sound_analyser

January 22, 2023

1 Analysing sound wave using fast Fourier transform

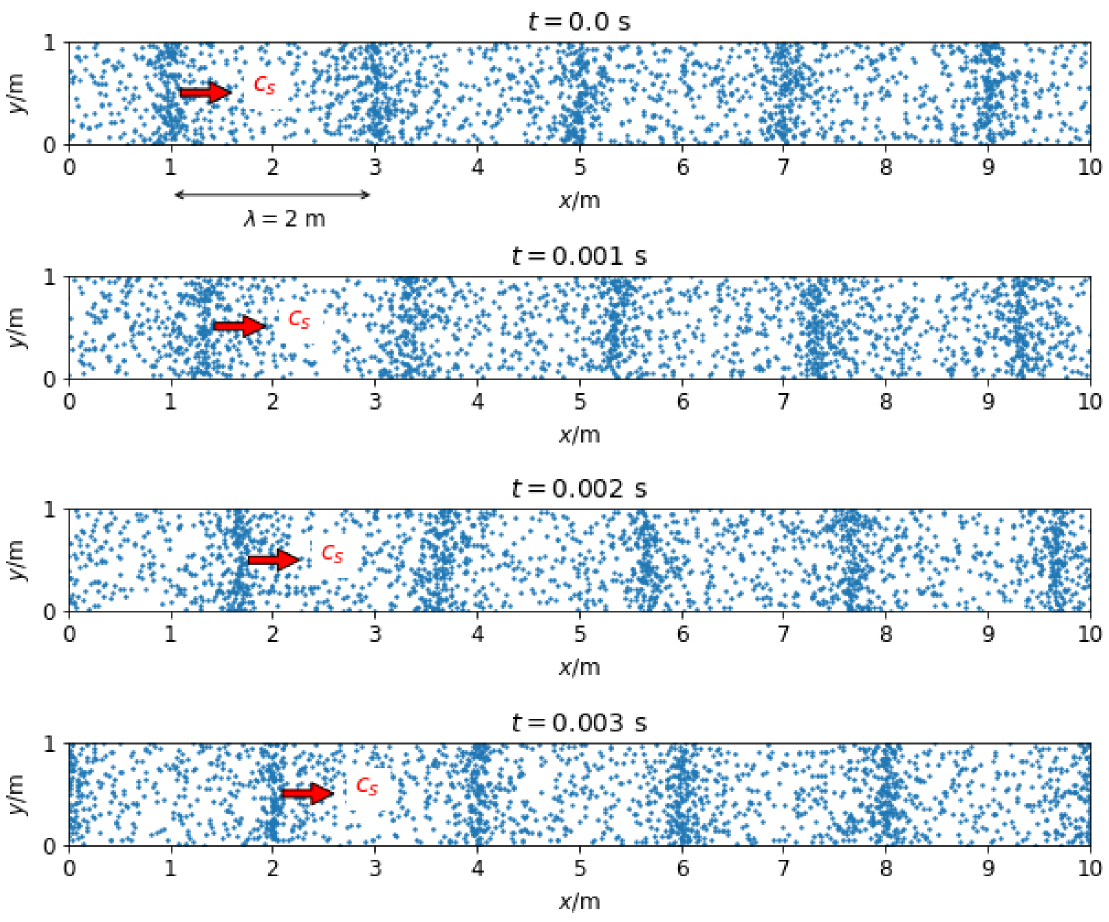
1.1 Sound wave

A sound wave consists of compression and rarefaction of air molecules, which travels through space with sound speed c_s. The figure below depicts the propagation of a *sinusoidal* sound wave inside an infinitely long rectangular channel. The figure shows the snapshots of the air molecules (represented by blue dots in the figure) at four different instantaneous time $t = 0\,\mathrm{s}$, $0.001\,\mathrm{s}$, $0.002\,\mathrm{s}$, and $0.003\,\mathrm{s}$. The regions of high density are called the compression regions (these look like fuzzy vertical blue bands in the figure) and the regions of low density are called the rarefaction regions. In the figure, these bands of compression and rarefaction regions travel along the positive x-direction with speed equals to the sound speed c_s. The wavelength of this sinusoidal sound wave is defined by the distance between two nearest compression bands, which is equal to $\lambda = 2\,\mathrm{m}$. As we can see from the figure, after time $t = 0.003\,\mathrm{s}$, the bands will have travelled a distance of $1\,\mathrm{m}$. Therefore the speed of sound in this example is equal to:

$$c_s = \frac{1\,\mathrm{m}}{0.003\,\mathrm{s}} \simeq 333\,\mathrm{ms}^{-1}. \tag{1}$$

In general, the speed of sound depends on various factors such as ambient pressure and temperature.

[ ]:
```python
import matplotlib.pyplot as plt
import numpy as np

Np = 6000 # number of particles
Lx, Ly = 10, 1
Dx = 0.25 # maximum displacement
lamb = 1.0 # wavelength
phi = 0.0 # phase difference

t = np.arange(0, 10, 0.001)

fig, ax = plt.subplots(4, figsize=(10,9))

for n in range(0, 4, 1):
    x = np.random.uniform(-Lx, 2*Lx, Np)
    y = np.random.uniform(0, Ly, Np)
    phi = np.pi/3*n
    x = x + Dx*np.sin(np.pi*x/lamb - phi)
    ax[n].set_title(f'$t = ${t[n]} s', fontsize=14)
    ax[n].set_xlabel('$x/$m', fontsize=12)
    ax[n].set_ylabel('$y/$m', fontsize=12)
    ax[n].set_xlim(0, Lx)
    ax[n].set_ylim(0, Ly)
    ax[n].set_aspect(1)
    ax[n].tick_params(axis='both', which='major', labelsize=12)
    ax[n].set_yticks(np.arange(0, 2, 1))
    ax[n].set_xticks(np.arange(0, 11, 1))
    ax[n].scatter(x, y, s=2)
    ax[n].annotate('', xy=(1.6+n/3,0.5), xytext=(1.1+n/3,0.5),␣
↪arrowprops=dict(facecolor='red'))
    ax[n].annotate('$c_s$', fontsize=14, backgroundcolor='white', c='red',␣
↪xy=(1.8+n/3,0.5))

ax[0].annotate('', c='black',
            xy=(1.0,-0.5),  # location of the arrow tip
            xytext=(3.0,-0.5),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
↪arrowstyle='<->'),
            annotation_clip=False)

ax[0].annotate('$\lambda=2$ m', c='black', fontsize=12, xy=(1.7,-0.8),␣
↪annotation_clip=False)

plt.show()
```



Now suppose that that we place a microphone at the end of the channel. The compression and rarefaction of air will cause the diaphragm inside the microphone to vibrate. This vibration is then converted into an electrical signal, which is shown in the figure below. The horizontal axis in the figure represents the time t (in units of seconds). The vertical axis represents the voltage of the electrical signal produced by the microphone $V(t)$ (in some rescaled units, which we do not need to worry about). As we can see in this example, the audio signal in the figure below has a sinusoidal form, which can be descibed by a trigonometric function:

$$V(t) = A\cos(\omega t + \phi), \tag{2}$$

where $V(t)$ is the audio signal (in some rescaled units), t is time (in seconds), A is the amplitude, ω is the angular frequency, and ϕ is the phase difference. The angular frequency ω is related to the frequency f and period T of the sound wave through this relation:

$$\omega = 2\pi f = \frac{2\pi}{T}. \tag{3}$$

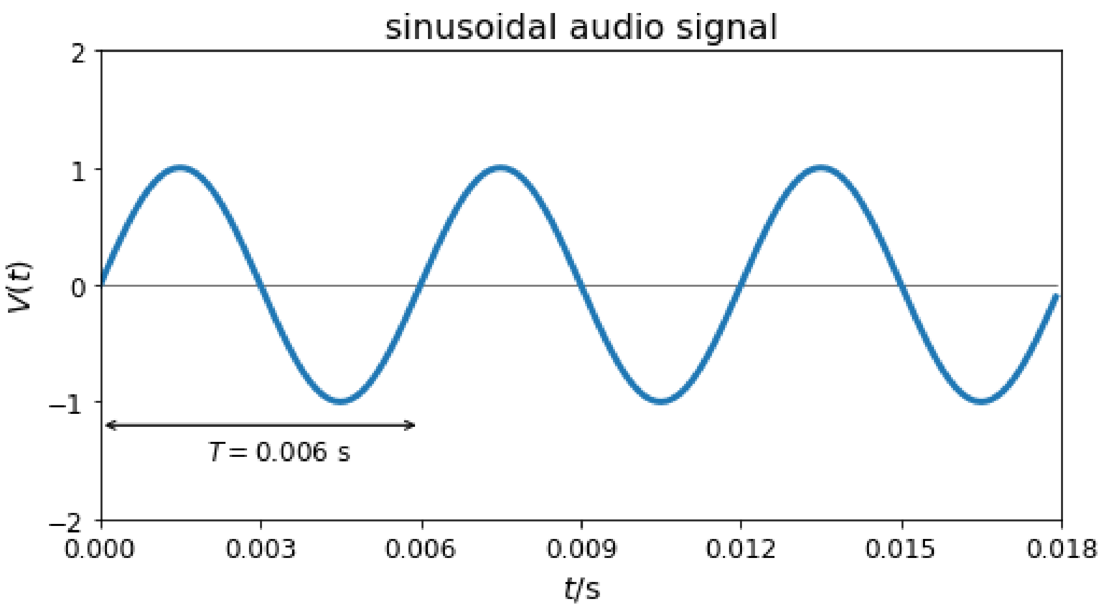
From the plot below, we can measure the period to be $T = 0.006\,\mathrm{s}$, which translates to audio frequency of $f \simeq 167\,\mathrm{Hz}$. Hz (prounounced as Hertz) is the SI unit of frequency, defined to be $\mathrm{Hz} = \mathrm{s}^{-1}$.

In the equation above, V, A, ω, ϕ, and t are all real. However, sometimes it might be useful to write the audio signal in a complex form (as we shall see later in Fourier series) as follows:

$$V(t) = Ce^{i\omega t} + C^{*}e^{-i\omega t}, \text{ where } C = \frac{A}{2}e^{i\phi} \text{ is the complex amplitude (the rest of the variables are real).} \tag{4}$$

The star $*$ above C indicates complex conjugate operation.

```
[ ]: T = 0.006 # period
     dt = 0.0001 # timestep
     t = np.arange(0, 0.018, dt)
     omega = np.pi*2/T
     V = np.sin(omega*t)

     fig, ax = plt.subplots(figsize=(8, 4))

     ax.set_title('sinusoidal audio signal', fontsize=16)
     ax.set_xlabel('$t/$s', fontsize=14)
     ax.set_ylabel('$V(t)$', fontsize=14)
     ax.set_xlim(0, 0.018)
     ax.set_ylim(-2, 2)
     ax.tick_params(axis='both', which='major', labelsize=12)
     ax.set_yticks(np.arange(-2, 2.1, 1))
     ax.set_xticks(np.arange(0, 0.019, 0.003))

     ax.plot(t, t*0, c='black', linewidth=0.5)
     ax.plot(t, V, linewidth=3)

     ax.annotate('', c='black',
                 xy=(0.0,-1.2),  # location of the arrow tip
                 xytext=(0.006,-1.2),  # location of the text
                 arrowprops=dict(edgecolor='black', facecolor='red',␣
      ↪arrowstyle='<->'),
                 annotation_clip=False)

     ax.annotate('$T = 0.006$ s', c='black', fontsize=12, xy=(0.002,-1.5),␣
      ↪annotation_clip=False)

     plt.show()
```



1.2 Fourier series

In the plot above, the audio signal can be described by a single trigonometric function. This is because we have assumed the sound wave to be sinusoidal. However this is not true in general. For example, the sound wave coming from a tuning fork is very close to a sinusoidal wave, on the other hand, the sound wave from a saxophone is far from being sinusoidal.

Let us consider another audio signal from an unknown musical instrument, which is depicted in the figure below. We can immediately tell that the signal is periodic with the same period $T = 0.006\,\mathrm{s}$ (or fundamental frequency $f \simeq 167\,\mathrm{Hz}$) as the one above. However, the shape of the audio signal is much more complicated than a sinusoidal wave and cannot be simply described by a single trigonometric function. Luckily, Fourier series allows us to decompose this periodic signal into a sum of trigonometric functions as follows:

$$V(t) = \sum_{p=-\infty}^{\infty} C_p e^{i\omega_p t}, \text{ where } \omega_p = \frac{2\pi p}{T} \text{ and } p \in \mathbb{Z}. \tag{5}$$

Each term in the Fourier series is a simple sinusoidal wave with angular frequency ω_p and complex amplitude C_p's. The first non-constant term in the Fourier series corresponds to the *fundamental frequency* $\omega_1 = \frac{2\pi}{T}$ shown by the first figure in the second row below. The next term in the Fourier series has double the fundamental frequency $\omega_2 = \frac{4\pi}{T}$ and is sometimes called the second harmonic (see the second figure in the second row below). The next next term has triple the fundamental frequency $\omega_3 = \frac{6\pi}{T}$ and is sometimes called the third harmonic (see the third figure in the second row below).

To find the complex amplitudes C_p's (or Fourier coefficients) we multiply the above equation by

$e^{-i\omega_q t}$ and then integrate with respect to t from $t = 0$ to $t = T$.

$$\int_0^T V(t)e^{-i\omega_q t}\,dt = \sum_{p=-\infty}^{\infty} C_p \int_0^T e^{i(\omega_p - \omega_q)t}\,dt, \tag{6}$$

where p and q are integers. We note that the integral:

$$\int_0^T e^{i(\omega_p-\omega_q)t}\,dt = \int_0^T e^{i\frac{2\pi}{T}(p-q)t}\,dt = \begin{cases} T & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}. \tag{7}$$

More succintly, we can write,

$$\int_0^T e^{i(\omega_p-\omega_q)t}\,dt = T\delta_{pq}, \tag{8}$$

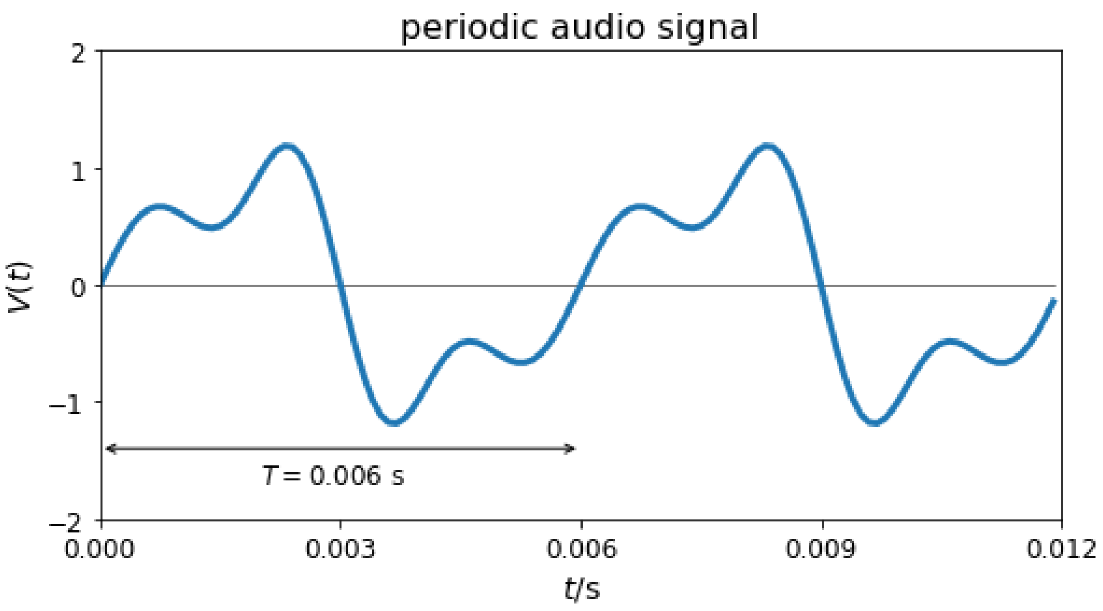
where the Kronecker delta δ_{pq} is defined to be equal to 1 if $p = q$ and 0 if $p \neq q$. Therefore,

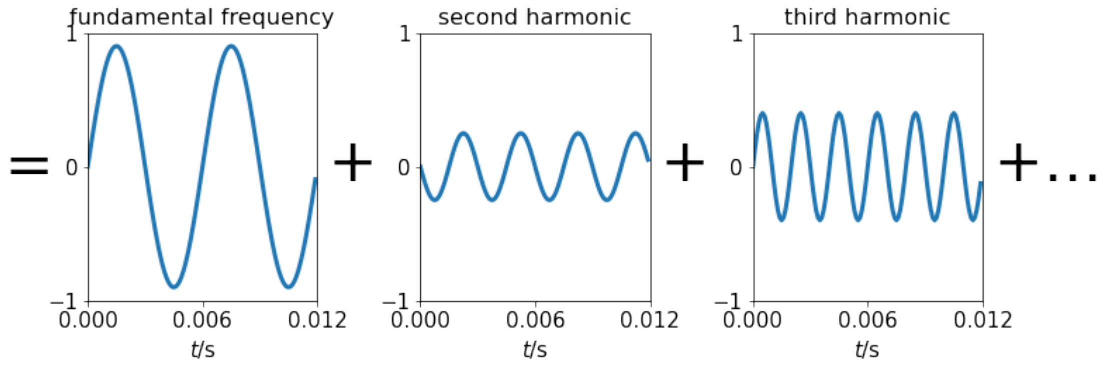
$$\int_0^T V(t)e^{-i\omega_q t}\,dt = \sum_{p=-\infty}^{\infty} C_p T\delta_{pq} = TC_q \tag{9}$$

$$\Rightarrow C_q = \frac{1}{T}\int_0^T V(t)e^{-i\omega_q t}\,dt. \tag{10}$$

Question 1. (a) Show that if $C_{-p} = C_p^*$ for all $p \in \mathbb{Z}$, then $V(t)$ is real. (b) Show that if $V(t)$ is real, then $C_{-p} = C_p^*$ for all $p \in \mathbb{Z}$.

[ ]:
```
dt = 0.0001
T = 0.006
omega = np.pi*2/T
t = np.arange(0, 0.012, dt)
V = 0.9*np.sin(omega*t) - 0.25*np.sin(2*omega*t) + 0.4*np.sin(3*omega*t) - 0.
↪05*np.sin(4*omega*t)
sin1 = 0.9*np.sin(omega*t)
sin2 = - 0.25*np.sin(2*omega*t)
sin3 = 0.4*np.sin(3*omega*t)

fig, ax = plt.subplots(figsize=(8, 4))

ax.set_title('periodic audio signal', fontsize=16)
ax.set_xlabel('$t/$s', fontsize=14)
ax.set_ylabel('$V(t)$', fontsize=14)
ax.set_xlim(0, 0.012)
ax.set_ylim(-2, 2)
ax.tick_params(axis='both', which='major', labelsize=12)
ax.set_yticks(np.arange(-2, 2.1, 1))
ax.set_xticks(np.arange(0, 0.013, 0.003))

ax.plot(t, t*0, c='black', linewidth=0.5)
ax.plot(t, V, linewidth=3)
ax.annotate('', c='black',
            xy=(0.0,-1.4),  # location of the arrow tip
            xytext=(0.006,-1.4),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
 ↪arrowstyle='<->'),
            annotation_clip=False)

ax.annotate('$T = 0.006$ s', c='black', fontsize=12, xy=(0.002,-1.7),␣
 ↪annotation_clip=False)

plt.show()

fig, ax = plt.subplots(1, 3, figsize=(12, 3.5))

for n in range(0, 3, 1):
    ax[n].set_xlabel('$t/$s', fontsize=15)
    ax[n].set_xlim(0, 0.012)
    ax[n].set_ylim(-1, 1)
    ax[n].set_aspect(0.007)
    ax[n].tick_params(axis='both', which='major', labelsize=15)
    ax[n].set_yticks(np.arange(-1, 1.1, 1))
    ax[n].set_xticks(np.arange(0, 0.013, 0.006))

ax[0].set_title('fundamental frequency', fontsize=16)
ax[0].plot(t, sin1, linewidth=3)
ax[0].annotate('$=$', c='black', fontsize=42, xy=(-0.005, -0.1),␣
 ↪annotation_clip=False)
ax[0].annotate('$+$', c='black', fontsize=42, xy=(0.0125, -0.1),␣
 ↪annotation_clip=False)
ax[1].set_title('second harmonic', fontsize=16)
ax[1].plot(t, sin2, linewidth=3)
ax[1].annotate('$+$', c='black', fontsize=42, xy=(0.0125, -0.1),␣
 ↪annotation_clip=False)
ax[2].set_title('third harmonic', fontsize=16)
ax[2].plot(t, sin3, linewidth=3)
ax[2].annotate('$+\dots$', c='black', fontsize=42, xy=(0.0125, -0.1),␣
 ↪annotation_clip=False)

plt.show()
```





1.3 Fourier transform

In reality, the audio signal is not perfectly periodic since it might be contaminated by a background noise. Furthermore the oscillation of the sound wave itself decays gradually to zero, so a realistic representation of an audio signal recorded from a microphone might look something like the plot below.

To analyse this audio signal, we shall use Fourier transform, which is an extension of the Fourier series above by taking the limit $T \to \infty$ (loosely speaking, the function $V(t)$ is no longer periodic). In this case the angular frequency of each Fourier component becomes continuous $\omega_p \to \omega \in \mathbb{R}$. The Fourier transform of $V(t)$ is another function $\tilde{V}(\omega)$, which is a function of angular frequency

ω. $V(t)$ and $\tilde{V}(\omega)$ are related through the following relations:

$$V(t) = \int_{-\infty}^{\infty} \tilde{V}(\omega) e^{i\omega t} d\omega \quad \text{(inverse Fourier transform)} \tag{11}$$

$$\tilde{V}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} V(t) e^{-i\omega t} dt \quad \text{(Fourier transform)}, \tag{12}$$

where $\omega \in \mathbb{R}$. The above relations can be derived by taking the limit $T \to \infty$ in the definition of Fourier series, in which case the summation over ω_p becomes an integral over ω, and then relabelling $C_p \to \tilde{V}(\omega_p) \to \tilde{V}(\omega)$.

Now let us define the Dirac delta function $\delta(x - y)$, where $x, y \in \mathbb{R}$ such that

$$\delta(x - y) = \begin{cases} 0 & \text{if } x \neq y \\ \infty & \text{if } x = y \end{cases}. \tag{13}$$

The Dirac delta function has the following properties:

$$\int_{-\infty}^{\infty} \delta(x - y)\, dx = 1, \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x - y) f(x)\, dx = f(y). \tag{14}$$

In continous space, though it is ill-defined, its graph would take the shape of an infinitely thin curve that peaks at a single point, $x = y$, and is zero elsewhere. The area (integral) under this curve is equal to 1, see the second figure below.

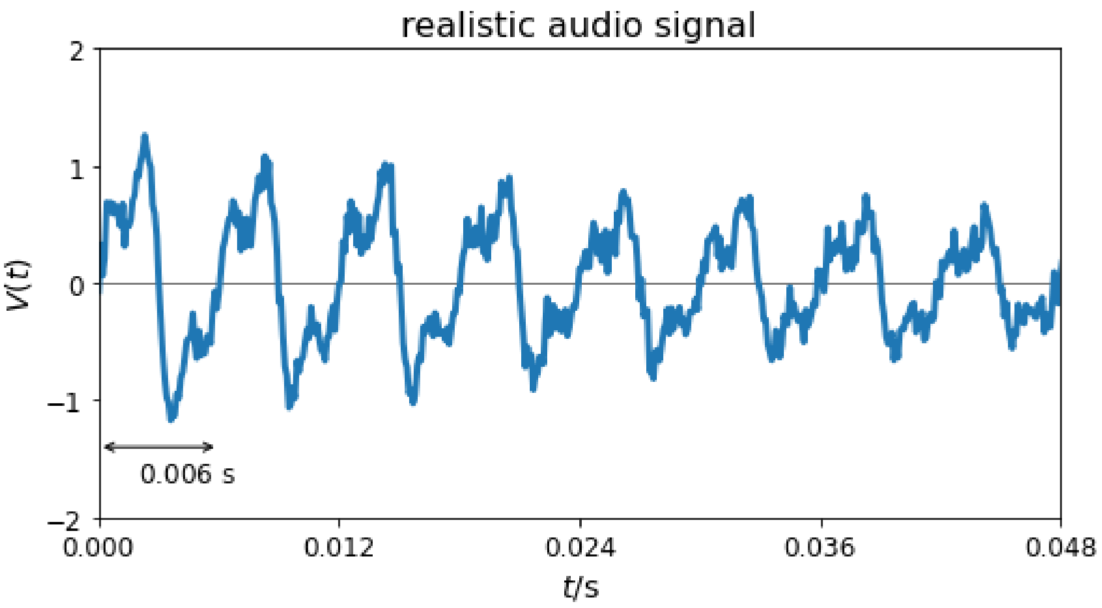
Question 2. Using the definition of Fourier transform, show that

$$\int_{-\infty}^{\infty} e^{i(\alpha - \beta)t}\, dt = 2\pi\delta(\alpha - \beta), \tag{15}$$

for all $\alpha, \beta, t \in \mathbb{R}$.

Question 3. Show that if $V(t)$ is real then $\tilde{V}(-\omega) = \tilde{V}(\omega)^*$ for all $\omega \in \mathbb{R}$ and *vice versa*.

```
[ ]: dt = 0.0001
     T = 0.006
     omega = np.pi*2/T
     t = np.arange(0, 0.192, dt)
     V = (0.9*np.sin(omega*t) - 0.25*np.sin(2*omega*t) + 0.4*np.sin(3*omega*t) - 0.
      ↪05*np.sin(4*omega*t))*np.exp(-t/0.05) + 0.1*np.random.normal(0,1, np.
      ↪shape(t))

     fig, ax = plt.subplots(figsize=(8, 4))

     ax.set_title('realistic audio signal', fontsize=16)
     ax.set_xlabel('$t/$s', fontsize=14)
     ax.set_ylabel('$V(t)$', fontsize=14)
     ax.set_xlim(0, 0.048)
     ax.set_ylim(-2, 2)
     ax.tick_params(axis='both', which='major', labelsize=12)
     ax.set_yticks(np.arange(-2, 2.1, 1))
ax.set_xticks(np.arange(0, 0.049, 0.012))

ax.plot(t, t*0, c='black', linewidth=0.5)
ax.plot(t, V, linewidth=3)

ax.annotate('', c='black',
            xy=(0.0,-1.4),  # location of the arrow tip
            xytext=(0.006,-1.4),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
 ↪arrowstyle='<->'),
            annotation_clip=False)

ax.annotate('$0.006$ s', c='black', fontsize=12, xy=(0.002,-1.7),␣
 ↪annotation_clip=False)

plt.show()
```



