INF5620

Autumn 2015

Finite difference simulation of 2D waves

John-Anders Stende

Date: October 11, 2015

Abstract

This project is a finite difference simulation of 2D waves with Neumann boundary conditions, damping and variable wave velocity. It is split into three parts: 1) Discretization of the wave equation and implementation of solver 2) Verification: Constant solution, 1d plug-wave, standing undamped waves, manufactured solution and convergence rate 3) Investigation of a specific physical problem

Discretization and implementation

To start with, I will present the complete mathematical problem and show how the equations can be discretized and implemented in a python script.

Mathematical problem

The project addresses the two-dimensional, standard, linear wave equation with damping

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

The associated Neumann boundary condition is

$$\frac{\partial u}{\partial n} = 0 \tag{2}$$

in a rectangular spatial domain $\Omega = [0, L_x] \times [0, L_y]$. The initial conditions are

$$u(x,y,0) = I(x,y) \tag{3}$$

$$u_t(x,y,0) = V(x,y) \tag{4}$$

Discretization

Using central differences for all derivates,

$$[D_t D_t u + bD_{2t} = D_x q^{-x} Dxu + D_y q^{-y} Dyu + f]^n_{i,j} \tag{5}$$

where q^{-x} is notation for the arithmetic mean of q.
This leads to the following equation for the interior spatial mesh points,

$$u^{n+1}_{i,j} = (1 + \frac{1}{2}b\Delta t)^{-1}\left[2u^n_{i,j} + u^{n-1}_{i,j}\left(\frac{1}{2}b\Delta t - 1\right) + \left(\frac{\Delta t}{\Delta x}\right)^2 dqdx + \left(\frac{\Delta t}{\Delta y}\right)^2 dqdy + \Delta t^2 f^n_{i,j}\right] \tag{6}$$

where

$$dqdx = [\frac{1}{2}(q_{i,j} + q_{i+1,j})(u^n_{i+1,j} - u^n_{i,j}) - \frac{1}{2}(q_{i-1,j} + q_{i,j})(u^n_{i,j} - u^n_{i-1,j})] \tag{7}$$

and

$$dqdy = [\frac{1}{2}(q_{i,j} + q_{i,j+1})(u^n_{i,j+1} - u^n_{i,j}) - \frac{1}{2}(q_{i,j-1} + q_{i,j})(u^n_{i,j} - u^n_{i,j-1})] \tag{8}$$

The discretized initial condition for u_t

$$D_{2t}u_{i,j} = \frac{u^1_{i,j} - u^{-1}_{i,j}}{2\Delta t} = V_{i,j} \tag{9}$$

yields

$$u^{-1}_{i,j} = u^1_{i,j} - 2\Delta t V_{i,j} \tag{10}$$

This equality is used to obtain the scheme for the first step ($n = 0$)

$$u^1_{i,j} = u^0_{i,j} - \frac{1}{2}\Delta t V_{i,j}(b\Delta t - 2) + \left(\frac{\Delta t}{\Delta x}\right)^2 dqdx + \left(\frac{\Delta t}{\Delta y}\right)^2 dqdy + \frac{1}{2}\Delta t^2 f^0_{N_x,j} \tag{11}$$

We have Neumann boundary condtions all along the rectangular spatial mesh described above. Assuming $\partial q/\partial x = \partial q/\partial y = 0$ at the boundaries, the scheme along the line $i = N_x, j = [0, N_y]$ becomes

$$u^{n+1}_{N_x,j} = - u^{n-1}_{N_x,j} + 2u^{n}_{N_x,j} + \left(\frac{\Delta t}{\Delta x}\right)^2 2q_{N_x-\frac{1}{2},j}(u^n_{N_x-1,j} - u^n_{N_x,j}) + \tag{12}$$

$$\left(\frac{\Delta t}{\Delta y}\right)^2 2q_{N_x,j-\frac{1}{2}}(u^n_{N_x,j-1} - u^n_{N_x,j}) + \Delta t^2 f^n_{N_x,j} \tag{13}$$

The corresponding equations for the other sides of the rectangular mesh is obtained by using the boundary relations

$$u^n_{i-1,j} = u^n_{i+1,j}, \qquad u^n_{i,j-1} = u^n_{i,j+1} \tag{14}$$

$$q_{i-\frac{1}{2},j} = q_{i+\frac{1}{2},j}, \qquad q_{i,j-\frac{1}{2}} = q_{i,j+\frac{1}{2}} \tag{15}$$

The boundary scheme (along the line $i = N_x, j = [0, N_y]$) for the first step is obtained in the same way as for the inner spatial points above,

$$u^1_{N_x,j} = - u^0_{N_x,j} + \frac{1}{2}\Delta t V_{N_x,j}(b\Delta t - 2) + \left(\frac{\Delta t}{\Delta x}\right)^2 q_{N_x-\frac{1}{2},j}(u^n_{N_x-1,j} - u^n_{N_x,j}) + \tag{16}$$

$$\left(\frac{\Delta t}{\Delta y}\right)^2 q_{N_x,j-\frac{1}{2}}(u^n_{N_x,j-1} - u^n_{N_x,j}) + \Delta t^2 f^n_{N_x,j} \tag{17}$$

Implementation

I have implmented both a scalar and a vectorized version of the above schemes in my solver. The equation for the inner spatial points can be reused at the boundaries if the indicies are changed, here shown for the line $i = N_x, j = [0, N_y]$

```
i = Ix[0]
ip1 = i+1
im1 = ip1
for j in Iy[1:-1]:
    dqdx = 0.5*(q[i,j] + q[ip1,j])*(u_1[ip1,j] - u_1[i,j]) - \
           0.5*(q[i,j] + q[im1,j])*(u_1[i,j] - u_1[im1,j])
    dqdy = 0.5*(q[i,j] + q[i,j+1])*(u_1[i,j+1] - u_1[i,j]) - \
           0.5*(q[i,j] + q[i,j-1])*(u_1[i,j] - u_1[i,j-1])
    u[i,j] = (1./(1+D))*(2*u_1[i,j] + u_2[i,j]*(D-1) +  \
              Cx2*dqdx + Cy2*dqdy + dt2*f(x[i], y[j], t[n]))
```

The vecorized version is

```
i = Ix[0]
ip1 = i+1
im1 = ip1
dqdx = 0.5*(q[i,1:-1] + q[ip1,1:-1])*(u_1[ip1,1:-1] - u_1[i,1:-1]) - \
       0.5*(q[i,1:-1] + q[im1,1:-1])*(u_1[i,1:-1] - u_1[im1,1:-1])
dqdy = 0.5*(q[i,1:-1] + q[i,2:])*(u_1[i,2:] - u_1[i,1:-1]) - \
       0.5*(q[i,1:-1] + q[i,:-2])*(u_1[i,1:-1] - u_1[i,:-2])
u[i,1:-1] = (1./(1+D))*(2*u_1[i,1:-1] + u_2[i,1:-1]*(D-1) +  \
            Cx2*dqdx + Cy2*dqdy + dt2*f_a[i,1:-1] )
```

Verification

It's important to implement methods that can verify that the code is correct. In this project I'm going to assert that the wave stays constant if the initial condition is a constant and that a wave-plug splits in two and meet again at the same spot it started from. I'm also going to find the convergence rate of the discretization for standing, undamped waves and and a manufactured solution.

Constant solution

$u(x, y, t) = c$ is a solution to the PDE problem if the source term f equals zero due to the fact that the derivative of a constant with respect to any variable is zero. To construct a test case, I thus set $I(x, y) = c$ and $V(x, y) = f(x, y, t) = 0$. Any value/expression can be chosen for b and $q(x, y)$ because they are both multiplied with zero when the constant solution is inserted in the PDE. This test case is implemented in the function *test_constant_solution*. Running a pytest verifies that u does indeed stay constant.

Standing, undamped waves