Wireless Communication and Networks Practical File

Submitted in partial fulfillment of the Requirements for the award of the degree

of

Bachelor of Technology

in

INFORMATION TECHNOLOGY

By:

Manas (IT-1)
00513203121

Department of Information Technology
Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University

Dwarka, New Delhi Year 2021-2025

INDEX

S. NO.	EXPERIMENTS	DATE	REMARKS
1	Stimulate the line coding techniques using MATLAB and Simulink.		
2	Simulate the Binary amplitude shift keying using MATLAB and Simulink.		
3	Simulate the Binary amplitude shift keying using MATLAB and Simulink.		
4	Write a Program to introduce an input box in a WML form.		
5	Write a Program to a variable and its value in WML form.		
6	Write a program to perform navigation between WML cards (forward and backward both)		
7	Write a Program to activate the current card in WML .		
8	Simulate the Binary phase shift keying using MATLAB and Simulink.		
9	Simulate the Direct sequence spread spectrum keying using MATLAB and Simulink.		
10	Simulate the Adaptive Delta Modulation using MATLAB and Simulink.		

Experiment - 1

Aim : Stimulate the line coding techniques using MATLAB and Simulink.

Code :

1. NRZ(Polar and Unipolar):

```
N = input('Enter the length of bitstream');
bitstream = rand(1,N) > 0.5
timeperiod = input('Enter the timeperiod (No. of samples)');
%samples
time = 0:(timeperiod*length(bitstream)-1);
%Time period = 100 samples
repeat = ones(1,timeperiod);
NRZP = bitstream.' * repeat;
NRZP = NRZP.';
NRZP = NRZP(:)';
%Unipolar
NRZU = NRZP;
figure(1)
subplot(2,2,1)
plot(time,NRZU);
axis([0,10*timeperiod,-1.5,1.5]);
title('NRZ Unipolar')
xlabel('Time(s)')
ylabel('Voltage(V)’)

%Polar
NRZP = (2.*NRZP) -ones(1,length(NRZP));
subplot(2,2,2)
plot(time,NRZP);
axis([0,10*timeperiod,-1.5,1.5]);
title('NRZ Polar')
xlabel('Time(s)')
ylabel('Voltage(V)')
```

2. RZ(Polar and Unipolar):

```
intermediate = ones(1,length(bitstream)*2);
intermediate(2:2:length(intermediate)) = 0;
```

```
intermediate = intermediate*
ones(1,(timeperiod/2));intermediate = intermediate.';
intermediate = intermediate(:)';%UnipolarRZU = NRZU .*
intermediate;
subplot(2,2,3)
plot(time,RZU);
axis([0,10*timeperiod,-1.5,1.5]);
title('RZ Unipolar')
xlabel('Time(s)')
ylabel('Voltage(V)')%PolarRZP = NRZP .* intermediate;
subplot(2,2,4)
plot(time,RZP);
axis([0,10*timeperiod,-1.5,1.5]);title('RZ Polar')
xlabel('Time(s)')
ylabel('Voltage(V)')
```

Output :

Experiment – 2

Aim : Simulate the Binary amplitude shift keying using MATLAB and Simulink.

MATLAB Code FOR ASK (Amplitude Shift Keying) :

```
clc %for clearing the command window
close all %for closing all the window except command window
clear all %for deleting all the variables from the memory
fc=input('Enter the freq of Sine Wave carrier:');
fp=input('Enter the freq of Periodic Binary pulse (Message):');
amp=input('Enter the amplitude (For Carrier & Binary Pulse Message):');
t=0:0.001:1; % For setting the sampling interval
c=amp.*sin(2*pi*fc*t);% For Generating Carrier Sine wave
subplot(3,1,1) %For Plotting The Carrier wave
plot(t,c)
xlabel('Time')
ylabel('Amplitude')
title('Carrier Wave')
m=amp/2.*square(2*pi*fp*t)+(amp/2);%For Generating Square wave message
subplot(3,1,2) %For Plotting The Square Binary Pulse (Message)
plot(t,m)
xlabel('Time')
ylabel('Amplitude')
title('Binary Message Pulses')
w=c.*m; % The Shift Keyed Wave
subplot(3,1,3) %For Plotting The Amplitude Shift Keyed Wave
plot(t,w)
xlabel('Time')
ylabel('Amplitude')
title('Amplitide Shift Keyed Signal')
```

Output:

Experiment - 3

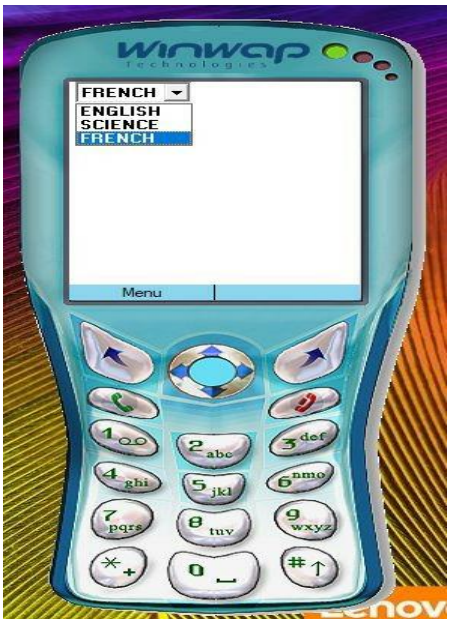
AIM: Write a program with two cards one for user input and other for displaying the result.

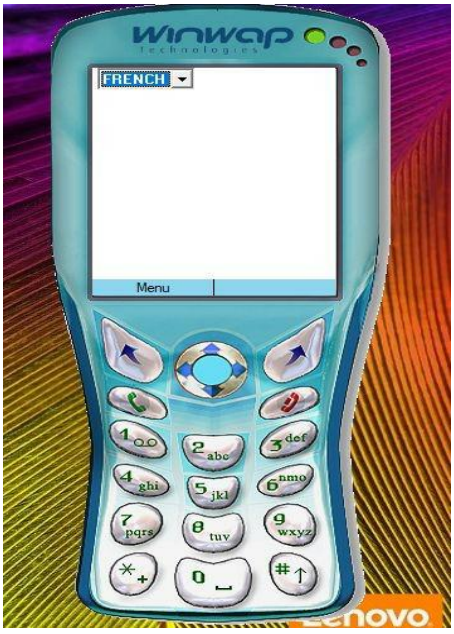
CODE:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="CARD1" title="TUTORIAL">
 <p>
                    <do type="ACCEPT" label="MY_SUBJECT">
                       <go href="#CARD2"/>
                         </do>
                       <select name="NAME">
                    <option value="English">ENGLISH </option>
                    <option value="Science">SCIENCE</option>
                    <option value="French">FRENCH</option>
```

```
                    </select>
</p>
   </card>
  <card id="CARD2" title="MY_SUBJECT">
   <p>YOU SELECTED: $(NAME)</p>
  </card></wml>
```

Output:





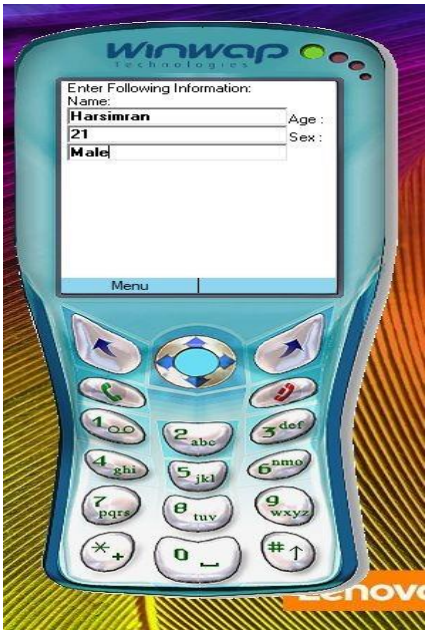
Experiment - 4

Aim- Write a Program to introduce an input box in a WML form.

Code

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"

  "http://www.wapforum.org/DTD/wml13.dtd">

<wml>

<card title="Input Fields">

<p >Enter Following Information:<"Br">

Name: <input name=”name” size="12"/>

Age:  <input name=”age” size="12" format=”*N”/>

Sex:  <input name=”sex” size="12"/>

</p>

</card>

</wml>
```

Output:



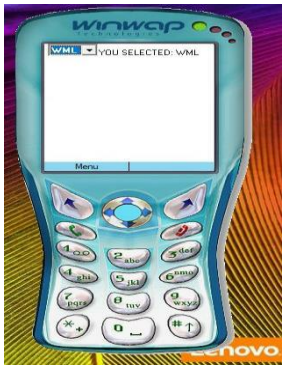
Experiment - 5

Aim Write a Program to a variable and its value in WML form.

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="CARD1" title="TUTORIAL">
 <p>
                                  <select name="NAME">
                                 <option value="WML">WML </option>
                                 <option value="HTML">HTML</option>
                                 <option value="Xml">XML</option>
                                 </select>
            <do type="ACCEPT" label="MY_SUBJECT">
                                 <go href="#CARD1"/>
                                 </do>
                       YOU SELECTED: $(NAME)</p>
 </card>
</wml>
```

Output:



Experiment - 6

Aim- Write a program to perform navigation between WML cards (forward and backward both)

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="c1" title="Card #1">
   <p align="center">
 <big><b>First Card</b></big><br/>
 Hello Everyone<br/>
<a href="#c2">Card2</a><br/>  <a
href="#c3">Card3</a><br/>
   </p>
 </card>
<card id="c2" title="Card #2">
      <p align="center">
    <big><b>Second Card</b></big><br/>
Welcome to WML<br/>
<a href="#c1">Back</a><br/>
<a href="#c3">Next</a><br/>
   </p>
</card>
<card id="c3" title="Card #3">
<p align="center">
```

<big>Third Card</big>

Have a good day

 Back to Card1

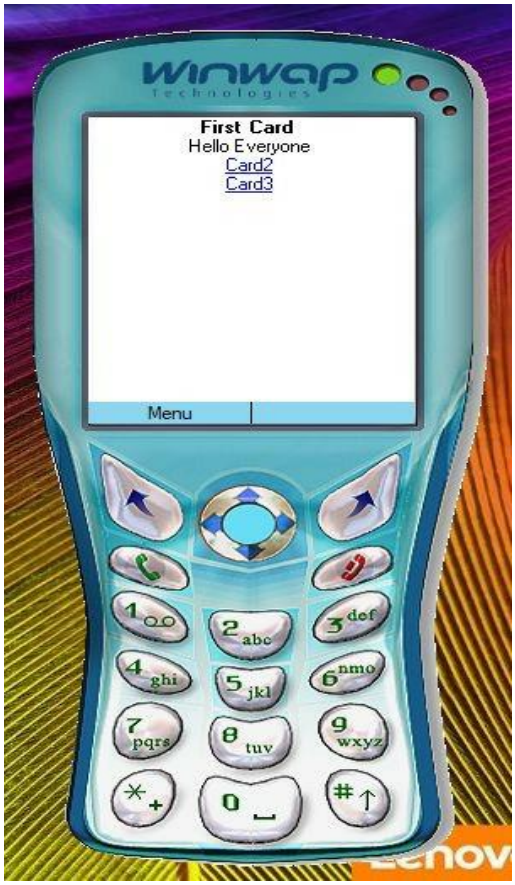
 Back to Card2

</p>

</card>

</wml>

Output:



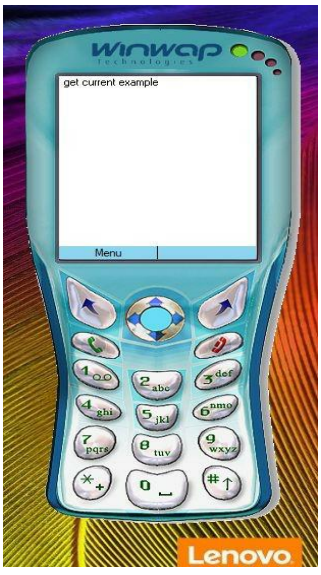
Experiment - 7

Aim - Write a Program to activate the current card in WML

Code

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"

    "http://www.wapforum.org/DTD/wml13.dtd">

<wml>

 <card id="card1" title="Card #1">

 <p> get current example</p>
<do type="accept">

<go href="GetCurrentCardEg.wmls#find()"/>

</do>

 </card>

 <card id="card2" title="Card #2">

      <p align="center">  My card
no. is two <br/>
current_card=$(currentcard);
   </p>

 </card>

</wml>
```

Output:



Experiment - 8

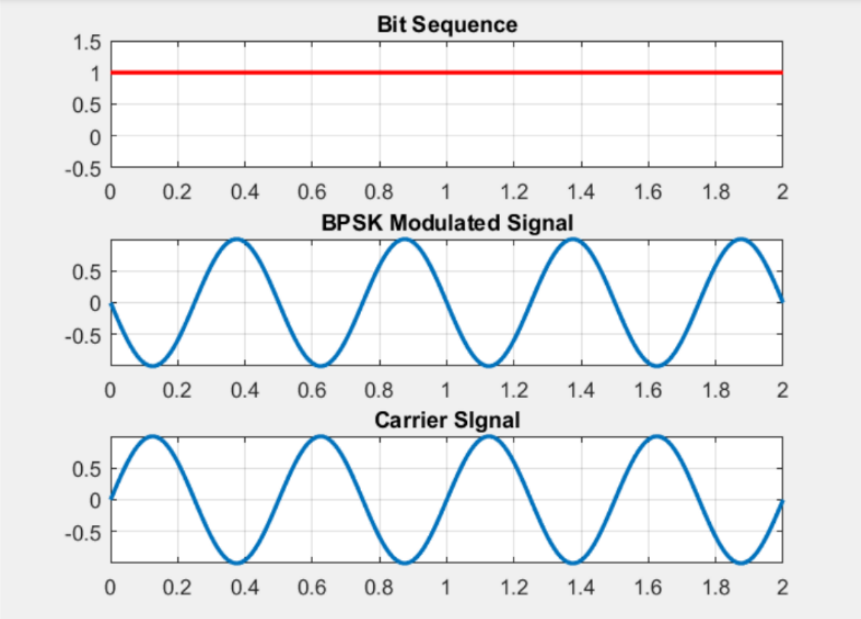
AIM: Simulate the Binary phase shift keying using MATLAB and Simulink.

Program Code

```
 Clc
 clear all;
 close all;
f = 2; %frequency of sine wave
 fs = 100; %sampling period of the sine wave
 t = 0:1/fs:1; %splitting time into segments of 1/fs
%setting the phase shifts for the different BPSK
signals
p1 = 0;
 p2 = pi;
 %getting the number bits to be modulated
 N = input('enter the number of bits to be modulated: N = ');
%generating the random signal
 bit_sequence=round(rand(1,N));
%allocating the dynamic variables
 time = [];
 digital_signal = [];
PSK = [];
carrier_signal = [];
%GENERATING THE SIGNALS
 for ii = 1:1:N
 %the original digital signal
is if bit_sequence(ii) == 0
 bit = zeros(1,length(t));
 else
 bit = ones(1,length(t));
end
digital_signal = [digital_signal bit];
%Generating the BPSK signal
if bit_sequence(ii) == 0
 bit = sin(2*pi*f*t+p1);
 else
 bit = sin(2*pi*f*t+p2);
end
PSK = [PSK bit];

 %Generating the carrier wave
```

```
carrier = sin(2*f*t*pi);
carrier_signal = [carrier_signal carrier];
time = [time t];
t = t + 1;
end
subplot(3,1,1);
plot(time,digital_signal,'r','linewidth',2);
grid on;
axis([0 time(end) -0.5 1.5]);
 title('Bit Sequence')
 subplot(3,1,2);
plot(time,PSK,'linewidth',2);
grid on;
axis tight;
 title('BPSK Modulated Signal')
subplot(3,1,3);
plot(time,carrier_signal,'linewidth',2);
 grid on;
 axis tight;
title('Carrier SIgnal')
```

Output:



Experiment - 9

AIM: Simulate the Direct sequence spread spectrum keying using MATLAB and Simulink.

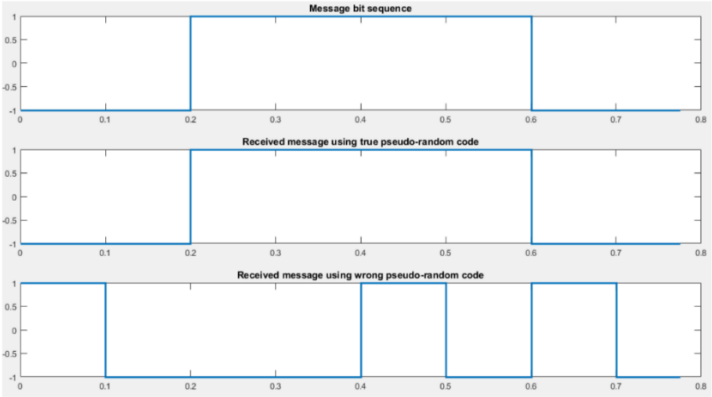
Program Code

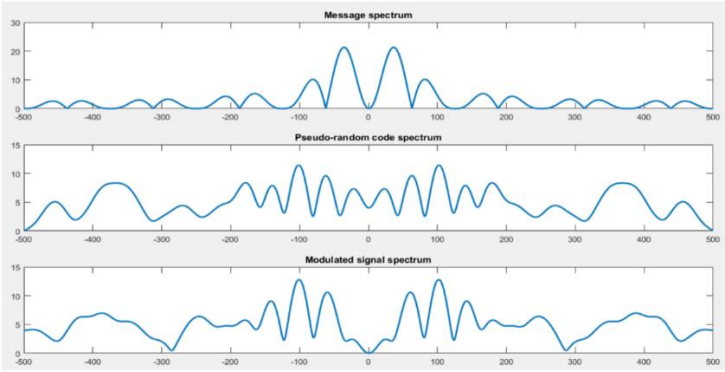
```
clear all;
%% parameters
Fs = 1000;
fc = 100;
fp = 4;
bit_t = 0.1;
%% message generation with BPSK
m = [0 0 1 1 1 1 0 0];
for bit = 1:length(m)
  if(m(bit)==0)
     m(bit) = -1;
  end
end
message =  repmat(m,fp,1);
message =  reshape(message,1,[]);
%% PN generation and multiply with message
pn_code = randi([0,1],1,length(m)*fp);
for bit = 1:length(pn_code)
  if(pn_code(bit)==0)
     pn_code(bit) = -1;
  end
end
DSSS = message.*pn_code;
%% create carrier and multipy with encoded sequence
t = 0:1/Fs:(bit_t-1/Fs);
s0 = -1*cos(2*pi*fc*t);
s1 = cos(2*pi*fc*t);
carrier = [];
BPSK = [];
for i = 1:length(DSSS)
   if (DSSS(i) == 1)
      BPSK = [BPSK s1];
   elseif (DSSS(i) == -1)
      BPSK = [BPSK s0];
   end
   carrier = [carrier s1];
```

```
end
message1 =  repmat(result,fp,1);
message1 =  reshape(message1,1,[]);
message2 =  repmat(resultWrong,fp,1);
message2 =  reshape(message2,1,[]);
%% Draw original message, PN code , encoded sequence on time domain
pn_size = length(pn_code);
tpn = linspace(0,length(m)*bit_t-bit_t/fp,pn_size);
tm = 0:bit_t/fp:length(m)*bit_t-bit_t/fp;
figure
subplot(311)
stairs(tm,message,'linewidth',2)
title('Message bit sequence')
axis([0 length(m)*bit_t -1 1]);
subplot(312)
stairs(tpn,pn_code,'linewidth',2)
title('Pseudo-random code');
axis([0 length(m)*bit_t -1 1]);
subplot(313)
stairs(tpn,DSSS,'linewidth',2)
title('Modulated signal');
axis([0 length(m)*bit_t -1 1]);
figure
subplot(311)
stairs(tm,message,'linewidth',2)
title('Message bit sequence')
axis([0 length(m)*bit_t -1 1]);
subplot(312)
stairs(tm,message1,'linewidth',2)
title('Received message using true pseudo-random code')
axis([0 length(m)*bit_t -1 1]);
subplot(313)
stairs(tm,message2,'linewidth',2)
title('Received message using wrong pseudo-random code')
axis([0 length(m)*bit_t -1 1]);
%% Draw original message, PN code , encoded sequence on frequency domain
f = linspace(-Fs/2,Fs/2,1024);
figure
subplot(311)
plot(f,abs(fftshift(fft(message,1024))),'linewidth',2);
```

```
title('Message spectrum')
subplot(312)
plot(f,abs(fftshift(fft(pn_code,1024))),'linewidth',2);
title('Pseudo-random code spectrum');
subplot(313)
plot(f,abs(fftshift(fft(DSSS,1024))),'linewidth',2);
title('Modulated signal spectrum');
figure;
subplot(311)
plot(f,abs(fftshift(fft(BPSK,1024))),'linewidth',2);
title('Transmitted signal spectrum');
subplot(312)
plot(f,abs(fftshift(fft(rx,1024))),'linewidth',2);
title('Received signal multiplied by pseudo code');
subplot(313)
plot(f,abs(fftshift(fft(demod,1024))),'linewidth',2);
title('Demodulated signal spectrum before decision device ');
```

Output:





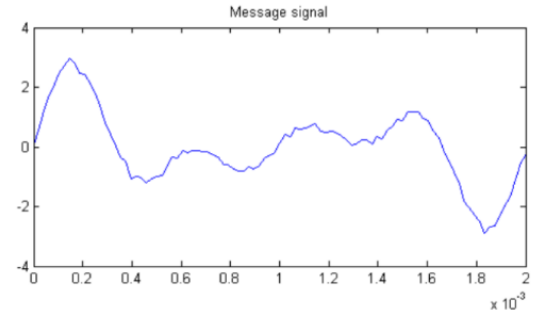
Experiment - 10

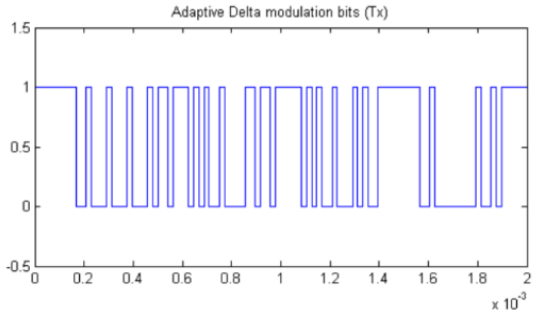
AIM: Simulate the Adaptive Delta Modulation using MATLAB and Simulink.

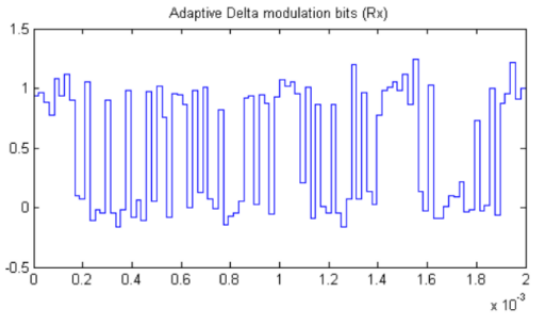
Program Code

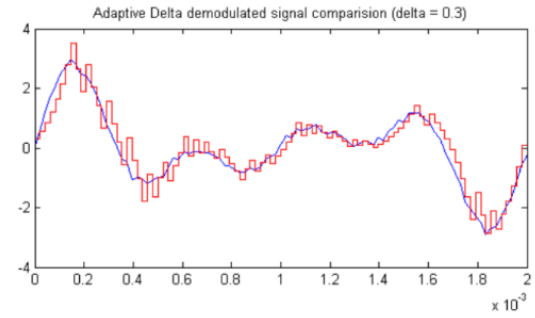
```
clc;
clear all;
close all;
% Simulation settings
tStart = 0;
tStop = 0.002;
Fs = 48 * 10^3;
delta = 0.3;
SNR = 20;
% Calculate
Ts = 1 / Fs;
t = tStart : Ts : tStop;
% Perform adaptive delta modulation and demodulation
xt_sig = awgn(sin(2*pi* 1000 * t) + sin(2*pi* 1500 * t) + sin(2*pi* 2000 * t), 20);
yt_sig = sig_mod_adaptivedelta(xt_sig, delta, 0, 1.2);
yr_sig = awgn(yt_sig, SNR);
xr_sig = sig_demod_adaptivedelta(yr_sig, delta, 0, 1.2);
% Plot results
subplot(2,2,1);
plot(t, xt_sig, 'b');
ylim([-4 4]);
title('Message signal');
subplot(2, 2, 2);
stairs(t, yt_sig);
ylim([-0.5 1.5]);
title('Adaptive Delta modulation bits (Tx)');
subplot(2, 2, 3);
stairs(t, yr_sig);
title('Adaptive Delta modulation bits (Rx)');
subplot(2, 2, 4);
plot(t, xt_sig, 'b');
hold on;
stairs(t, xr_sig, 'r');
ylim([-4 4]);
title('Adaptive Delta demodulated signal comparision (delta = 0.3)');
```

Output:


Message signal
4
2
0
-2
-4
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
x 10^-3


Adaptive Delta modulation bits (Tx)
1.5
1
0.5
0
-0.5
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
x 10^-3


Adaptive Delta modulation bits (Rx)
1.5
1
0.5
0
-0.5
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
x 10^-3


Adaptive Delta demodulated signal comparision (delta = 0.3)
4
2
0
-2
-4
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
x 10^-3