In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from lava.lib.dl import slayer
def process_spikes(time_range: np.array, spikes: np.array):
    spike_mask = np.argwhere(spikes)
    toas = time_range[spike_mask]
    spike_amps = spikes[spike_mask]
    return list(zip(toas, spike_amps))

def plot_spike_graph(time_range: np.array, signal: list, spikes: np.array, real_comp:
    plt.figure()
    spike_info = process_spikes(time_range, spikes)

    plt.plot(time_range, [complex_threshold] * len(time_range), label ="threshold", co
    for toa, amp in spike_info:
        plt.vlines(toa,complex_threshold, complex_threshold+amp, color = "black", line

    plt.plot(time_range, real_comp.flatten(), label = "Re (z(t))")

    plt.plot(time_range, signal - 2, color = "orange", label = "input pulse")
    plt.legend()
    plt.title(title)
    plt.xlabel("Time")
    plt.ylabel("Voltage")
    plt.savefig(save_name)
    plt.show()
```

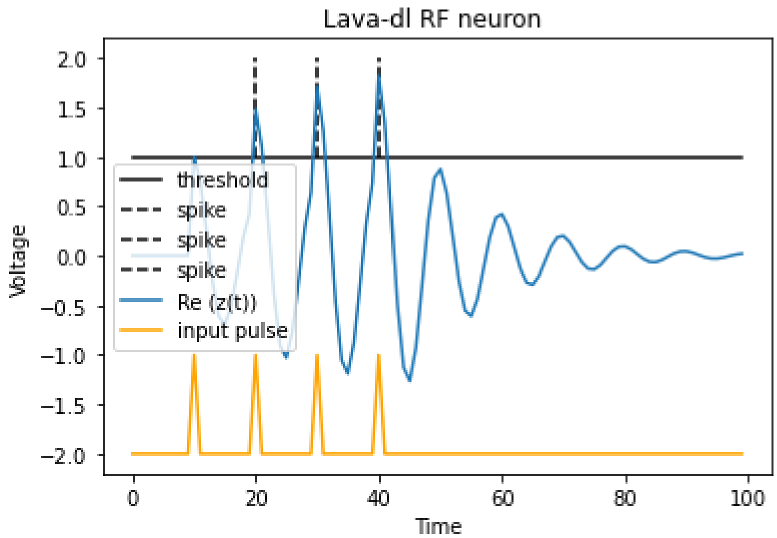
In [ ]:
```
import torch
n_neurons = 1 # 1 neuron
period = 10 # every ten timesteps complete and oscillation
alpha = .07
frequency = 1/period
radians_per_second = frequency * np.pi * 2
sin_decay = (1 - alpha) * np.sin(radians_per_second)
cos_decay = (1 - alpha) * np.cos(radians_per_second)

print(radians_per_second)
print(sin_decay, cos_decay)
vth = 1
num_steps = 100

# create spiking data that just spikes at timestep 1
incoming_spike_data = np.zeros((1, n_neurons, num_steps))
incoming_spike_data[:, :, [10, 20, 30, 40]] = 1  # neuron recieves input spikes at tim
re_input = torch.from_numpy(incoming_spike_data)
im_input = torch.zeros(re_input.shape)
rf_neuron_params = {
    'threshold': vth,
    "log_init": False,
    "shared_param": False,
    "period": period,
    "decay": alpha
}

neuron = slayer.neuron.rf.Neuron(**rf_neuron_params) #(1, 1/f * sampling_frequency, de
re, im = neuron.dynamics((re_input, im_input))  # simulation of subthreshold oscillati
spikes = neuron.forward(((re_input, im_input))).numpy().flatten()  # simulation with j
plot_spike_graph(np.arange(num_steps), incoming_spike_data.flatten(), spikes, re.numpy
```

```
0.6283185307179586
0.546640284632 0.7523858047687011
```



In [ ]:

In [ ]: