Chapter 2. Linear Space and Linear Mapping

2.1. Linear spaces

fig. 2.1 Vector sum fig2-1-left.py

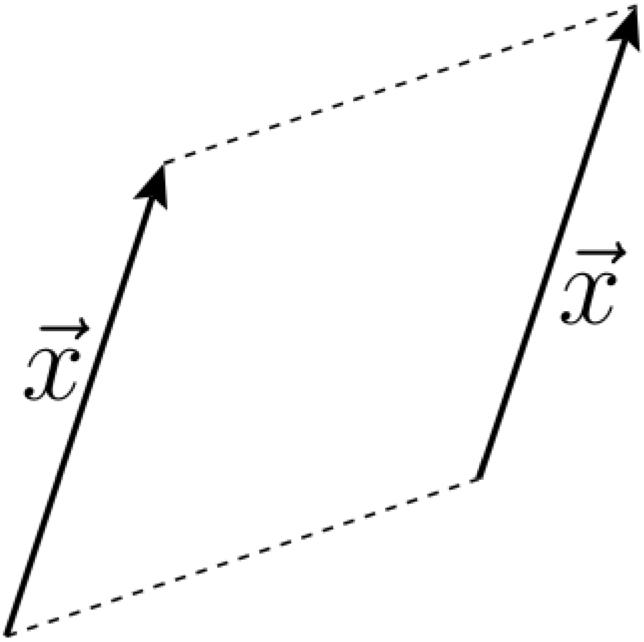


fig2-1-center.py

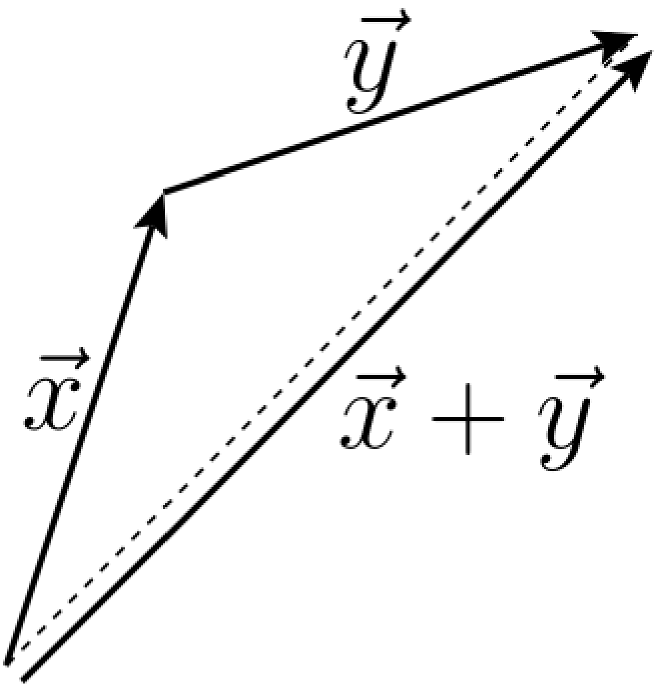
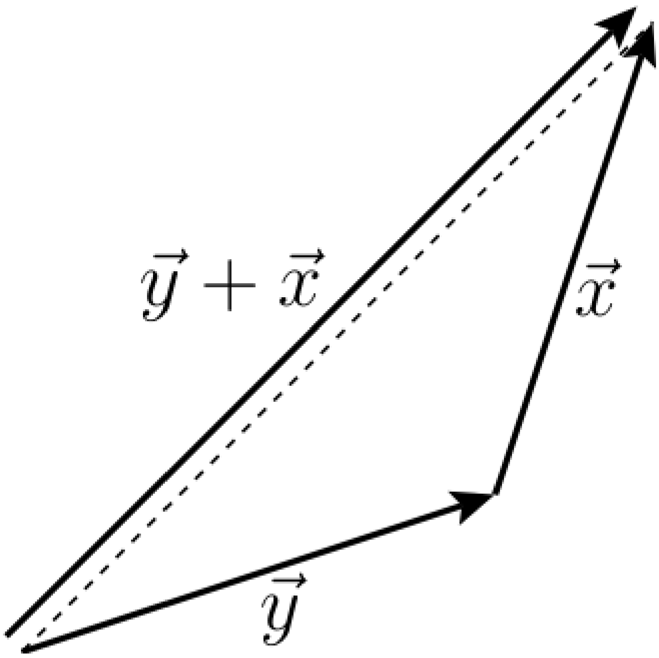


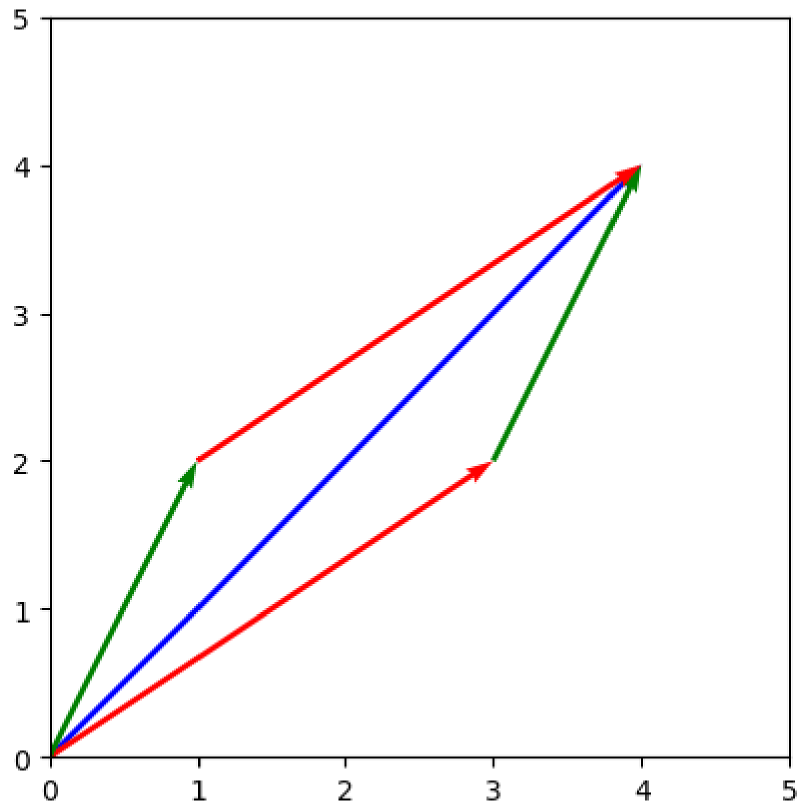
fig2-1-right.py



Program: vec2d.py

In [1]:
```
from numpy import array
import matplotlib.pyplot as plt

o, x, y = array([0, 0]), array([3, 2]), array([1, 2])
arrows = [(o, x + y, 'b'), (o, x, 'r'), (x, y, 'g'),
          (o, y, 'g'), (y, x, 'r')]
for p, v, c in arrows:
    plt.quiver(p[0], p[1], v[0], v[1],
               color=c, units='xy', scale=1)
plt.axis('scaled'), plt.xlim(0, 5), plt.ylim(0, 5)
```

Out[1]: ((-0.15000000000000002, 3.15, -0.1, 2.1), (0.0, 5.0), (0.0, 5.0))



Program: vec3d.py

In [1]:
```
from vpython import vec, arrow, mag

o = vec(0, 0, 0)
x, y, z = vec(1, 0, 0), vec(0, 1, 0), vec(0, 0, 1)
arrows = [(o, x + y), (x, y + z), (o, x + y + z), (o, x), (y, x),
          (z, x), (y + z, x), (o, y), (x, y), (z, y), (x + z, y),
          (o, z), (x, z), (y, z), (x + y, z)]
for p, v in arrows:
    arrow(pos=p, axis=v, color=v, shaftwidth=mag(v) / 50)
```

