In [6]:

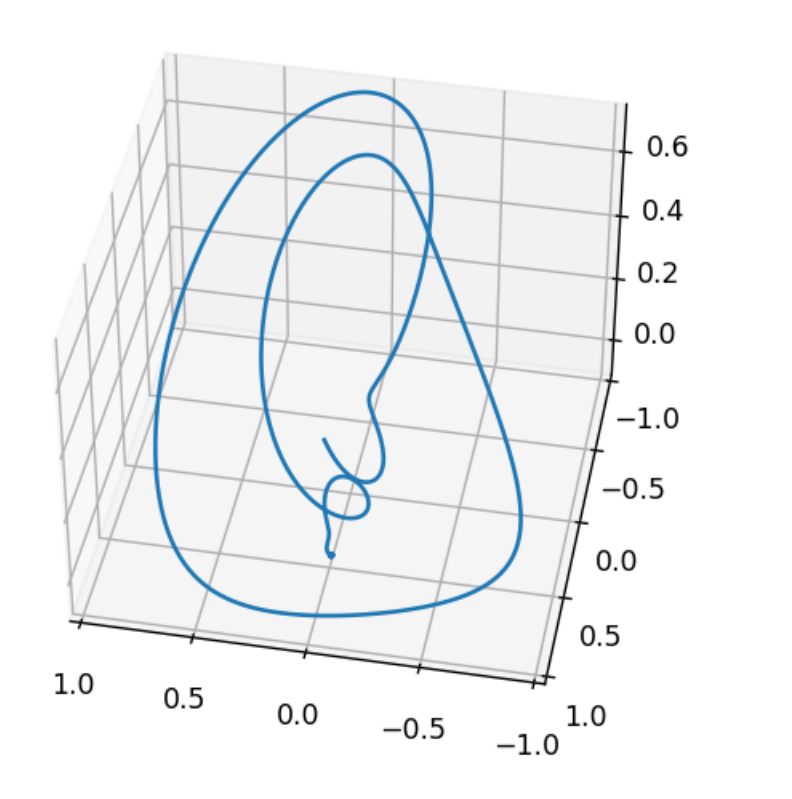
```python
import numpy as np
par_min = 0
par_max = 0.2
delta_par = 0.1

arg_min = 0
arg_max = 0.2
delta_arg = 0.1

max_attract = 100
min_attract = -100

h = 0.01
t_max = 10.0
from array import *
t = np.arange(0.0, t_max, h)

def x_t(x, y, z, w):
    return (-y - z - w)
def y_t(x, y, z, w):
    return (-x)
def z_t(x, y, z, w, a, b):
    return (a * (y - y**2) - b*z)
def w_t(x, y, z, w, c, d):
    return (c * (z/2 - z**2) - d*w)

par_min_int = int(par_min/delta_par)
par_max_int = int(par_max/delta_par)
arg_min_int = int(arg_min/delta_arg)
arg_max_int = int(arg_max/delta_arg)

x = array('d',[0])
y = array('d',[0])
z = array('d',[0])
w = array('d',[0])

for a1 in range (par_min_int, par_max_int):
    for b1 in range (par_min_int, par_max_int):
        print(b1)
        for c1 in range (par_min_int, par_max_int):
            for d1 in range (par_min_int, par_max_int):
                for x_01 in range (arg_min_int, arg_max_int):
                    for y_01 in range (arg_min_int, arg_max_int):
                        for z_01 in range (arg_min_int, arg_max_int):
                            for w_01 in range (arg_min_int, arg_max_int):
                                a = a1*delta_par
                                b = b1*delta_par
                                c = c1*delta_par
                                d = d1*delta_par
                                x_0 = x_01*delta_arg
                                y_0 = y_01*delta_arg
                                z_0 = z_01*delta_arg
                                w_0 = w_01*delta_arg

                                x = [x_0]
                                y = [y_0]
                                z = [z_0]
                                w = [w_0]
                                for i in range(1, int(t_max / h)):
                                    if (i == int(t_max / h) - 1):
                                        print('a={}, b={}, c={}, d={}, x_0={}, y_0={}, z_0={}, w_0={}'.format(a
                                    new_thing = x[i - 1] + h * x_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1])
                                    if new_thing < max_attract and new_thing > min_attract :
                                        x.append(new_thing)
                                    else:
                                        break
                                    new_thing = y[i - 1] + h * y_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1])
                                    if new_thing < max_attract and new_thing > min_attract :
                                        y.append(new_thing)
                                    else:
                                        break
                                    new_thing = z[i - 1] + h * z_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], a, b
                                    if new_thing < max_attract and new_thing > min_attract :
                                        z.append(new_thing)
                                    else:
                                        break
                                    new_thing = w[i - 1] + h * w_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], c, d
                                    if new_thing < max_attract and new_thing > min_attract :
                                        w.append(new_thing)
                                    else:
                                        break
```

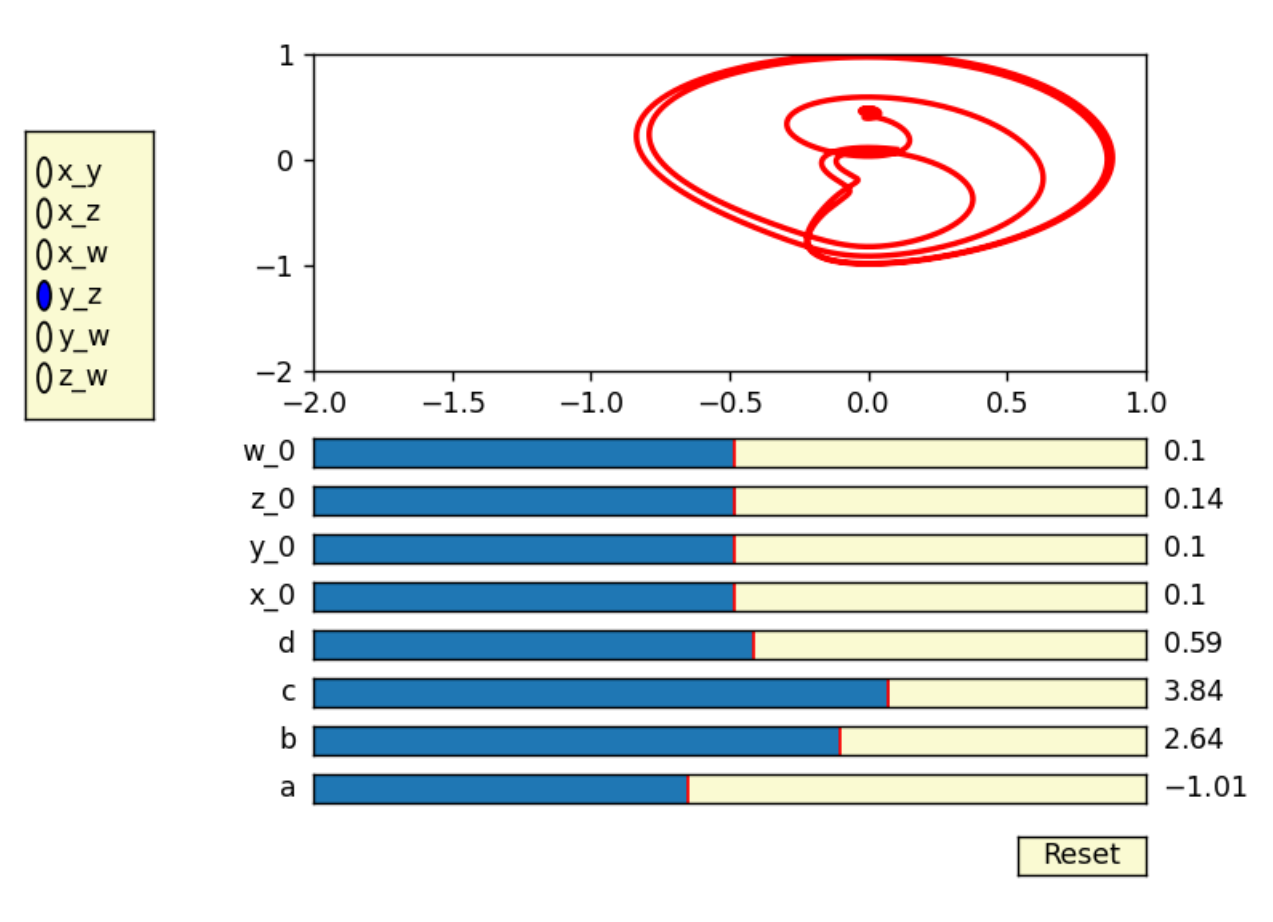
```
0
a=0.0, b=0.0, c=0.0, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.0, d=0.0, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.0, c=0.0, d=0.0, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.0, c=0.0, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.0, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.0, d=0.1, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.0, c=0.0, d=0.1, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.0, c=0.0, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.0, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.0, c=0.1, d=0.0, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.1, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.0, c=0.1, d=0.1, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.0, c=0.1, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
1
a=0.0, b=0.1, c=0.0, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.0, d=0.0, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.1, c=0.0, d=0.0, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.1, c=0.0, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.0, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.0, d=0.1, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.1, c=0.0, d=0.1, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.1, c=0.0, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.0, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.1, c=0.1, d=0.0, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.1, x_0=0.1, y_0=0.0, z_0=0.0, w_0=0.1
a=0.0, b=0.1, c=0.1, d=0.1, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.0, b=0.1, c=0.1, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
0
a=0.1, b=0.0, c=0.0, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.0, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.0, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.0, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.1, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.1, d=0.0, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
a=0.1, b=0.0, c=0.1, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.0, c=0.1, d=0.1, x_0=0.1, y_0=0.0, z_0=0.1, w_0=0.0
1
a=0.1, b=0.1, c=0.0, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.0, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.0, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.0, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.1, d=0.0, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.1, d=0.0, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.1, d=0.1, x_0=0.0, y_0=0.0, z_0=0.0, w_0=0.0
a=0.1, b=0.1, c=0.1, d=0.1, x_0=0.1, y_0=0.1, z_0=0.0, w_0=0.0
```

