OFDM数字通信系统的简单模拟

摘要

本文讨论了在给定多径信道参数的情况下OFDM算法的实现过程，并使用Matlab做了仿真，模拟出了给定条件下子载波个数与误码率的关系。需要说明的是，本文的介绍重点在于OFDM，因此没有考虑信源编解码、信道编解码、基带调制、载波调制等过程。

符号记号表

符号	意义
τ_{min}	多径信道时延的最小时间单位
S_k	待传输的第k 个符号
N_s	子载波的个数
T_{symbol}	一个OFDM符号的持续时间
T_{sample}	数字通信系统的采样时间间隔
Δf	相邻子载波的频率差
$S（t）$	模拟系统发送机发送的信号
$S[n]$	数字系统发送机发送的信号
f_k	第k 个子载波的频率
τ_k	第k 个路径的时间延迟
α_k	第k 个路径的幅度衰减
N_{path}	多径信道的路径个数
$\otimes$	循环卷积
$\mathcal{F}\{\cdot\}$	连续或离散傅立叶变换算子
$h[n]$	多径信道的单位冲激响应
L	$h[n]$有限序列的长度
$R[n]$	接收机接收到的信号序列
$\hat{S}[n]$	对发送信号的估值
$\hat{S}_k$	对传输符号的估值
$interval\_ratio$	一个采样周期对应的标准时长 $\frac{T_{sample}}{\tau_{min}}$

过程详细描述

多径信道的影响

为了在数字系统中考虑多径信道的影响，需要将多径信道模型离散化。我们用单位冲激响应$h[n]$ 来给信道建模，两个相邻的离散时间间隔为τ_{min} 根据信道的参数$\{\tau_k, \alpha_k\}$ 不难求出信道的单位冲激响应：

$$h[n] = \sum_{k=0}^{N_{path}-1} \delta[n - \frac{\tau_k}{\tau_{min}}] \cdot \alpha_k$$

信道对发送信号的作用用线性卷积来描述：

$$R[n] = S[n] * h[n] + Noise$$

消除多径效应的影响

在满足一定条件时，连续时间傅立叶变换有这样一条性质[2]：

$$\mathcal{F}[f * g] = \mathcal{F}[f] \cdot \mathcal{F}[g]$$

这启发我们，如果我们将接收信号看作$f * g$，对其做傅立叶变换后利用除法就可能消除掉信道的影响。

但是这里我们讨论的是数字系统，使用的是离散傅立叶变换，因此有些不同之处需要考虑。首先，离散傅立叶变换的对应的性质为：

$$\mathcal{F}\{x \otimes y\} = \mathcal{F}\{x\} \cdot \mathcal{F}\{y\}$$

为了使线性卷积等效于循环卷积，我们使用循环前缀技术[3]

我们在$S[n] = \{x[0], x[1] \cdots x[N_s - 1]\}$ 的前面填入长度为$L - 1$的尾部序列，填充完序列如下所示：

$$S[n] = \{x[N_s - L + 1], \cdots, x[N_s - 2], x[N_s - 1], x[0], x[1], \cdots x[N_s - 1]\}$$

此时有：

$$\mathcal{F}\{\mathcal{S}_n \otimes h[n]\} = \mathcal{F}\{\mathcal{S}_n\} \cdot \mathcal{F}\{h[n]\}$$

$$S_n = \mathcal{F}^{-1}\{\frac{\mathcal{F}\{\mathcal{R}[n]\}}{\mathcal{F}\{h[n]\}}\}$$

这样就可以消除多径效应的影响，得到原始的发送信号了。

发送机

模拟OFDM系统中发送机发送的信号为[1]：

$$S(t) = \sum_{k=0}^{N_s - 1} S_k \cdot e^{j2\pi f_k t}$$

为了将此模拟系统转换为数字系统，我们以T_{sample} 的采样周期进行理想采样，此时上式成为：

$$S(nT_{sample}) = \sum_{k=0}^{N_s - 1} S_k \cdot e^{j2\pi f_k n T_{sample}}$$

为了使子载波之间相互正交[1]，我们令$f_k = k\Delta f$；同时我们选择在T_{symbol}的时间内采样N_s 次，即$T_{sample} = \frac{T_{symbol}}{N_s}$；令$\Delta f = \frac{1}{T_{symbol}}$此时上式成为：

$$S[n] = \sum_{k=0}^{N_s - 1} S_k \cdot e^{j2\pi k \frac{n}{N_s}}$$

注意到上式在形式上类似$IDFT$，具体的：

$$S[n] = N_s \cdot \mathcal{F}^{-1} S_k$$

([1]中PPT关于DFT、IDFT的公式好像写错了)

为了消除多径效应的影响，我们对$S[n]$ 做循环前缀处理，即将$S[n]$ 填充为下式所示：

$$S[n] = \{x[N_s - L + 1], \cdots, x[N_s - 2], x[N_s - 1], x[0], x[1], \cdots x[N_s - 1]\}$$

这就是发送机最终发送的信号。

接收机

当$interval_ratio > 1$ 时，发送的数据量大于计算所需，接收机可以选择采样。 最简单的可以选择仍按T_{sample} 的速率采样，丢弃多余的数据。

对接受信号$R[n]$做一些处理不难得到原始发送信号$\hat{S}[n]$:

$$\hat{S}_n = \mathcal{F}^{-1}\{\frac{\mathcal{F}\{\mathcal{R}[n]\}}{\mathcal{F}\{h[n]\}}\}$$

接着从$\hat{S}[n]$ 中恢复出$\hat{S}_k$:

$$\hat{S}_k = \mathcal{F}\{\frac{S[n]}{N_s}\}$$

Matlab仿真及结果

定义及说明

对时间的定义

为了便于仿真，我们将时间的最小单位定义成τ_{min}，因此，信号$S[n]$ 相邻两个元素的时间间隔为$\tau_{min} \times interval\ ratio$ 。发送N_s个符号花费的时间是（$N_s + L - 1$）$\times interval\ ratio \times \tau_{min}$

对信道的定义

定义$h[n] = \{0.2, 0, 0, 0.7, 0, 0.1\}$

对发送信号的定义

随机生成N_s 元素，元素 $\in \{1, 2\}$

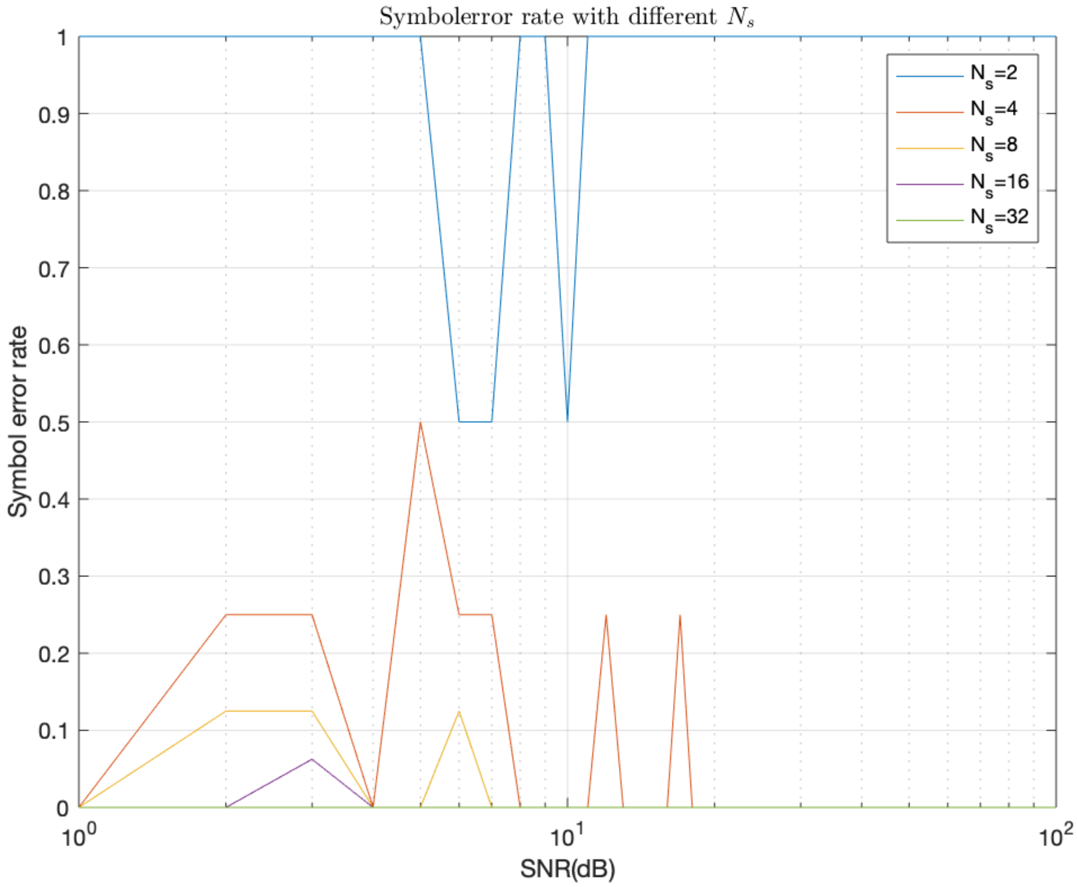
接收机采样

仍按照T_{sample} 的速率采样，丢弃多余数据。

定义噪声

可以根据信噪比调整的加性高斯白噪声

仿真结果



可以发现，子载波个数比较少的时候并不能消除码间串扰。这是由于发送的序列长度相比$h[n]$ 不够长，在卷积过程中码间仍有干扰。相比之下，噪声对误码率的影响不是十分明显。可能是由于相同条件下只计算了一次误码率，随机性太大。但是整体的趋势是信噪比越大，误码率越小。

参考文献

[1] 郭俊奇. 通信原理1-8[R].北京：北京师范大学人工智能学院，2020.

[2] "傅立叶变换" Wikipedia, The Free Encyclopedia. 22 July 2004, 10:55 UTC. Wikimedia Foundation, Inc. 10 Aug. 2004.

[3] "Cyclic prefix" Wikipedia, The Free Encyclopedia. 22 July 2004, 10:55 UTC. Wikimedia Foundation, Inc. 10 Aug. 2004.

附录

```
function error_rate = cal_OFDM(interval_ratio, N_s, SNR)
% Brief: Roughly simulate OFDM transmission process and calculate symbol error rate
```

```
% return symbol error rate
% Author: Dongxu Guo
% Date: 3.29.2020

% Definition and Declaration
% Assume there are N_s subcarrier wave
% N_s different symbols represented by integer (1, 2)
S_k = randi(2, N_s, 1)';

% impulse response of the channel
h_channel = [0.7, 0.3, 0.1];
H_k = fft(h_channel, N_s);
L = 3;

% signal-to-noise ratio(dB)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Transmitter
S_n = N_s*ifft(S_k);
S_n = [S_n(N_s-L+2:end), S_n];  % Cyclic prefix

% Transmission through the channel
C_r = zeros(1, interval_ratio*(N_s+L-1));
for i = 1:(N_s+L-1)
    C_r(interval_ratio*i-interval_ratio+1:interval_ratio*i) = S_n(i)*ones(1,
interval_ratio); % Padding S_n to the corresponding size
end

S_r = conv(C_r, h_channel);            % Multipath effect
S_r_withNoise = awgn(S_r, SNR);        % Gaussian Noise

% Receiver
Receiver_Sample = zeros(1, N_s+L-1);
for i = 1:(N_s+L-1) % Sample discarding L-1 elements caused by the convolution
    Receiver_Sample(i) = S_r_withNoise(interval_ratio*i-
floor(interval_ratio/2));
end
Demodulation = fft(Receiver_Sample(L:end))./N_s;

S_n_bar = round(abs(Demodulation./H_k));



% Calculate bit error rate
error = sum(S_n_bar ~= S_k);
error_rate = error/N_s;

end
```