Line Coding Project

October 24, 2018

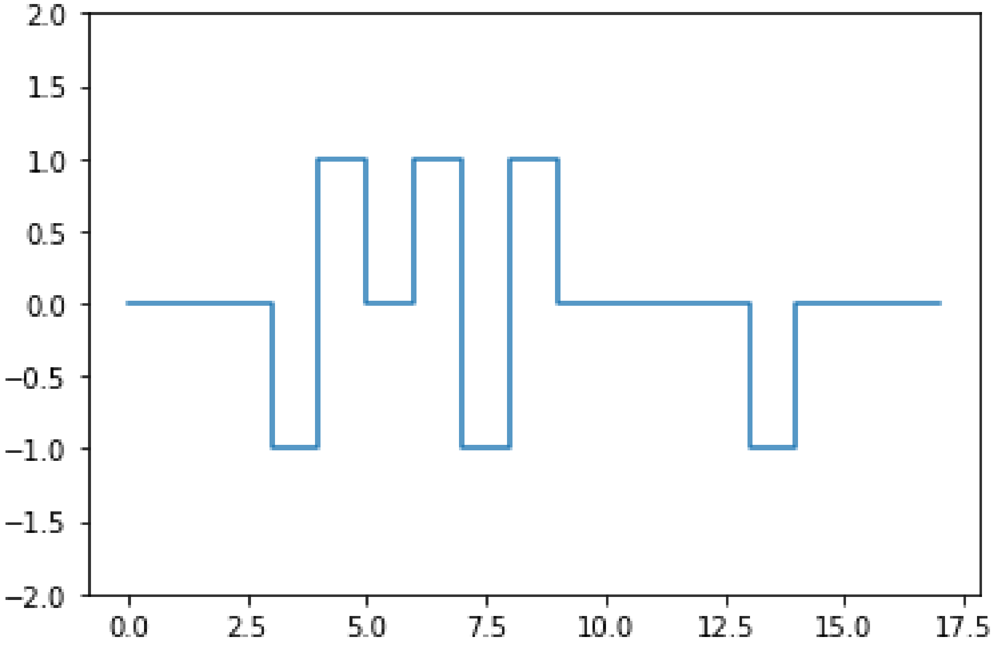
```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        from random import *
        randBinList = lambda n: [randint(0,1) for b in range(1,n+1)]
        print("1. user input")
        print("2. random binary input")
        x=int(input())
        if(x==1):
            print("Enter the binary input")
            s=input()
        else:
            print("Enter 1 if you want fixed subsequence(size must be greater than 8) otherwis
            bla = int(input())
            print("Enter size of binary input")
            size = int(input())
            kla = randint(0,9)
            if(bla==1):
                size = size - 8
            kla = kla%size
            #print(kla)
            s=[]
            s=randBinList(size)
            saurabh = "00000000"
            #print(*s)
            s="".join(str(i) for i in s)
            if(bla==1):
                s=s[:kla]+saurabh+s[kla:]
            print("Input is: ")
            print(s)

        #palindrome code
        print("Longest palindrome substring is: ")
        import sys

        def printSubStr(st,low,high) :
                sys.stdout.write(st[low : high + 1])
                sys.stdout.flush()
                return ''

        def longestPalSubstr(st) :
                n = len(st)
                table = [[0 for x in range(n)] for y
                                                        in range(n)]

                maxLength = 1
                i = 0
                while (i < n) :
                        table[i][i] = True
                        i = i + 1

                # check for sub-string of length 2.
                start = 0
                i = 0
                while i < n - 1 :
                        if (st[i] == st[i + 1]) :
                                table[i][i + 1] = True
                                start = i
                                maxLength = 2
                        i = i + 1

                # Check for lengths greater than 2.
                # k is length of substring
                k = 3
                while k <= n :
                        # Fix the starting index
                        i = 0
                        while i < (n - k + 1) :

                                # Get the ending index of
                                # substring from starting
                                # index i and length k
                                j = i + k - 1

                                # checking for sub-string from
                                # ith index to jth index iff
                                # st[i+1] to st[(j-1)] is a
                                # palindrome
                                if (table[i + 1][j - 1] and
                                                st[i] == st[j]) :
                                        table[i][j] = True

                                        if (k > maxLength) :
                                                start = i
                                                maxLength = k
                                i = i + 1
                        k = k + 1
                printSubStr(st, start, start + maxLength - 1)

                return maxLength # return length


        # Driver program to test above functions
        st = s
        l = longestPalSubstr(st)
        print("Length is:", l )
            #pal end
        print("\n1.NRZ-L\n","2.NRZ-I\n","3.Manchester\n","4.Differantial Manchester\n","5.other
        n=int(input())

        if(n==1):
            print("Perform NRZ-L")
            ls=list()
            for i in range(len(s)):
                if(s[i]=='0' or s[i]==0):
                    ls.append(-1)
                else:
                    ls.append(1)
            xs = np.repeat(range(len(s)), 2)
            ys = np.repeat(ls, 2)
            xs=xs[1:]
            xs=np.append(xs,(xs[len(xs)-1]+1))
            ys=ys[:-1]
            ys=np.append(ys,(ys[len(ys)-1]))
            plt.step(xs,ys)
            plt.ylim(-2,2)
            plt.show()
            print("Positive logic")

            ls1=list()
            for i in range(len(s)):
                if(s[i]=='0' or s[i]==0):
                    ls1.append(1)
                else:
                    ls1.append(-1)
            xs1 = np.repeat(range(len(s)), 2)
            ys1 = np.repeat(ls1, 2)
            xs1=xs1[1:]
            xs1=np.append(xs1,(xs1[len(xs1)-1]+1))
            ys1=ys1[:-1]
            ys1=np.append(ys1,(ys1[len(ys1)-1]))
            plt.step(xs1,ys1)
            plt.ylim(-2,2)
            plt.show()
            print("Negative logic")
        elif(n==2):
            print("Perform NRZ-I")
            Is=list()
            if(s[0]=='0' or s[0]==0):
                Is.append(-1)
            else:
                Is.append(1)
            k=len(s)
            i=1
            while(i<k):
                if(int(s[i])==0):
                    Is.append(Is[i-1])
                else:
                    Is.append(-Is[i-1])
                i=i+1
            xs = np.repeat(range(len(s)), 2)
            ys = np.repeat(Is, 2)
            xs=xs[1:]
            xs=np.append(xs,(xs[len(xs)-1]+1))
            ys=ys[:-1]
            ys=np.append(ys,(ys[len(ys)-1]))
            plt.step(xs,ys)
            plt.ylim(-2,2)
            plt.show()
            print("Positive logic")
            Is=list()
            if(s[0]=='0' or s[0]==0):
                Is.append(1)
            else:
                Is.append(-1)
            k=len(s)
            i=1
            while(i<k):
                if(int(s[i])==0):
                    Is.append(Is[i-1])
                else:
                    Is.append(-Is[i-1])
                i=i+1
            xs1 = np.repeat(range(len(s)), 2)
            ys1= np.repeat(Is, 2)
            xs1=xs1[1:]
            xs1=np.append(xs1,(xs1[len(xs1)-1]+1))
            ys1=ys1[:-1]
            ys1=np.append(ys1,(ys1[len(ys1)-1]))
            plt.step(xs1,ys1)
            plt.ylim(-2,2)
            plt.show()
            print("Negative logic")
        elif(n==3):
            print("Perform Manchester")
            pm=list()
            for j in range(len(s)):
                if(s[j]=='0' or s[j]==0):
                    pm.append(-1)
                    pm.append(1)
                else:
                    pm.append(1)
                    pm.append(-1)
            xs=[x*0.5 for x in range(0,(2*len(s)))]
            xs=np.repeat(xs,2)
            ys = np.repeat(pm, 2)
            xs=xs[1:]
            xs=np.append(xs,(xs[len(xs)-1]+0.5))
            ys=ys[:-1]
            ys=np.append(ys,(ys[len(ys)-1]))
            plt.step(xs,ys)
            plt.ylim(-2,2)
            plt.show()
            print("Positive logic")

            pm=list()
            for j in range(len(s)):
                if(s[j]=='0' or s[j]==0):
                    pm.append(1)
                    pm.append(-1)
                else:
                    pm.append(-1)
                    pm.append(1)
            xs1=[x*0.5 for x in range(0,(2*len(s)))]
            xs1=np.repeat(xs1,2)
            ys1= np.repeat(pm, 2)
            xs1=xs1[1:]
            xs1=np.append(xs1,(xs1[len(xs1)-1]+0.5))
            ys1=ys1[:-1]
            ys1=np.append(ys1,(ys1[len(ys1)-1]))
            plt.step(xs1,ys1)
            plt.ylim(-2,2)
            plt.show()
            print("Negative logic")
        elif(n==4):
            print("Perform Differantial Manchester")
            pdm=list()
            pdm.append(1)
            pdm.append(-1)
            i=1
            k=len(s)
            while(i<k):
                if(int(s[i])==1):
                    pdm.append(pdm[len(pdm)-1])
                    pdm.append(-pdm[len(pdm)-1])
                else:
                    pdm.append(-pdm[len(pdm)-1])
                    pdm.append(-pdm[len(pdm)-1])
                i=i+1
            print(pdm)
            xs=[x*0.5 for x in range(0,(2*len(s)))]
            xs=np.repeat(xs,2)
            ys = np.repeat(pdm, 2)
            xs=xs[1:]
            xs=np.append(xs,(xs[len(xs)-1]+0.5))
            ys=ys[:-1]
            ys=np.append(ys,(ys[len(ys)-1]))
            plt.step(xs,ys)
            plt.ylim(-2,2)
            plt.show()
            print("Positive logic")
            pdm=list()
            pdm.append(-1)
            pdm.append(1)
            i=1
            k=len(s)
            while(i<k):
                if(int(s[i])==1):
                    pdm.append(pdm[len(pdm)-1])
                    pdm.append(-pdm[len(pdm)-1])
                else:
                    pdm.append(-pdm[len(pdm)-1])
                    pdm.append(-pdm[len(pdm)-1])
                i=i+1
            print(pdm)
            xs1=[x*0.5 for x in range(0,(2*len(s)))]
            xs1=np.repeat(xs1,2)
            ys1 = np.repeat(pdm, 2)
            xs1=xs1[1:]
            xs1=np.append(xs1,(xs1[len(xs1)-1]+0.5))
            ys1=ys1[:-1]
            ys1=np.append(ys1,(ys1[len(ys1)-1]))
            plt.step(xs1,ys1)
            plt.ylim(-2,2)
            plt.show()
            print("Negative logic")
        else:
```

```
print("0.No Scrambling(AMI)","1.Scrambling(HDB3/B8ZS)")
q=int(input())
if(q==0):
    print("Perform AMI")
    am=list()
    m=1
    for i in range(len(s)):
        if(int(s[i])==0):
            am.append(0)
        else:
            if(m%2==1):
                am.append(1)
            else:
                am.append(-1)
            m=m+1
    xs = np.repeat(range(len(s)), 2)
    ys = np.repeat(am, 2)
    xs=xs[1:]
    xs=np.append(xs,(xs[len(xs)-1]+1))
    ys=ys[:-1]
    ys=np.append(ys,(ys[len(ys)-1]))
    plt.step(xs,ys)
    plt.ylim(-2,2)
    plt.show()
    print("Positive logic")
    am=list()
    m=1
    for i in range(len(s)):
        if(int(s[i])==0):
            am.append(0)
        else:
            if(m%2==1):
                am.append(-1)
            else:
                am.append(1)
            m=m+1
    xs1 = np.repeat(range(len(s)), 2)
    ys1 = np.repeat(am, 2)
    xs1=xs1[1:]
    xs1=np.append(xs1,(xs1[len(xs1)-1]+1))
    ys1=ys1[:-1]
    ys1=np.append(ys1,(ys1[len(ys1)-1]))
    plt.step(xs1,ys1)
    plt.ylim(-2,2)
    plt.show()
    print("Negative logic")
else:
    print("Perform Scrambling")
    print("1.B8ZS","2.HDB3")
    p=int(input())
    q=len(s)
    if(p==1):
        print("Perform B8ZS")
        bz=list()
        m=1
        s1=s.replace("00000000","000vb0vb")
        for i in range(len(s1)):
            if(s1[i]=='0' or s1[i]==0):
                bz.append(0)
            elif(s1[i]=='1'):
                if(m%2==1):
                    bz.append(1)
                else:
                    bz.append(-1)
                m=m+1
            elif(s1[i]=='v'):
                if(m%2==1):
                    bz.append(-1)
                else:
                    bz.append(1)
            else:
                if(m%2==1):
                    bz.append(1)
                else:
                    bz.append(-1)
                m=m+1
        xs = np.repeat(range(len(s)), 2)
        ys = np.repeat(bz, 2)
        xs=xs[1:]
        xs=np.append(xs,(xs[len(xs)-1]+1))
        ys=ys[:-1]
        ys=np.append(ys,(ys[len(ys)-1]))
        plt.step(xs,ys)
        plt.ylim(-2,2)
        plt.show()
        print("Positive logic")
        bz=list()
        m=1
        for i in range(len(s1)):
            if(s1[i]=='0'):
                bz.append(0)
            elif(s1[i]=='1'):
                if(m%2==1):
                    bz.append(-1)
                else:
                    bz.append(1)
                m=m+1
            elif(s1[i]=='v'):
                if(m%2==1):
                    bz.append(1)
                else:
                    bz.append(-1)
            else:
                if(m%2==1):
                    bz.append(-1)
                else:
                    bz.append(1)
                m=m+1
        xs1 = np.repeat(range(len(s)), 2)
        ys1 = np.repeat(bz, 2)
        xs1=xs1[1:]
        xs1=np.append(xs1,(xs1[len(xs1)-1]+1))
        ys1=ys1[:-1]
        ys1=np.append(ys1,(ys1[len(ys1)-1]))
        plt.step(xs1,ys1)
        plt.ylim(-2,2)
        plt.show()
        print("Negative logic")
    else:
        print("Perform HDB3")
        m=0
        hd=list()

        f=s.find("0000")
        if(f==-1):
            f=len(s)
        i=0
        k=len(s)
        d=1
        p=0
        while(i<k):
            if(s[i]=='1' or s[i]==1):
                m=m+1
                p=p+1
                if(m%2==1):
                    hd.append(d)
                    d=1
                else:
                    hd.append(-d)
                    d=-d
            else:
                if(i<f):
                    hd.append(0)
                elif(i==f):
```

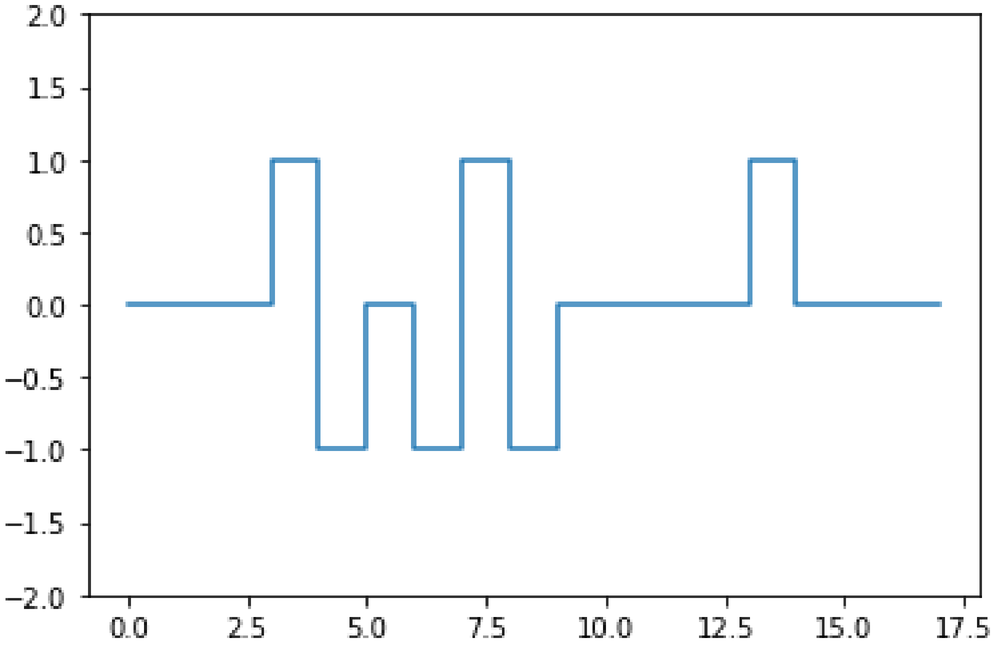
```
            i=i+3
            if(p%2==0):
                hd.append(-d)
                hd.append(0)
                hd.append(0)
                hd.append(-d)
                d=-d
                p=p+2
                m=m+1
            else:
                hd.append(0)
                hd.append(0)
                hd.append(0)
                hd.append(d)
                p=p+1
            jk=s[i+1:(i+1)+(k-i-1)]
            x=jk.find("0000")
            if(x==-1):
                f=k
            else:
                f=i+1+x
    i=i+1
xs = np.repeat(range(len(s)), 2)
ys = np.repeat(hd, 2)
xs=xs[1:]
xs=np.append(xs,(xs[len(xs)-1]+1))
ys=ys[:-1]
ys=np.append(ys,(ys[len(ys)-1]))
plt.step(xs,ys)
plt.ylim(-1.5,1.5)
plt.show()
print(hd)
```

```
1. user input
2. random binary input
2
Enter 1 if you want fixed subsequence(size must be greater than 8) otherwise press 2
1
Enter size of binary input
17
Input is:
00000000100001000
Longest palindrome substring is:
000100001000Length is: 12

1.NRZ-L
```

```
 2.NRZ-I
 3.Manchester
 4.Differantial Manchester
 5.others(AMI/B8ZS/HDB3)

6
0.No Scrambling(AMI) 1.Scrambling(HDB3/B8ZS)
1
Perform Scrambling
1.B8ZS 2.HDB3
1
Perform B8ZS
```



```
Positive logic
```



```
Negative logic
```