```
 1 import numpy as np
 2 from matplotlib import pyplot as plt
 3 from scipy import fft
 4 from scipy import signal
 5
 6 def bin_array(num, m):
 7     """Convert a positive integer
   num into an m-bit bit vector"""
 8     return np.array(list(np.
   binary_repr(num).zfill(m))).astype(
   np.bool_)
 9
10 # import 24 bit digital data
11 id_num = 2802461
12 Nbits = 24
13 tx_bin = bin_array(id_num, Nbits)
14
15 # initialise constants and variables
16 fc = 0.125
17 bit_period = 16
18 tx_mod = np.empty(0)
19
20 # BPSK modulation
21 for i in range(Nbits):
22     for j in range(bit_period):
23         tx_mod = np.append(tx_mod, (
   2 * tx_bin[i] - 1) *
24                                 np.cos(2
    * np.pi * fc * (i * bit_period + j
```

```
24 )))
25
26 plt.figure()
27 plt.plot(tx_mod)
28 plt.show()
29 plt.figure()
30 plt.plot(np.abs(fft.fft(tx_mod)))
31 plt.show()
32
33 # low-pass filter
34 numtaps = 32
35 cutoff = 0.1
36 b1 = signal.firwin(numtaps, cutoff)
37
38 # Digital filter frequency response
39 mixed = np.zeros(numtaps)
40 w1, h1 = signal.freqz(b1)
41
42 plt.title("Digital filter frequency
   response")
43 plt.plot(w1 / 2 / np.pi, 20 * np.
   log10(np.abs(h1)))
44 plt.ylabel("Amplitude Response/dB")
45 plt.xlabel("Frequency/sample rate")
46 plt.grid()
47 plt.show()
48
49 # Demodulation
50 rx_mixed = np.empty(0)
```

```
51 for i in range(Nbits):
52     for j in range(bit_period):
53         rx_mixed = np.append(
   rx_mixed, tx_mod[i * bit_period + j
   ] *
54                              np.cos(2
    * np.pi * fc * (i * bit_period + j
   )))
55
56 rx_lpf = signal.lfilter(b1, 1,
   rx_mixed)
57 rx_lpf = np.append(rx_lpf, np.ones(
   numtaps // 2))
58
59 plt.figure()
60 plt.plot(rx_lpf)
61 plt.show()
62
63 rx_bin = np.empty(0)
64 for i in range(Nbits):
65     t = (2 * i + 1) * bit_period //
   2 + numtaps // 2
66     # rx_bin = np.append(rx_bin, np.
   heaviside(rx_lpf[t], 0))
67     rx_bin = np.append(rx_bin,
   rx_lpf[t] > 0.1)
68
69 print(tx_bin, "\n", rx_bin)
70 print(np.sum(rx_bin != tx_bin))
```

```
71
72 fig, (ax1, ax2) = plt.subplots(1, 2)
73 ax1.plot(rx_bin, color="blue", label
   ="rx_bin")
74 ax2.plot(tx_bin, color="red", label=
   "tx_bin")
75 fig.tight_layout()
76 plt.show()
```

```
 1 import numpy as np
 2 from matplotlib import pyplot as plt
 3 from scipy import fft
 4 from scipy import signal
 5
 6 def bin_array(num, m):
 7     """Convert a positive integer
   num into an m-bit bit vector"""
 8     return np.array(list(np.
   binary_repr(num).zfill(m))).astype(
   np.bool_)
 9
10 # import 24 bit digital data
11 id_num = 2802461
12 Nbits = 24
13 tx_bin = bin_array(id_num, Nbits)
14
15 # initialise constants and variables
16 fc = 0.125
17 bit_period = 16
18 tx_mod = np.empty(0)
19
20 # QPSK modulation
21 for i in range(0, Nbits, 2):
22     for j in range(bit_period):
23         tx_mod = np.append(tx_mod, (
   2 * tx_bin[i] - 1) *
24                                 np.cos(2
    * np.pi * fc * (i * bit_period + j
```

```
24 )) +
25                                  (2 *
   tx_bin[i + 1] - 1) * np.sin(2 * np.
   pi * fc * (i * bit_period + j)))
26
27 plt.figure()
28 plt.plot(tx_mod)
29 plt.show()
30 plt.figure()
31 plt.plot(np.abs(fft.fft(tx_mod)))
32 plt.show()
33
34 # low-pass filter
35 numtaps = 8
36 cutoff = 0.1
37 b1 = signal.firwin(numtaps, cutoff)
38
39 # Demodulation
40 rx_mixed_i = np.empty(0)
41 rx_mixed_q = np.empty(0)
42
43 for i in range(0, Nbits, 2):
44     for j in range(bit_period):
45         rx_mixed_i = np.append(
   rx_mixed_i, tx_mod[(i // 2) *
   bit_period + j] *
46                                 np.cos(2
    * np.pi * fc * (i * bit_period + j
   )))
```

```
47          rx_mixed_q = np.append(
    rx_mixed_q, tx_mod[(i // 2) *
    bit_period + j] *
48                                      np.
    sin(2 * np.pi * fc * (i * bit_period
     + j)))
49
50 rx_filt_i = signal.lfilter(b1, 1,
    rx_mixed_i)
51 rx_filt_i = np.append(rx_filt_i, np.
    zeros(numtaps // 2) / 2)
52 rx_filt_q = signal.lfilter(b1, 1,
    rx_mixed_q)
53 rx_filt_q = np.append(rx_filt_q, np.
    zeros(numtaps // 2) / 2)
54
55 plt.figure()
56 plt.plot(rx_filt_i, color="blue",
    label="In-phase")
57 plt.plot(rx_filt_q, color="red",
    label="Quadrature")
58 plt.show()
59
60 rx_bin = np.empty(0)
61 for i in range(0, Nbits, 2):
62     t = (i + 1) * bit_period // 2 +
    numtaps // 2
63     rx_bin = np.append(rx_bin,
    rx_filt_i[t] > 0)
```

```
64      rx_bin = np.append(rx_bin,
    rx_filt_q[t] > 0)
65
66 print(tx_bin, "\n", rx_bin)
67 print(np.sum(rx_bin != tx_bin))
68
69 fig, (ax1, ax2) = plt.subplots(1, 2)
70 ax1.plot(rx_bin, color="blue", label
    ="rx_bin")
71 ax2.plot(tx_bin, color="red", label=
    "tx_bin")
72 fig.tight_layout()
73 plt.show()
```