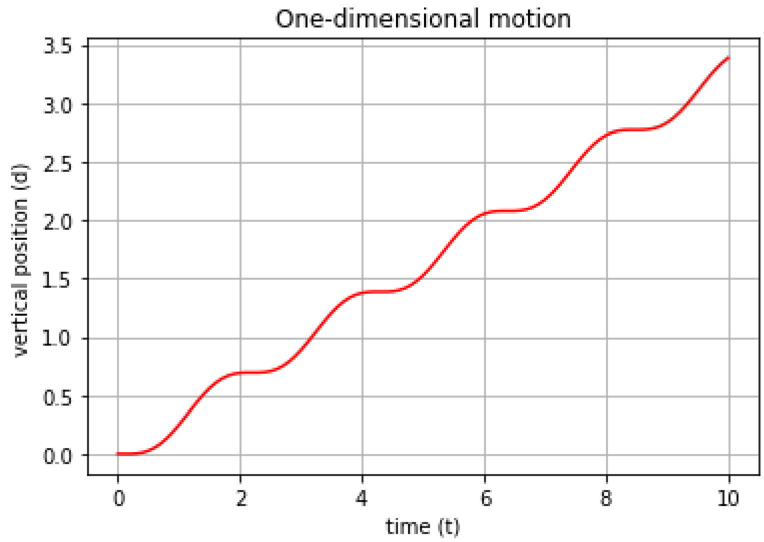
In [33]:

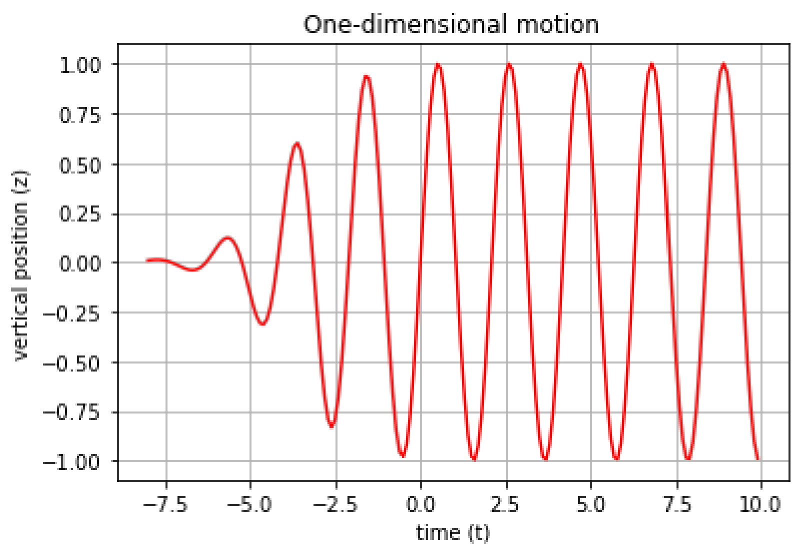
```
import numpy as np
import matplotlib.pyplot as plt
import math

def electron():
    # work in MKS units
    T = 2 * math.pi  # seconds
    Dt = 0.1  # seconds
    a = 1
    w = 3
    N = 100
    intX = 0
    intY = 0
    t = np.zeros(N, )
    z = np.zeros(N, )

#a)
    pos = 0
    pos = -(1/3)*T + (1/9)*(math.sin(3*T))

    print("a) The electron position is: ", pos);

#b)
    total = 0
    velocity = 0
    position = 0
    accel = 0
    for T in range(0, 63):
        position = position + velocity * Dt
        velocity = velocity + accel * Dt
        accel = -(math.sin(3*T*Dt))

    print("b) The estimated value is: ", position);

    dif = pos - position

    print("The difference is: ", dif);

    velocity = 0
    position = 0
    accel = 0

    time = np.zeros(101,)
    d = np.zeros(101,)

    for t in range(0, 101):
        position = position + velocity * Dt
        velocity = velocity + accel * Dt
        accel = math.sin(3 * t/10)
        time[t] = t/10
        d[t] = position


    # fig = plt.figure(figsize=(10, 6))  # create a fig object named "fig"
    fig = plt.figure()  # create a fig object named "fig"
    Plot = fig.add_subplot(111)  # create an axis object within "fig" named
```

```
    Plot.plot(time, d, 'r-')  # Add a curve described by the arrays t and z
    Plot.set(xlabel='time (t)', ylabel='vertical position (d)',  # add a ti
             title='One-dimensional motion')
    Plot.grid()  # superimpose a grid on Plot
    # fig.savefig("test.png") # save the figure as a png file.
    plt.show(fig)  # Plot the figure here.
    # Use a breakpoint in the code line below to debug your script.


def electron2():
    # work in MKS units
    Dt = 0.1  # seconds
    a = 1  # fix initial position
    w = 3  # fix initial velocity
    N = 180
    t = np.zeros(N, )  # numpy array holding the Nstep+1 values of the time
    z = np.zeros(N, )  # numpy array holding the Nstep+1 values of the vert
    time = 0

    for n in range(0, 80):
        t[n] = (Dt * n) - 8.0
        time = (n - 80) * 0.1
        tenT = math.pow(10, 0.5*time)
        z[n] = tenT / (0.01 + tenT) * (math.sin(3*(time)))

    for n in range(80, 180):
        t[n] = (Dt * n) - 8.0
        time = (n - 80) * 0.1
        z[n] = math.sin(3 * time)

    print("c)");

    # fig = plt.figure(figsize=(10, 6))  # create a fig object named "fig"
    fig = plt.figure()  # create a fig object named "fig"
    Plot = fig.add_subplot(111)  # create an axis object within "fig" named
    Plot.plot(t, z, 'r-')  # Add a curve described by the arrays t and z to
    Plot.set(xlabel='time (t)', ylabel='vertical position (z)',  # add a ti
             title='One-dimensional motion')
    Plot.grid()  # superimpose a grid on Plot
    # fig.savefig("test.png") # save the figure as a png file.
    plt.show(fig)  # Plot the figure here.




# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    electron()
    electron2()
```

```
a) The electron position is:  -2.0943951023931953
b) The estimated value is:  -2.0765319197737755
The difference is:  -0.01786318261941977
```


One-dimensional motion
vertical position (d)
time (t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
2
4
6
8
10

c)


One-dimensional motion
vertical position (z)
time (t)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [ ]: