Project: Finite difference simulation of 2D waves. INF5620

Shafa Aria

October 13, 2014

1 Introduction

In this project we will discretize a 2D wave equation

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

and implement it in a python code and solve it numerically for various waves using Neumann boundary conditions, $\frac{\partial u}{\partial n} = 0$ in spatial domain $\Omega = [0, L_x] \times [0, L_y]$. The initial conditions being:

$$u(x,y,0) = I(x,y)$$

$$u_t(x,y,0) = V(x,y)$$

We will start off discritizing our PDE then using the discritized solution to implement it in a program using the `wave2D_u0.py` as a starting point. Later on we will go ahead with verification and investigating a physical problem.

2 Discretization

We will in this section proceed to discritize the above equation to be implemented numerically. We start off by discritizing the LHS and then the RHS. Starting off taking the first term:

$$\frac{\partial^2 u}{\partial t^2} = \frac{u_{i,j}^{n+1} - 2u_{i,j}^{n} + u_{i,j}^{n-1}}{\Delta t^2}$$

second term:

$$b\frac{\partial u}{\partial t} = b\frac{u_{i,j}^{n+1} - u_{i,j}^{n-1}}{2\Delta t}$$

Where the b is the damping constant. Moving on to the RHS, we can find a nice discritized version of the first and second term in the lecture notes for 1D, but we can expand it to 2D:

$$\left[\frac{\partial}{\partial x}\left(q(x,y)\frac{\partial u}{\partial x}\right)\right]_i^n \approx \frac{1}{\Delta x^2}\left(q_{i+\frac{1}{2},j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-\frac{1}{2},j}(u_{i,j}^n - u_{i-1,j}^n)\right)$$

Doing the same for the y direction:

$$\left[\frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right)\right]_j^n \approx \frac{1}{\Delta y^2}\left(q_{i,j+\frac{1}{2}}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-\frac{1}{2}}(u_{i,j}^n - u_{i,j-1}^n)\right)$$

We will from here by denote the above two equations as u_{xx} and u_{yy} respectively. Adding together the LHS and RHS and solving for $u_{i,j}^{n+1}$ results in (after some algebra):

$$u_{i,j}^{n+1} = \frac{1}{2 + b\Delta t}\left[(u_{xx} + u_{yy} + f(x,y,t))2\Delta t^2 + u_{i,j}^{n-1}(b\Delta t - 2) + 4u_{i,j}^n\right]$$

$$u_{i,j}^{n+1} = \frac{1}{1 + \frac{b\Delta t}{2}}\left[(u_{xx} + u_{yy} + f(x,y,t))\Delta t^2 + u_{i,j}^{n-1}(\frac{1}{2}b\Delta t - 1) + 2u_{i,j}^n\right] \quad (1)$$

Lastly we note the term $q_{i+\frac{1}{2},j}$ and $q_{i-\frac{1}{2},j}$ in the equation u_{xx}.We would need to take the arithmetic mean of this for our scheme and thus retain: $q_{i+\frac{1}{2},j} = \frac{1}{2}(q_{i,j} + q_{i+1,j})$ and $q_{i-\frac{1}{2},j} = \frac{1}{2}(q_{i,j} + q_{i-1,j})$ respectively. Same goes for the y component where we use the j indice instead of the i.

Now we need an equation for the first time step. Starting with $n = 0$ we have the relation

$$[D_{2t}u = V]^0_{i,j}$$
$$u^{-1}_{i,j} = u^1_{i,j} - 2\Delta t V_{i,j}$$

Inserting the above into equation (1) and solving for $u^1_{i,j}$:

$$u^1_{i,j} = \frac{1}{1+\frac{b\Delta t}{2}}\left[(u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 + (u^1_{i,j} - 2\Delta t V_{i,j})(\frac{1}{2}b\Delta t - 1) + 2u^0_{i,j}\right]$$
$$u^1_{i,j} = \frac{1}{1+\frac{b\Delta t}{2}}\left[(u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 + u^1_{i,j}\frac{1}{2}b\Delta t - u^1_{i,j} - b\Delta t^2 V_{i,j} + 2\Delta t V_{i,j} + 2u^0_{i,j}\right]$$
$$u^1_{i,j}\left(1+\frac{b\Delta t}{2}\right) = (u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 + u^1_{i,j}\frac{1}{2}b\Delta t - u^1_{i,j} - b\Delta t^2 V_{i,j} + 2\Delta t V_{i,j} + 2u^0_{i,j}$$
$$u^1_{i,j} + u^1_{i,j}\frac{b\Delta t}{2} - u^1_{i,j}\frac{1}{2}b\Delta t + u^1_{i,j} = (u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 - b\Delta t^2 V_{i,j} + 2\Delta t V_{i,j} + 2u^0_{i,j}$$
$$2u^1_{i,j} = (u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 - b\Delta t^2 V_{i,j} + 2\Delta t V_{i,j} + 2u^0_{i,j}$$
$$u^1_{i,j} = \frac{1}{2}\left[(u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 - \Delta t V_{i,j}(b\Delta t - 2) + 2u^0_{i,j}\right]$$

With the final result:

$$u^{n+1}_{i,j} = \frac{1}{2}\left[(u_{xx}+u_{yy}+f(x,y,t))\Delta t^2 - \Delta t V_{i,j}(b\Delta t - 2) + 2u^n_{i,j}\right]$$

3 Implementation

We will use the file `wave2D_u0.py` as our starting point and removing most of the useless stuff. The important thing here is to change first the boundary conditions from Dirichlet to Neumann, by using ghost cells. We then write the discretized equations for our wave equation both in scalar and vectorized format.

We will not go into details, but will point out the vital parts of the code. The ghost cells:

```
i = Ix[0];                j = Iy[0]
u[i-1,:] = u[i+1,:];      u[:,j-1] = u[:,j+1]

i = Ix[-1];               j = Iy[-1]
u[i+1,:] = u[i-1,:];      u[:,j+1] = u[:,j-1]
```

The scalar loop for the wave equation for the subsequent steps:

```
for i in Ix[:]:
            for j in Iy[:]:
                u_xx = Cx*((q[i,j]+q[i+1,j])*(u_1[i+1,j]-u_1[i,j])\
                -(q[i,j]+q[i-1,j])*(u_1[i,j]-u_1[i-1,j]))
                u_yy = Cy*((q[i,j]+q[i,j+1])*(u_1[i,j+1]-u_1[i,j])\
                -(q[i,j]+q[i,j-1])*(u_1[i,j]-u_1[i,j-1]))
                u[i,j] = 1.0/(1+0.5*b*dt)*((u_xx+u_yy+f(x[i-Ix[0]],y[j-Iy[0]],t[n]))\
                *dt2 + u_2[i,j]*(0.5*(b*dt)-1.0) + 2.0*u_1[i,j])
```

For the rest of the details please refer to references.

4 References

1. https://github.com/Ace-/INF5620-Shafa