Finite difference simulation of 2D waves

Compulsory project in INF5620 by Florian Arbes

September 2019

Introduction

In this project I tried to implement a simulation of a two dimensional wave using the finite difference methods. Then using the simulation to study the behavior of waves as they pass through different medium with different velocities.

The core parts of the project

Discretization of the PDE

The following PDE is addressed in this project:

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

The boundary condition is given as:

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

with initial conditions:

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

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

The parts of the equation can be discretized as following:

$$\frac{\partial^2 u}{\partial t^2} = \frac{u_{i,j}^{n+1} - 2u_{i,j}^{n} + u_{i,j}^{n-1}}{\Delta t^2} \tag{5}$$

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n-1}}{2\Delta t} \tag{6}$$

$$\frac{\partial}{\partial x}(q(x,y)\frac{\partial u}{\partial x}) = \frac{1}{\Delta x^2}[q_{i+.5,j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-.5,j}(u_{i,j}^n - u_{i-1,j}^n)] \tag{7}$$

$$\frac{\partial}{\partial y}(q(x,y)\frac{\partial u}{\partial y}) = \frac{1}{\Delta y^2}[q_{i,j+.5}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-.5}(u_{i,j}^n - u_{i,j-1}^n)] \tag{8}$$

These equations can be plugged into 1. I used SymPy to find $u_{i,j}^{n+1}$:

```
t1 = ((b*dt-2)*u_nm1[i, j] +
      2*dt**2*f(dx*i, dy*j, t_1) +
      4*u_n[i, j])
t2 = dtdx2*(- q(dx*(i-.5), dy*j)*u_n[i, j] +
             q(dx*(i-.5), dy*j)*u_n[im1, j] -
             q(dx*(i+.5), dy*j)*u_n[i, j] +
             q(dx*(i+.5), dy*j)*u_n[ip1, j])
t3 = dtdy2*(- q(dx*i, dy*(j-.5))*u_n[i, j] +
             q(dx*i, dy*(j-.5))*u_n[i, jm1] -
             q(dx*i, dy*(j+.5))*u_n[i, j] +
             q(dx*i, dy*(j+.5))*u_n[i, jp1])
u[i, j, n+1] = 1/(b*dt + 2)*(t1 + 2*t2 + 2*t3)

```

This means:

$$\begin{aligned} u_{i,j}^{n+1} = \frac{1}{b\Delta t + 2}(\\ & 2\frac{\Delta t^2}{\Delta x^2}[q_{i+.5,j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-.5,j}(u_{i,j}^n - u_{i-1,j}^n)] + \\ & 2\frac{\Delta t^2}{\Delta y^2}[q_{i,j+.5}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-.5}(u_{i,j}^n - u_{i,j-1}^n)] + \\ & 2\Delta t^2 f_{i,j}^n + b\Delta t u_{i,j}^{n-1} + 4u_{i,j}^n - 2u_{i,j}^{n-1}) \end{aligned} \tag{9}$$

Using the discretised initial condition, a special formula for the first step can be derived:

```
t1 = (2*dt - b*dt**2)*V(i, j) + \
    dt**2*f(dx*i, dy*j, 0) + \
    2*u_n[i, j]
t2 = dtdx2*(- q(dx*(i-.5), dy*j)*u_n[i, j] +
             q(dx*(i-.5), dy*j)*u_n[im1, j] -
             q(dx*(i+.5), dy*j)*u_n[i, j] +
             q(dx*(i+.5), dy*j)*u_n[ip1, j])
t3 = dtdy2*(- q(dx*i, dy*(j-.5))*u_n[i, j] +
             q(dx*i, dy*(j-.5))*u_n[i, jm1] -
             q(dx*i, dy*(j+.5))*u_n[i, j] +
             q(dx*i, dy*(j+.5))*u_n[i, jp1])
u[i, j, 1] = 0.5 * (t1 + t2 + t3)

```

At the boundary points, the scheme has to be modified. This was done with the Neumann conditions and modifying indices:

- $u_{i-1,j}^n = u_{i+1,j}^n; i = 0$
- $u_{i+1,j}^n = u_{i-2,j}^n; i = N_x$
- $u_{i,j-1}^n = u_{i,j+1}^n; j = 0$
- $u_{i,j+1}^n = u_{i,j-1}^n; j = N_y$

Implementation

The scheme is implemented in the functions `scheme_ijn` and `scheme_ij1` in the file `wave2D.py`. The vectorized version is quite simple, as it can be achieved with index lists and ”advanced indexing” (see: https://docs.scipy.org/doc/numpy-1.17.0/reference/arrays.indexing.html#advanced-indexing)

Verification

Constant solution

Let $u(x,y,t) = c$ be the exact solution.
This means $\frac{\partial u}{\partial t} = 0$ and $\frac{\partial^2 u}{\partial t^2} = 0$. Therefore $\frac{\partial}{\partial x}(q(x,y)\frac{\partial u}{\partial x}) = \frac{\partial}{\partial y}(q(x,y)\frac{\partial u}{\partial y}) = 0$.
The remaining term $f$in the wave equation must be 0 as well. $q(x,y)$ could be any arbitrary function.
The constant solution is also a solution of the discrete equations:

$$\begin{aligned} u_{i,j}^{n+1} = \frac{1}{b\Delta t + 2}\{ & \\ & 2\frac{\Delta t^2}{\Delta x^2}[q_{i+.5,j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-.5,j}(u_{i,j}^n - u_{i-1,j}^n)] + \\ & 2\frac{\Delta t^2}{\Delta y^2}[q_{i,j+.5}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-.5}(u_{i,j}^n - u_{i,j-1}^n)] + \\ & 2\Delta t^2 f_{i,j}^n + b\Delta t u_{i,j}^{n-1} + 4u_{i,j}^n - 2u_{i,j}^{n-1}\} \end{aligned} \tag{10}$$

As $u_{i,j,n}^n = c$:

$$\begin{aligned} u_{i,j}^{n+1} = \frac{1}{b\Delta t + 2}\{ & \\ & 2\frac{\Delta t^2}{\Delta x^2}[q_{i+.5,j}(c-c) - q_{i-.5,j}(c-c)] + \\ & 2\frac{\Delta t^2}{\Delta y^2}[q_{i,j+.5}(c-c) - q_{i,j-.5}(c-c)] + \\ & 2\Delta t^2 f_{i,j}^n + b\Delta t c + 4c - 2c\} \end{aligned} \tag{11}$$

As $f_{i,j,n}^n = 0$:

$$u_{i,j}^{n+1} = \frac{1}{b\Delta t + 2}(2\Delta t^2 0 + b\Delta t c + 4c - 2c)$$

$$u_{i,j}^{n+1} = \frac{1}{b\Delta t + 2}(b\Delta t c + 2c) = c$$

This was implemented. Please run `nosetests test_3_1()`

Possible bugs are:

- Arguments of f in the wrong order. Test passes.
- In the first step, I(i, j) is called instead of V(i, j). Test failes.
- Wrong formula for the initial condition. $u^{-1}_{i,j} = u^{1}_{i,j} - 2\Delta t u^{0}_{i,j}$ rather than $u^{-1}_{i,j} = u^{1}_{i,j} - 2\Delta t V_{i,j}$. Test failes.
- Initial condition wrong. Test failes.
- Boundary conditions on the left side not implemented. Test passes.
- Boundary conditions on the right side not implemented. Test fails.

Exact 1D plug-wave solution in 2D

The `pulse()` function was adjusted and implemented. Please run
`nosetests pulse(Nx=100, Ny=0, pulse_tp='plug', T=15, medium=[-1, -1])`
or
`nosetests pulse(Nx=0, Ny=100, pulse_tp='plug', T=15, medium=[-1, -1])`.
You might want to adjust the speed of the visualization. The delay between the frames is specified in ms on top of the file. Every time step, exactly 4 cells change value. If the wave is at the boundary, only two cells change value.

Standing, undamped waves

The exact solution of the PDE is given as

$$u_e(x, y, t) = Acos(k_x x)cos(k_y y)cos(\omega t), k_x = \frac{m_x \pi}{L_x}, k_y = \frac{m_y \pi}{L_y}$$

c should be constant, $f(x, y)$, $I(x, y)$, $V(x, y)$ are determined using SymPy:

$$I(x, y) = Acos(\frac{m_x \pi}{L_x} x)cos(\frac{m_y \pi}{L_y} y)$$

$$V(x, y) = 0.0$$

$$q(x, y) = c^2$$

$$f(x, y) = A(-L_x^2 L_y^2 w(bsin(t\omega) + \omega cos(t\omega)) + \pi^2 Lx^2 c^2 m_y^2 cos(t\omega) + \pi^2 Ly^2 c^2 m_x^2 cos(t\omega))\frac{cos(k_x x)cos(k_y y)}{(L_x^2 L_y^2)}$$

In 2D $C = c\frac{\Delta t^2}{\Delta x^2} + c\frac{\Delta t^2}{\Delta x^2}$, which means:

$$\Delta t = \frac{C}{c}\frac{1}{\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta x^2}}}$$

A common discretization parameter h is introduced, such that $h = \delta x$. For the sake of simplicity I set $\Delta x = \Delta y$. This means Δt is also proportional to the common discretization parameter h:

$$\Delta t = \frac{C}{c}\frac{1}{\sqrt{2}}h$$

This leads to the simple error model

$$E = \hat{C}h^r$$

From two consecutive experiments with different h, the convergence rate r can be computed as

$$r = \frac{log\frac{E_2}{E_1}}{log\frac{h_2}{h_1}}$$

where E_1 and E_2 are the computed errors from the experiments. I computed the following error:

$$E = \sqrt{\Delta x \Delta y \Delta t \sum (u\_e_{i,j,t} - u_{i,j,t})^2}$$

In the experiment, the following parameters were set:
$A = 2.3$, $m_x = 3$, $m_y = 4$, $w = \pi$, $c = 1.0$, $C = 1.0$, $b = 1.0$, $Lx = 10$, $Ly = 10$, $T = 2\frac{10}{\sqrt{2}}$. The experiments show a convergence rate of $r = 2$ if $\Delta x < 1$ and thus $N_x > 10$:

```
E =  33.29835815362203
E1 = 502.6910, E2 = 33.2984
h1 = 2.0000, h2 = 1.0000
dx1 = 2.0000, dx2 = 1.0000
dy1 = 2.0000, dy2 = 1.0000
dt1 = 1.4142, dt2 = 0.7071
convergence rate:  3.916149031955887

E =  5.621684081060024
E1 = 33.2984, E2 = 5.6217
h1 = 1.0000, h2 = 0.5000
dx1 = 1.0000, dx2 = 0.5000
dy1 = 1.0000, dy2 = 0.5000
dt1 = 0.7071, dt2 = 0.3536
convergence rate:  2.5663767571978973

E =  1.2511877561621625
E1 = 5.6217, E2 = 1.2512
h1 = 0.5000, h2 = 0.2500
```

```
dx1 = 0.5000, dx2 = 0.2500
dy1 = 0.5000, dy2 = 0.2500
dt1 = 0.3536, dt2 = 0.1768
convergence rate:  2.1677040816493616

E =  0.299841717263914
E1 = 1.2512, E2 = 0.2998
h1 = 0.2500, h2 = 0.1250
dx1 = 0.2500, dx2 = 0.1250
dy1 = 0.2500, dy2 = 0.1250
dt1 = 0.1768, dt2 = 0.0884
convergence rate:  2.061025274043297

E =  0.07366495802300047
E1 = 0.2998, E2 = 0.0737
h1 = 0.1250, h2 = 0.0625
dx1 = 0.1250, dx2 = 0.0625
dy1 = 0.1250, dy2 = 0.0625
dt1 = 0.0884, dt2 = 0.0442
convergence rate:  2.025150714520014

E =  0.018273090079556038
E1 = 0.0737, E2 = 0.0183
h1 = 0.0625, h2 = 0.0312
dx1 = 0.0625, dx2 = 0.0312
dy1 = 0.0625, dy2 = 0.0312
dt1 = 0.0442, dt2 = 0.0221
convergence rate:  2.0112578789296505

```

Manufactured solution

The exact solution of the PDE is given as

$$u_e(x,y,t) = Acos(k_x x)cos(k_y y)cos(\omega t), k_x = \frac{m_x \pi}{L_x}, k_y = \frac{m_y \pi}{L_y}$$

the wave velocity q should be variable, $f(x,y)$, $I(x,y)$, $V(x,y)$ are determined using SymPy. $q(x,y)$ was chosen in a way, that f(x,y,t) would be simple.

$$q(x,y) = \frac{1}{sin(k_x x)}\frac{1}{sin(k_y y)}$$

$f(x,y) =$

$$(A+B)(-b(ccos(t\omega) + \omega sin(t\omega)) + c^2 cos(t\omega) + 2c\omega sin(t\omega) - \omega^2 cos(t\omega))$$
$$e^{-ct} cos(k_x x)cos(k_y y) \tag{12}$$

Using SymPy I got the following results:

$$q(x,y) = c^2$$

$$I(x,y) = (A+B) * cos(\pi * m_x * x/L_x) * cos(\pi * m_y * y/L_y)$$
$$V(x,y) = -c * (A+B) * cos(\pi * m_x * x/L_x) * cos(\pi * m_y * y/L_y)$$

However, i couldn't find any values, in order to get a stable numerical solution. Therefore I used the equations from the previous task:

$$q(x,y) = k$$
$$\omega = \sqrt{k_x^2 + k_y^2 - c^2}$$
$$c = b/2$$
$$u_t(x,y,0) = 0$$
$$f(x,y,t) = 0$$

With SymPy i found a equation for b:

$$b = \sqrt{2kk_x^2 + 2kk_y^2}$$

After three experiments, the convergence rate was found to be 2:

```
E1 = 0.1457, E2 = 0.0328
h1 = 0.5000, h2 = 0.2500
dx1 = 0.5000, dx2 = 0.2500
dy1 = 0.5000, dy2 = 0.2500
dt1 = 0.3536, dt2 = 0.1768
convergence rate:  2.1508650974923853

E1 = 0.0328, E2 = 0.0078
h1 = 0.2500, h2 = 0.1250
dx1 = 0.2500, dx2 = 0.1250
dy1 = 0.2500, dy2 = 0.1250
dt1 = 0.1768, dt2 = 0.0884
convergence rate:  2.0731804640721454

E1 = 0.0078, E2 = 0.0019
h1 = 0.1250, h2 = 0.0625
dx1 = 0.1250, dx2 = 0.0625
dy1 = 0.1250, dy2 = 0.0625
dt1 = 0.0884, dt2 = 0.0442
convergence rate:  2.036265823276573

```