**东南大学**

**《协作通信与网络》**

**实验报告**

**论文题目：差错检测下DF中继协议网络信噪比性能**

**姓 名 ：段斯琪**

**学 号 ：04016407**

**专业班级：040614班**

**学院名称：信息科学与工程学院**

**2018年12月**

## 一、实验目的

实验工具：Matlab R2016a

实验目的：了解协作中继的模式之一—DF协作及其基本原理，进一步掌握MATLAB的使用方法。

## 二、实验要求

理解DF协作的基本原理，独立完成DF基本程序的编写，根据实验结果进行分析，完成实验报告。

## 三、实验内容

1.实验原理

协作通信是指允许用户帮忙传输彼此间消息至目的地的系统或技术。协作中继是其基本类型之一。协作中继的基本模型是：一个源节点（S），一个目的节点（D），若干个中继节点（R）。最简单的情况是只有一个中继节点，我们称之为S—R—D模型。

DF也称再生中继。在DF协议下，中继要先对接收到的信号进行解调、采样判决、存储、译码等数字处理，然后将处理后的数据进行编码调制后再转发，即中继节点接收到信号后先解码再转发。

由于DF为再生中继方式，因此，DF本质上是一种数字信号处理方式。DF方式通过译码，避免了噪声对下一跳的影响。但是，如果中继节点对所接收到的信号直接进行译码，之后将所译出的信号转发给目的节点，将有可能引起错误传播。这是由于中继节点有可能得到的是错误信息，这样对协作传输反而是不利因素。

下面将详细介绍单中继DF协作通信过程。

（1）源端广播过程

源节点S以广播的方式向周围发送信号，其中一路直接发送到目的节点D，一路发送到中继节点R。

（2）中继端解码重传过程

中继节点R直接将接收到来自源节点S的信号直接解调译码，并通过某种方式校验译码是否正确，如果错误则中继不再发送该信息，如果正确。则该信号重新编码调制，然后转发给目的节点D。在编码过程中，可以选择与源节点一样的编码方案，或者采取不同的编码方式。

有两种简单的方法可以降低错误解码所带来的不利影响：

第一种方法是信号在发射之前先进行循环冗余校验（Cyclical Redundancy Check,CRC）码处理。这样，中继节点接收到信号后，将对信号进行解调和解码（如果没有编码则不需要解码），并对数据进行CRC校验，判别接收到的信息比特里是否存在错误。如果数据正确，就按照信源的编码调制方式将信号进行编码和调制，再将数据发射出去；如果数据不正确即将帧丢弃。但是CRC的引入将降低信息的传输速率。

第二种方法不需要对源信号进行CRC编码操作，只需在每个节点处设定一个门限值。在对接收信号译码处理之前，先比较它的等效信噪比与门限值得大小。如果大于门限值，中继节点将进行译码处理，并进行信息转发；反之不对信号处理。基于门限的方法虽然简单，但是门限值的选择至关重要。如果太小，中继节点译出的信息很可能存在错误；如果太大，每个中继节点可能都不会进行信息转发，这样协作将失去意义。另外，即使是等效信噪比大于门限值，也并不能保证中继节点译出信息一定正确。DF方式不会带来噪声传播问题，但受源—中继端信道传输性能影响较大，若编码方式不采用CRC码，得不到满分集阶数。

经过中继重新编码调制信号为，目的节点接收到的来自中继的信号变为:

（3—1）

（3）目的端接收处理

DF协议下的目的节点按照MRC方案得到合并信:

（3—2）

加权系数、取决于两路信号的信噪比最优值：

（3—3）

（3—4）

对于DF模式，中继节点对源节点信息解码错误所带来的误差会随着跳数的增加而不断积累，从而影响到分集效果和中继性能。这表明源—中继节点信道传输特性的好坏对DF方式协同通信系统的性能有很大影响。

2.实验原理图



3.实验代码

N=100;

L=65; %一帧长度

BerSnrTable=zeros(20,5);

for snr=0:25

BerSnrTable(snr+1,1) = snr;

sig=1/sqrt(10^(snr/10)); %将dB转化为十进制数值

temp=0;

temp1=0;

for i=1:N

BitsTx = floor(rand(1,L)\*2); %向下取整；BitsTx为初始输入信号

BitsTxcrc=CrcEncode(BitsTx); %循环检测编码；BitsTxcrc为循环编码后的信号

BitsTxcnv=cnv(BitsTxcrc); %二进制卷积编码

Mod8Tx=mod\_8psk(BitsTxcnv); %8PSK编码

M=length(Mod8Tx); %8PSK编码后的序列长度

%以下为假设信道模型和噪声模型，由于本次仿真重点不在于此，所以做以下简化

%sr表示信源--中继；sd表示信源--目的；rd表示中继--目的

Hsd=RayleighCH(); %源节点和目的节点之间的信道

Hsr=RayleighCH(); %源节点和中继节点之间的信道

Hrd=RayleighCH(); %中继节点和目的节点之间的信道

Nsd=randn(1,M)+1i\*randn(1,M); %源节点和目的节点之间的噪声

Nsr=randn(1,M)+1i\*randn(1,M); %源节点和中继节点之间的噪声

Nrd=randn(1,M)+1i\*randn(1,M); %中继节点和目的节点之间的噪声

% 目的节点接收到用户1即源节点的功率

Ysd=Hsd.\*Mod8Tx+sig\*Nsd; %目的节点接收到的源节点的信号功率和噪声功率之和

%中继接收并解码

Ysr=Hsr.\*Mod8Tx+sig\*Nsr; %中继节点接收源节点的信号功率和噪声功率之和

Rsr=conj(Hsr).\*Ysr; %conj为取共轭

BitRsr=demod\_8psk(Rsr); %8PSK方式解码，BitRsr为解码后的码

BitRsrviterbi=viterbi(BitRsr); %最优路径\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*不懂viterbi函数具体干嘛的\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BitRsrviterbi=BitRsrviterbi(1:length(BitRsrviterbi)-1);

[BitRsrdecrc,error]=CrcDecode(BitRsrviterbi); %BitRsrdecrc为中继解码后的码；error为1则有错，为0则无错

%非协作情况

if(error==1)

Rsd=conj(Hsd).\*Ysd;

BitRsd=demod\_8psk(Rsd); %目的节点接收到的源节点的解码

BitRsdviterbi=viterbi(BitRsd); %viterbi译码

BitRsdviterbi=BitRsdviterbi(1:length(BitRsdviterbi)-1);

BitR1ddecrc=CrcDecode(BitRsdviterbi); %循环解码BitR1ddecrc为循环解码得到的码

[Num,Ber] = symerr(BitR1ddecrc,BitsTx); %symerr函数用来计算错误符号的个数和误符号率

%Num指出了两组数据集相比不同符号的个数；Ber为误码率，它等于Num除以总符号数

BerSnrTable(snr+1,2)=BerSnrTable(snr+1,2)+Num;

end

%协作情况

if(error==0)

Bitsrd=BitRsrdecrc; %中继解码后准备发送至目的节点的码

Bitsrdcrc=CrcEncode(Bitsrd); %对解码后的码进行循环编码

Bitsrdcnv=cnv(Bitsrdcrc); %二进制卷积编码

Mod8\_rd=mod\_8psk(Bitsrdcnv); %对编码后的码进行8PSK调制

Yrd=Hrd.\*Mod8\_rd+sig\*Nrd; %目的节点接收到的中继节点的功率

%最大合并比在此处的简化形式

Rd=conj(Hrd).\*Yrd+conj(Hsd).\*Ysd;

BitRd=demod\_8psk(Rd); %目的节点对接收到的中继节点的码进行8PSK方式解码

BitRdviterbi=viterbi(BitRd); %viterbi译码

BitRdviterbi=BitRdviterbi(1:length(BitRdviterbi)-1);%这句老师说是编程习惯.....

BitRddecrc=CrcDecode(BitRdviterbi); %循环解码

[Num,Ber] = symerr(BitRddecrc,BitsTx); %symerr函数用来计算错误符号的个数和误符号率

%Num指出了两组数据集相比不同符号的个数；Ber为误码率，它等于Num除以总符号数

BerSnrTable(snr+1,2)=BerSnrTable(snr+1,2)+Num;

temp=temp+1;

end

end

BerSnrTable(snr+1,3)=BerSnrTable(snr+1,2)/(L\*N); %此处将M改成N

BerSnrTable(snr+1,4)=temp;

end

semilogy(BerSnrTable(:,1),BerSnrTable(:,3),'r\*-');

grid on;

figure

semilogy(BerSnrTable(:,1),BerSnrTable(:,4),'g\*-');

grid on;

time\_of\_sim = toc; %记录程序完成时间

echo on;

4.实验结果