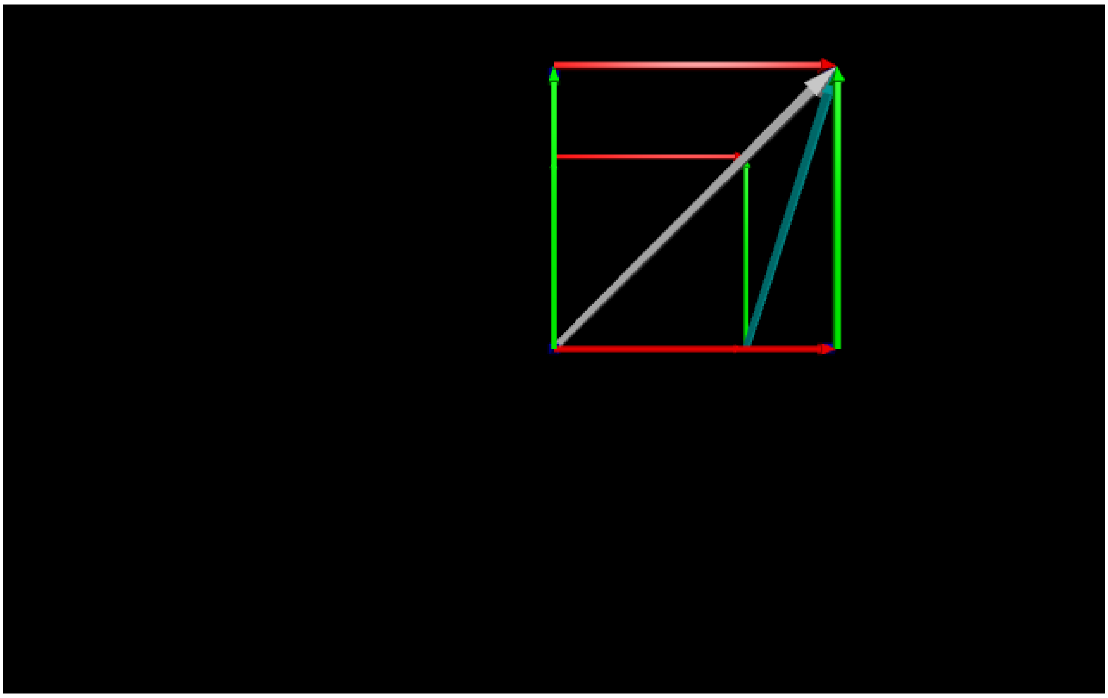


fig. 2.3 2-dimensional, 3-dimensional and n-dimensional vectors

fig2-3-left.py

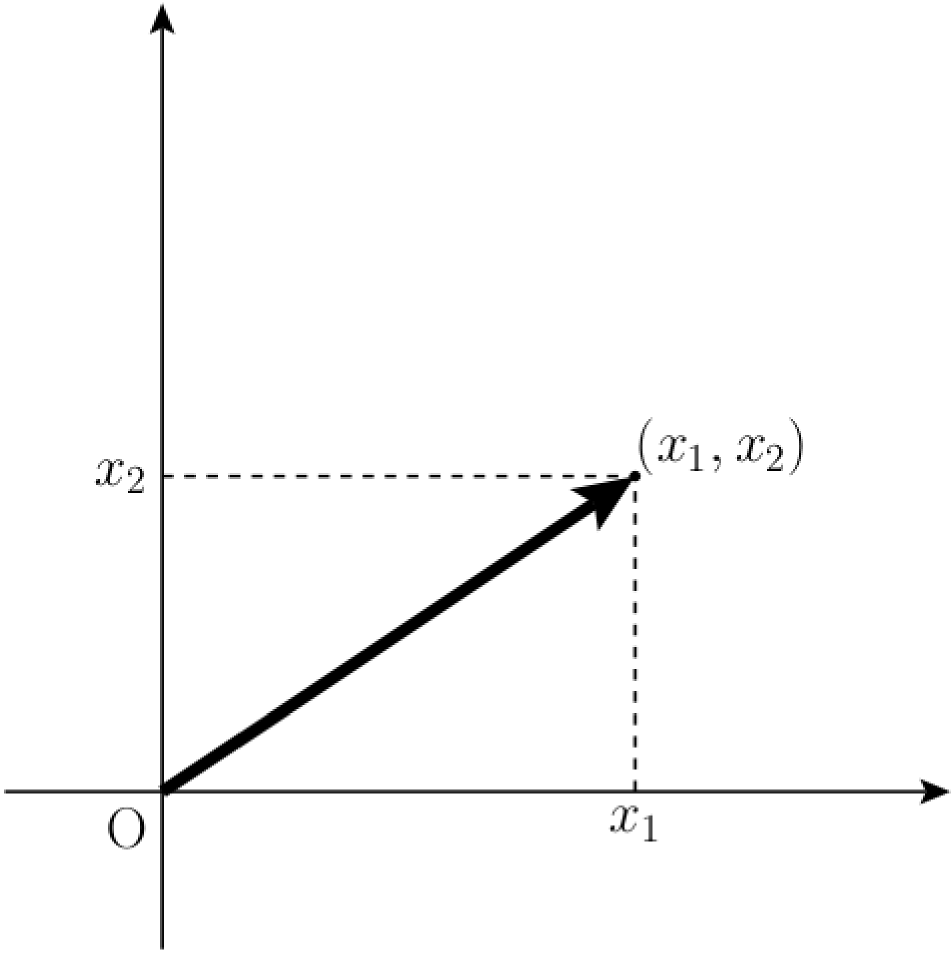


fig2-3-center.py

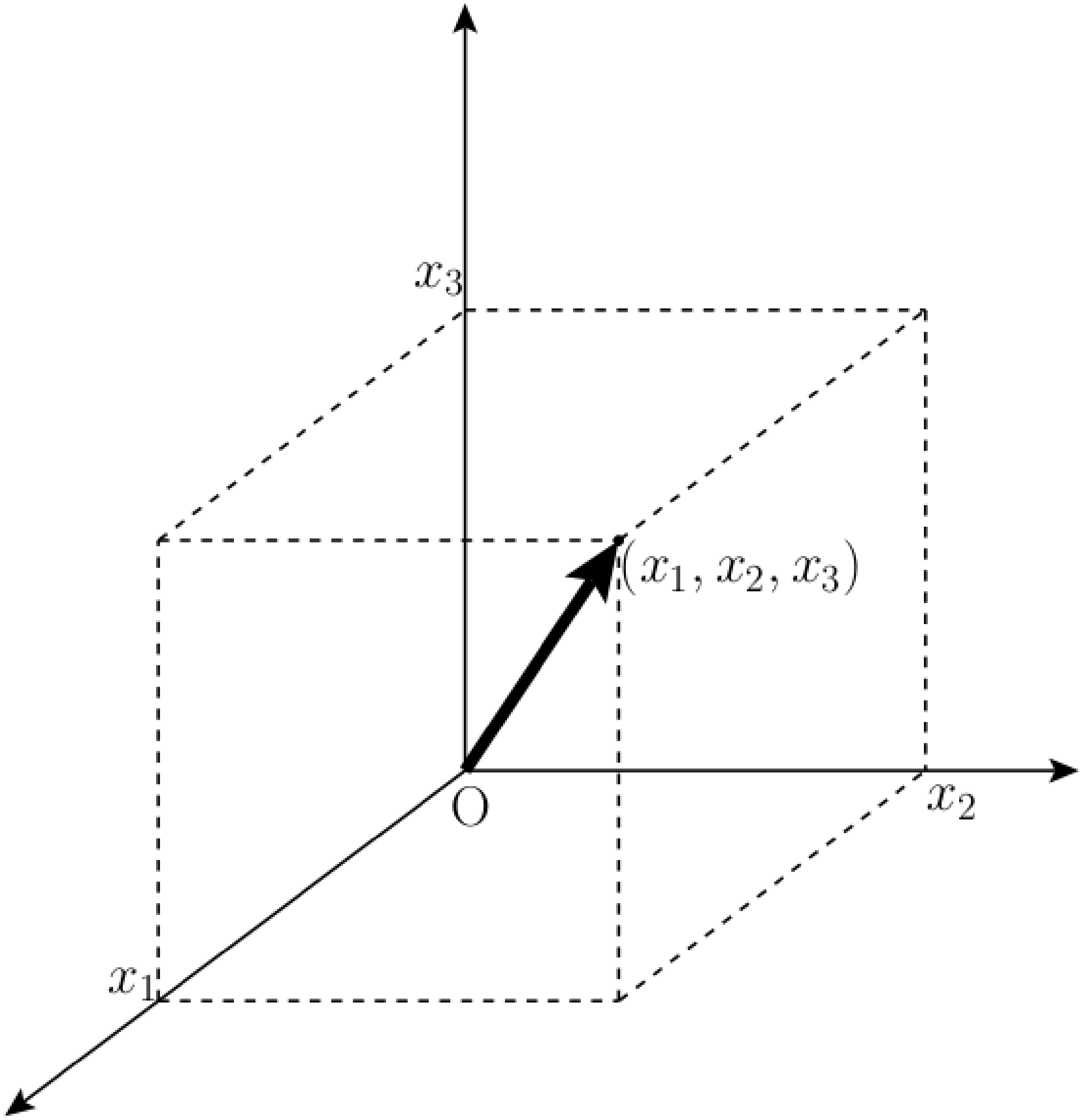
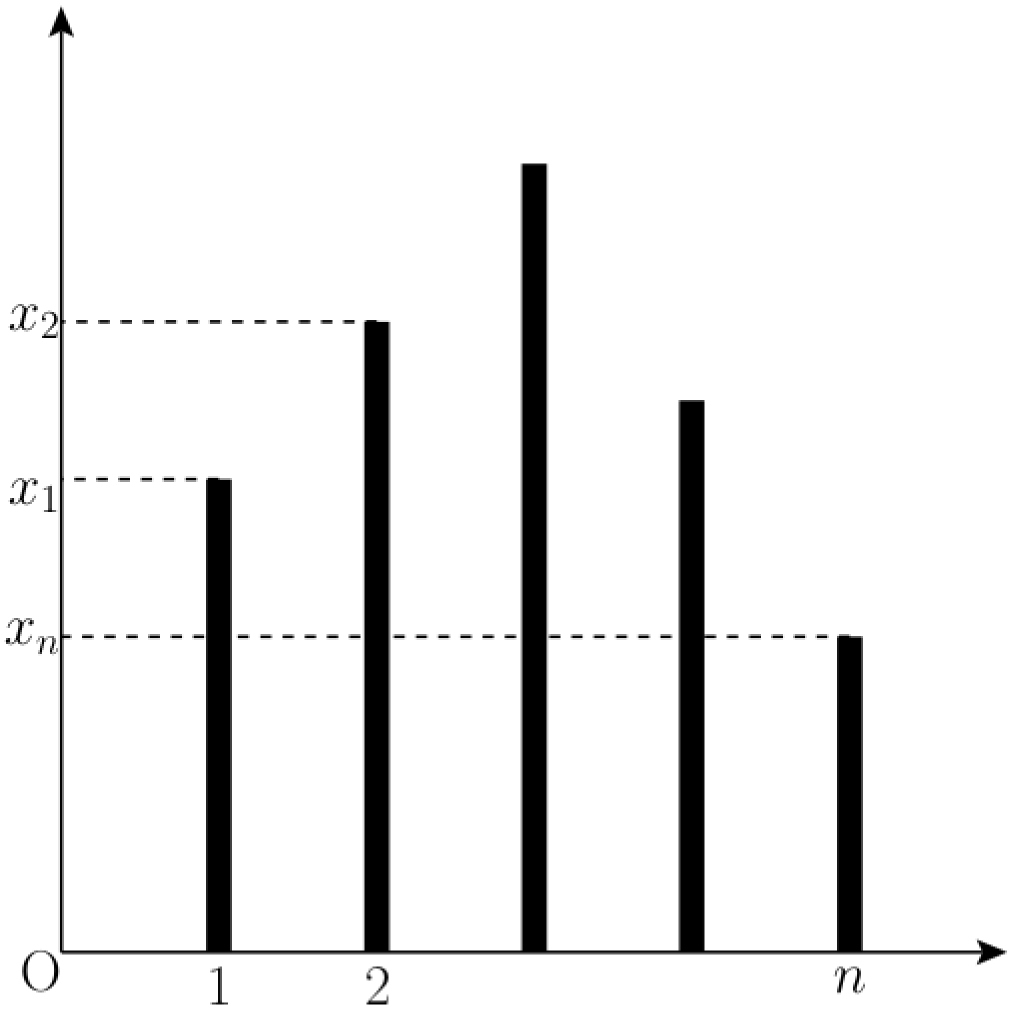


fig2-3-right.py



Program: func.py

In [1]:
```
from numpy import pi, sin, cos, linspace
import matplotlib.pyplot as plt

zero = lambda x: 0*x
f = lambda x: x**2 - 1
g = lambda x: 2*sin(2*x)
fig = plt.figure(figsize=(20, 5))
x = linspace(-pi, pi, 101)
ax1 = fig.add_subplot(131)
for y in [zero(x), f(x), g(x), f(x) + g(x)]:
    ax1.plot(x, y)
ax2 = fig.add_subplot(132)
for y in [zero(x), f(x), -f(x), f(x)/2]:
    ax2.plot(x, y)
ax3 = fig.add_subplot(133)
for y in [zero(x), g(x), -g(x), 3*g(x)]:
    ax3.plot(x, y)
```

fig. 2.4 Vector sum,schalar multiple, zero vector and inverse vecto on $\mathbb{R}^{[-\pi,\pi]}$

fig2-4-left.py

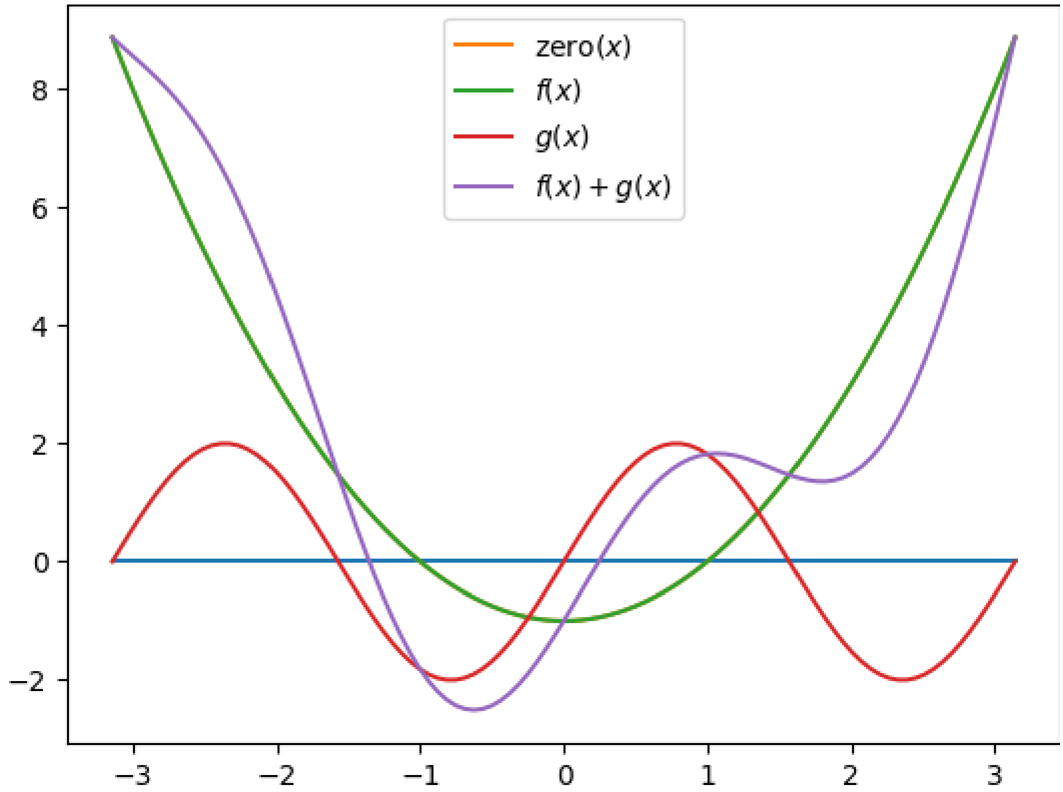


fig2-4-center.py

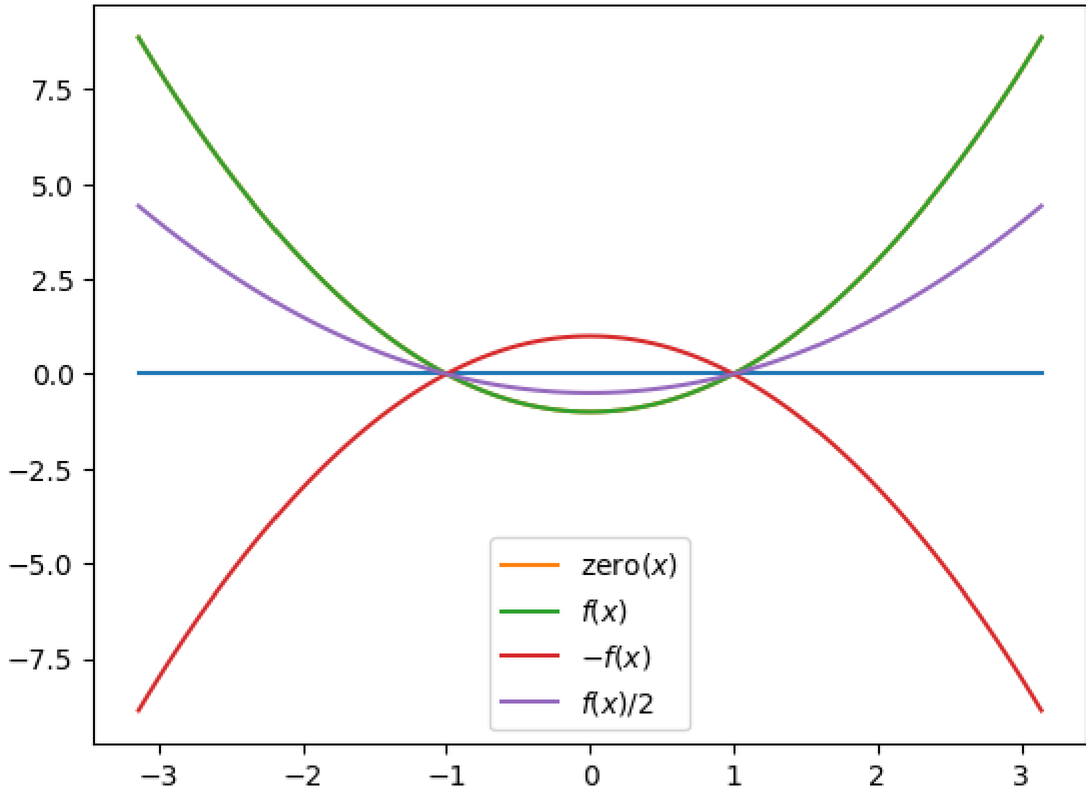
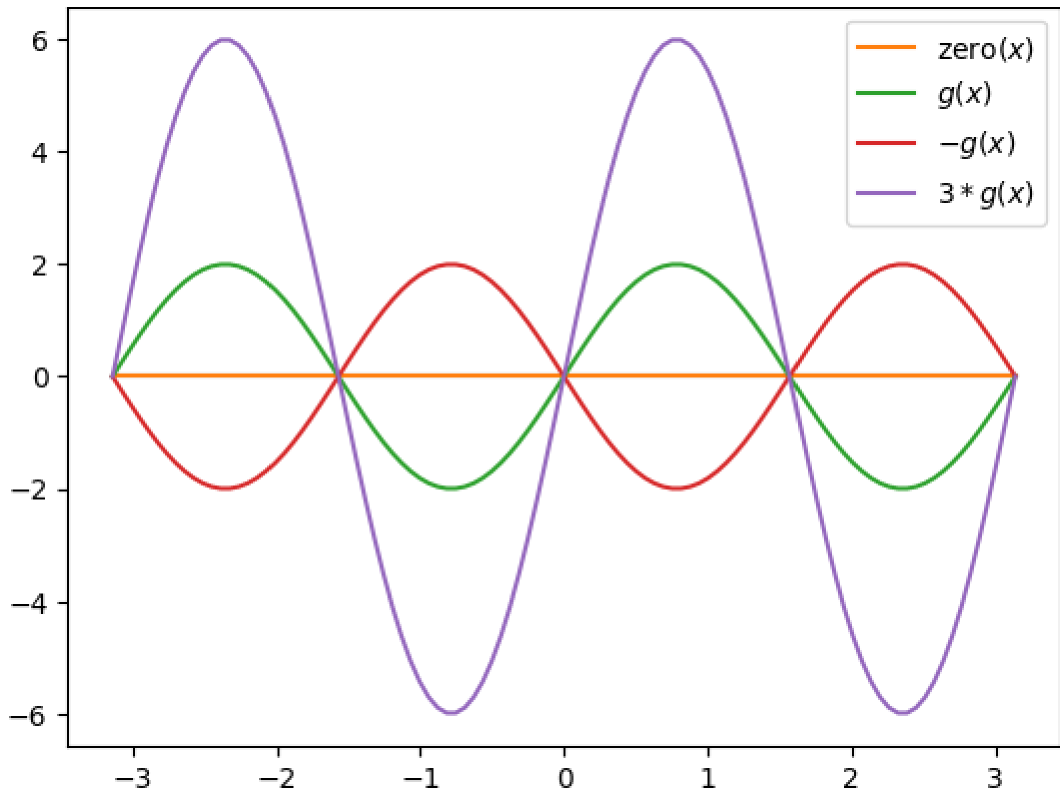


