangular
domain of the form

 utt + s ut = P uxx + Q uyy + p ux + q uy + r u + eta,
                                a<=x<=b, c<=y<=d, 0<=t<=T

 u(a,y,t) = ga(y,t), u(b,y,t) = gb(y,t), c<=y<=d, 0<=t<=T
 u(x,c,t) = gc(x,t), u(x,d,t) = gd(x,t), a<=x<=b, 0<=t<=T

 u(x,y,0) = f(x,y), a<=x<=b, c<=y<=d
 ut(x,y,0) = gamma(x,y), a<=x<=b, c<=y<=d


Inputs:
  PDEeval Function to evaluate P,Q,p,q,r,s,eta
  BCeval Function to evaluate ga,gb,gc,gd
  ICeval Function to evaluate f,gamma
  a,b,c,d Space domain parameters
  N,M Number of interior x,y pts (N+2,M+2 total pts)
  x Grid point vector: x(i)=a+i*dx, i=0...N+1
  y Grid point vector: y(j)=c+j*dy, j=0...M+1
  t,dt Current time t and time step dt
  uold Approx soln at time t-dt
  u Approx soln at time t: u(i,j) is soln
              at x(i),y(j), i=0...N+1, j=0...M+1

Outputs:
  x Grid point vector: x(i)=a+i*dx, i=0...N+1
  y Grid point vector: y(j)=c+j*dy, j=0...M+1
  uold Approx soln at time t
  u Approx soln at time t+dt: u(i,j) is soln
              at x(i),y(j), i=0...N+1, j=0...M+1


Note 1: This function is incomplete; you'll need to finish
coding the method as indicated below.

Note 2: The functions PDEeval, BCeval and ICeval are assumed
to be defined externally (e.g. by calling program).

Note 3: Rather than use the single-label index l=1...NM
for the interior xy-grid, it is convenient to use the
double-label indices i=0...N+1, j=0...M+1 for the entire
xy-grid.
*************************************************************/
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include "matrix.h"
using namespace std;


/*** Ext fun: PDEeval(x,y,t,P,Q,p,q,r,s,eta) ***/
void PDEeval(const double&, const double&, double&,
                   double&, double&, double&, double&,
                             double&, double&, double&) ;

/*** Ext fun: BCeval(x,y,t,gLeft,gRight,gBottom,gTop) ***/
void BCeval(const double&, const double&, double&,
                   double&, double&, double&, double&) ;

/*** Ext fun: ICeval(x,y,f,gamma) ***/
```

```
void ICeval(const double&, const double&, double&, double&) ;


/*** Main function: central-difference method ***/
int ctrdiff2D(int N, int M,
              double a, double b, double c, double d,
              vector& x, vector& y, double& t, double& dt,
                                   matrix& uold, matrix& u){

  int success_flag=0 ;
  double P, Q, p, q, r, s, eta, tn, tnn ;
  double gLeft, gRight, gBottom, gTop, f, gamma ;
  double uLeft, uRight, uBottom, uTop, uCenter ;
  double dx=(b-a)/(N+1), dy=(d-c)/(M+1) ; //grid params
  double ahatij, bhatij, chatij, dhatij, ehatij ; //ctr-diff params
  matrix unew(N+2,M+2) ; //temporary variable

  /*** Compute u^{n+1} at interior grid points ***/
  for(int i=1; i<=N; i++) { //actual i-index on grid
    for(int j=1; j<=M; j++) { //actual j-index on grid

      tn = t ;
      PDEeval(x(i),y(j),tn,P,Q,p,q,r,s,eta) ;
      BCeval(x(i),y(j),tn,gLeft,gRight,gBottom,gTop) ;
      dhatij = Q/(dy*dy) + q/(2.0*dy) ;
      ehatij = Q/(dy*dy) - q/(2.0*dy) ;
      chatij = P/(dx*dx) + p/(2.0*dx) ;
      ahatij = P/(dx*dx) - p/(2.0*dx) ;
      bhatij = r - 2.0*P/(dx*dx) - 2.0*Q/(dy*dy) ;

      if( j<M ){
        uTop = u(i,j+1) ;
      }
      else {
        uTop = gTop ;
      }

      if( i>1 ){
        uLeft = u(i-1,j) ;
      }
      else {
        uLeft = gLeft ;
      }

      uCenter = u(i,j) ;

      if( i<N ){
        uRight = u(i+1,j) ;
      }
      else {
        uRight = gRight ;
      }

      if( j>1 ){
        uBottom = u(i,j-1) ;
      }
      else {
        uBottom = gBottom;
      }

      if(t==0){
        /*** Compute starting value unew = u^{1} using Taylor
        expansion formula with u = u^{0} = f and (du/dt)^{0} = gamma.
        At t=0, the u-matrix has the f-values, but we need to call
        ICeval to get the gamma-values. This part is complete. ***/
```

```
      ICeval(x(i),y(j),f,gamma) ;
      unew(i,j) = u(i,j)
                     + dt*gamma
                     + (dt*dt/2)*dhatij*uTop
                     + (dt*dt/2)*ahatij*uLeft
                     + (dt*dt/2)*bhatij*uCenter
                     + (dt*dt/2)*chatij*uRight
                     + (dt*dt/2)*ehatij*uBottom
                     + (dt*dt/2)*eta
                     - (dt*dt/2)*s*gamma ;
    } else {
      /*** Compute unew = u^{n+1} using central-difference formula
      with u = u^{n} and uold = u^{n-1}. THIS PART NEEDS TO BE
      COMPLETED. ***/
      // unew(i,j) = 0 ;
  unew(i,j) = (2*u(i,j) - (1-s*dt/2)*uold(i,j) + dt*dt
         *(dhatij*uTop + ahatij*uLeft
           + bhatij*uCenter + chatij*uRight
           + ehatij*uBottom + eta))/(1 + s*dt/2);
    }

