

PH4606 - Lecture 2

Claus-Dieter Ohl

February 23, 2016

1 Harmonic Plane Waves

In the following we will consider a constant speed of sound and start with the description of a plane wave propagation. Planar waves have a constant amplitude and phase perpendicular to the direction of propagation. Thus if the wave moves in the x -plane we have:

$$\frac{\partial^2 p}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} \quad . \quad (2.1)$$

The complex solution to the harmonic wave equation is

$$p = A e^{i(\omega t - kx)} + B e^{i(\omega t + kx)} \quad . \quad (2.2)$$

Using the linear Euler Eq. (1.11) we can obtain the particle velocity in a plane wave as

$$\mathbf{u} = u \vec{e}_1 = \left[\frac{A}{\rho_0 c} e^{i(\omega t - kx)} - \frac{B}{\rho_0 c} e^{i(\omega t + kx)} \right] \vec{e}_1 \quad (2.3)$$

Please note that the constants A and B , pressures and velocities are now all complex. We can separate the wave travelling in positive x -direction from the the wave travelling in negative x -direction and designate them with the subscript $+$ and $-$, respectively.

The wave travelling in positive x -direction is

$$p_+ = A e^{i(\omega t - kx)} \quad (2.4)$$

while in negative x -direction is

$$p_- = B e^{i(\omega t + kx)} \quad (2.5)$$

These two waves can be formally written for the other acoustic variables:

$$u_{\pm} = \pm \frac{p_{\pm}}{\rho_0 c} \quad (1)$$

$$s_{\pm} = \frac{p_{\pm}}{\rho_0 c^2} \quad (2)$$

$$\Phi_{\pm} = -\frac{p_{\pm}}{i\omega \rho_0} \quad (2.8)$$

Plane waves travelling in an arbitrary direction can be written as

$$p = Ae^{i(\omega t - k_x x - k_y y - k_z z)} \quad (2.9)$$

If we insert Eq. (2.9) into the wave equation, Eq. (1.17) we obtain the condition

$$\left(\frac{\omega}{c}\right)^2 = k_x^2 + k_y^2 + k_z^2 \quad (2.10)$$

Now we can introduce the propagation vector, \mathbf{k} of the wave as

$$\mathbf{k} \equiv k_x \mathbf{e}_1 + k_y \mathbf{e}_2 + k_z \mathbf{e}_3 \quad (2.11)$$

with

$$|\mathbf{k}| = \frac{\omega}{c} \quad (2.12)$$

Thus using the position vector $\mathbf{r} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$ we can rewrite the plane wave solution, Eq. (2.9), as

$$p = Ae^{i(\omega t - \mathbf{k} \cdot \mathbf{r})} \quad (2.13)$$

The propagation vector points in the direction of wave propagation which can be simply shown. Surfaces of constant phase are defined by the scalar product $\mathbf{k} \cdot \mathbf{r} = \text{const.}$ The vector $\nabla(\mathbf{k} \cdot \mathbf{r})$ is a vector in the normal direction to the constant phase surface. But $\mathbf{k} = \nabla(\mathbf{k} \cdot \mathbf{r})$, thus \mathbf{k} is pointing normal to the plane of constant phase and thus in the direction of propagation.

1.1 Example

Let a plane wave have a constant phase parallel to the z-axis, thus the wave travels in the xy-plane as sketched in Fig. 2.1. Formally the surface of constant phase is $\mathbf{k} \cdot \mathbf{r} = 0$; thus if the phase is independent of z , \mathbf{k} can only have k_x and k_y components. We can thus write Eq. (2.9) as

$$p = Ae^{i(\omega t - k_x x - k_y y)} \quad (2.14)$$

and the equation for the surface of constant phase is

$$k_x x + k_y y = \text{const.} \quad (2.15)$$

or

$$y = -\frac{k_x}{k_y} x + \text{const.} \quad (2.16)$$

Equation (2.16) is line with a slope $-k_x/k_y$. The surfaces are thus oriented as sketched in Fig. 2.1.

Figure 2.1

Now let's study this wave in the x-direction while $y = 0$

$$p = Ae^{i(\omega t - k_x x)} \quad (2.17)$$

where $k_x = 2\pi/\lambda_x$ with λ_x being the apparent wavelength in x -direction. Note that this wavelength is larger than the wavelength in \vec{k} -direction. We can relate λ_x with the help of Fig. 1 to $\cos \phi = \lambda/\lambda_x$ and thus $k_x = k \cos \phi$. Applying the same arguments in the y -direction gives

$$\mathbf{k} = k \cos \phi \mathbf{e}_1 + k \sin \phi \mathbf{e}_2 \quad . \quad (2.18)$$

Thus an harmonic plane wave in 2-dimensions propagating under an angle ϕ to the x -axis can be written as

$$p = Ae^{i(\omega t - kx \cos \phi - ky \sin \phi)} \quad (2.19)$$

```

1  ## Simulation of a 2-dimensional wave
2
3  %matplotlib notebook
4  import math as m
5  import numpy #array operations
6  import matplotlib.pyplot as plt #plotting
7  from ipywidgets import widgets #for the widgets
8  #from ipykernel.client import display
9  from IPython import display #for continuous display
10 #from PIL import Image #to export images
11
12 nimg=0
13
14 def savemyimage(visual):
15     global nimg
16     visual = (visual + 2.)/4.
17     #result = Image.fromarray((visual * 255).astype(numpy.uint8))
18     #result.save('out{:03d}.bmp'.format(nimg))
19     nimg=nimg+1
20
21 def plotwave(u,time,px,py,pp,pt):
22     plt.figure(1)
23     plt.clf()
24     plt.subplot2grid((4,4),(0,0), colspan = 4, rowspan = 3)
25     plt.imshow(u, origin='upper', extent=[0., 2., 0., 2.], vmax=2, vmin=-2) #plot the wave field
26     plt.text(0.1,1.8,"time {0:.5f}".format(time)) #annotate the time
27     plt.plot(w_probex.value,1.,'o') #position of probe
28     plt.subplot2grid((4,4),(3,0), colspan = 4)
29     plt.plot(pt,pp) #plot the pressure at the probe
30     plt.gca().set_ylim([-2,2])
31     display.clear_output(wait=True)
32     display.display(plt.gcf())
33
34 def solvewave(b):
35     tabs.visible=False
36     #computational domain
37     nx = ny = 381
38     size=2. #size of the domain
39     #parameters of the wave
40     c = 5. #speed of sound
41     l=w_wavelength.value #wavelength
42     nu=c/l #frequency
43     omega=nu*2.*m.pi #angular frequency
44     duration=w_sourceduration.value/nu #duration of source
45     #position
46     emissionlength=(w_sizeemit.value/100.)*nx
47     startx=int(nx/2-emissionlength/2)
48     endx=int(nx/2+emissionlength/2)

```

```

47
48 #further variables
49 dx = size/(nx-1)
50 CFL=0.1 #CFL number <1
51 dt = CFL*dx/c
52 nt=int(w_simulation.value/dt) #number of time steps
53 if w_position.value=='Left':
54     sourcepos=0
55 else:
56     sourcepos=int(nx/2)
57
58 #arrays for measuring the pressure at a position
59 pt=numpy.arange(nt+1)*dt
60 pp=numpy.zeros(nt+1)
61 px=int(w_probex.value*(nx-1)/2.)
62 py=int(ny/2)
63
64 #every xx times over the total nt timesteps an output should be generated
65 output=map(int,list(numpy.linspace(1,nt,int(nt/50))))
66
67 u = numpy.zeros((nx,ny)) #pressure at t
68 un = numpy.zeros((nx,ny)) #pressure at t-dt
69 unn= numpy.zeros((nx,ny)) #pressure at t-2*dt
70
71 row, col = u.shape
72 #Assign initial conditions, for a sine wave
73 un[startx:endx,0]=0. #amplitude is sin(omega*t) with t=0
74 unn[startx:endx,0]=1. #velocity is cos(omega*t) with t=0
75
76 C=c*c*dt*dt/dx/dx
77
78 plt.figure(1, figsize=(8, 8), dpi=300)
79
80 for n in range(nt+1): ##loop across number of time steps
81     #this line computes the finite differences of the wave equation
82     u[1:-1,1:-1]=2.*un[1:-1,1:-1]-unn[1:-1,1:-1]+C*(un[1:-1,:-2]+\
83         un[:-2,1:-1]+un[2:,1:-1]+un[1:-1,2:]-4.*un[1:-1,1:-1])
84
85     #hard reflective boundary conditions
86     u[0,:] = u[1,:]
87     u[-1,:] = u[-2,:]
88     u[:,0] = u[:,1]
89     u[:, -1] = u[:, -2]
90
91     #pressure source
92     #if float(n*dt<duration):
93     u[startx:endx,sourcepos]=m.sin(omega*n*dt)*float(n*dt<duration)
94
95     pp[n]=u[py,px]
96
97     #save values for the time derivative
98     unn=un.copy() #n-1 time steop
99     un=u.copy() #n time step
100
101     if (n in output):
102         plotwave(u,n*dt,px,py,pp,pt)
103         if (w_saveplots.value):
104             savemyimage(u)
105

```

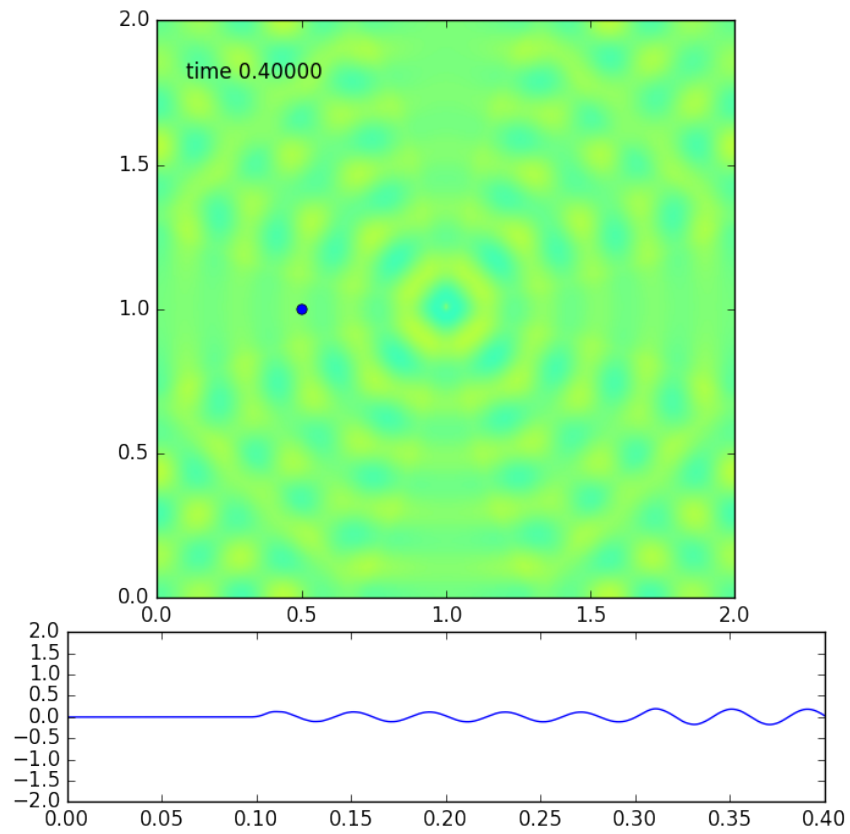


Figure 1:

```

106  #and plot the last figure
107  plotwave(u,n*dt,px,py,pp,pt)
108  tabs.visible=True

```

1