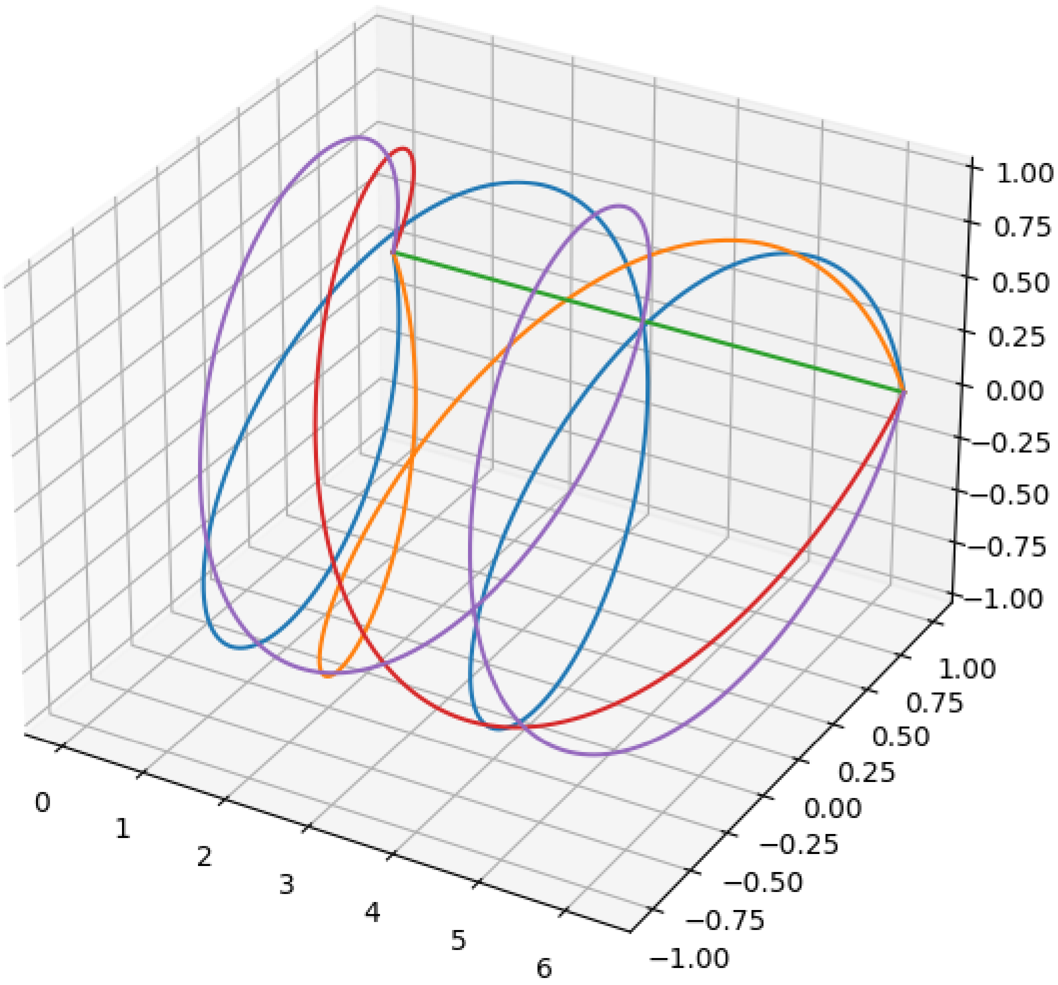
fig2-4-right.py



Program: cfunc.py

In [1]:
```
from numpy import exp, pi, linspace
import matplotlib.pyplot as plt

f = lambda n, t: exp(1j * n * t)
```

```
t = linspace(0, 2 * pi, 1001)

fig = plt.figure(figsize=(7, 7))
ax = fig.add_subplot(111, projection='3d')
for n in range(-2, 3):
    z = f(n, t)
    ax.plot(t, z.real, z.imag)
```



2.2. Subspaces

Empty

2.3. Linear mappings

fig. 2.6 Preservation of vector sum

fig2-6-left.py

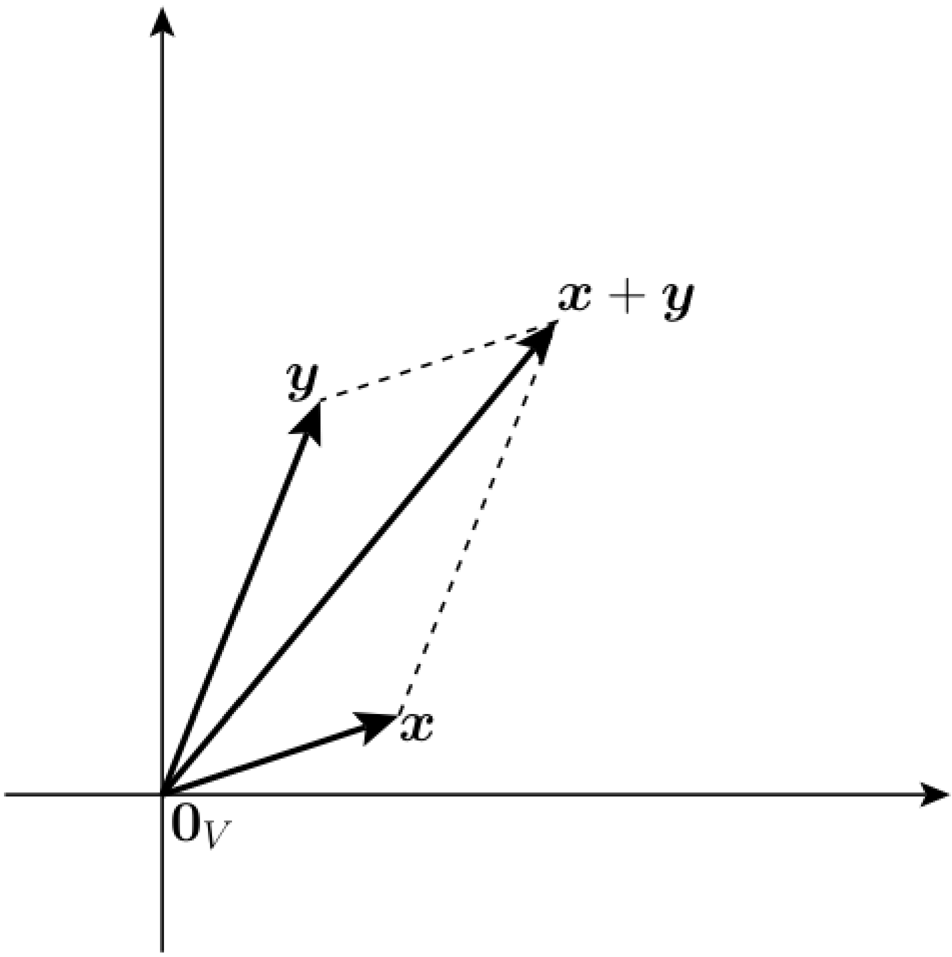


fig2-6-right.py

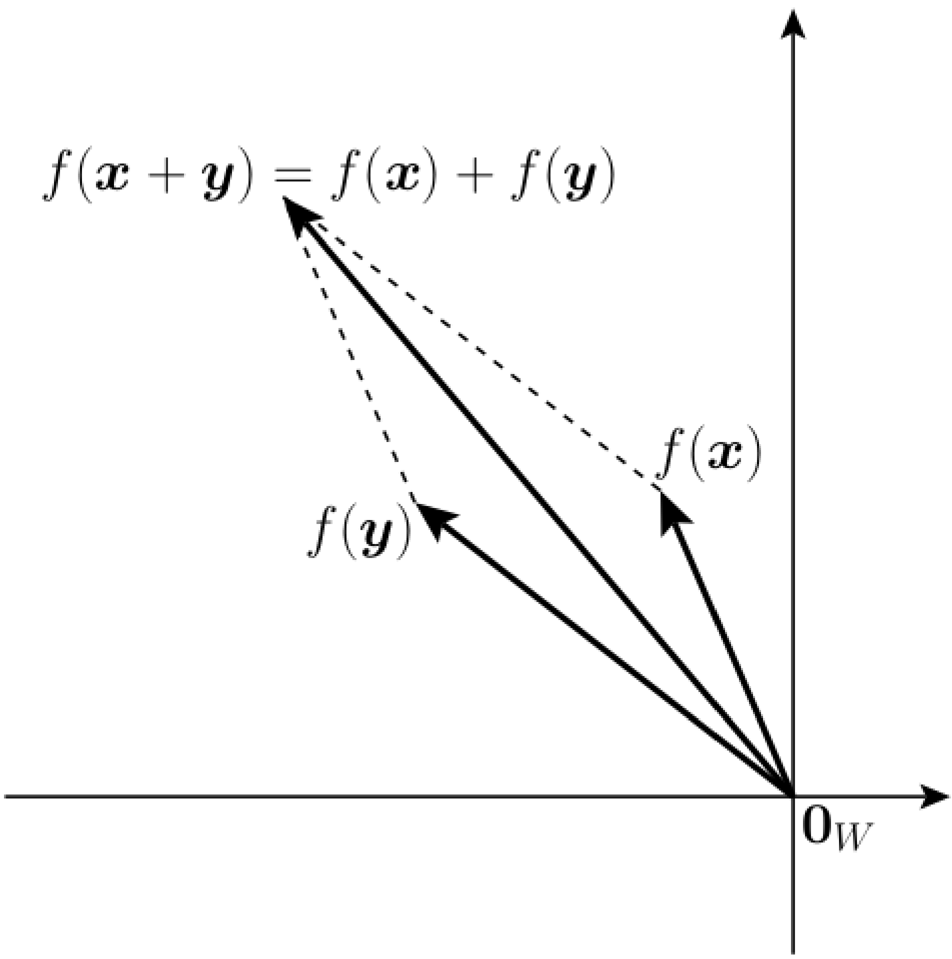
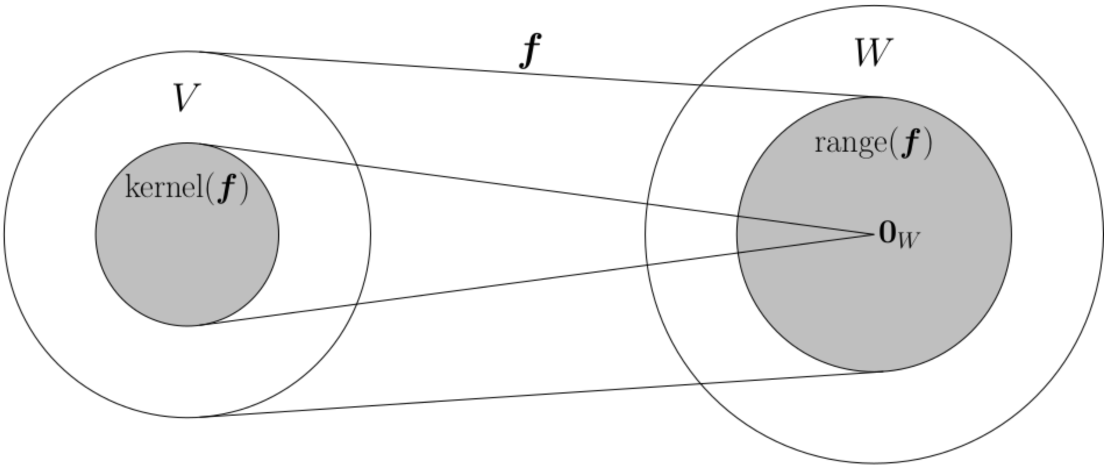
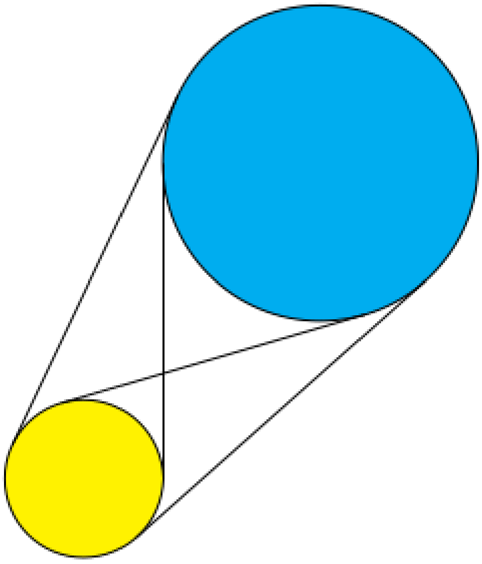


fig. 2.6 Kernel and range

fig2-7.py



tangent.py is used in the above as a library which draws a common tangent line between two given circles.



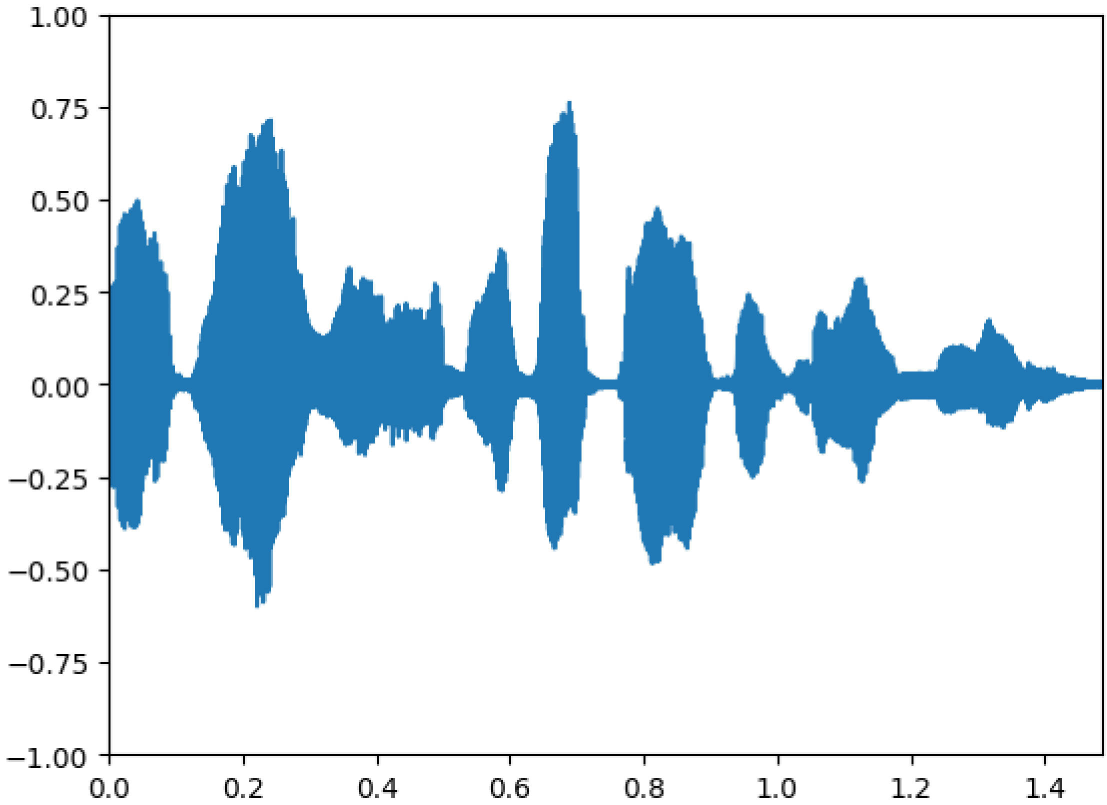
2.4. Application: Visualizing sounds

Program: sound2.py

In [1]:
```
import scipy.io.wavfile as wav
import numpy as np
import matplotlib.pyplot as plt

file_name = 'mono'
rate, Data = wav.read(file_name + '.wav')
print(rate, Data.shape)
n = len(Data)
t = n / rate
dt = 1 / rate
print(t, dt)

#x = np.arange(n) / n
```

```
x = np.arange(0, n*dt, dt)
#x = np.linspace(0, t, n, endpoint=False)
y = Data / 32768

if len(Data.shape) == 1:
    plt.plot(x, y)
    plt.xlim(0, t), plt.ylim(-1, 1)
elif len(Data.shape) == 2:
    fig, ax = plt.subplots(2)
    for i in range(2):
        ax[i].plot(x, y[:, i])
        ax[i].set_xlim(0, t), ax[i].set_ylim(-1, 1)
```

```
22050 (32768,)
1.486077097505669 4.5351473922902495e-05
```



Program: cord.py

In [1]:
```
from numpy import arange, pi, sin
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt

xmax, rate = 2, 22050
x = arange(0, xmax, 1 / rate)


def f(hz):
    return [sin(2 * pi * hz * x) * 0.9 for n in range(-2, 3)]


A = f(440.000000)
B = f(493.883301)
C = f(523.251131)
```

```
D = f(587.329536)
E = f(659.255114)
F = f(698.456463)
G = f(783.990872)
CEG = (C[2] + E[2] + G[2]) / 3
Data = (CEG * 32768).astype('int16')
wav.write('CEG.wav', rate, Data)
for y in [C[2], E[2], G[2], CEG]:
    plt.plot(x, y)
plt.xlim(1, 1.01)
```

Out[1]: (1.0, 1.01)