Wave Equation

Abushet Wosene Simanesew,
Daniel Alexander Mo Søreide Houshmand,
Imran Ali

October 21, 2013

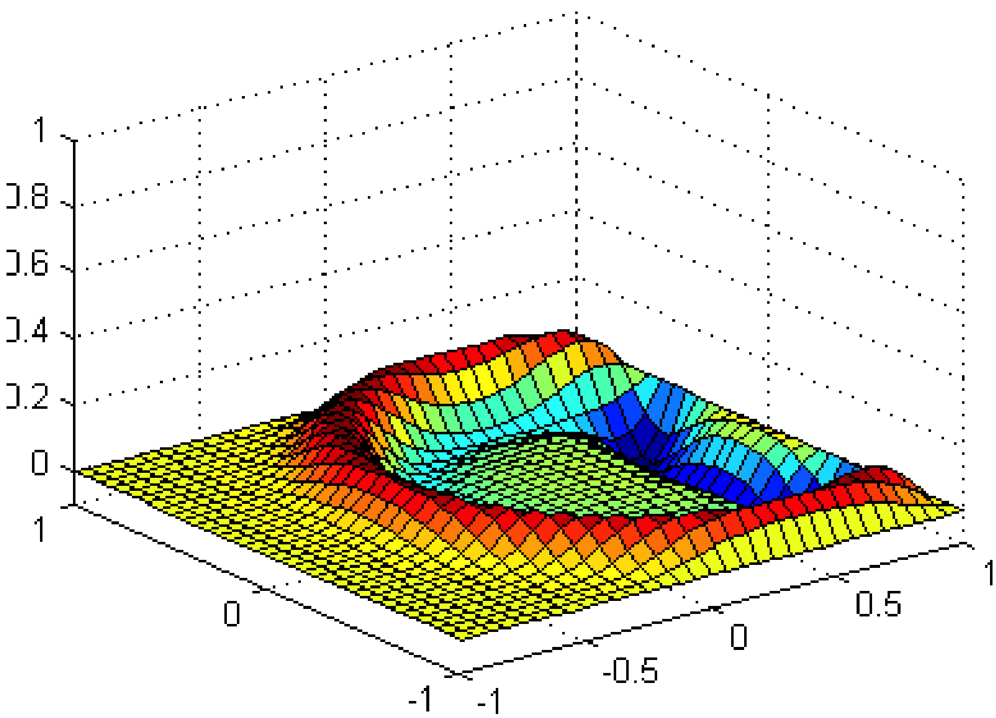

Figure 1: Courtesy of Mathworks

Contents

1 Mathematical Problem

We address the initial-value problem

$$\frac{\partial^2 u}{\partial t^2} + b\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(q(x,y)\frac{\partial u}{\partial x}\right) \tag{1}$$

$$+ \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right) + f(x,y,t), \quad t \in (0,T], \tag{2}$$

$$u(x,y,0) = I, \text{ on } \partial\Omega \tag{3}$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega \tag{4}$$

in a rectangular spatial mesh domain $\Omega = [0, L_x]\text{x}[0, L_y]$. The initial conditions are

$$u(x,y,0) = I(x,y), \tag{5}$$

$$u(x,y,0) = V(x,y), \tag{6}$$

where b is damping coefficient, $q(x,y)$ is the square of wave velocity, T the total period and $u(x,y,t)$ is the unknown function to be estimated. This mathematical model addresses the two-dimensional, linear wave equation with damping.

2 Descretization

2.1 Descritizing the domain

The temporal domain $[0,T]$ is represented by a finite number of mesh points

$$0 = t_0 < t_1 \cdots < t_{N_t+1} = T. \tag{7}$$

Similarly the 2D spatial domain $\Omega = [0, L_x]x[0, L_y]$ is replaced by spatial mesh points

$$0 = x_0 < x_1 \cdots < x_{N_x-1} < x_{N_x+1} = L_x, \tag{8}$$

$$0 = y_0 < y_1 \cdots < y_{N_y-1} < y_{N_y+1} = L_y, \tag{9}$$

The mesh points are defined as (x_i, y_j, t_n), with indices $i = 0, \dots, N_x + 1$, $y_j = 0, \dots, N_y + 1$ and $n = 0, \dots, N_t + 1$. For uniformly distributed mesh points one can introduce constant mesh spacings Δt, Δx and Δy, such that the mesh points can be written as follows

$$x_i = i\Delta x, i = 0, \dots, N_x + 1, \tag{10}$$

$$y_j = i\Delta y, j = 0, \dots, N_y + 1, \tag{11}$$

$$t_n = n\Delta t, n = 0, \dots, N_t + 1, \tag{12}$$

2.2 The discrete solution

The exact solution $u(x, y, t)$ is now to be approximated by the mesh function $u^n_{i,j}$ at the mesh points (x_i, y_j, t_n) defined in the domain above. The numerical solution of the partial differential equation (2) is fulfilled at the interior mesh points:

$$\begin{aligned}\frac{\partial^2 u(x_i, y_j, t_n)}{\partial t^2} + b\frac{\partial u(x_i, y_j, t_n)}{\partial t} &= \frac{\partial}{\partial x}\left(q(x_i, y_j)\frac{\partial u(x_i, y_j, t_n)}{\partial x}\right)\\ &+ \frac{\partial}{\partial y}\left(q(x_i, y_j)\frac{\partial u(x_i, y_j, t_n)}{\partial y}\right)\\ &+ f(x_i, y_j, t_n), \quad t \in (0, T],\end{aligned} \tag{13}$$

for $i = 1, \dots, N_x + 1$, $j = 1, \dots, N_y + 1$ and $n = 1 \dots, N_t$. For $n = 0$ we have the initial conditions (5) and (9) and at the boundaries $i = 0, N_x + 1$ and $j = 0, N_y + 1$ we have the boundary condition (4). The first and second order derivatives involved in the equation can now be replaced by finite differences. The second-order time derivative to the left side of the equality sign in equation (13) can be replaced by central differences

$$\frac{\partial^2}{\partial t^2}u(x_i, y_j, t_n) \approx \frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2} \tag{14}$$

The term containing damping is replaced by centered difference as

$$b\frac{\partial}{\partial t}u(x_i, y_j, t_n) \approx b\frac{u^{n+1}_{i,j} - u^{n-1}_{i,j}}{2\Delta t} \tag{15}$$

The derivatives containing the the variable coefficients are replaced by centered difference. The variable coefficient at the center of the mesh are estimated by arithimetic average of the mesh points bounding the center point.

$$\frac{\partial}{\partial x}(q\frac{\partial u}{\partial x}) = \frac{q_{i+1/2,j}\frac{u^n_{i+1,j}-u^n_{i,j}}{\Delta x} - q_{i-1/2,j}\frac{u^n_{i,j}-u^n_{i-1,j}}{\Delta x}}{\Delta x} \tag{16}$$

$$\frac{\partial}{\partial x}(q\frac{\partial u}{\partial x}) = \frac{\frac{1}{2}(q_{i,j}+q_{i+1,j})(u^n_{i+1,j}-u^n_{i,j}) - \frac{1}{2}(q_{i,j}+q_{i-1,j})(u^n_{i,j}-u^n_{i-1,j})}{\Delta x^2} \tag{17}$$

Similarly

$$\frac{\partial}{\partial y}(q\frac{\partial u}{\partial y}) = \frac{\frac{1}{2}(q_{i,j}+q_{i,j+1})(u^n_{i,j+1}-u^n_{i,j}) - \frac{1}{2}(q_{i,j}+q_{i,j-1})(u^n_{i,j}-u^n_{i,j-1})}{\Delta y^2} \tag{18}$$

Before implementation we gather all the terms to solve for $u^{n+1}_{i,j}$:

$$\begin{aligned}
u^{n+1}_{i,j} &= (1+\frac{\Delta t}{2}b)^{-1} * ((\frac{\Delta t b}{2}-1)u^{n-1}_{i,j} + 2u^n_{i,j} \\
&\quad + \frac{\Delta t^2}{\Delta x^2}\left(\frac{1}{2}(q_{i,j}+q_{i+1,j})(u^n_{i+1,j}-u^n_{i,j}) - \frac{1}{2}(q_{i,j}+q_{i-1,j})(u^n_{i,j}-u^n_{i-1,j})\right) \\
&\quad + \frac{\Delta t^2}{\Delta y^2}\left(\frac{1}{2}(q_{i,j}+q_{i,j+1})(u^n_{i,j+1}-u^n_{i,j}) - \frac{1}{2}(q_{i,j}+q_{i,j-1})(u^n_{i,j}-u^n_{i,j-1})\right)) \quad +\Delta t^2 f^n_{i,j}
\end{aligned}$$

When $n = 0$, a special formula is used for the the initial condition (9). The discretization of the initial condition is then:

$$\frac{\partial}{\partial t}u(x_i, y_j, t_n) \approx \frac{u^1_{i,j} - u^{-1}_{i,j}}{2\Delta t} = V_{i,j} \to u^{-1}_{i,j} = u^1_{i,j} - 2\Delta t V_{i,j}. \tag{19}$$

The modified scheme for the first step is found by inserting (9) into the main scheme:

$$\begin{aligned}
u^1_{i,j} &= u^0_{i,j} + ((\Delta t - \frac{\Delta t^2 b}{4})V_{i,j} \\
&\quad + \frac{\Delta t^2}{\Delta x^2}\left(\frac{1}{4}(q_{i,j}+q_{i+1,j})(u^0_{i+1,j}-u^0_{i,j}) - \frac{1}{4}(q_{i,j}+q_{i-1,j})(u^0_{i,j}-u^0_{i-1,j})\right) \\
&\quad + \frac{\Delta t^2}{\Delta y^2}\left(\frac{1}{4}(q_{i,j}+q_{i,j+1})(u^0_{i,j+1}-u^0_{i,j}) - \frac{1}{4}(q_{i,j}+q_{i,j-1})(u^0_{i,j}-u^0_{i,j-1})\right)) \quad +\Delta t^2 f^0_{i,j}
\end{aligned}$$

2.3 Discretization of derivatives at the boundary

Since the main equation is discretized by the centeral difference method, a way to implement the boundary conditions (4) should be devised. At $x = 0 (i = 0)$ and $y = 0 (j = 0)$ for $t = t_n$ the difference is written as

$$\frac{u^n_{-1,j} - u^n_{1,j}}{2\Delta x} = 0, \quad \rightarrow \quad u^n_{-1,j} = u^n_{1,j} \tag{20}$$

$$\frac{u^n_{i,-1} - u^n_{i,1}}{2\Delta y} = 0, \quad \rightarrow \quad u^n_{i,-1} = u^n_{i,1} \tag{21}$$

At the boundaries, the conditions $\frac{\partial q}{\partial x} = 0$ and $\frac{\partial q}{\partial y} = 0$ are imposed on variable coefficient q, to ease the implementation.The corresponding boundary conditions of the scheme for $x = L_x(i = Nx)$ and $y = L_y(j = N_y)$ are respectively

$$\frac{u^n_{N_x+1,j} - u^n_{N_x-1,j}}{2\Delta x} = 0, \quad \rightarrow \quad u^n_{N_x+1,j} = u^n_{N_x-1,j} \tag{22}$$

$$\frac{u^n_{i,N_y+1} - u^n_{i,N_y-1}}{2\Delta y} = 0, \quad \rightarrow \quad u^n_{i,N_y+1} = u^n_{i,N_y-1} \tag{23}$$

The Neumann boundary condition is implemented by updating ghost cells in the simulation. Part of the code looks as follows

```
def NeumannBC(self,u,Ix,Iy,version="scalar"):
        if version=="scalar":
            for j in range(0,Iy[-1]+1):
                i = Ix[0] # physical grid point 0
                u[i-1,j] = u[i+1,j]
                i = Ix[-1] # physical grid point Nx+1
                u[i+1,j] = u[i-1,j]

            for i in range(0,Ix[-1]+1):
                j = Iy[0] # physical grid point 0
                u[i,j-1] = u[i,j+1]
                j = Iy[-1] # physical grid point Ny+1
                u[i,j+1] = u[i,j-1]

        elif version=="vectorized":
            i = Ix[0]
            u[i-1,:] = u[i+1,:]
            i = Ix[-1]
            u[i+1,:] = u[i-1,:]
            j = Iy[0]
            u[:,j-1] = u[:,j+1]
            j = Iy[-1]
            u[:,j+1] = u[:,j-1]
        return u
```

3 Truncation Error

Suppose that we want to measure the closeness with which the polynomial $P_n(x)$ approximates the function $u(t)$. This closeness can then be measured by the difference

$$R_n(t) = u(t) - P_n(t)$$

This difference $R_n(x)$ is the nth-degree remainder for $u(t)$ at $t = a$. It is the error made if the value $u(t)$ is replaced with the approximation $P_n(t)$, and it is called the truncation error. More spesifically, we have for central difference approximation, equations on the form:

$$R^n = [D_k u]^n - u'(k_n)$$

The common way of calculating R^n in a centered difference scheme is to:

1. Expand $u(t_{n+\frac{1}{2}})$ and $u(t_{n+\frac{1}{2}})$ in a Taylor series arround the point t_n where the derivative is evaluated.
2. Insert this Taylor series in $R^n = [D_k u]^n - u'(k_n)$
3. Collect terms that cancel and simplify the expression

The result is an expression for R^n in terms of power series in Δt. The truncation error is the residual R in the equation and most of the terms needed in our equation i taken from main_trunc and been modified to fit our problem . Showing here term by term calculations.

$$[D_t D_t u_e + b D_{2t} u_e = D_x \bar{q}^x D_x u_e + D_y \bar{q}^y D_y u_e + f + R]^n_{i,j} \tag{24}$$

The first term on the left hand side (lhs)

$$[D_t D_t u_e]^n_{i,j} = \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} = u''(t_n) + R^n \tag{25}$$

$$R^n = \frac{1}{12} u^{(4)}(t_n)\Delta t^2 + O(\Delta t^4) \tag{26}$$

The second term on lhs

$$[D_{2t} u_e]^n_{i,j} = \frac{u^{n+1} - u^{n-1}}{2\Delta t} = u'(t_n) + R^n \tag{27}$$

$$R^n = \frac{1}{6} u'''(t_n)\Delta t^2 + O(\Delta t^4) \tag{28}$$

The x-term on rhs with variable coefficient

$$[D_x \bar{q}^x D_x u_e]^n_{i+\frac{1}{2},j} = \frac{1}{\Delta x}\left([\bar{q}^x D_x u_e]^n_{i+\frac{1}{2},j} - [\bar{q}^x D_x u_e]^n_{i-\frac{1}{2},j}\right) \tag{29}$$

$$[D_x u_e]^n_{i+\frac{1}{2},j} = u_{e,x}(x_{i+\frac{1}{2}}, y_j, t_n) + \frac{1}{24} u_{e,xxx}(x_{i+\frac{1}{2}}, y_j, t_n)\Delta x^2 + O(\Delta x^4) \tag{30}$$

$$[\bar{q}^x]_{i+\frac{1}{2},j} = q(x_{i+\frac{1}{2}},y_j) + \frac{1}{8}q''(x_{i+\frac{1}{2}},y_j)\Delta x^2 + O(\Delta x^4) \tag{31}$$

$$\begin{aligned}
[\bar{q}D_x u_e]^n_{i+\frac{1}{2},j} = &\left[q(x_{i+\frac{1}{2}},y_j) + \frac{1}{8}q''(x_{i+\frac{1}{2}},y_j)\Delta x^2 + O(\Delta x^4)\right]\cdot \\
&\left[u_{e,x}(x_{i+\frac{1}{2}},y_j,t_n) + \frac{1}{24}u_{e,xxx}(x_{i+\frac{1}{2}},y_j,t_n)\Delta x^2 + O(\Delta x^4)\right] \\
= &\, q(x_{i+\frac{1}{2}},y_j,t_n)u_{e,x}(x_{i+\frac{1}{2}},y_j,t_n) + q(x_{i+\frac{1}{2}},y_j)\frac{1}{24}u_{e,xxx}(x_{i+\frac{1}{2}},y_j,t_n)\Delta x^2 \\
&+ u_{e,x}(x_{i+\frac{1}{2}},y_j,t_n)\frac{1}{8}q''(x_{i+\frac{1}{2}},y_j)\Delta x^2 + O(\Delta x^4) \\
= &\, [qu_{e,x}]^n_{i+\frac{1}{2},j} + G^n_{i+\frac{1}{2},j}\Delta x^2 + O(\Delta x^4)
\end{aligned} \tag{32}$$

with introduction of the short form

$$G^n_{i+\frac{1}{2},j} = \left(\frac{1}{24}u_{e,xxx}(x_{i+\frac{1}{2}},y_j,t_n)q(x_{i+\frac{1}{2}},y_j) + u_{e,x}(x_{i+\frac{1}{2}},y_j,t_n)\frac{1}{8}q''(x_{i+\frac{1}{2}},y_j)\right)\Delta x^2.$$

Similary, we find that

$$[\bar{q}D_x u_e]^n_{i-\frac{1}{2},j} = [qu_{e,x}]^n_{i-\frac{1}{2},j} + G^n_{i-\frac{1}{2},j}\Delta x^2 + O(\Delta x^4) \tag{33}$$

Inserting the expression in equation (29)

$$\begin{aligned}
[D_x\bar{q}^x D_x u_e]^n_{i,j} &= \frac{1}{\Delta x}\left([\bar{q}^x D_x u_e]^n_{i+\frac{1}{2},j} - [\bar{q}^x D_x u_e]^n_{i-\frac{1}{2},j}\right) \\
&= \frac{1}{\Delta x}[qu_{e,x}]^n_{i+\frac{1}{2},j} + G^n_{i+\frac{1}{2},j}\Delta x^2 - [qu_{e,x}]^n_{i-\frac{1}{2},j} - G^n_{i-\frac{1}{2},j}\Delta x^2 + O(\Delta x^4)) \\
&= [D_x qu_{e,x}]^n_{i,j} + [D_x G]^n_{i,j}\Delta x^2 + O(\Delta x^4)
\end{aligned}$$

$$\begin{aligned}
[D_x qu_{e,x}]^n_{i,j} &= \frac{\partial}{\partial x}q(x_i,y_j)u_{e,x}(x_i,y_j,t_n) + \frac{1}{24}G_{xxxx}(x_i,y_j,t_n)\Delta x^2 + O(\Delta x^4) \\
[D_x G]^n_{i,j}\Delta x^2 &= G_x(x_i,y_j,t_n)\Delta x^2 + \frac{1}{24}G_{xxx}(x_i,y_j,t_n)\Delta x^4 + O(\Delta x^4) \\
[D_x\bar{q}^x D_x u_e]^n_{i,j} &= \frac{\partial}{\partial x}q(x_i,y_j)u_{e,x}(x_i,y_j,t_n) + O(\Delta x^2)
\end{aligned}$$

Similar procedure for the y-component follows. Writing out the result from equation (29)

$$\begin{aligned}
[D_y\bar{q}^y D_y u_y]^n_{i,j} &= \frac{1}{\Delta y}\left([\bar{q}^y D_y u_e]^n_{i,j+\frac{1}{2}} - [\bar{q}^y D_y u_e]^n_{i,j-\frac{1}{2}}\right) \\
&= \frac{1}{\Delta y}[qu_{e,y}]^n_{i+\frac{1}{2},j} + G^n_{i+\frac{1}{2},j}\Delta y^2 - [qu_{e,y}]^n_{i-\frac{1}{2},j} - G^n_{i-\frac{1}{2},j}\Delta y^2 + O(\Delta y^4)) \\
&= [D_y qu_{e,y}]^n_{i,j} + [D_y G]^n_{i,j}\Delta x^2 + O(\Delta y^4)
\end{aligned}$$

$$
\begin{aligned}
[D_y q u_{e,y}]^n_{i,j} =& \frac{\partial}{\partial y} q(x_i, y_j) u_{e,y}(x_i, y_j, t_n) + \frac{1}{24} G_{yyyy}(x_i, y_j, t_n)\Delta y^2 + O(\Delta y^2) \\
[D_y G]^n_{i,j} \Delta y^2 =& G_y(x_i, y_j, t_n)\Delta y^2 + \frac{1}{24} G_{yyy}(x_i, y_j, t_n)\Delta y^4 + O(\Delta y^4) \\
[D_y \bar{q}^y D_y u_e]^n_{i,j} =& \frac{\partial}{\partial y} q(x_i, y_j) u_{e,y}(x_i, y_j, t_n) + O(\Delta y^2)
\end{aligned}
$$

$\downarrow$

$$
\begin{aligned}
R^n_{i,j} =& \frac{1}{12} u_{e,tttt}(x_i, y_j, t_n)\Delta t^2 + O(\Delta t^4) + b\left(\frac{1}{6} u'''(t_n)\Delta t^2 + O(\Delta t^4)\right) \\
&+ \frac{\partial}{\partial x} q(x_i, y_j) u_{e,x}(x_i, y_j, t_n) + O(\Delta x^2) + \frac{\partial}{\partial y} q(x_i, y_j) u_{e,y}(x_i, y_j, t_n) + O(\Delta y^2) \\
&+ f(x_i, y_j, t_n) \\
=& \Delta t^2 \left[\left(\frac{1}{12} u_{e,tttt}(x_i, y_j, t_n)\right) + b\left(\frac{1}{6} u'''(t_n)\right)\right] \\
&+ \left(\frac{\partial}{\partial x} q(x_i, y_j) u_{e,x}(x_i, y_j, t_n)\right) + \left(\frac{\partial}{\partial y} q(x_i, y_j) u_{e,y}(x_i, y_j, t_n)\right) \\
&+ O(\Delta t^4, \Delta x^2, \Delta y^2) + f(x_i, y_j, t_n)
\end{aligned}
\tag{34}
$$

The error is in second order in space, that is $\Delta x^2, \Delta y^2$, and fourth order in time Δt^4.

4 Verification: Standing Undamped Waves

Neglecting damping and with constant wave velocity, we can assume standing wave solution without source term, on the form

$$
\begin{aligned}
u_e(x, y, t) =& A cos(k_x x) cos(k_y y) cos(\omega t) \\
k_x =& \frac{m_x \pi}{L_x} \\
k_y =& \frac{m_y \pi}{L_y}
\end{aligned}
\tag{35}
$$

for arbitrary amplitude A, arbitrary integers m_x and m_y, and a suitable choice of ω the solution can be used to test the convergence of the numerical method. Even though the truncation error is not the true error $e^n_{i,j} = u_e(x_i, y_j, t_n) - u^n_{i,j}$, we can assume that $e^n_{i,j}$ depends on th discretization parameters in the same way. We also assume that a norm $e^n_{i,j}$, e.g

$$
E = \|e^n_{i,j}\|_{l^\infty} = \max_i \max_j \max_t \mid e_{i,j} \mid
$$

Inserting equation (35) in equation (34) and assuming constant q, our truncation error takes the form

$$
\begin{aligned}
R_{i,j}^n =& \frac{1}{12}\left[\left(u_{e,tttt}(x_i,y_j,t_n)\Delta t^2\right) + \left(u_{e,xxxx}(x_i,y_j,t_n)\right) + \left(u_{e,yyyy}(x_i,y_j,t_n)\right)\right] \\
e_{i,j}^n = R_{i,j}^n =& \Delta t^2\left(\frac{1}{12}\omega^4 A cos(k_x x)cos(k_y y)cos(\omega t)\right) + \left(Ak_x^4 cos(k_x x)cos(k_y y)cos(\omega t)\right) \\
&+ \left(Ak_y^4 cos(k_x x)cos(k_y y)cos(\omega t)\right)
\end{aligned}
$$

The norm then takes the form

$$
E = \max_i \max_j \max_t \ \mid e_{i,j} \mid = \frac{1}{12}\left(\Delta t^2\omega^4 + q(\Delta x^2 k_x^4 + \Delta y^2 k_y^4)\right)
$$

and for h $= \Delta x = \Delta y$, $E = Ch^2$.

The following output has been taken from the implementation for the undamped standing wave

h	E/h^2	r
0.500000	0.572456	———
0.250000	1.006119	1.186437
0.125000	0.730571	2.461704
0.062500	0.713948	2.033207
0.031250	0.771668	1.887838
0.015625	0.732676	2.074804

We see that the convergence rate is of order 2, which is what we expected for our implementation. We also see that E/h^2 is almost a constant.

5 Verification: Standing Damped Waves

Finding the analytical solution of damped waves using an ansatz of the type

$$
\begin{aligned}
u_e(x,y,t) =& (A\cos(\omega t) + B\sin(\omega t))e^{-ct}\cos(k_x x)\cos(k_y y), \\
k_x =& \frac{m_x\pi}{L_x}, \quad k_y = \frac{m_y\pi}{L_y}.
\end{aligned} \tag{36}
$$

The aim is to find A, B, ω, and c such that (36) solves the PDE with constant q, no source term and initial condition $u_t(x,y,0) = 0$. The first derivative of (36) with respect to time is

$$
\frac{\partial u_e}{\partial t} = (-c(A\cos(\omega t)+B\sin(\omega t))+\omega(-A\sin(\omega t)+B\cos(\omega t)))e^{-ct}\cos(k_x x)\cos(k_y y). \tag{37}
$$

Imposing the initial condition on (37) gives the relation $B = \frac{c}{\omega}A$ and hence B can be eliminated from the equation. The second derivative of (36) with respect to time is

$$\frac{\partial^2 u_e}{\partial t^2} = (c^2(A\cos(\omega t) + B\sin(\omega t)) - 2c\omega(-A\sin(\omega t) \tag{38}$$

$$+ B\cos(\omega t)) - \omega^2(A\cos(\omega t) + B\sin(\omega t)))e^{-ct}\cos(k_x x)\cos(k_y y). \tag{39}$$

The second derivative of (36) with respect to x and y are respectively

$$u_e(x, y, t) = -k_x^2(A\cos(\omega t) + B\sin(\omega t))e^{-ct}\cos(k_x x)\cos(k_y y), \tag{40}$$

and

$$u_e(x, y, t) = -k_y^2(A\cos(\omega t) + B\sin(\omega t))e^{-ct}\cos(k_x x)\cos(k_y y). \tag{41}$$

Finally inserting (37), (39), (40) and (41) in to PDE (2) result in

$$\omega^2 = qk_x^2 + qk_y^2 - c^2. \tag{42}$$

The result in (42) is acheived by assuming that q is constant, with zero source term and using $c = b/2$. The final equation is then

$$u_e(x, y, t) = A(\cos(\omega t) + \frac{c}{\omega}\sin(\omega t))e^{-ct}\cos(k_x x)\cos(k_y y), \quad k_x = \frac{m_x \pi}{L_x}, \quad k_y = \frac{m_y \pi}{L_y}. \tag{43}$$

Here is the result from the simulation of damped wave

$h = \Delta x = \Delta y$	E/h^2	r
0.500000	1.319609	——
0.250000	1.240769	2.088876
0.125000	1.109111	2.161831
0.062500	1.104624	2.005848
0.031250	1.127713	1.970156
0.015625	1.117418	2.013230

Just like before, we have managed to achieve the correct convergence rate for this test. Again, we also see that E/h^2 is almost a constant.

6 Verification : Manufactured Solution

The variable q is choosen such that there will be a linear inclination in one direction while there is no variation in the y-direction, $q = x$. To find $f(x, y, t)$ the wave equation solution (36) is inserted into the PDE and solve for the source term $f(x, y, t) = u_{tt} + bu_t - (qu_x)_x - (qu_y)_y$. Using sympy, the reuslt becomes

$$
\begin{aligned}
f(x,y,t) = &((-Ac^2 + Ak_x^2 x + Ak_y^2 x - Aw^2)\cos(k_x x)\cos(\omega t) + Ak_x \sin(k_x x)\cos(\omega t) \\
&+ Bk_x \sin(k_x x)\sin(\omega t) + (-Bc^2 + Bk_x^2 x + Bk_y^2 x - Bw^2)\cos(k_x x)\sin(\omega t))e^{-ct}\cos(k_y y) \\
&- (Ac + B\omega)\cos(k_x x)\cos(k_y y)
\end{aligned}
$$

The initial conditions are

$$I(x,y) = A\cos(k_x x)\cos(k_y y) \tag{44}$$

and

$$V(x,y) = (-Ac + B\omega)\cos(k_x x)\cos(k_y y) \tag{45}$$

In python, the following is performed to find $f(x,y,t)$

```
from sympy import *

x = Symbol('x')
y = Symbol('y')
A = Symbol('A')
B = Symbol('B')
c = Symbol('c')
kx = Symbol('kx')
ky = Symbol('ky')
t = Symbol('t')
w = Symbol('w')
q = x
u = (A*cos(w*t) + B*sin(w*t))*exp(-c*t)*cos(kx*x)*cos(ky*y)
ut = diff(u, t)  #Derivative u_t
utt = diff(diff(u, t), t)  #Derivative u_tt
rxx = diff(q*diff(u, x), x)  #Derivative (q*u_x)_x
ryy = diff(q*diff(u, y), y)  #Derivative (q*u_y)_y
fxyt = utt + 2*c*ut - rxx - ryy
```

The rusult is

```
fxyt = (-A*c**2*cos(kx*x)*cos(t*w) + A*kx**2*x*cos(kx*x)*cos(t*w)
       +A*kx*sin(kx*x)*cos(t*w) + A*ky**2*x*cos(kx*x)*cos(t*w)
       -A*w**2*cos(kx*x)*cos(t*w) - B*c**2*sin(t*w)*cos(kx*x)
       +B*kx**2*x*sin(t*w)*cos(kx*x) + B*kx*sin(kx*x)*sin(t*w)
       +B*ky**2*x*sin(t*w)*cos(kx*x) - B*w**2*sin(t*w)*cos(kx*x))
       *exp(-c*t)*cos(ky*y) - A*c*cos(kx*x)*cos(ky*y) + B*w*cos(kx*x)*cos(ky*y).
```