INF5620
Project 1

Sebastian Gjertsen

October 11, 2015

In this project we use the linear, two-dimensional wave equation:

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

Our general scheme for internal points is:

$$K = \frac{\Delta t * b}{2}$$

$$u_{i,j}^{n+1} = \frac{K-1}{K+1}u_{i,j}^{n-1} + (\frac{2}{1+K}) * u_{i,j}^{n}$$

$$0.5(\frac{\frac{\Delta t^2}{\Delta x^2}}{K+1}(q_{i+1,j} + q_{i,j})(u_{i+1,j}^n - u_i^n) - (q_{i,j} + q_{i-1,j})(u_{i,j}^n - u_{i+1,j}^n)) +$$

$$0.5(\frac{\frac{\Delta t^2}{\Delta y^2}}{K+1}(q_{i,j+1} + q_j)(u_{i,(j+1)}^n - u_{i,j}^n) - (q_{i,j} + q_{i,j-1})(u_{i,j}^n - u_{i,j+1}^n)) + f_{i,j}^n$$

I have used ghost points so i only need to update the ghost point at each edge with the line in from the edge. Where 0 and N are the ghost points in each respective end.

$$u_{i,0} = u_{i,2}$$
$$u_{i,N} = u_{i,N-2}$$
$$u_{0,j} = u_{2,j}$$
$$u_{N,j} = u_{N-2,j}$$

Since we have a second order equation, we need to calculate the first time step using $\frac{\partial u}{\partial t} = 0@t = 0$ By using this with a centred difference we get: $u_{i,j}^{-1} = u_{i,j}^1 - 2\delta t V$ This can be put in to our scheme giving the first step scheme:

$$u_{i,j}^1 = u_{i,j}^0 + dt * V(2 + deltatb)$$

$$\frac{\Delta x^2}{4\Delta t^2}((q_{i+1,j} + q_{i,j})(u_{i+1,j}^0 - u_i^0) - (q_{i,j} + q_{i-1,j})(u_{i,j}^0 - u_{i+1,j}^0)) +$$

$$\frac{\Delta y^2}{4\Delta t^2}((q_{i,j+1} + q_j)(u_{i,(j+1)}^0 - u_{i,j}^0) - (q_{i,j} + q_{i,j-1})(u_{i,j}^0 - u_{i,j+1}^0)) + \frac{1}{2}f_{i,j}^n$$

Using the the same approach as above with the ghost points.

3

3.1 Constant Solution

Here I have used the constant solution of $u = 2$, as we see from the print out of the error, the numerical solution is correct: we see that we need f to be zero.
Terminal

```
[fenics@ubuntu64-box] ../INF5620/Wave_project > python Project_1.py
Error in constant solution: 0.0000 with dt = 0.1000 and dx = dy = 1.0000
Error in constant solution: 0.0000 with dt = 0.0500 and dx = dy = 0.5000
Error in constant solution: 0.0000 with dt = 0.0250 and dx = dy = 0.2500
Error in constant solution: 0.0000 with dt = 0.0125 and dx = dy = 0.1250
Error in constant solution: 0.0000 with dt = 0.0063 and dx = dy = 0.0625
Error in constant solution: 0.0000 with dt = 0.0031 and dx = dy = 0.0312
Error in constant solution: 0.0000 with dt = 0.0016 and dx = dy = 0.0156
Error in constant solution: 0.0000 with dt = 0.0008 and dx = dy = 0.0078
```

When changing V to be non zero, we get errors in the solver. Terminal

```
Error in constant solution: 1.8757 with dt = 0.1000 and dx = dy = 1.0000
Error in constant solution: 1.8024 with dt = 0.0500 and dx = dy = 0.5000
Error in constant solution: 1.7658 with dt = 0.0250 and dx = dy = 0.2500
Error in constant solution: 1.7476 with dt = 0.0125 and dx = dy = 0.1250
Error in constant solution: 1.7384 with dt = 0.0063 and dx = dy = 0.0625
Error in constant solution: 1.7339 with dt = 0.0031 and dx = dy = 0.0312
Error in constant solution: 1.7316 with dt = 0.0016 and dx = dy = 0.0156
Error in constant solution: 1.7305 with dt = 0.0008 and dx = dy = 0.0078
```

3.4

Here I have inserted the exact solution into the pde, to get $\omega = \sqrt{q * (k_x^2 + k_y^2)}$. In the gif standing wave, I have plotted the numerical and exact solution on top of eachother, and as we can see there is no difference so this tells me that the solver is accurate.
Terminal

```
[fenics@ubuntu64-box] ../INF5620/Wave_project > python Project_1.py
Error is  1.80901699437494767330747 44 with dt: 0.100000, dx:1.0000 , dy:1.0000
Error is  1.97075324492716896784827 44 with dt: 0.050000, dx:0.5000 , dy:0.5000
Error is  1.06298606600204181837909 80 with dt: 0.025000, dx:0.2500 , dy:0.2500
Error is  0.27289557172939582496695 72 with dt: 0.012500, dx:0.1250 , dy:0.1250
Error is  0.06817983559487152123157 47 with dt: 0.006250, dx:0.0625 , dy:0.0625
Error is  0.01703060515442979261369 06 with dt: 0.003125, dx:0.0312 , dy:0.0312
Error is  0.00425694910017332395213 43 with dt: 0.001563, dx:0.0156 , dy:0.0156
The convergence rate: ..
[-0.12354118892171646, 0.89062446436077825,
1.9617017963021701, 2.0009319584616945,
2.0012154209330246, 2.000237954508191]
As we can see the convergence tends toward 2 which again assures
me that the solver is correct.
```

4

In this section I have modelled different ocean floors and look at how it effects the wave. I have used more or less the same wave for every run. The first one is a simulation of a beach, found in

$beach_1.gif$ The red is the seafloor and the blue is the wave. We see that it acts the way

we would expect a wave to hit a beach. It increases in amplitude as it hits the beach and gets pulled down a bit as it moves out. The next simulation was of a wide ridge in the middle of the domain,$ridge_1.gif$. We see that the wave gets pulled down a bit from ocean floor, as expected. Now we see in $ridge_2.gif$, that when we increase the height of the ridge and increase the amplitude we get some errors in the solution. Its not too bad, but it indicates at what capacity our solver starts making errors. In the $peak_smooth.gif$ with a little dampning, the effects look natural and we get a pull from the geometry on the wave when it moves over. With the $peak_crash.gif$ we see a that if we choose a peak thats too high the solver crashes. In the $rough_mesh.gif$, and $super_rough_mesh.gif$ we see that the solver delivers a good solution even with a coarse mesh.