In [8]:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, RadioButtons
%matplotlib notebook
a0 = -1.01
b0 = 2.64
c0 = 3.8
d0 = 0.59
a = a0
b = b0
c = c0
d = d0
h = 0.01
t_max = 1000.0
from array import *
t = np.arange(0, t_max, h)
x_0 = 0.1
y_0 = 0.1
z_0 = 0.1
w_0 = 0.1
x = array('d',[x_0]);
y = array('d',[y_0]);
z = array('d',[z_0]);
w = array('d',[w_0]);

def x_t(x, y, z, w):
    return (-y - z - w)
def y_t(x, y, z, w):
    return (x)
def z_t(x, y, z, w, a, b):
    return (a * (y - y**2) - b*z)
def w_t(x, y, z, w, c, d):
    return (c * (z/2 - z**2) - d*w)

for i in range(1, int(t_max / h)):
    x.append(x[i - 1] + h * x_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
    y.append(y[i - 1] + h * y_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
    z.append(z[i - 1] + h * z_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], a, b))
    w.append(w[i - 1] + h * w_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], c, d))
    if x[i] < -100 or x[i] > 100 or y[i] < -100 or y[i] > 100 or z[i] < -100 or z[i] > 100 or w[i] < -100 or w
        break
print(len(x))
print(len(y))
print(len(z))
print(len(w))
#print(x)
#print(y)
#print(z)
#print(w)
import matplotlib.pyplot as plt_3d
from mpl_toolkits.mplot3d import Axes3D
fig_3d = plt_3d.figure()
ax_3d = fig_3d.add_subplot(111, projection='3d')
ax_3d.plot(x, y, z, label='parametric curve')
plt_3d.axis([-1, 1, -1, 1])


fig, ax = plt.subplots()
plt.subplots_adjust(left=0.25, bottom=0.55)

l, = plt.plot(x, y, lw=2, color='red')
plt.axis([-2, 1, -2, 1])

axcolor = 'lightgoldenrodyellow'
ax_param_a = plt.axes([0.25, 0.1, 0.65, 0.03], facecolor=axcolor)
ax_param_b = plt.axes([0.25, 0.15, 0.65, 0.03], facecolor=axcolor)
ax_param_c = plt.axes([0.25, 0.2, 0.65, 0.03], facecolor=axcolor)
ax_param_d = plt.axes([0.25, 0.25, 0.65, 0.03], facecolor=axcolor)

s_param_a = Slider(ax_param_a, 'a', -10, 10.0, valinit=a0)
s_param_b = Slider(ax_param_b, 'b', -10, 10.0, valinit=b0)
s_param_c = Slider(ax_param_c, 'c', -10, 10.0, valinit=c0)
s_param_d = Slider(ax_param_d, 'd', -10, 10.0, valinit=d0)

ax_param_x0 = plt.axes([0.25, 0.3, 0.65, 0.03], facecolor=axcolor)
ax_param_y0 = plt.axes([0.25, 0.35, 0.65, 0.03], facecolor=axcolor)
ax_param_z0 = plt.axes([0.25, 0.4, 0.65, 0.03], facecolor=axcolor)
ax_param_w0 = plt.axes([0.25, 0.45, 0.65, 0.03], facecolor=axcolor)

s_param_x0 = Slider(ax_param_x0, 'x_0', -10, 10.0, valinit=x_0)
s_param_y0 = Slider(ax_param_y0, 'y_0', -10, 10.0, valinit=y_0)
s_param_z0 = Slider(ax_param_z0, 'z_0', -10, 10.0, valinit=z_0)
s_param_w0 = Slider(ax_param_w0, 'w_0', -10, 10.0, valinit=w_0)




def update(val):
    plt_3d.show()
    a = s_param_a.val
    b = s_param_b.val
    c = s_param_c.val
    d = s_param_d.val

    x_0 = s_param_x0.val
    y_0 = s_param_y0.val
    z_0 = s_param_z0.val
    w_0 = s_param_w0.val

    x = [x_0]
    y = [y_0]
    z = [z_0]
    w = [w_0]

    for i in range(1, int(t_max / h)):
        x.append(x[i - 1] + h * x_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
        y.append(y[i - 1] + h * y_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
        z.append(z[i - 1] + h * z_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], a, b))
        w.append(w[i - 1] + h * w_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], c, d))
        if x[i] < -100 and x[i] > 100 and y[i] < -100 and y[i] > 100 and z[i] < -100 and z[i] > 100 and w[i] <
            break

    l.set_ydata(y)
    l.set_xdata(x)
    fig.canvas.draw_idle()


resetax = plt.axes([0.8, 0.025, 0.1, 0.04])
button = Button(resetax, 'Reset', color=axcolor, hovercolor='0.975')
s_param_a.on_changed(update)
s_param_b.on_changed(update)
s_param_c.on_changed(update)
s_param_d.on_changed(update)
s_param_x0.on_changed(update)
s_param_y0.on_changed(update)
s_param_z0.on_changed(update)
s_param_w0.on_changed(update)

def reset(event):
    s_param_a.reset()
    s_param_b.reset()
    s_param_c.reset()
    s_param_d.reset()
    s_param_x0.reset()
    s_param_y0.reset()
    s_param_z0.reset()
    s_param_w0.reset()
button.on_clicked(reset)
rax = plt.axes([0.025, 0.5, 0.1, 0.3], facecolor=axcolor)
radio = RadioButtons(rax, ('x_y', 'x_z', 'x_w', 'y_z', 'y_w','z_w'), active=0)


def type_graph(label):
    if label == 'x_y':
        l.set_ydata(y)
        l.set_xdata(x)
    elif label == 'x_z':
        l.set_ydata(z)
        l.set_xdata(x)
    elif label == 'x_w':
        l.set_ydata(w)
        l.set_xdata(x)
    elif label == 'y_z':
        l.set_ydata(z)
        l.set_xdata(y)
    elif label == 'y_w':
        l.set_ydata(w)
        l.set_xdata(y)
    elif label == 'z_w':
        l.set_ydata(w)
        l.set_xdata(z)
    fig.canvas.draw_idle()

radio.on_clicked(type_graph)
```

```
100000
100000
100000
100000
```

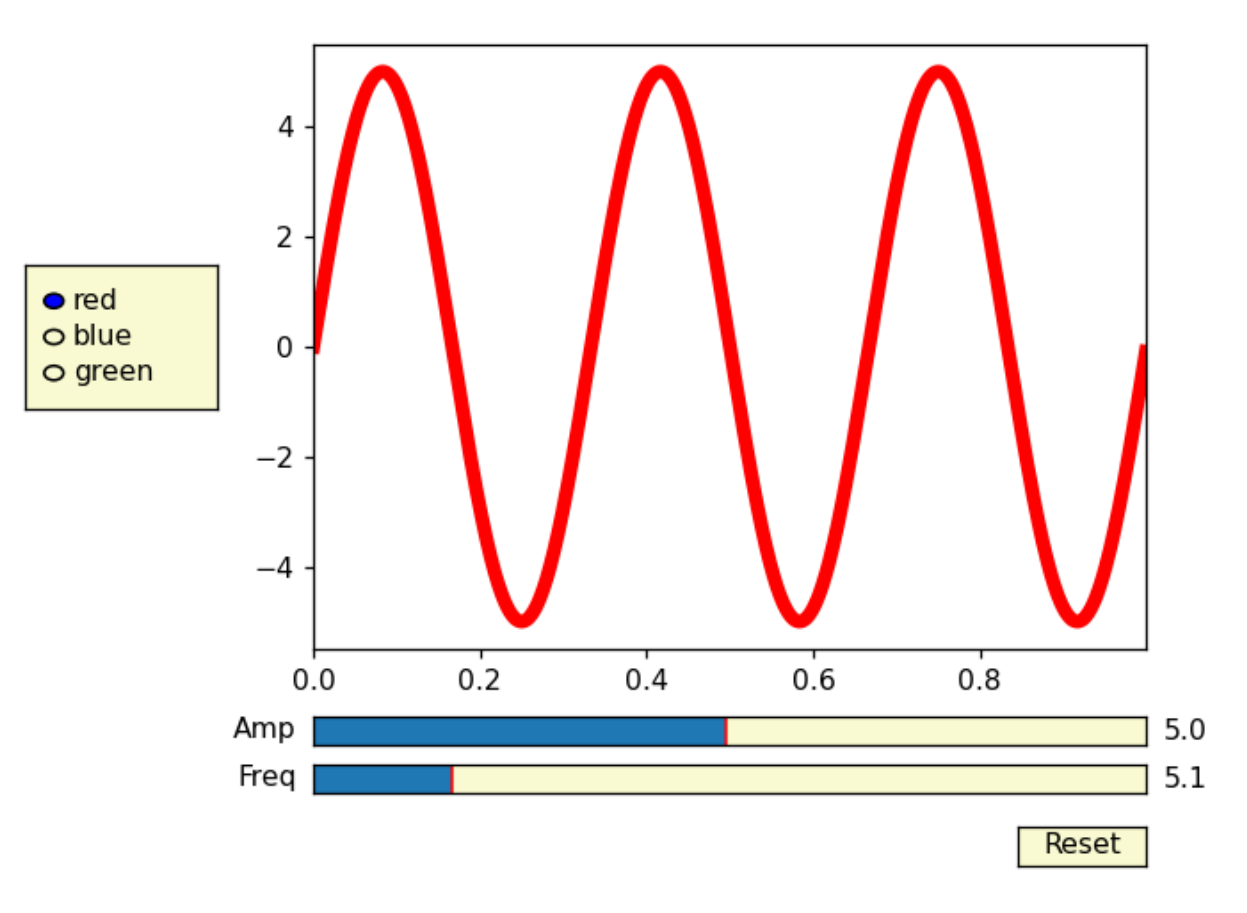




Out[8]: 0

In [1]:

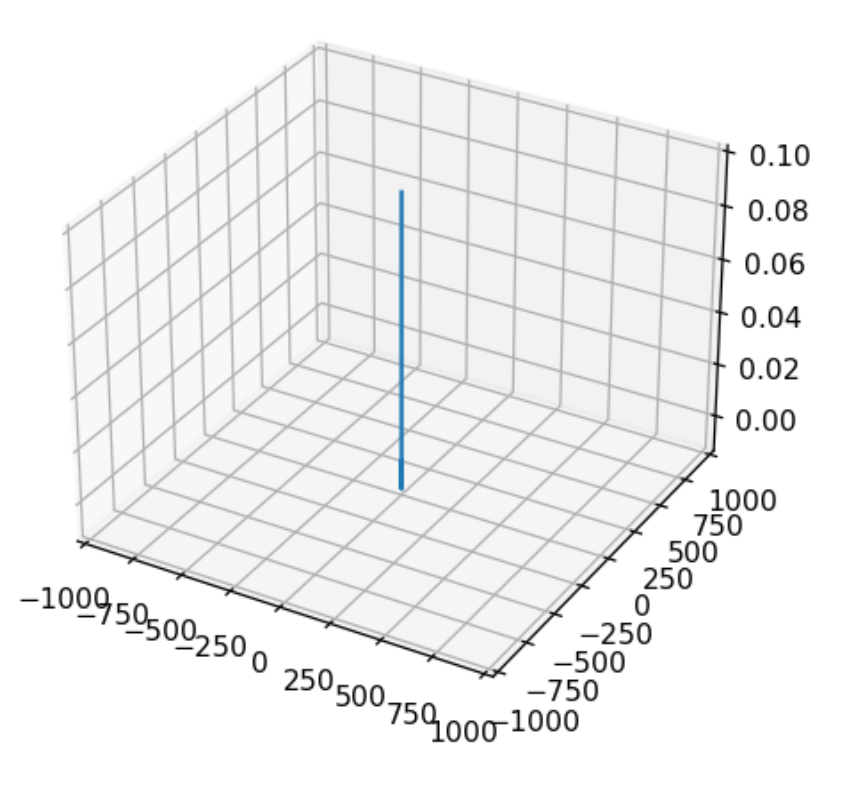
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, RadioButtons
%matplotlib notebook

fig, ax = plt.subplots()
plt.subplots_adjust(left=0.25, bottom=0.25)
t = np.arange(0.0, 1.0, 0.001)
a0 = 5
f0 = 3
delta_f = 5.0
s = a0 * np.sin(2 * np.pi * f0 * t)
l, = plt.plot(t, s, lw=5)
ax.margins(x=0)

axcolor = 'lightgoldenrodyellow'
axfreq = plt.axes([0.25, 0.1, 0.65, 0.03], facecolor=axcolor)
axamp = plt.axes([0.25, 0.15, 0.65, 0.03], facecolor=axcolor)

sfreq = Slider(axfreq, 'Freq', 0.1, 30.0, valinit=f0, valstep=delta_f)
samp = Slider(axamp, 'Amp', 0.1, 10.0, valinit=a0)


def update(val):
    amp = samp.val
    freq = sfreq.val
    l.set_ydata(amp*np.sin(2*np.pi*freq*t))
    fig.canvas.draw_idle()


sfreq.on_changed(update)
samp.on_changed(update)

resetax = plt.axes([0.8, 0.025, 0.1, 0.04])
button = Button(resetax, 'Reset', color=axcolor, hovercolor='0.975')


def reset(event):
    sfreq.reset()
    samp.reset()
button.on_clicked(reset)

rax = plt.axes([0.025, 0.5, 0.15, 0.15], facecolor=axcolor)
radio = RadioButtons(rax, ('red', 'blue', 'green'), active=0)


def colorfunc(label):
    l.set_color(label)
    fig.canvas.draw_idle()
radio.on_clicked(colorfunc)

# Initialize plot with correct initial active value
colorfunc(radio.value_selected)

plt.show()
```

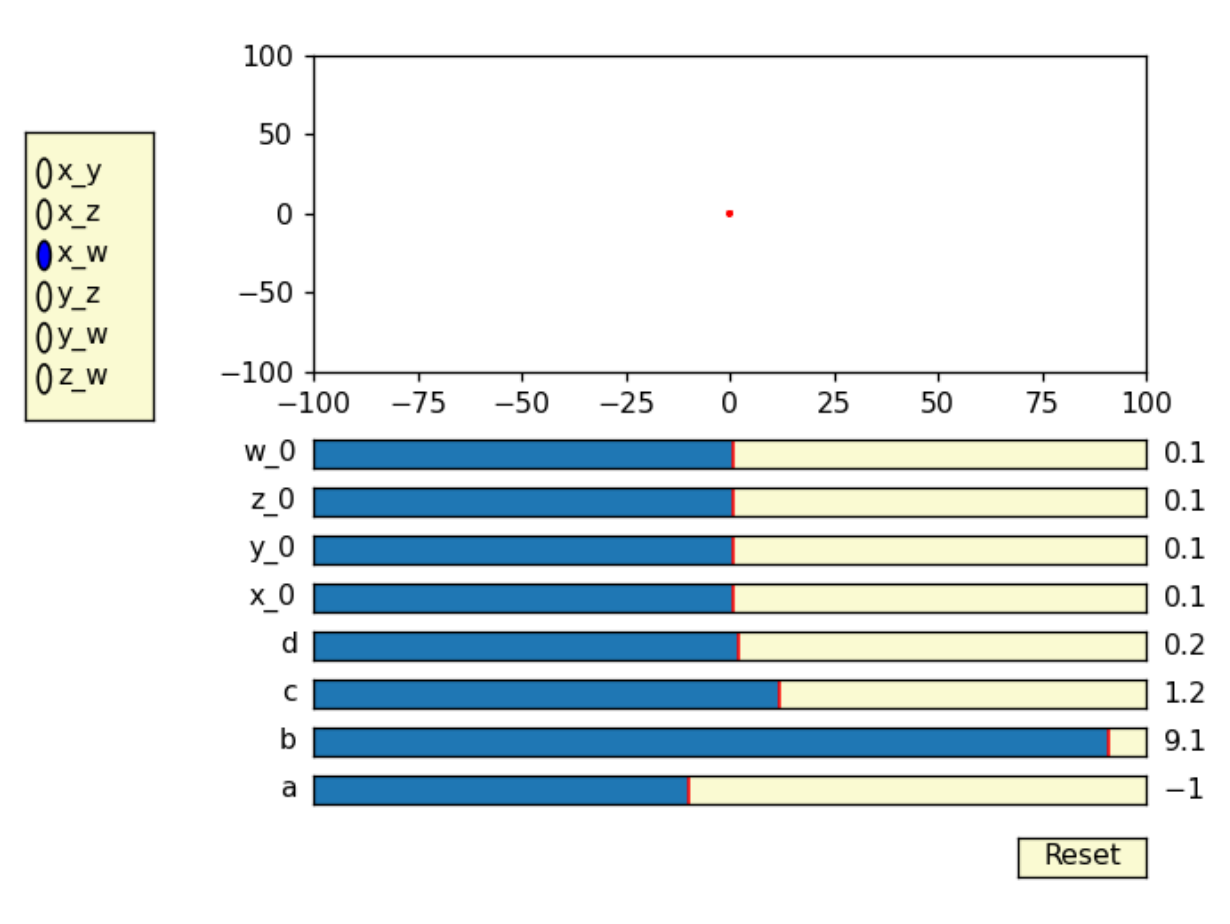


In [2]:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, RadioButtons
%matplotlib notebook
a0 = -1
b0 = 9.1
c0 = 1.2
d0 = 0.2
a = a0
b = b0
c = c0
d = d0
h = 0.01
t_max = 100.0
from array import *
t = np.arange(0.0, t_max, h)
x_0 = 0.1
y_0 = 0.1
z_0 = 0.1
w_0 = 0.1
x = array('d',[x_0]);
y = array('d',[y_0]);
z = array('d',[z_0]);
w = array('d',[w_0]);

def x_t(x, y, z, w):
    return (-y - z - w)
def y_t(x, y, z, w):
    return (x)
def z_t(x, y, z, w, a, b):
    return (a * (y - y**2) - b*z)
def w_t(x, y, z, w, c, d):
    return (c * (z/2 - z**2) - d*w)

for i in range(1, int(t_max / h)):
    x.append(x[i - 1] + h * x_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
    y.append(y[i - 1] + h * y_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
    z.append(z[i - 1] + h * z_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], a, b))
    w.append(w[i - 1] + h * w_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], c, d))
    if x[i] < -100 or x[i] > 100 or y[i] < -100 or y[i] > 100 or z[i] < -100 or z[i] > 100 or w[i] < -100 or w
        break
print(len(x))
print(len(y))
print(len(z))
print(len(w))
#print(x)
#print(y)
#print(z)
#print(w)
import matplotlib.pyplot as plt_3d
from mpl_toolkits.mplot3d import Axes3D
fig_3d = plt_3d.figure()
ax_3d = fig_3d.add_subplot(111, projection='3d')
ax_3d.plot(x, y, z, label='parametric curve')
plt_3d.axis([-1000, 1000, -1000, 1000])


fig, ax = plt.subplots()
plt.subplots_adjust(left=0.25, bottom=0.55)

l, = plt.plot(x, y, lw=2, color='red')
plt.axis([-100, 100, -100, 100])

axcolor = 'lightgoldenrodyellow'
ax_param_a = plt.axes([0.25, 0.1, 0.65, 0.03], facecolor=axcolor)
ax_param_b = plt.axes([0.25, 0.15, 0.65, 0.03], facecolor=axcolor)
ax_param_c = plt.axes([0.25, 0.2, 0.65, 0.03], facecolor=axcolor)
ax_param_d = plt.axes([0.25, 0.25, 0.65, 0.03], facecolor=axcolor)

s_param_a = Slider(ax_param_a, 'a', -10, 10.0, valinit=a0)
s_param_b = Slider(ax_param_b, 'b', -10, 10.0, valinit=b0)
s_param_c = Slider(ax_param_c, 'c', -10, 10.0, valinit=c0)
s_param_d = Slider(ax_param_d, 'd', -10, 10.0, valinit=d0)

ax_param_x0 = plt.axes([0.25, 0.3, 0.65, 0.03], facecolor=axcolor)
ax_param_y0 = plt.axes([0.25, 0.35, 0.65, 0.03], facecolor=axcolor)
ax_param_z0 = plt.axes([0.25, 0.4, 0.65, 0.03], facecolor=axcolor)
ax_param_w0 = plt.axes([0.25, 0.45, 0.65, 0.03], facecolor=axcolor)

s_param_x0 = Slider(ax_param_x0, 'x_0', -10, 10.0, valinit=x_0)
s_param_y0 = Slider(ax_param_y0, 'y_0', -10, 10.0, valinit=y_0)
s_param_z0 = Slider(ax_param_z0, 'z_0', -10, 10.0, valinit=z_0)
s_param_w0 = Slider(ax_param_w0, 'w_0', -10, 10.0, valinit=w_0)




def update(val):
    plt_3d.show()
    a = s_param_a.val
    b = s_param_b.val
    c = s_param_c.val
    d = s_param_d.val

    x_0 = s_param_x0.val
    y_0 = s_param_y0.val
    z_0 = s_param_z0.val
    w_0 = s_param_w0.val

    x = [x_0]
    y = [y_0]
    z = [z_0]
    w = [w_0]

    for i in range(1, int(t_max / h)):
        x.append(x[i - 1] + h * x_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
        y.append(y[i - 1] + h * y_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1]))
        z.append(z[i - 1] + h * z_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], a, b))
        w.append(w[i - 1] + h * w_t(x[i - 1], y[i - 1], z[i - 1], w[i - 1], c, d))
        if x[i] < -100 and x[i] > 100 and y[i] < -100 and y[i] > 100 and z[i] < -100 and z[i] > 100 and w[i] <
            break

    l.set_ydata(y)
    l.set_xdata(x)
    fig.canvas.draw_idle()


resetax = plt.axes([0.8, 0.025, 0.1, 0.04])
button = Button(resetax, 'Reset', color=axcolor, hovercolor='0.975')
s_param_a.on_changed(update)
s_param_b.on_changed(update)
s_param_c.on_changed(update)
s_param_d.on_changed(update)
s_param_x0.on_changed(update)
s_param_y0.on_changed(update)
s_param_z0.on_changed(update)
s_param_w0.on_changed(update)

def reset(event):
    s_param_a.reset()
    s_param_b.reset()
    s_param_c.reset()
    s_param_d.reset()
    s_param_x0.reset()
    s_param_y0.reset()
    s_param_z0.reset()
    s_param_w0.reset()
button.on_clicked(reset)
rax = plt.axes([0.025, 0.5, 0.1, 0.3], facecolor=axcolor)
radio = RadioButtons(rax, ('x_y', 'x_z', 'x_w', 'y_z', 'y_w','z_w'), active=0)


def type_graph(label):
    if label == 'x_y':
        l.set_ydata(y)
        l.set_xdata(x)
    elif label == 'x_z':
        l.set_ydata(z)
        l.set_xdata(x)
    elif label == 'x_w':
        l.set_ydata(w)
        l.set_xdata(x)
    elif label == 'y_z':
        l.set_ydata(z)
        l.set_xdata(y)
    elif label == 'y_w':
        l.set_ydata(w)
        l.set_xdata(y)
    elif label == 'z_w':
        l.set_ydata(w)
        l.set_xdata(z)
    fig.canvas.draw_idle()

radio.on_clicked(type_graph)
```

```
10000
10000
10000
10000
```





Out[2]: 0

In [ ]: