Data Communication Assignment 2

1 NRZ-L Encoding and NRZ-I Encoding

NRZ-L Encoding is a line encoding technique, specifically a serializer line code used to send information bitwise. Conventionally, 1 is represented by one physical level -1, while 0 is represented by another level 1.

In bipolar NRZL encoding, the signal essentially swings from one level to another.

NRZ-I Encoding is another serializer line encoding technique, used to send information bitwise.

The two-level NRZI signal distinguishes data bits by the presence or absence of a transition, meaning that a 1 is represented by a transition from the previous encoded bit, while 0 is represented by no transition.

NRZ-I encoding is used in USBs, but the opposite convention i.e. “change on 0” is used for encoding.

1.0.1 A-1 : Take input as bit sequence of n-bits, plot its NRZ-L and NRZ-I line coding.

1.0.2 Keep time axis resolution in milli-seconds.

[13]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
import ipywidgets as wd
from IPython.display import display

%matplotlib nbagg
```

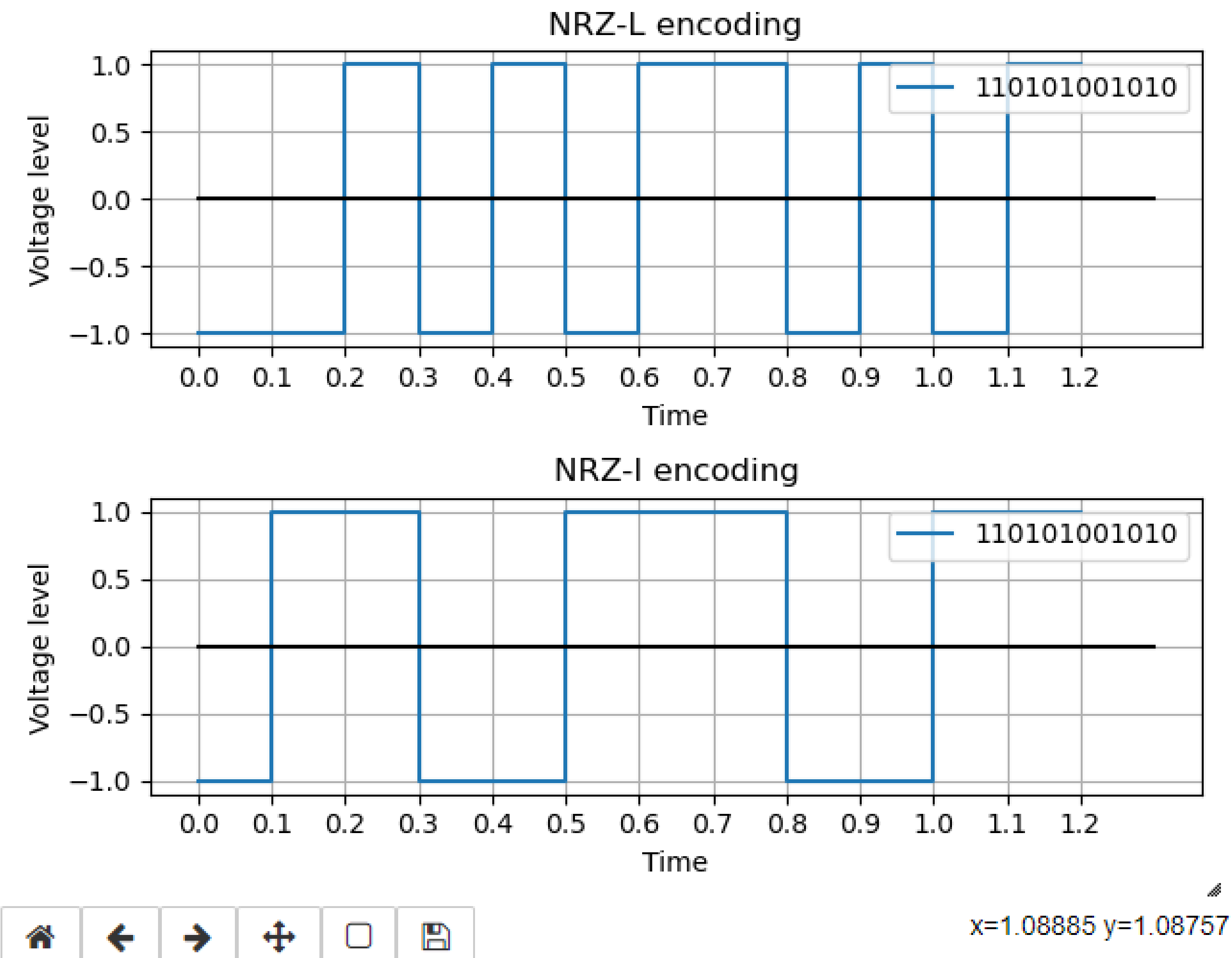
[14]:
```
print("Enter the bit sequence")
# bit_data=input("Enter the bit sequence")


def draw_plot(bit_data):

    plt.clf()

    bit_data = str(bit_data)

    # Creating two plots for two graphs
```

```
fig,(ax1,ax2)=plt.subplots(nrows=2,ncols=1)
x=[]
nrzl=[]
nrzi=[]
x_val=0


# Generating values for nrzl and nrzi
for it in range(len(bit_data)):

    if int(bit_data[it])==0:
        if len(nrzi) == 0:
            val=1
        else:
            val=nrzi[-1]
        for _ in range(8):
            x+=[round(random.uniform(x_val,x_val+0.1),4)]
            nrzl+=[1]
            nrzi+=[val]
        x+=[x_val,x_val+0.1]
        nrzl+=[1,1]
        nrzi+=[val,val]
        x_val+=0.1
    else:
        if len(nrzi)==0:
            val=-1
        else:
            val=-nrzi[-1]
        for _ in range(8):
            x += [round(random.uniform(x_val, x_val + 0.1),4)]
            nrzl += [-1]
            nrzi+=[val]
        x+=[x_val,x_val+0.1]
        nrzl+=[-1,-1]
        nrzi+=[val,val]
        x_val+=0.1
x. sort()


# list of values to be plotted on x-axis
val_to_be_pl=[0]
x_val=0.1
for _ in range(len(bit_data)):
      val_to_be_pl+=[x_val]
      x_val+=0.1
ax1.plot(x,nrzl,label=bit_data)
ax1.set_title("NRZ-L encoding")
```

```
    ax1.set_xlabel("Time")
    ax1.set_ylabel("Voltage level")
    ax1.plot((0,(len(bit_data)+1)*0.1),(0,0),color="k")
    ax1.set_xticks(val_to_be_pl)
    ax1.legend(loc = 1)
    ax1.grid(axis="both")

    ax2.plot(x,nrzi,label=bit_data)
    ax2.set_title("NRZ-I encoding")
    ax2.set_xlabel("Time")
    ax2.set_ylabel("Voltage level")
    ax2.plot((0,(len(bit_data)+1)*0.1),(0,0),color="k")
    ax2.set_xticks(val_to_be_pl)
    ax2.legend(loc = 1)
    ax2.grid(axis="both")


    plt.tight_layout()
    plt.show()


bit_data = wd.IntText(value=110101001010,  description="Bit Data:
 ↪",disabled=False)

# time_duration = wd.FloatSlider(min=0,max=10,value=2,description = "Time$(s)$")



wd.interactive(draw_plot,bit_data=bit_data)
```

```
Enter the bit sequence
```

Bit Data: 110101001010



1.0.3 A-2: Add random noise to above generated Digital Signal and plot the noisy signal

[15]:
```python
import matplotlib.pyplot as plt
import numpy as np
import random
import ipywidgets as wd
from IPython.display import display

%matplotlib nbagg
```

[16]:
```
print("Enter the bit sequence")

def draw_plot(bit_data,vol_pos,vol_neg):

    plt.clf()

    bit_data = str(bit_data)



    # Creating Three Plots
    fig2, ax = plt.subplots(nrows=1,ncols=1,figsize=(7,2))

    fig1,(ax1,ax2)=plt.subplots(nrows=2,ncols=1)

    fig3,(ax3,ax4)=plt.subplots(nrows=2,ncols=1)

    x=[]
    nrzl=[]
    nrzi=[]
    x_val=0

    # Generating values for nrzl and nrzi
    for it in range(len(bit_data)):
        if int(bit_data[it])==0:
            if len(nrzi) == 0:
                val=1
            else:
                val=nrzi[-1]
            for _ in range(18):
                x+=[round(random.uniform(x_val,x_val+0.1),4)]
                nrzl+=[1]
                nrzi+=[val]
            x+=[x_val,x_val+0.1]
            nrzl+=[1,1]
            nrzi+=[val,val]
            x_val+=0.1
        else:
            if len(nrzi)==0:
                val=-1
            else:
                val=-nrzi[-1]
            for _ in range(18):
                x += [round(random.uniform(x_val, x_val + 0.1),4)]
                nrzl += [-1]
                nrzi+=[val]
            x+=[x_val,x_val+0.1]
```

```
        nrzl+=[-1,-1]
        nrzi+=[val,val]
        x_val+=0.1
x. sort()
nrzl_with_noise=[]
nrzi_with_noise=[]
rand_noise=[]


