2023-07-03_Interactive_02_Matplotlib_Interactive

July 3, 2023

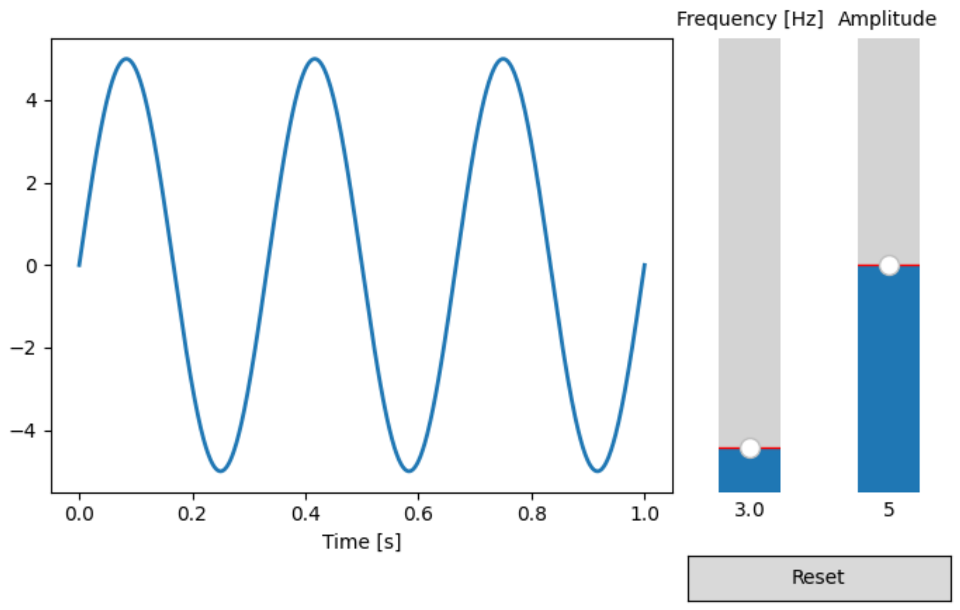
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
```

1 Matplotlib: interactive

[2]:
```
# adaped from: https://matplotlib.org/stable/gallery/widgets/slider_demo.html
```

[3]:
```
# The parametrized function to be plotted
def f(t, amplitude, frequency):
    return amplitude * np.sin(2 * np.pi * frequency * t)

# points at which to evaluate curve
t = np.linspace(0, 1, 1000)
# initial parameters
init_amplitude = 5
init_frequency = 3
```

[5]:
```
%matplotlib widget

fig=plt.figure(figsize=(8,5));
gs = fig.add_gridspec(3, 3,  width_ratios=(5,1,1), height_ratios=(5,0.5,0.5),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.05, hspace=0.05)

mainax=fig.add_subplot(gs[0,0])
line, = mainax.plot(t, f(t, init_amplitude, init_frequency), lw=2)
mainax.set_xlabel('Time [s]')

axfreq = fig.add_subplot(gs[0,1])
freq_slider = Slider(
    ax=axfreq,
    label='Frequency [Hz]',
    valmin=0.1,
    valmax=30,
    valinit=init_frequency,
    orientation="vertical"
    )

axamp = fig.add_subplot(gs[0,2])
amp_slider = Slider(
    ax=axamp,
    label="Amplitude",
    valmin=0,
    valmax=10,
    valinit=init_amplitude,
    orientation="vertical"
)

# callback function for sliders
def update(val):
    line.set_ydata(f(t, amp_slider.val, freq_slider.val))
    fig.canvas.draw_idle()
# register the update function with each slider
freq_slider.on_changed(update)
amp_slider.on_changed(update)


# reset button
resetax = fig.add_subplot(gs[2,1:3])
button = Button(resetax, 'Reset')

def reset(event):
    freq_slider.reset()
    amp_slider.reset()
button.on_clicked(reset)

plt.show()
```


Frequency [Hz]
Amplitude
3.0
5
Reset
Time [s]
4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```