

# PH4606 - Lecture 4

Claus-Dieter Ohl

April 12, 2016

## 1 Spherical Waves

In spherical coordinates  $(r, \phi, \theta)$  with

$$x = r \sin \theta \cos \phi \quad (1)$$

$$y = r \sin \theta \sin \phi \quad (2)$$

$$z = r \cos \theta \quad (3)$$

the Laplacian operator is

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

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

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

Thus the wave equation becomes

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

We can rewrite Eq. (6) as

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

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

$$pr = f_1(ct - r) + f_2(ct + r) \quad (8)$$

or

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

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.

```

1  %matplotlib notebook
2  import math as m
3  import numpy #array operations
4  import matplotlib.pyplot as plt #plotting
5  from IPython import display #for continous display
6  from ipywidgets import widgets #for the widgets
7
8  nimg=0
9
10 def plotwave(u,time,r):
11     plt.figure(1)
12     plt.clf()
13     plt.plot(r,u)
14     plt.text(0.1,3.4,"time {0:.5f}".format(time)) #annotate the time
15     plt.xlabel(r"r")
16     plt.gca().set_ylim([-4,4])
17     display.clear_output(wait=True)
18     display.display(plt.gcf())
19
20 def solvewave(b):
21     tabs.visible=False
22     #computational domain
23     nx = 381
24     size=2. #size of the domain
25     #parameters of the wave
26     c = 5. #speed of sound
27     l=w_wavelength.value #wavelength
28     nu=c/l #frequency
29     omega=nu*2.*m.pi #angular frequency
30     duration=w_sourceduration.value/nu #duration of source
31
32     #further variables
33     dx = size/(nx-1)
34     CFL=0.1 #CFL number <1
35     dt = CFL*dx/c
36     nt=int(w_simduration.value/dt) #number of time steps
37     if w_position.value=='Left':
38         sourcepos=1
39         ampl=100.
40     else:
41         sourcepos=int(nx/2)
42         ampl=1.
43     r=numpy.arange(dx,nx*dx,dx) #radius
44
45     #every xx times over the total nt timesteps an output should be generated
46     output=map(int,list(numpy.linspace(1,nt,int(nt/50))))
47
48     u = numpy.zeros(nx) #pressure at t
49     un = numpy.zeros(nx) #pressure at t-dt
50     unn= numpy.zeros(nx) #pressure at t-2*dt
51     C=c*c*dt*dt/dx/dx
52
53     #Assign initial conditions, for a sin^2 wave
54     if w_source_type.value=="Time Dependent":

```

```

55     un[sourcepos]=0.      #amplitude is sin(omega*t) with t=0
56     unn[sourcepos]=1.*r[sourcepos] #velocity is cos(omega*t) with t=0
57 else: #or set it as an initial value
58     for xx in range(nx):
59         x=float(xx)*dx-size/2. #0 at the center
60         unn[xx]=(m.cos(2.*m.pi/l*x)**2)*float(abs(x)<(c*duration/2.))
61     for xx in range(1,nx-1): #calculate t=0 time step
62         un[xx] = unn[xx] - 0.5*C*(unn[xx+1]-2.*unn[xx]+unn[xx-1])
63
64 plt.figure(1, figsize=(8, 4), dpi=300)
65
66 for n in range(nt+1): ##loop across number of time steps
67     #this line computes the finite differences of the wave equation
68     u[1:-1]=2.*un[1:-1]-unn[1:-1]+C*(un[:-2]+un[2:]-2.*un[1:-1])
69
70     #Boundary conditions right
71     if w_boundary_r.value=='Open':
72         u[-1] = un[-1]-dt*c/dx*(un[-1]-un[-2])
73     else:
74         u[-1] = u[-2]
75     #Boundary conditions right
76     u[0] = u[1]
77
78     #pressure source
79
80     if w_source_type.value=="Time Dependent":
81         if float(n*dt<duration):
82             u[sourcepos]=ampl*m.sin(omega*n*dt)**2*float(n*dt<duration)*r[sourcepos]
83
84     #save values for the time derivative
85     unn=un.copy() #n-1 time step
86     un=u.copy() #n time step
87
88     if (n in output):
89         plotwave(u[1:]/r,n*dt,r)
90
91 #and plot the last figure
92 plotwave(u[1:]/r,n*dt,r)
93 tabs.visible=True

```

1

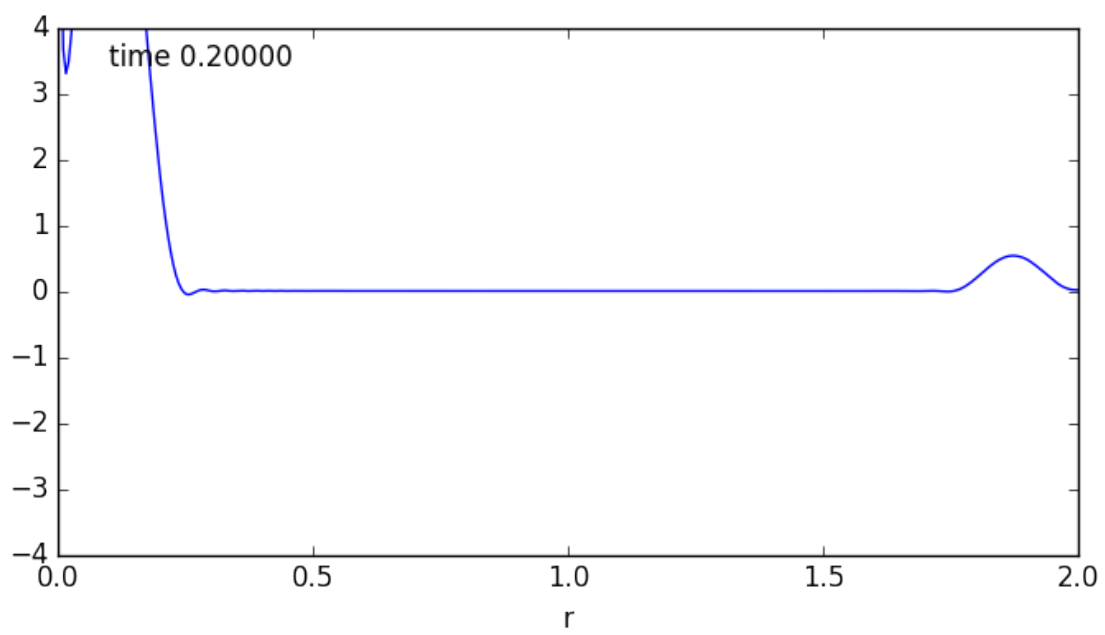


Figure 1: