PH4606 - Lecture 4

Claus-Dieter Ohl

February 16, 2016

1 Spherical Waves

In spherical coordinates (r, ϕ, θ) with

$$\begin{aligned} x &= r\sin\theta\cos\phi & (1)\\ y &= r\sin\theta\sin\phi & (2)\\ z &= r\cos\theta & (3) \end{aligned}$$

the Laplacian operator is

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2} \tag{4}$$

If we assume spherical symmetry the the acoustic pressure is a function of r only and the Laplacian simplifies to

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} \quad . \tag{5}$$

Thus the wave equation becomes

$$\frac{\partial^2 p}{\partial r^2} + \frac{2}{r}\frac{\partial p}{\partial r} = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} \quad . \tag{6}$$

We can rewrite Eq. (6) as

$$\frac{\partial^2 (pr)}{\partial r^2} = \frac{1}{c^2}\frac{\partial^2 (rp)}{\partial t^2} \quad , \tag{7}$$

which resembles a plane wave equation for the product of the variable p and r. Thus the solution to this equation is

$$pr = f_1(ct - r) + f_2(ct + r) \tag{8}$$

or

$$p = \frac{1}{r}f_1(ct - r) + \frac{1}{r}f_2(ct + r) \quad , \tag{9}$$

for all $r > 0$, the solution is not valid at $r = 0$. Equation (9) is an outgoing and incoming wave where the amplitude is proportional to $1/r$. The incoming wave diverges at $r = 0$. The reason why our equation becomes invalid is that the small amplitude limit is not valid anymore, nonlinearities build up and we need a better description which accounts for finite amplitude effects.

```
%matplotlib notebook
import math as m
import numpy #array operations
import matplotlib.pyplot as plt #plotting
from IPython import display #for continous display
from ipywidgets import widgets #for the widgets

nimg=0

def plotwave(u,time,r):
  plt.figure(1)
  plt.clf()
  plt.plot(r,u)
  plt.text(0.1,3.4,"time {0:.5f}".format(time)) #annotate the time
  plt.xlabel(r"r")
  plt.gca().set_ylim([-4,4])
  display.clear_output(wait=True)
  display.display(plt.gcf())

def solvewave(b):
  tabs.visible=False
  #computational domain
  nx = 381
  size=2. #size of the domain
  #parameters of the wave
  c = 5. #speed of sound
  l=w_wavelength.value #wavelength
  nu=c/l #frequency
  omega=nu*2.*m.pi #angular frequency
  duration=w_sourceduration.value/nu #duration of source

  #further variables
  dx = size/(nx-1)
  CFL=0.1 #CFL number <1
  dt = CFL*dx/c
  nt=int(w_simduration.value/dt) #number of time steps
  if w_position.value=='Left':
      sourcepos=1
      ampl=100.
  else:
      sourcepos=int(nx/2)
      ampl=1.
  r=numpy.arange(dx,nx*dx,dx) #radius

  #every xx times over the total nt timesteps an output should be generated
  output=map(int,list(numpy.linspace(1,nt,int(nt/50))))

  u  = numpy.zeros(nx) #pressure at t
  un = numpy.zeros(nx) #pressure at t-dt
  unn= numpy.zeros(nx) #pressure at t-2*dt
  C=c*c*dt*dt/dx/dx

  #Assign initial conditions, for a sin^2 wave
  if w_source_type.value=="Time Dependent":
```

```
    un[sourcepos]=0.    #amplitude is sin(omega*t) with t=0
    unn[sourcepos]=1.*r[sourcepos]   #velocity  is cos(omega*t) with t=0
else: #or set it as an initial value
    for xx in range(nx):
        x=float(xx)*dx-size/2. #0 at the center
        unn[xx]=(m.cos(2.*m.pi/l*x)**2)*float(abs(x)<(c*duration/2.))
    for xx in range(1,nx-1): #calculate t=0 time step
        un[xx] = unn[xx] - 0.5*C*(unn[xx+1]-2.*unn[xx]+unn[xx-1])

plt.figure(1, figsize=(8, 4), dpi=300)

for n in range(nt+1): ##loop across number of time steps
    #this line computes the finite differences of the wave equation
    u[1:-1]=2.*un[1:-1]-unn[1:-1]+C*(un[:-2]+un[2:]-2.*un[1:-1])

    #Boundary conditions right
    if w_boundary_r.value=='Open':
        u[-1] = un[-1]-dt*c/dx*(un[-1]-un[-2])
    else:
        u[-1] = u[-2]
    #Boundary conditions right
    u[0] = u[1]

    #pressure source

    if w_source_type.value=="Time Dependent":
        if float(n*dt<duration):
            u[sourcepos]=ampl*m.sin(omega*n*dt)**2*float(n*dt<duration)*r[sourcepos]

    #save values for the time derivative
    unn=un.copy() #n-1 time step
    un=u.copy()   #n time step

    if (n in output):
        plotwave(u[1:]/r,n*dt,r)

#and plot the last figure
plotwave(u[1:]/r,n*dt,r)
tabs.visible=True
```

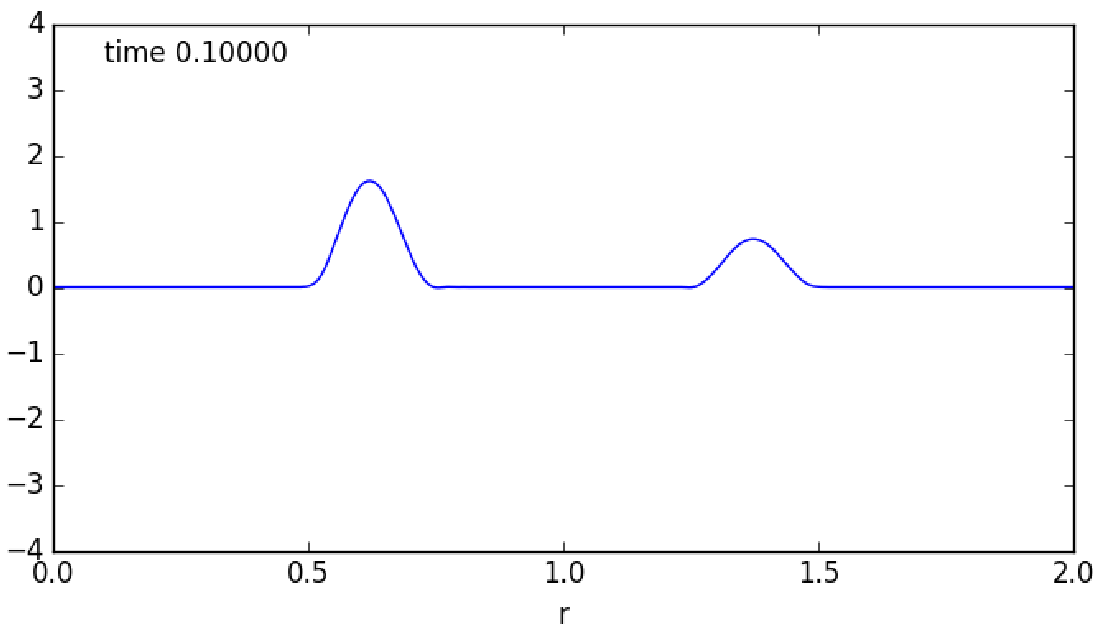

Figure 1: