现代通信系统与网络仿真技术

大作业

Cantjie
cantjie@stu.xjtu.edu.cn

Problem1

题干略

After sampling, we have the energy spectral density

$$G_{MSKsampling}(f) = f_s^2 \sum_{n=-\infty}^{\infty} G_{MSK}(f - nf_s)\,.$$

Hence we get the energy of signal and aliasing noise

$$E_{signal} = \int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} f_s^2 G_{MSK}(f)\,df\,,$$

$$E_{noise} = \int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} f_s^2 \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} G_{MSK}(f - nf_s)\,df\,,$$

separately. By observing the representation of E_{nosie} and the propriety of $G_{MSK}(f)$, we derive

$$E_{noise} = \int_{\frac{f_s}{2}}^{\infty} 2 f_s^2 G_{MSK}(f)\,df\,,$$

Then the resultant $(SNR)_a = \frac{E_{signal}}{E_{noise}}$. However, the integral is hard to solve, so we derive an approximation of $(SNR)_a$ by numerical integration:

$$(SNR)_a \approx \frac{\sum_{j=0}^{\frac{kf_s}{2}} G_{MSK}\left(\frac{j}{k}\right)}{\sum_{j=\frac{kf_s}{2}}^{\infty} G_{MSK}\left(\frac{j}{k}\right)}\,.$$

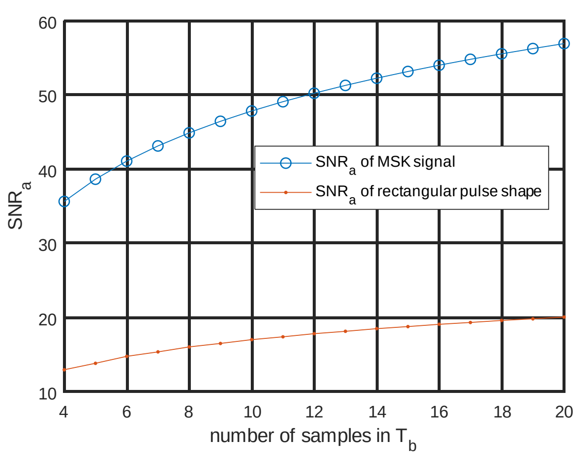


Figure P1.a

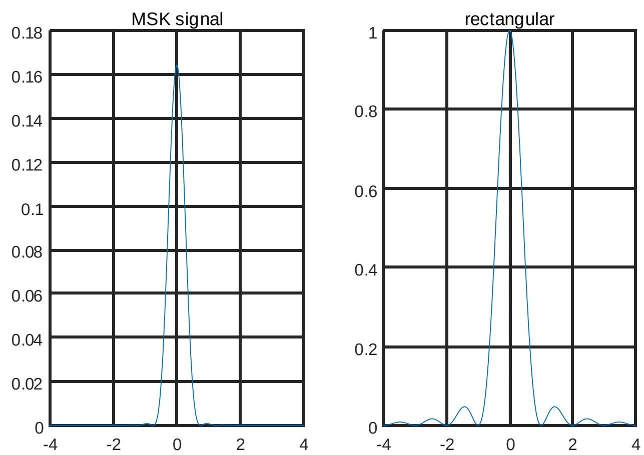


Figure P1.b

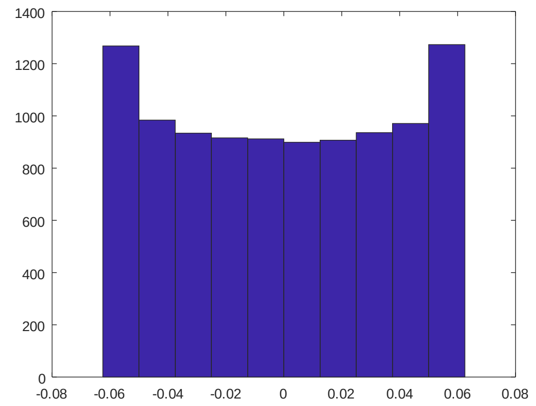


Figure P2.a

The comparison of SNR is shown in Figure P1.a, which shows the signal to aliasing noise ratio of MSK signal is much higher than that of rectangular shape pulse signal, under same sampling frequency.

By plotting the energy spectral density in Figure P1.b, we can easily explain the gap: Energy spectral density of the rectangular shape pulse has a relative much higher sidelobe compared with that of MSK signal, whose sidelobe is very close to zero.

Problem 2

题干略

(1) To avoid the vector being periodic, which will reduce the valid data.

(3) Quantizing level $\Delta = \frac{2}{2^4} = 0.125$. Theoretical values:

$$E\{e[k]\} = 0.$$

$$E\{e^2[k]\} = \frac{\Delta^2}{12} = 0.0013.$$

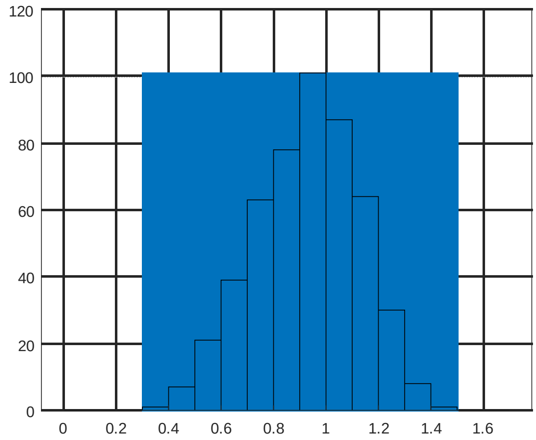
Experimental values:

$$\widehat{E}\{e[k]\} = \ -5.5155 \times 10^{-5} \approx 0$$

$$\widehat{E}\{e^2[k]\} = 0.0015.$$

The theoretical values and experimental values are close.

(4) Histogram is shown as Figure P2.a. We conclude that it's a proper model to view quantizing error variant $e[k] = x_q[k] - x[k]$ as a uniformly distributed variant. But in this case of sin function, the probability of large error is greater than that of small error.



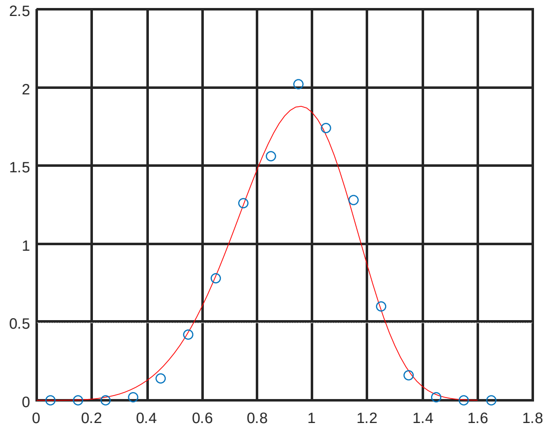


Figure P3.a

Figure P3.b

Problem 3

题干略

Solution:

$$
\begin{aligned}
f_X(x) &= a x^{a-1} e^{-x^a} u(x) \\
F_X(x) &= \int f_X(x) dx \\
&= \int_0^\infty a x^{a-1} e^{-x^a} dx \\
&= 1 - e^{-x^a} (x > 0)
\end{aligned}
$$

Let $U = F_X(X)$, then $X = F_X^{-1}(U)$

$$X = \sqrt[a]{-\ln(1-U)}$$

where U is uniformly distributed in (0,1).

Histogram is shown in Figure P3.a and the PDF is shown in P3.b. To compare them, we use points in Figure P3.b to represent the relative value of histogram bins. Generating only 500 samples makes the gap between PDF and experimental values significant, but it still shows the correspondence between them, indicating that the algorithm is correct.

Problem 4

题目内容略

总服务时间见图P4.a，平均吞吐量见P4.b，平均接入时延见P4.c

可以看出在总服务时间随MTCD数量增加浮动较大，但总体趋势为MTCD数量越多，总服务时间越长。

在设备数量并不很多时，随着设备数量增多，平均吞吐量和和平均接入时延都较快增加，但到当设备数量

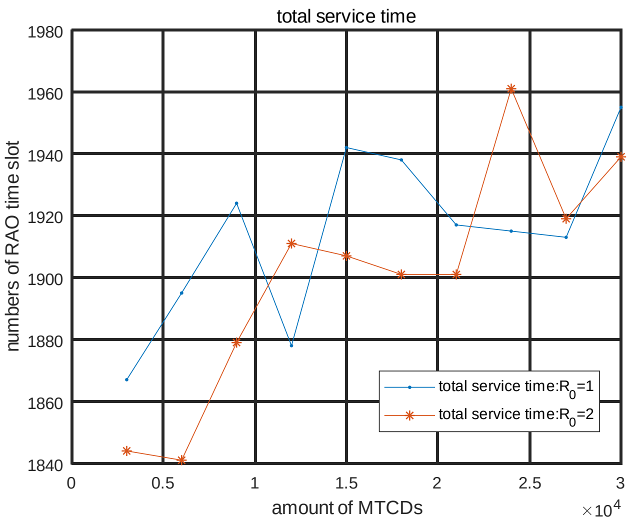


Figure P4.a

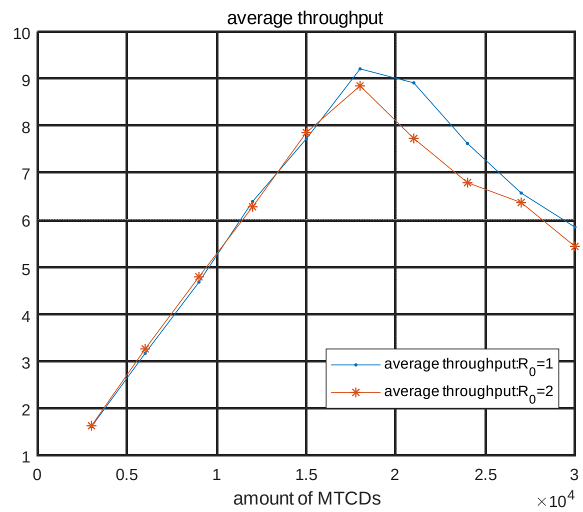


Figure P4.b

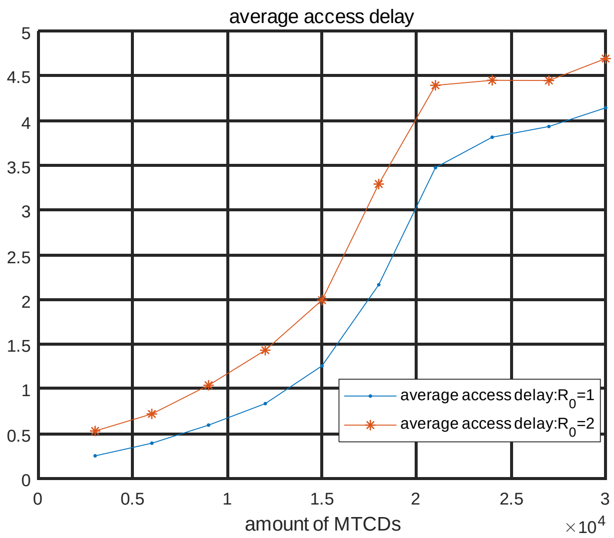


Figure P4.c

进一步增加时，平均吞吐量反而下降，而平均接入时延增速放缓。

对比不同最小传输速率约束R_0下的系统性能，发现其对总服务时间的影响规律性不强；而对于平均吞吐量，在设备数量较小时，R_0对平均吞吐量基本没有影响，而在设备数量较大时，$R_0 = 2$相较$R_0 = 1$时的吞吐量更小；R_0对于平均接入时延的影响的规律性最强，$R_0 = 2$时的接入时延显著高于$R_0 = 1$时的平均接入时延。

源码：

Problem1.m

```
% Problem1
clc,clear,clf

infty_n = 1e5;
approx_sample_n = 30;

m = 4:20;
T_b = 1;
f_s = m ./ T_b;

SNR_db_MSK = zeros(size(m));
SNR_db_rec = zeros(size(m));
for idx = 1:length(m)
    approx_points = 0: 1/approx_sample_n : f_s(idx)/2;
```

```
    E_signal_MSK = sum(G_MSK(approx_points,T_b));
    E_signal_rec = sum(G_rec(approx_points,T_b));

    approx_points = f_s(idx)/2 : 1/approx_sample_n : infty_n;
    E_noise_MSK = sum(G_MSK(approx_points,T_b));
    E_noise_rec = sum(G_rec(approx_points,T_b));

    SNR_db_MSK(idx) = 10 * log10(E_signal_MSK / E_noise_MSK);
    SNR_db_rec(idx) = 10 * log10(E_signal_rec / E_noise_rec);
end

plot(m,SNR_db_MSK,'o-','DisplayName','SNR_a of MSK signal');
hold on
plot(m,SNR_db_rec,'.-','DisplayName','SNR_a of rectangular pulse shape');
xlabel('number of samples in T_b');
ylabel('SNR_a');
legend
grid on

figure(2)
f = -4:0.02:4;
y = G_MSK(f,1);
subplot(1,2,1)
plot(f,y,'DisplayName','MSK signal');
title('MSK signal');
grid on

subplot(1,2,2);
y = G_rec(f,1);
plot(f,y);
title('rectangular');
grid on

function y = G_MSK(f,T_b)
    y = 16 .* T_b .* (cos(2 .* pi .* T_b .* f)).^2 ./ ( pi^2 .* (1-(4 .* T_b .* f).^2)).^2;
end

function y = G_rec(f,T_b)
    y = (sinc(f .* T_b)).^2;
end
```

Problem2.m

```
clc,clear,clf

n_sample = 1e4;
t = 0 : 1/20 : n_sample / 20 - 1e-8;
signal = sin(6 .* t);

bias = 1/16;
q = quantizer('fixed','Round','saturate',[4,3]);
signal_q = quantize(q,signal - bias) + bias;
error = signal_q - signal;

% [signal_q_custom, error_custom] = quantize_costum(signal,1,3,bias);

mean_error = mean(error)
mean_error_square = mean(error .^ 2)

hist(error,30)


% 下面这个函数只是检验一下自己对quantizer函数的理解。
% function [y,error] = quantize_costum(x,intLength,fracLength,bias)
%     error = zeros(size(x));
%     y = zeros(size(x));
```

```
%       step = 2^(-fracLength);
%       levels = -2^(intLength - 1) : step : 2^(intLength - 1) - 1e-8;
%       levels = levels + bias;
%       levels = levels(:);
%       % levels is a column vector
%       % suppose x is a row vector
%       x_rep =   repmat(x,length(levels),1);
%       err = x_rep -   levels;
%       [~,min_idx] = min(abs(err));
%       for idx = 1:length(x)
%             error(idx) = err(min_idx(idx),idx);
%             y(idx) = levels(min_idx(idx));
%       end
% end
```

Problem3.m

```
clc,clear,clf

num = 500;
a = 5;

figure(1)
interval_len = 0.1;
intervals = [0:interval_len:1.7];
U = rand([1,num]);
X = F_weibull_inverse(U,a);
h = histogram(X,intervals);
grid on

figure(2)
x = 0:0.02:1.6;
f = f_weibull(x,a);
plot(x,f,'r')
hold on
grid on
scatter(intervals(1:end-1) + interval_len /2 , h.Values ./ num ./ interval_len);

function f = f_weibull(x,a)
      f = a .* x.^(a-1) .* exp(-x.^a);
end

function X = F_weibull_inverse(U,a)
      X = (-(log(1-U))).^(1/a);
end
```

Problem4.m

```
clear,clc,clf


N_D_array = 3e3:3e3:3e4;
N_p = 54;
T = 10; % 10s
T_RAO = 5e-3; % 5ms, totally 2000 RAO time slot
beta_a = 3; % paramaters of Beta distribution
beta_b = 4;
J = 9;
target_SNR = 10;
R_0 = 1;
filename_prefix = 'P4_R_0_1_';
total_service_time = zeros(size(N_D_array));
```

```
average_throughput = zeros(size(N_D_array));
average_access_delay = zeros(size(N_D_array));

for N_D_idx = 1:length(N_D_array)
    N_D = N_D_array(N_D_idx);

    % get the first activation time
    activate_time = betarnd(beta_a,beta_b,1,N_D) * T;
    activate_time = floor(activate_time ./ T_RAO);
    first_activate_time = activate_time;

    time_slot_idx = min(activate_time);
    is_done = 0;
    J_local = ones(1,N_D) * J; % when J_local(idx) == -1, MTCD of idx won't access
    sending_preamble_time = zeros(1,N_D);
    accessed_time = zeros(1,N_D);
    while is_done ~= 1
        % activate MTCD according to activate_time
        activate_request_MTCD_idxs = find(activate_time == time_slot_idx);
        N_a = length(activate_request_MTCD_idxs);

        % ACB test
        ACB_factor = min(1, N_p / N_a);
        rand_theta = rand(1,N_a);
        bad_request_mask = rand_theta > ACB_factor;

        % bad request should request later
        delays = randi([1,4],size(activate_request_MTCD_idxs));
        delays = delays .* bad_request_mask;
        activate_time(activate_request_MTCD_idxs) = activate_time(activate_request_MTCD_idxs) + delays;

        % delete bad request ones and remains are activated successfully.
        activated_MTCD_idxs = find(activate_time == time_slot_idx);

        sending_preamble_time(activated_MTCD_idxs) = time_slot_idx;
        sending_preamble_MTCD_idxs = find(sending_preamble_time == time_slot_idx);

        % J validity check
        J_local(sending_preamble_MTCD_idxs) = J_local(sending_preamble_MTCD_idxs) - 1;
        sending_preamble_MTCD_idxs = J_validity_check(J_local,sending_preamble_MTCD_idxs);

        % activated MTCDs choose preamble code
        [succeed_to_send_MTCD_idxs, fail_to_send_MTCD_idxs] =
preamble_code_uniqueness_check(sending_preamble_MTCD_idxs,N_p);
        % those who fail to send unique preamble code should send codes later.
        delays = randi([1,4],size(fail_to_send_MTCD_idxs));
        sending_preamble_time(fail_to_send_MTCD_idxs) = time_slot_idx + delays;

        % those succeed to send unique preamble code will face the SNR check
        SNR_threshold = 2^R_0 - 1;
        [succeed_to_access_MTCD_idxs, fail_to_access_MTCD_idxs] =
SNR_threshold_check(succeed_to_send_MTCD_idxs,target_SNR,SNR_threshold);
        delays = randi([1,4],size(fail_to_access_MTCD_idxs));
        sending_preamble_time(fail_to_access_MTCD_idxs) = time_slot_idx + delays;

        accessed_time(succeed_to_access_MTCD_idxs) = time_slot_idx;
        accessed_number = sum(accessed_time ~= 0);
        if (accessed_number + sum(J_local < 0)) == N_D
            is_done = 1;
            % evalute
            total_service_time(N_D_idx) = time_slot_idx;
            average_throughput(N_D_idx) = accessed_number / time_slot_idx;
            access_delay = accessed_time - first_activate_time;
            access_delay(access_delay < 0) = []; % delete those failed
            average_access_delay(N_D_idx) = mean(access_delay);
        end
```

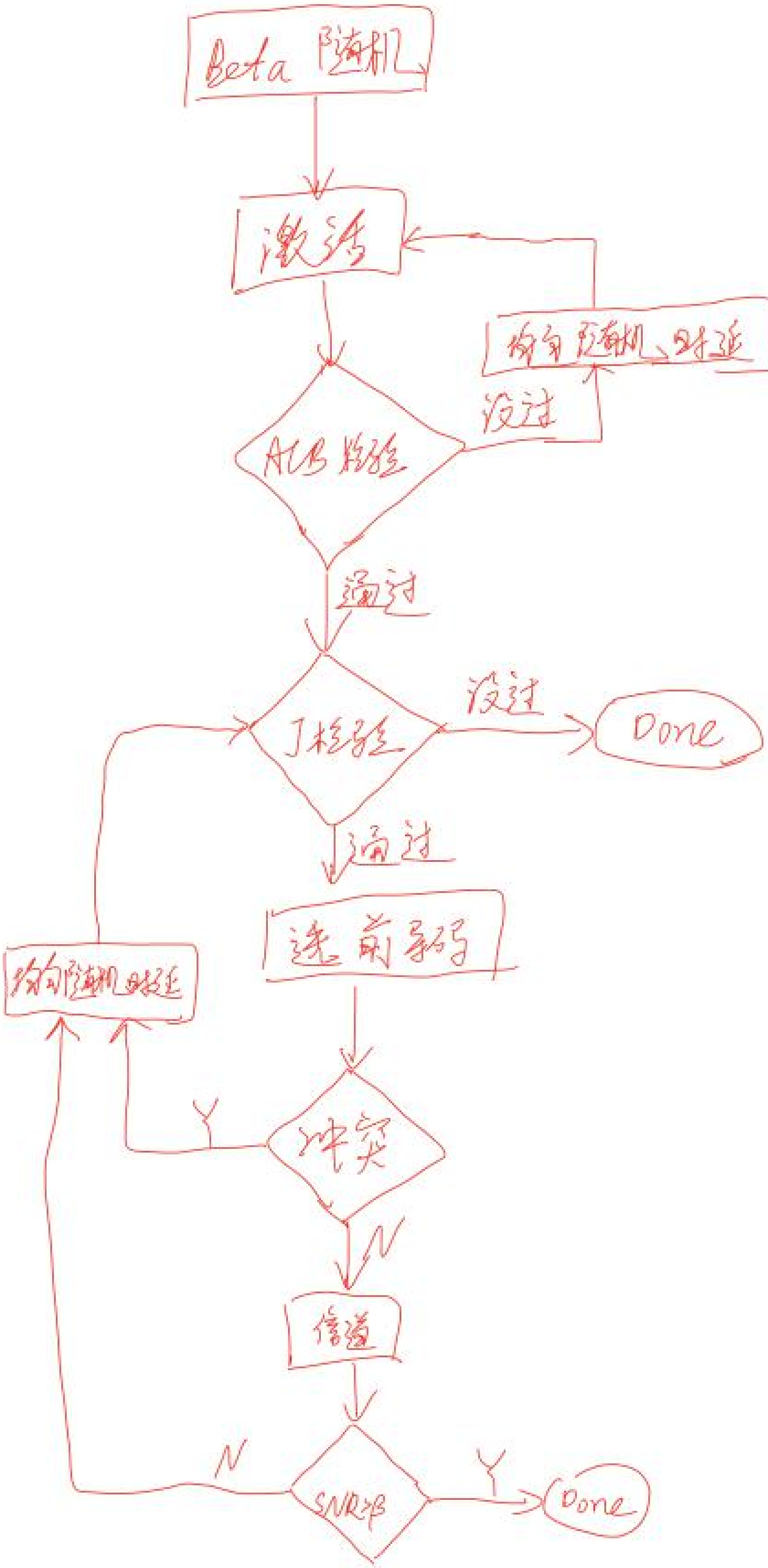
```
            time_slot_idx = time_slot_idx + 1;
        end
        N_D_idx
end

figure(1)
plot(N_D_array,total_service_time,'DisplayName','total service time:R_0=1');
beautify_figure();

figure(2)
plot(N_D_array,average_throughput,'DisplayName','average throughput:R_0=1');
beautify_figure();

figure(3)
plot(N_D_array,average_access_delay,'DisplayName','average access delay:R_0=1');
beautify_figure();

csvwrite([filename_prefix,'TST.csv'], total_service_time);
csvwrite([filename_prefix,'AT.csv'], average_throughput);
csvwrite([filename_prefix,'AAD.csv'], average_access_delay);

function beautify_figure()
        grid on
        legend
end

function [success_MTCD_idxs,failure_MTCD_idxs] = SNR_threshold_check(total_MTCD_idxs, target_SNR,
SNR_threshold)
        mu = 2;
        channel_gains = exprnd(mu,size(total_MTCD_idxs));
        channel_gains = channel_gains * target_SNR;
        success_MTCD_idxs = total_MTCD_idxs(channel_gains >= SNR_threshold);
        failure_MTCD_idxs = total_MTCD_idxs(channel_gains < SNR_threshold);
end

function [success_MTCD_idxs, failure_MTCD_idxs] =
preamble_code_uniqueness_check(total_MTCD_idxs,N_p)
        codes_chosen = randi([1,N_p],size(total_MTCD_idxs));
        [codes_unique,success_MTCD_idxs,~] = unique(codes_chosen);
        codes_chosen = codes_chosen';
        failure_mask = sum(codes_unique == codes_chosen) - 1; % it's a row vector
        success_MTCD_idxs = success_MTCD_idxs' .* (~failure_mask);
        success_MTCD_idxs(success_MTCD_idxs == 0) = [];
        success_MTCD_idxs = total_MTCD_idxs(success_MTCD_idxs);
        failure_MTCD_idxs = setdiff(total_MTCD_idxs,success_MTCD_idxs);
end

function idx = J_validity_check(J_local,pre_idx)
        J_exceed_mask = J_local(pre_idx) < 0;
        idx = pre_idx .* (~J_exceed_mask);
        idx(idx == 0) = [];
end
```

Problem4_plot.m

```
clear,clc

N_D_array = 3e3:3e3:3e4;
filename_prefix = 'P4_R_0_1_';
tst_1 = csvread([filename_prefix,'TST.csv']);
at_1 = csvread([filename_prefix,'AT.csv']);
aad_1 = csvread([filename_prefix,'AAD.csv']);

filename_prefix = 'P4_R_0_2_';
tst_2 = csvread([filename_prefix,'TST.csv']);
at_2 = csvread([filename_prefix,'AT.csv']);
```

```
aad_2 = csvread([filename_prefix,'AAD.csv']);

figure(1);clf;
plot(N_D_array,tst_1,'.-','DisplayName','total service time:R_0=1');
hold on
plot(N_D_array,tst_2,'*-','DisplayName','total service time:R_0=2');
title('total service time');
ylabel('numbers of RAO time slot');
beautify_figure();

figure(2);clf;
plot(N_D_array,at_1,'.-','DisplayName','average throughput:R_0=1');
hold on
plot(N_D_array,at_2,'*-','DisplayName','average throughput:R_0=2');
title('average throughput');
beautify_figure();

figure(3);clf;
plot(N_D_array,aad_1,'.-','DisplayName','average access delay:R_0=1');
hold on
plot(N_D_array,aad_2,'*-','DisplayName','average access delay:R_0=2');
title('average access delay');
beautify_figure();

function beautify_figure()
    xlabel('amount of MTCDs');
    grid on
    legend
end
```

Problem4 流程图：


Beta 随机
激活
均匀随机时延
没过
ACB 检验
通过
没过
Done
通过
选前导码
均匀随机时延
Y
冲突
N
信道
N
Y
Done