#Generating random noise signal
for j in range(len(x)):
    rand_noise+=[round(random.uniform(vol_neg,vol_pos),4)]
    nrzl_with_noise+=[rand_noise[j]+nrzl[j]]
    nrzi_with_noise+=[rand_noise[j]+nrzi[j]]

# list of values to be plotted on x-axis
val_to_be_pl=[0]
x_val=0.1
for _ in range(len(bit_data)):
     val_to_be_pl+=[x_val]
     x_val+=0.1



# Noise Plot
ax.plot(x,rand_noise)
ax.set_title("Random noise")
ax.set_xlabel("Time")
ax.set_ylabel("Voltage level")
ax.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
ax.set_xticks(val_to_be_pl)
ax.grid(axis='both')

#fig1 will consists of ax1,ax2
#ax1 will plot nrzl+noise
ax1.plot(x,nrzl,label=bit_data)
ax1.plot(x,rand_noise,color='lightcoral', linewidth=1, linestyle='dotted')
ax1.set_title("NRZ-L encoding + random noise")
ax1.set_xlabel("Time")
ax1.set_ylabel("Voltage level")
ax1.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
ax1.set_xticks(val_to_be_pl)
ax1.legend()
ax1.grid(axis='both')


#ax2 will plot the resultant nrzl due to noise i.e noisy signal
```

```
    ax2.plot(x,nrzl_with_noise)
    ax2.plot(x,nrzl,color='lightcoral', linewidth=1, linestyle='-')
    ax2.set_title("Noisy signal")
    ax2.set_xlabel("Time")
    ax2.set_ylabel("Voltage level")
    ax2.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax2.set_xticks(val_to_be_pl)
    ax2.grid(axis='both')




    #fig3 will consists of ax3,ax4
    #ax3 will plot nrzi+noise
    #ax4 will plot the resultant nrzi due to noise i.e noisy signal
    ax3.plot(x,nrzi,label=bit_data)
    ax3.plot(x,rand_noise,color='lightcoral', linewidth=1, linestyle='dotted')
    ax3.set_title("NRZ-I encoding + noise")
    ax3.set_xlabel("Time")
    ax3.set_ylabel("Voltage level")
    ax3.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax3.set_xticks(val_to_be_pl)
    ax3.legend(loc = 1)
    ax3.grid(axis='both')


    ax4.plot(x,nrzi_with_noise)
    ax4.plot(x,nrzi,color='lightcoral', linewidth=1, linestyle='-')
    ax4.set_title("Noisy signal")
    ax4.set_xlabel("Time")
    ax4.set_ylabel("Voltage level")
    ax4.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax4.set_xticks(val_to_be_pl)
    ax4.grid(axis='both')


    plt.tight_layout()
    plt.show()


bit_data = wd.IntText(value=110101001010,  description='Bit Data:
 ↪',disabled=False)


vol_pos = wd.FloatSlider(min=0, max=5, value=0.5, description="Maximum Positive␣
 ↪voltage of noise:")
```

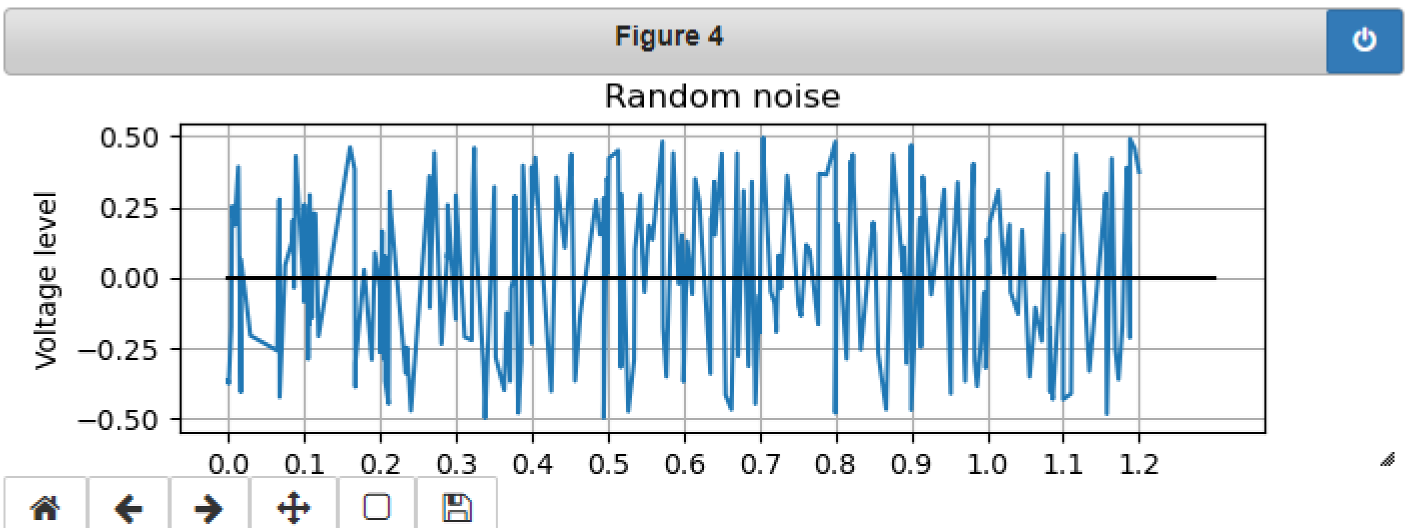
```
vol_neg = wd.FloatSlider(min=-5, max=0, value=-0.5, description="Maximum␣
↪Negative voltage of noise:")

bit_d = str(bit_data)

wd.interactive(draw_plot,bit_data=bit_data,vol_pos=vol_pos,vol_neg=vol_neg)
```

```
Enter the bit sequence
```

Bit Data: 110101001010

Maximum ... 0.50

Maximum ... -0.50



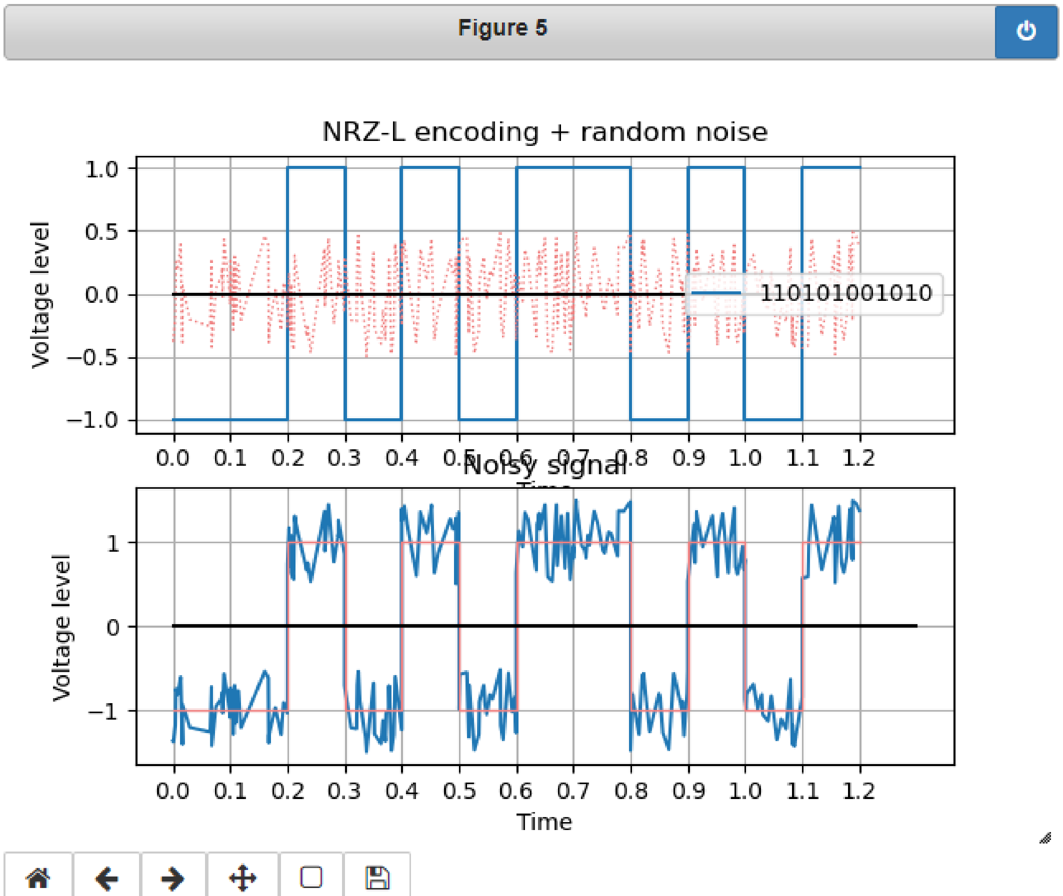


Figure 5
NRZ-L encoding + random noise
Voltage level
1.0
0.5
0.0
−0.5
−1.0
110101001010
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2
Noisy signal
Voltage level
1
0
−1
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2
Time

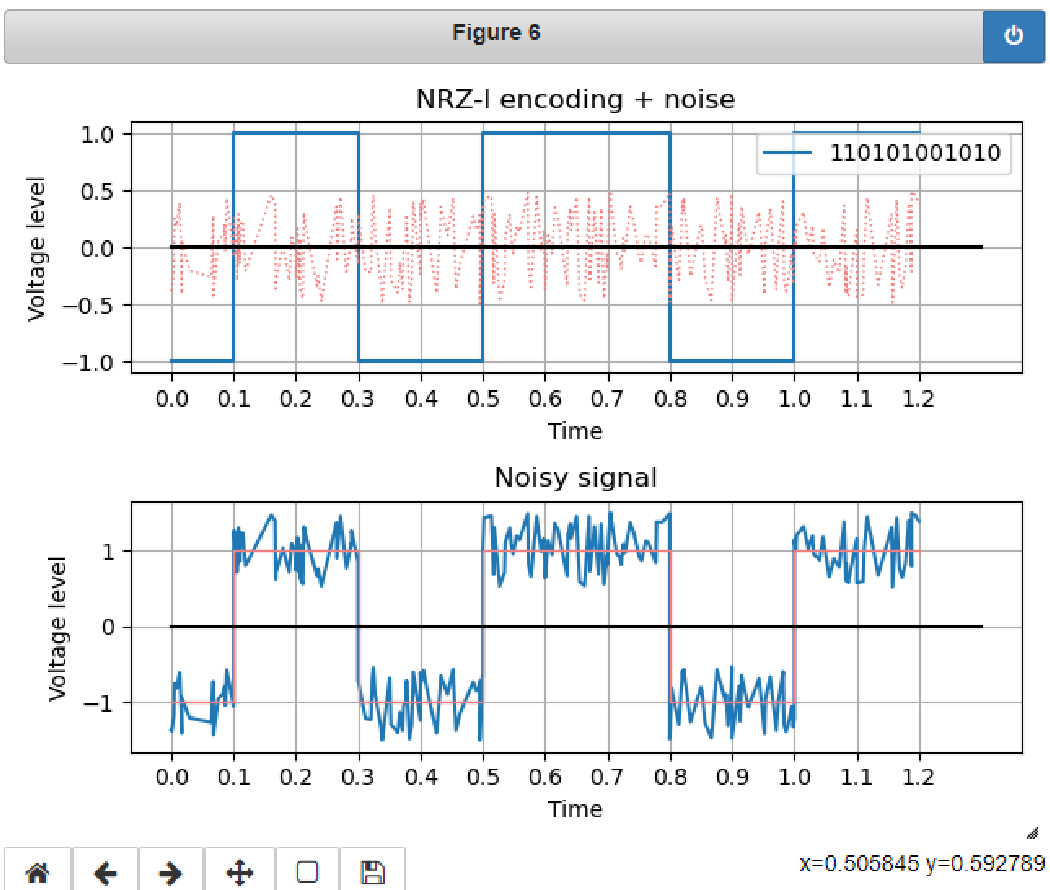


Figure 6
NRZ-I encoding + noise
Voltage level
1.0
0.5
0.0
−0.5
−1.0
110101001010
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2
Time
Noisy signal
Voltage level
1
0
−1
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2
Time
x=0.505845 y=0.592789

1.0.4 A-3: Decide threshold for detection of bit sequence back and calculate number of bits in error

[17]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
import math
import ipywidgets as wd
from IPython.display import display

%matplotlib nbagg
```

[18]:
```
print("Enter the bit sequence")

def draw_plot(bit_data,vol_pos,vol_neg,threshold_pos,threshold_neg):

    plt.clf()

    bit_data = str(bit_data)



    # Creating Two Plots

    fig1,(ax1,ax3)=plt.subplots(nrows=2,ncols=1)
    fig2,(ax2,ax4)=plt.subplots(nrows=2,ncols=1)

    x=[]
    nrzl=[]
    nrzi=[]
    x_val=0

    # Generating values for nrzl and nrzi
    for it in range(len(bit_data)):
        if int(bit_data[it])==0:
            if len(nrzi) == 0:
```

```
                val=1
            else:
                val=nrzi[-1]
            for _ in range(18):
                x+=[round(random.uniform(x_val,x_val+0.1),4)]
                nrzl+=[1]
                nrzi+=[val]
            x+=[x_val,x_val+0.1]
            nrzl+=[1,1]
            nrzi+=[val,val]
            x_val+=0.1
        else:
            if len(nrzi)==0:
                val=-1
            else:
                val=-nrzi[-1]
            for _ in range(18):
                x += [round(random.uniform(x_val, x_val + 0.1),4)]
                nrzl += [-1]
                nrzi+=[val]
            x+=[x_val,x_val+0.1]
            nrzl+=[-1,-1]
            nrzi+=[val,val]
            x_val+=0.1
x. sort()
nrzl_with_noise=[]
nrzi_with_noise=[]
rand_noise=[]


#Generating random noise signal
for j in range(len(x)):
    rand_noise+=[round(random.uniform(vol_neg,vol_pos),4)]
    nrzl_with_noise+=[rand_noise[j]+nrzl[j]]
    nrzi_with_noise+=[rand_noise[j]+nrzi[j]]


#bit sequence received
bit_seq_in_nrzi=""
bit_seq_in_nrzl=""
ambg_in_nrzl=0

#it is just any number, to calculate nrzi
prev_mean=100
i=0
while i<len(nrzl):
    mean1=(sum(nrzl_with_noise[i:i+20]))/10
```

```
        mean2=(sum(nrzi_with_noise[i:i+20]))/10
        if mean1>=threshold_pos:
            bit_seq_in_nrzl+="0"
        elif mean1<=threshold_neg:
            bit_seq_in_nrzl += "1"
        else:#if it does not fall in any limit then
             #we will consider the bit as 1 and we we
             #can have count of such bits
            bit_seq_in_nrzl+="1"
            ambg_in_nrzl+=1
        if prev_mean==100:
            if mean2>=threshold_pos:
                bit_seq_in_nrzi+="0"
            else:
                bit_seq_in_nrzi+="1"
        else:
            if abs(prev_mean+mean2)>abs(mean2) and␣
↪abs(prev_mean+mean2)>abs(mean1):#if they have same sign
                bit_seq_in_nrzi += "0"
            else:#therefore they have opp sign
                bit_seq_in_nrzi += "1"
        prev_mean = mean2
        i+=20

    bit_err_in_nrzi=0
    bit_err_in_nrzl=0
    for j in range(len(bit_data)):
        if int(bit_data[j])!=int(bit_seq_in_nrzi[j]):
            bit_err_in_nrzi+=1
        if int(bit_data[j])!=int(bit_seq_in_nrzl[j]):
            bit_err_in_nrzl+=1
    print("No. of bits which we were in ambiguous to indentify in␣
↪NRZ-L=",ambg_in_nrzl)
    print("No.of bits error in NRZ-L = ",bit_err_in_nrzl)
    print("No.of bits error in NRZ-I = ",bit_err_in_nrzi)

    # list of values to be plotted on x-axis
    val_to_be_pl=[0]
    x_val=0.1
    for _ in range(len(bit_data)):
          val_to_be_pl+=[x_val]
          x_val+=0.1


    #fig1 will consists of ax2,ax4
    #ax1 will plot nrzl+noise
    #ax3 will plot noisy signal
```

```
    ax1.plot(x,nrzl,label=bit_data)
#   ax1.plot(x,rand_noise)
    ax1.set_title("NRZ-L encoding + random noise")
    ax1.set_xlabel("Time")
    ax1.set_ylabel("Voltage level")
    ax1.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax1.set_xticks(val_to_be_pl)
    ax1.legend(loc=1)
#   ax1.tight_layout()
    ax1.grid(axis='both')


    ax3.plot(x,nrzl_with_noise,label="bit rec={}".format(bit_seq_in_nrzl))
    ax3.plot(x,nrzl,color='lightcoral', linewidth=1, linestyle='-')
    ax3.set_title("Noisy signal")
    ax3.set_xlabel("Time")
    ax3.set_ylabel("Voltage level")
    ax3.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax3.set_xticks(val_to_be_pl)
    ax3.legend(loc=1)
#   ax3.tight_layout()
    ax3.grid(axis='both')


    #fig2 will consists of ax2,ax4
    #ax2 will plot nrzi+noise
    #ax4 will plot noisy signal
    ax2.plot(x,nrzi,label=bit_data)
#   ax2.plot(x,rand_noise)
    ax2.set_title("NRZ-I encoding + noise")
    ax2.set_xlabel("Time")
    ax2.set_ylabel("Voltage level")
    ax2.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax2.set_xticks(val_to_be_pl)
    ax2.legend(loc=1)
#   ax2.tight_layout()
    ax2.grid(axis='both')


    ax4.plot(x,nrzi_with_noise,label="bit rec={}".format(bit_seq_in_nrzi))
    ax4.plot(x,nrzi,color='lightcoral', linewidth=1, linestyle='-')
    ax4.set_title("Noisy signal")
    ax4.set_xlabel("Time")
    ax4.set_ylabel("Voltage level")
    ax4.plot((0,(len(bit_data)+1)*0.1),(0,0),color='k')
    ax4.set_xticks(val_to_be_pl)
    ax4.legend(loc=1)
```

```
    ax4.grid(axis='both')

    fig1.tight_layout()
    fig2.tight_layout()
    plt.tight_layout()
    plt.show()


bit_data = wd.IntText(value=110101001010,  description='Bit Data: ',disabled=False)

vol_pos = wd.FloatSlider(min=0, max=5, value=0.5,description="Maximum Positive voltage of noise:")
vol_neg = wd.FloatSlider(min=-5, max=0, value=-0.5,description="Maximum Negative voltage of noise:")

threshold_pos = wd.FloatSlider(min=0, max=5, value=0.5,description="Positive threshold amplitude")
threshold_neg = wd.FloatSlider(min=-5, max=0, value=-0.5,description="Negative threshold amplitude")

wd.interactive(draw_plot, bit_data=bit_data, vol_pos=vol_pos, vol_neg=vol_neg,
                threshold_pos=threshold_pos,threshold_neg=threshold_neg)
```

```
Enter the bit sequence
```

Bit Data: 110101001010

Maximum ... 0.50

Maximum ... -0.50

Positive thr... 0.50

Negative th... -0.50

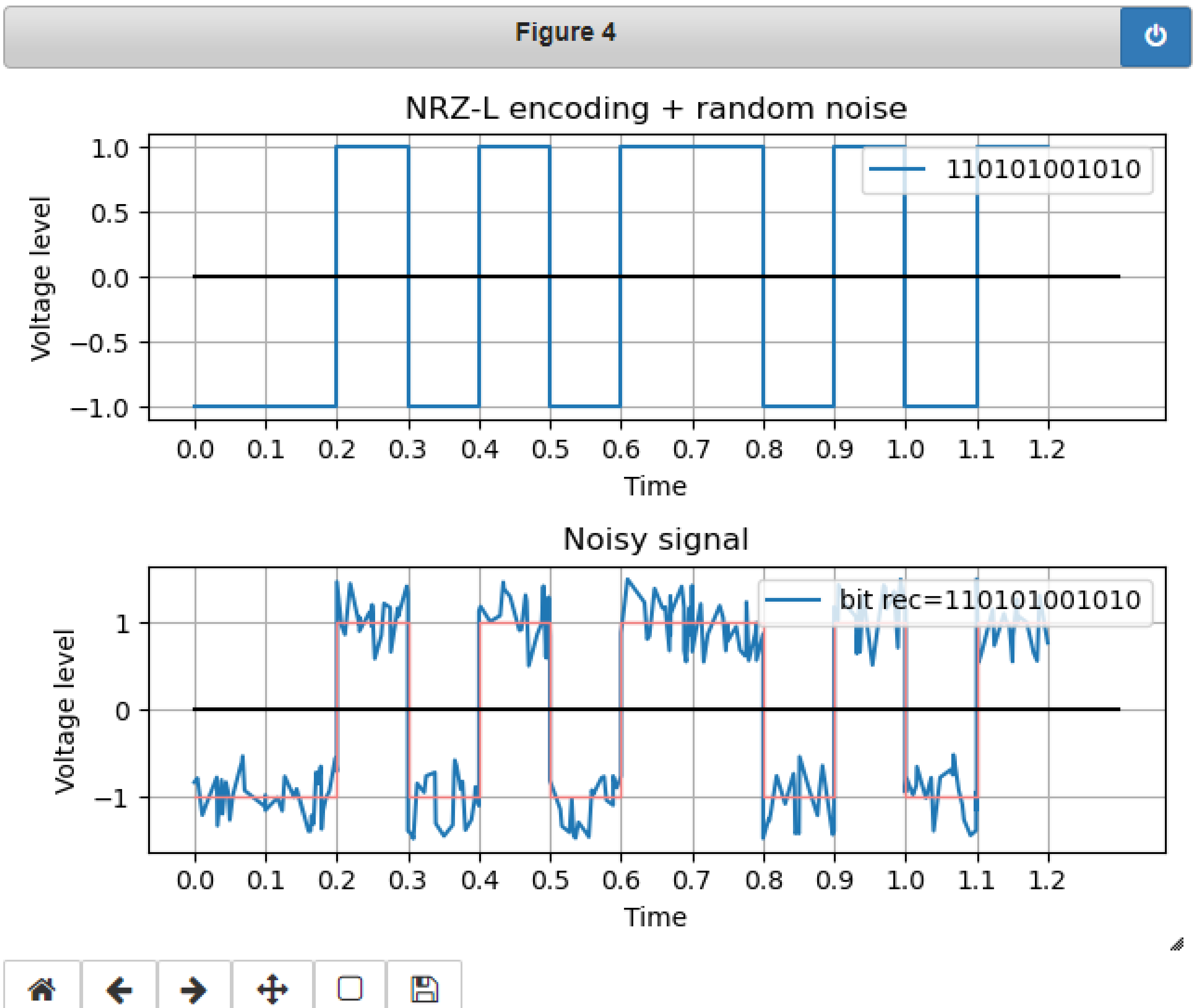
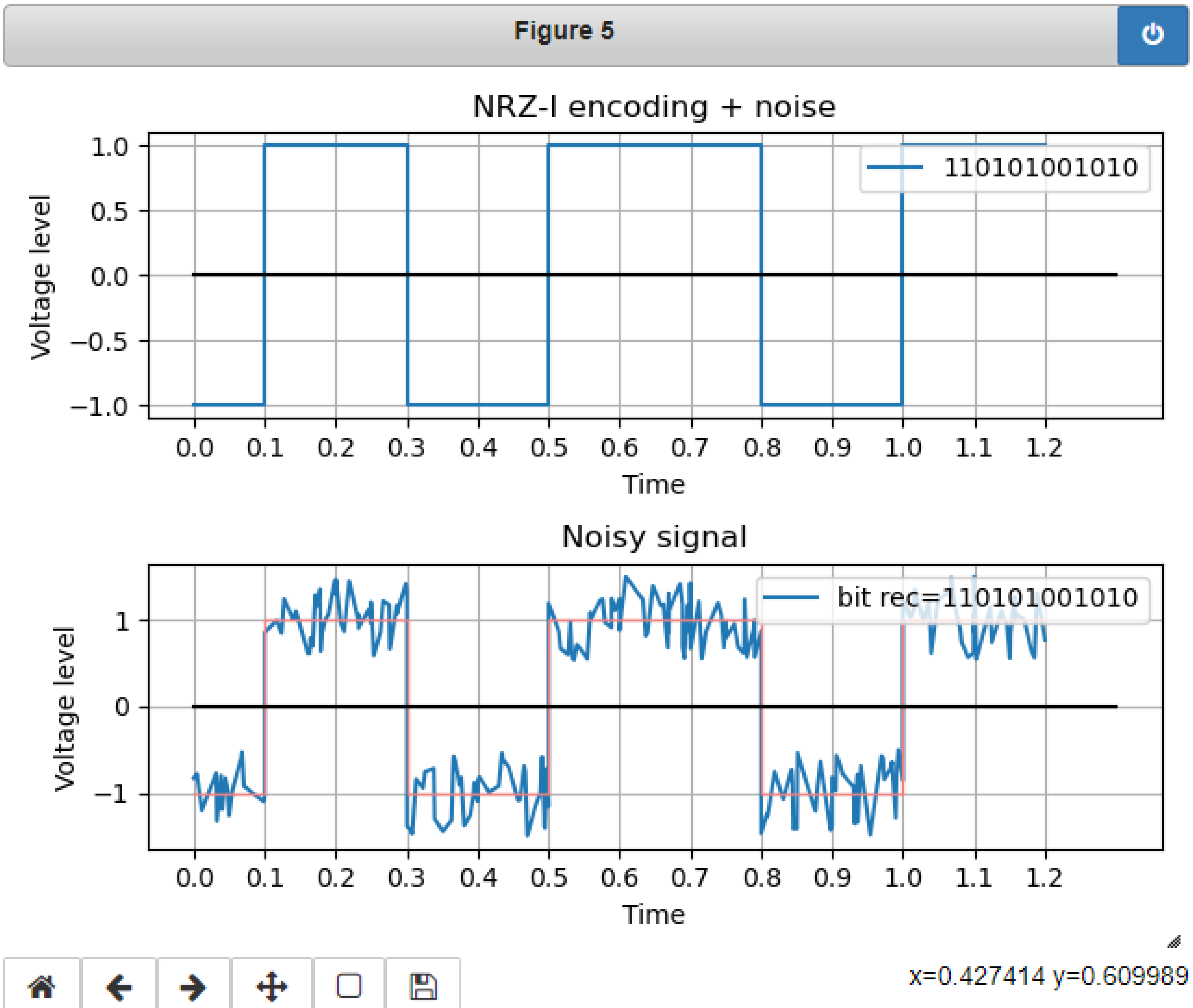


Figure 4
NRZ-L encoding + random noise
Voltage level
Time
110101001010
Noisy signal
bit rec=110101001010
1.0
0.5
0.0
−0.5
−1.0
1
0
−1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2




```
No. of bits which we were in ambiguous to indentify in NRZ-L= 0
No.of bits error in NRZ-L =  0
No.of bits error in NRZ-I =  0
```