1.4 Energy spectrum

From electromagnetism, we learnt that the rate of power dissipation is given by:

$$P(t) = \frac{V(t)^2}{R}, \quad \text{where } R \text{ is the electrical resistance.} \tag{16}$$

The total energy dissipation is then given by the integral

$$E = \frac{1}{R} \int_{-\infty}^{\infty} V(t)^2 \, dt \tag{17}$$

Substituting the definition for Fourier transform to the above, we get:

$$E = \frac{1}{R}\int dt \int d\omega \int d\omega' \, \tilde{V}(\omega)\tilde{V}(\omega')e^{i(\omega+\omega')t} \tag{18}$$

$$= \frac{2\pi}{R}\int d\omega \int d\omega' \, \tilde{V}(\omega)\tilde{V}(\omega')\delta(\omega+\omega') \tag{19}$$

$$= \frac{2\pi}{R}\int d\omega \, \tilde{V}(\omega)\tilde{V}(-\omega) \tag{20}$$

Since $V(t)$ is real, we have $\tilde{V}(-\omega) = \tilde{V}(\omega)^*$ from the previous exercise and thus the total energy dissipation can be written as:

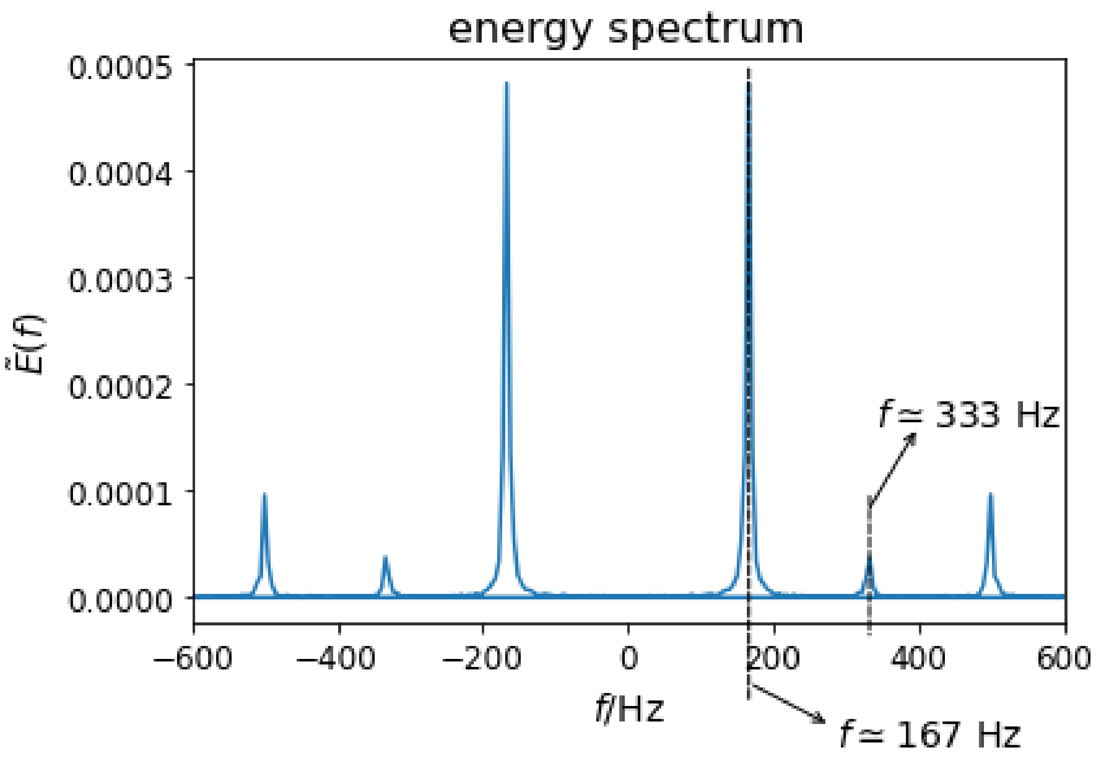
$$E = \frac{2\pi}{R}\int_{-\infty}^{\infty} d\omega \, |\tilde{V}(\omega)|^2 \tag{21}$$

Now we can define the energy spectrum to be $\tilde{E}(\omega) = |\tilde{V}(\omega)|^2$. Physically, Fourier transform allows us to decompose an electrical signal into an infinite sum of sinusoidal oscillations with different angular frequencies ω's. The energy spectrum $\tilde{E}(\omega)$ gives the energy contribution from a single oscillation with particular angular frequency ω.

The figure below shows the energy spectrum $\tilde{E}(f)$ as a function of frequency f, which corresponds to the realistic audio signal $V(t)$, shown in the previous figure. Note that the frequency f is related to the angular frequency by a factor of 2π, *i.e.* $\omega = 2\pi f$. As we can see in the figure below, the energy spectrum spectrum is symmetric with respect to $f \to -f$ (and for this reason, $\tilde{E}(f)$ is usually plotted on the positive x-axis only). Furthermore we also observe several sharp peaks in the energy spectrum. The first peak $f \simeq 167$ Hz corresponds to the fundamental frequency of the signal. (Although the signal is no longer periodic, it still retains some underlying periodic characteristics.) The second peak $f \simeq 333$ Hz (which is double the fundamental frequency) corresponds to the second harmonic and so on.

Question 4. Show that the energy spectrum is symmetric, *i.e.* $\tilde{E}(\omega) = \tilde{E}(-\omega)$.

```
[ ]: N = np.shape(V)[0]
     f = 1/(N*dt)*np.concatenate((np.arange(0, N/2, 1), np.arange(-N/2, 0, 1)))

     Vtilde = np.fft.fft(V)*dt # Fourier transform of V(t), Vtilde(omega)

     fig, ax = plt.subplots(figsize=(6, 4))

     ax.set_title('energy spectrum', fontsize=16)
     ax.set_xlabel('$f/$Hz', fontsize=14)
     ax.set_ylabel('$\\tilde{E}(f)$', fontsize=14)
     ax.set_xlim(-600, 600)
     #ax.set_ylim(0, 0.0005)
     ax.tick_params(axis='both', which='major', labelsize=12)
     #ax.set_yticks(np.arange(-2, 2.1, 1))
     #ax.set_xticks(np.arange(0, 0.049, 0.012))

     ax.plot(f, np.real(Vtilde*np.conjugate(Vtilde)))

     ax.annotate('', c='black',
            xy=(167,-0.0001),  # location of the arrow tip
            xytext=(167,0.0005),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red', arrowstyle='-',␣
↪linestyle='--'),
            annotation_clip=False)

ax.annotate('', c='black',
            xy=(280,-0.00012),  # location of the arrow tip
            xytext=(167,-0.00008),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
↪arrowstyle='->'),
            annotation_clip=False)

ax.annotate('', c='black',
            xy=(333,-0.00004),  # location of the arrow tip
            xytext=(333,0.0001),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red', arrowstyle='-',␣
↪linestyle='--'),
            annotation_clip=False)

ax.annotate('', c='black',
            xy=(400,0.00016),  # location of the arrow tip
            xytext=(330,0.00008),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
↪arrowstyle='->'),
            annotation_clip=False)

ax.annotate('$f\simeq 167$ Hz', c='black', fontsize=14, xy=(285,-0.00014),␣
↪annotation_clip=False)
ax.annotate('$f\simeq 333$ Hz', c='black', fontsize=14, xy=(340,0.00016),␣
↪annotation_clip=False)

plt.show()
```



1.5 Storing audio wave into a computer

Let's say we record a sound on our microphone for the total duration of t_N (also called the sampling time). How is this audio signal $V(t)$ stored inside our computer? Obviously a computer cannot store an infinite amount of information, so the computer has to divide the signal $V(t)$ into discrete values $V(t_0), V(t_1), V(t_2), \ldots, V(t_{N-1})$ at discrete times t_n's, as we can see in the figure below. This also means that the time is discretized into: $t \to t_n = n\Delta t$, where $n = 0, 1, 2, \ldots, N-1$ and Δt is the timestep. In the figure below, the total time (or sampling time) is $t_N = 0.018\,\text{s}$, the timestep is $\Delta t = 0.001\,\text{s}$, and the total number of points is $N = 18$. We also define the framerate to be the total number of points N per unit time. In the figure below, we can calculate the framerate to be:

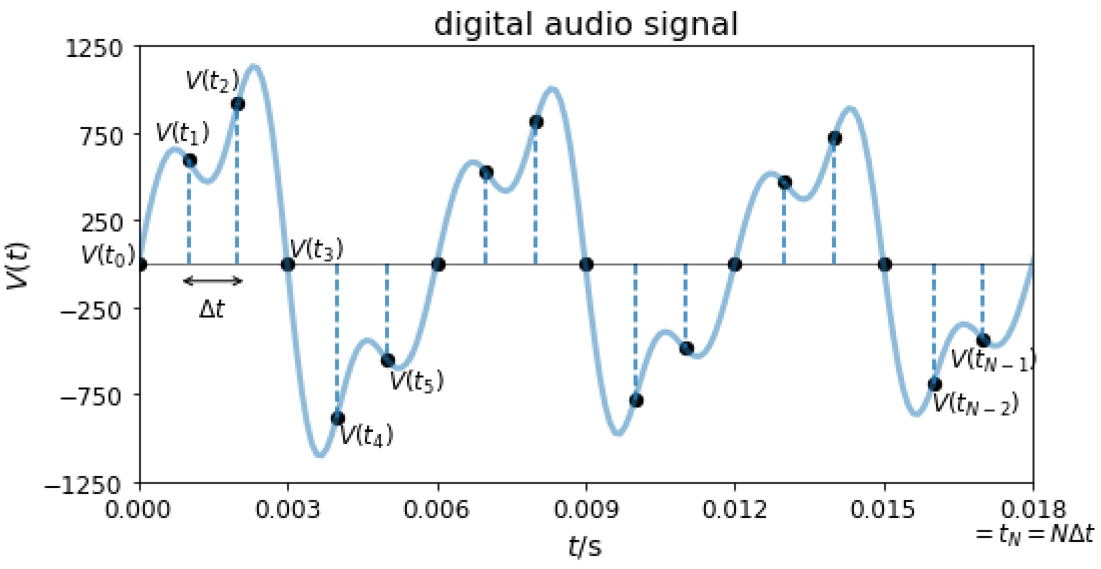
$$\text{framerate} = \frac{N}{t_N} = \frac{18}{0.018\,\text{s}} = 1000\,\text{s}^{-1}. \tag{22}$$

Usually, when we record a sound using recording software such as Audacity, we need to specify this framerate. Higher framerate will give a better sound quality but the file size will also be bigger! For a 16-bit digital audio, the values of V ranges from -32768 to 32767 in integer steps (note that $V(t)$ is in some rescaled units). Therefore the vertical V-axis is also discrete.

```
[ ]: t = np.arange(0, 0.192, 0.0001)
     t_discrete = np.arange(0, 0.018, 0.001)
     omega = np.pi*2/T
     V = (0.9*np.sin(omega*t) - 0.25*np.sin(2*omega*t) + 0.4*np.sin(3*omega*t) - 0.
      ↪05*np.sin(4*omega*t))*np.exp(-t/0.05)*1000
```

```
V_discrete = (0.9*np.sin(omega*t_discrete) - 0.25*np.sin(2*omega*t_discrete) +␣
↪0.4*np.sin(3*omega*t_discrete) - 0.05*np.sin(4*omega*t_discrete))*np.
↪exp(-t_discrete/0.05)*1000

fig, ax = plt.subplots(figsize=(8, 4))

ax.set_title('digital audio signal', fontsize=16)
ax.set_xlabel('$t/$s', fontsize=14)
ax.set_ylabel('$V(t)$', fontsize=14)
ax.set_xlim(0, 0.018)
ax.set_ylim(-1250, 1250)
ax.tick_params(axis='both', which='major', labelsize=12)
ax.set_yticks(np.arange(-1250, 1300, 500))
ax.set_xticks(np.arange(0, 0.019, 0.003))

ax.plot(t, t*0, c='black', linewidth=0.5)
ax.plot(t, V, linewidth=3, alpha=0.5)
ax.scatter(t_discrete, V_discrete, c='black', s=40)
ax.stem(t_discrete, V_discrete, markerfmt=' ', basefmt=' ', linefmt='--',␣
↪use_line_collection=True)

ax.annotate('', c='black',
            xy=(0.0008,-100),  # location of the arrow tip
            xytext=(0.0022,-100),  # location of the text
            arrowprops=dict(edgecolor='black', facecolor='red',␣
↪arrowstyle='<->'),
            annotation_clip=False)

ax.annotate('$\Delta t$', c='black', fontsize=12, xy=(0.0012,-300),␣
↪annotation_clip=False)
ax.annotate('$= t_N = N\Delta t$', c='black', fontsize=12, xy=(0.0167,-1600),␣
↪annotation_clip=False)
ax.annotate('$V(t_0)$', c='black', fontsize=12, xy=(-0.0012,0.0),␣
↪annotation_clip=False)
ax.annotate('$V(t_1)$', c='black', fontsize=12, xy=(0.0003,700),␣
↪annotation_clip=False)
ax.annotate('$V(t_2)$', c='black', fontsize=12, xy=(0.0009,1000),␣
↪annotation_clip=False)
ax.annotate('$V(t_3)$', c='black', fontsize=12, xy=(0.003,40),␣
↪annotation_clip=False)
ax.annotate('$V(t_4)$', c='black', fontsize=12, xy=(0.004,-1030),␣
↪annotation_clip=False)
ax.annotate('$V(t_5)$', c='black', fontsize=12, xy=(0.005,-720),␣
↪annotation_clip=False)
ax.annotate('$V(t_{N-1})$', c='black', fontsize=12, xy=(0.0163,-595),␣
↪annotation_clip=False)
```

```
ax.annotate('$V(t_{N-2})$', c='black', fontsize=12, xy=(0.01595,-850),␣
↪annotation_clip=False)

plt.show()
```



1.6 Importing .wav file

There are different audio formats in computer such as `.mp3` and `.wav`. The difference is that the former is a compressed file while the later is an uncompressed file. In this Tutorial, we will only consider `.wav` files.

Let us now put everything we have learnt above into practice. Inside the folder `./samples/` we have various `.wav` files from different notes from different musical instruments. First let us have a look at the file called `./samples/piano-C4.wav`. This is a recording of the note C4 in a piano. This note has a fundamental frequency of 261.63 Hz. Let us now analyse this wave file below. First we need to import the essential libraries such as `numpy`, `matplotlib`, `sys` and `wave` into Python, as you can see in the first few lines of the code below. The library `sys` is used to access the filesystem inside our computer and the library `wave` is used to import and export `.wav` files. Next we will open the file `./samples/piano-C4.wav` and store it into an object, called `inputfile`, using the method `wave.open('./samples/piano-C4.wav', 'r')`. The letter `'r'` indicates that we are reading the file (letter `'w'` indicates writing into a file). Before we proceed further, we need some information about this audio signal, which we just imported. For example, we need the framerate, which we can obtain using the method `.getframerate()`. In this case the framerate is $11025\,\mathrm{s}^{-1}$, which we can confirm by printing the value into the screen. We also need the total number of points, *i.e.* N, which we can obtain using the method `.getnframes()`. In this case we get $N = 29750$. Hence we can calculate the total recording time (or sampling time) t_N:

$$t_N = \frac{N}{\text{framerate}} \simeq 2.70\,\mathrm{s}. \tag{23}$$

We also need to compute the timestep Δt:

$$\Delta t = \frac{t_N}{N} = \frac{1}{\text{framerate}} \simeq 0.0000907\,\text{s}. \tag{24}$$

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     import wave  # this library is to import and export .wav files
     import sys  # this library is to access the filesystem

     inputfile = wave.open('./samples/piano-C4.wav', 'r')  # read a .wav file and␣
      ↪store it into an object

     framerate = inputfile.getframerate()  # get the framerate of the audio file (in␣
      ↪units of seconds^-1)
     print(f'framerate = {framerate} s^-1')  # print the framerate into the computer␣
      ↪screen

     N = inputfile.getnframes() # get the total number of points
     print(f'N = {N}')

     tN = N/framerate  # calculate the total time
     print(f't_N = {N/framerate} s')

     dt = 1/framerate  # calculate the timestep
     print(f'dt = {dt} s')
```

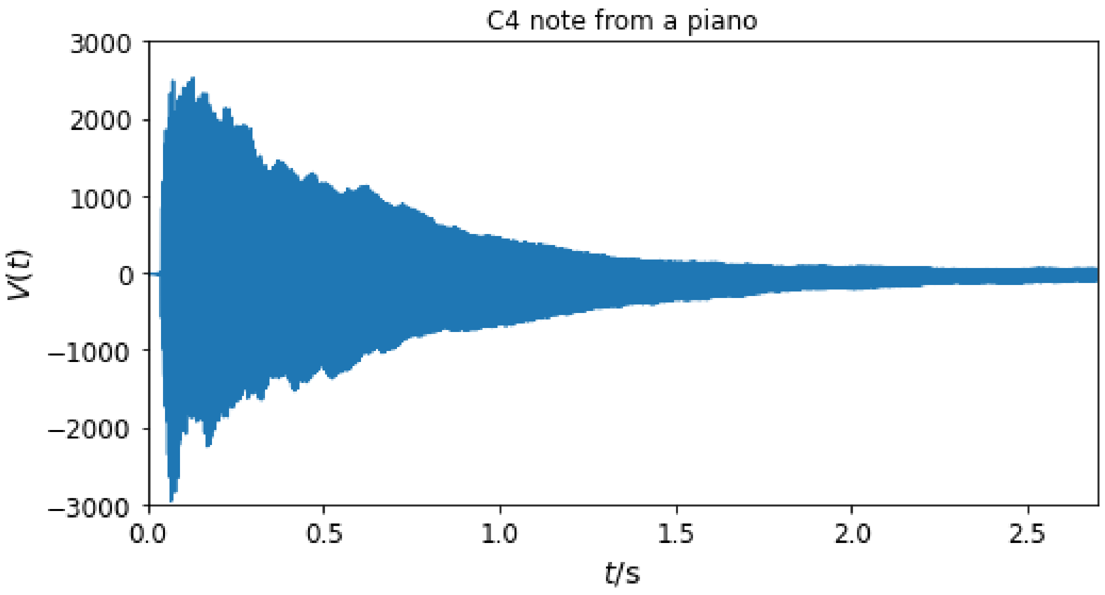
```
framerate = 11025 s^-1
N = 29750
t_N = 2.6984126984126986 s
dt = 9.070294784580499e-05 s
```

To read the actual audio signal itself from the `.wav` file, we use the method `.readframes(-1)`. If you try to print the output you will get something like this: `b'\xf6\xf1\xff\xf1\xff...\xf2\xff'`. The strange combination of three characters, separated by backslash, actually represents a binary number. We need to convert these binary numbers, called bytes objects, into decimals using the numpy method `np.frombuffer(V, dtype=np.int16)`. The `int16` data type indicates that it is a 16-bit audio, so that the values of V ranges from -32768 to 32767 in integer steps.

```
[ ]: V = inputfile.readframes(-1)  # read audio signal from the input .wav file
     V = np.frombuffer(V, dtype=np.int16)  # convert the binary format into integ
```

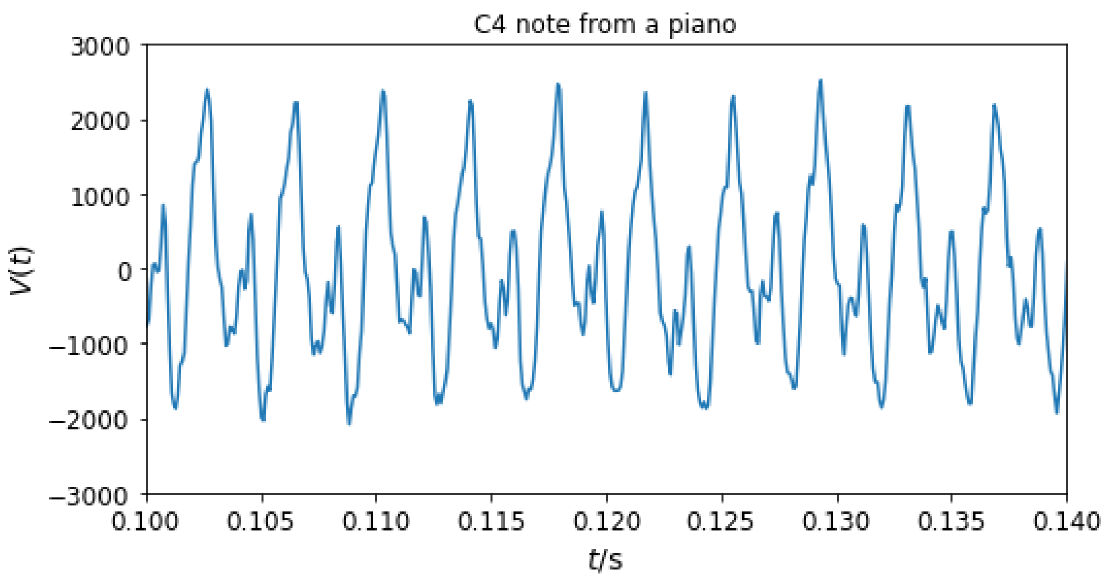
Next we can then plot the the audio signal $V(t)$ as a function of time t as shown below.

```
[ ]: t = np.linspace(0, len(V) / framerate, num=len(V))

     fig, ax = plt.subplots(figsize=(8, 4))

     ax.set_title("C4 note from a piano")
     ax.set_xlabel('$t/$s', fontsize=14)
     ax.set_ylabel('$V(t)$', fontsize=14)
     ax.set_xlim(0, 2.7)
     ax.set_ylim(-3000, 3000)
     ax.tick_params(axis='both', which='major', labelsize=12)

     plt.plot(t, V)
     plt.show()
```

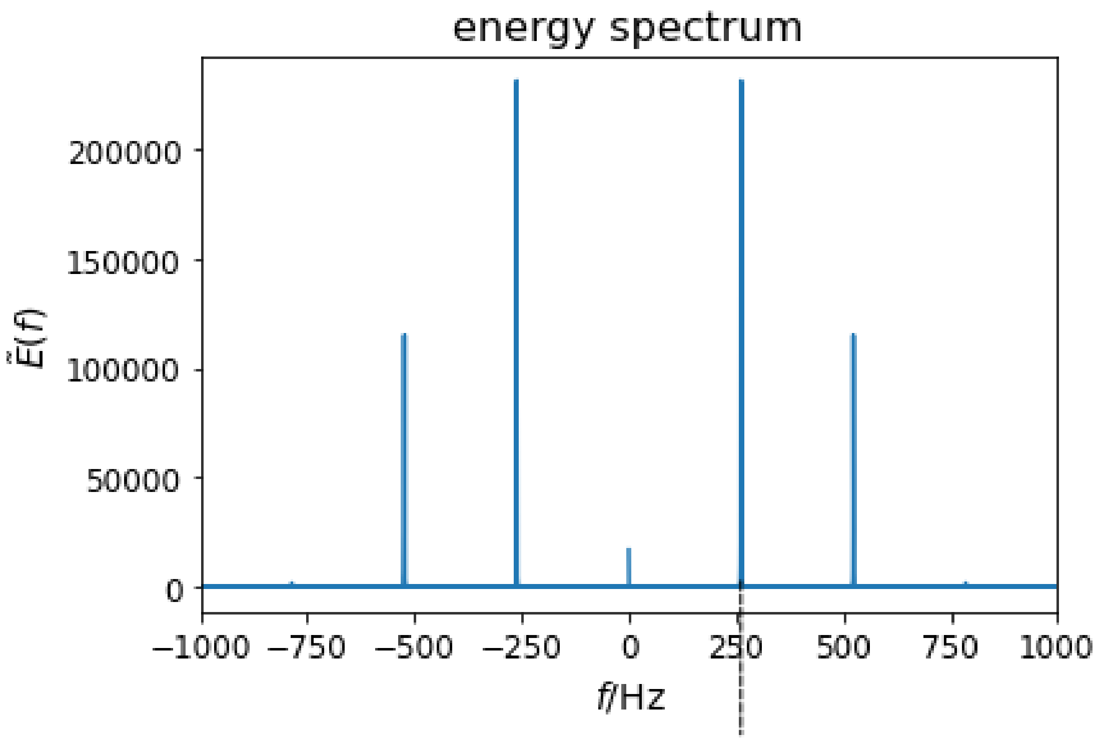


We can also zoom into a narrow interval such as $t \in [0.1\,\mathrm{s}, 0.14\,\mathrm{s}]$.

[ ]:
```
fig, ax = plt.subplots(figsize=(8, 4))

ax.set_title("C4 note from a piano")
ax.set_xlabel('$t/$s', fontsize=14)
ax.set_ylabel('$V(t)$', fontsize=14)
ax.set_xlim(0.1, 0.14)
ax.set_ylim(-3000, 3000)
ax.tick_params(axis='both', which='major', labelsize=12)

plt.plot(t, V)
plt.show()
```



```
[ ]: N = np.shape(V)[0]
     f = 1/(N*dt)*np.concatenate((np.arange(0, N/2, 1), np.arange(-N/2, 0, 1)))

     Vtilde = np.fft.fft(V)*dt # Fourier transform of V(t), Vtilde(omega)

     fig, ax = plt.subplots(figsize=(6, 4))

     ax.set_title('energy spectrum', fontsize=16)
     ax.set_xlabel('$f/$Hz', fontsize=14)
     ax.set_ylabel('$\\tilde{E}(f)$', fontsize=14)
     ax.set_xlim(-1000, 1000)
     #ax.set_ylim(0, 0.0005)
     ax.tick_params(axis='both', which='major', labelsize=12)
     #ax.set_yticks(np.arange(-2, 2.1, 1))
     #ax.set_xticks(np.arange(0, 0.049, 0.012))

     ax.plot(f, np.real(Vtilde*np.conjugate(Vtilde)))

     ax.annotate('', c='black',
                 xy=(261.63,-70000),  # location of the arrow tip
                 xytext=(261.63,5000),  # location of the text
                 arrowprops=dict(edgecolor='black', facecolor='red', arrowstyle='-',␣
      ↪linestyle='--'),
                 annotation_clip=False)

     plt.show()
```



1.6.1 References

1. Digital Audio Fundamentals, Audacity
2. Sound Examples, Dan Ellis, Columbia University
3. Playing and Recording Sound in Python, Real Python