   }
  }

  /*** Compute u^{n+1} at boundary grid points ***/
  for(int i=0; i<=N+1; i++){ //actual i-index on grid
    tnn = t + dt ;
    BCeval(x(i),y(M+1),tnn,gLeft,gRight,gBottom,gTop) ;
    unew(i,M+1) = gTop ;
    BCeval(x(i),y(0),tnn,gLeft,gRight,gBottom,gTop) ;
    unew(i,0) = gBottom ;
  }
  for(int j=0; j<=M+1; j++){ //actual j-index on grid
    tnn = t + dt ;
    BCeval(x(N+1),y(j),tnn,gLeft,gRight,gBottom,gTop) ;
    unew(N+1,j) = gRight ;
    BCeval(x(0),y(j),tnn,gLeft,gRight,gBottom,gTop) ;
    unew(0,j) = gLeft ;
  }

  uold = u ; //update uold-matrix
  u = unew ; //update u-matrix

  return success_flag=1 ;
}
```

Listing 3: `ctrdiff2D.cpp`

```perl
#!/usr/bin/perl -w

use warnings;
use strict;

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
  die("usage: <format> <data files...>");
}

my $format = shift(@ARGV);

foreach my $file (@ARGV) {
  open(my $in, "<$file") or die("Could not open $file for reading.");
  my $basename = $file;
  $basename =~ s/(^.*)\..*?$/$1/;
  $basename =~ s/_/-/g;

  my $outFile = $basename.".tmp";
  my $gnuOutputFile = $basename.".".$format;

  open(my $out, ">$outFile") or die("Could not open $outFile for writing");

  my $old1;
  while (<$in>) {
    if (defined($old1) && $1 ne $old1) {
      print($out "\n")
    }
    if ($_ =~ m/(-?\d+\.?\d+)\s+(-?\d+\.?\d+)\s+(-?\d+\.?\d+)/g) {
      print($out "$1 $2 $3\n");
    }
    $old1 = $1;
  }

  my $plotcmd = <<PERLEOF;
#!/bin/sh
gnuplot << GNUEOF
set style line 11 lc rgb 'black' lt 1
set style line 12 lc rgb '#808080' lt 0 lw 1
set tics nomirror

set nokey

set terminal $format
set output "$gnuOutputFile"

set style line 1 lc rgb 'black' lt 1 lw 2

set pm3d map

# Matlab-style colorbars
# Source: http://www.gnuplotting.org/matlab-colorbar-with-gnuplot/
set palette defined ( 0 "#000090", 1 "#000fff", 2 "#0090ff",\\
                      3 "#0fffee", 4 "#90ff70", 5 "#ffee00",\\
                      6 "#ff7000", 7 "#ee0000", 8 "#7f0000")

splot '$outFile'

GNUEOF

PERLEOF

  system("$plotcmd");
}
```

```
printf("done\n");
```

Listing 4: `minilab_contour.pl`

```
#!/usr/bin/gnuplot

set object 1 polygon from 0,.5 to \
    0.25,0.75 to \
    .5,.5 to \
    0,.5 \
    fs transparent solid .25 fc rgb '#0060ad'
set object 2 polygon from 0,.5 to \
    0.25,0.75 to \
    0,1 to 0,.5 \
    fs transparent solid .25 fc rgb '#0060ad'

set xlabel "x" offset 2
set ylabel "y" offset 2
set xrange [-.5:1.0]
set yrange [-.25:1.25]
set xtics 0.5,.5,.5 axis
set ytics 0.5,.5,1 axis
set style line 11 lc rgb 'black' lt 1
set style line 12 lc rgb '#808080' lt 0 lw 1
set tics nomirror
set xzeroaxis ls 11
set yzeroaxis ls 11
set arrow from 1,0 to 1.05,0 size screen 0.025,15,60 filled ls 11
set arrow from 0,1.25 to 0,1.3 size screen 0.025,15,60 filled ls 11
set noborder
set border 0 back ls 11

set label '$E_1$' center at .30,.8
set label '$E_2$' center at .05,.45
set label '$E_3$' center at .05,1.05
set label '$E_4$' center at .55,.55
set label '$T_1$' center at .0975,.75
set label '$T_2$' center at .25,.6125

set terminal tikz
set output 'triangles.tikz'
set nokey

plot '-' with linespoints lc rgb '#0060ad' pt 7 lt 1 lw 2 ps 1.5, \
     '-' with linespoints lc rgb '#0060ad' pt 7 lt 1 lw 2 ps 1.5, \
     '-' with linespoints lc rgb '#0060ad' pt 7 lt 1 lw 2 ps 1.5, \
     '-' with linespoints lc rgb '#0060ad' pt 7 lt 1 lw 2 ps 1.5
0 0.5
0 1
e
0 0.5
.5 .5
e
0 1
.5 .5
e
0 0.5
.25 .75
e
```

Listing 5: plot_1.gp