**Lab 7: LTE - Cell -Scanner**

|  |  |
| --- | --- |
| **Author** | Name： 吉辰卿 Student ID: 11911303 |
| **Introduction:**  In this experiment, we try to interpret a complete MATLAB program. The function of this program is to search the cell without calling the LTE toolbox in MATLAB. Therefore, this experiment we focus on the study of LTE cell search MATLAB programming ideas. In this experimental report, we focus on the analysis and interpretation of the SSS detection part of the program, and describe the process through a flow chart.  **Theoretical analysis:**   1. **LTE Cell Search**   The cell search process of LTE is mainly shown in the figure below:  20220501163500  Figure. The main steps of cell search in LTE system  The brief steps of cell search in LTE system are as follows:   1. Search for primary synchronization sequence and secondary synchronization sequence. 2. Do Channel estimation and time synchronization through reference signal (CRS). 3. Retrieve MIB information and obtain system frame number by PBCH channel. 4. SIB information retrieval: As data is transmitted on PDSCH, its location is informed by PDCCH, and the mode of PDCCH is determined by PCFICH. Therefore, SIB information retrieval includes decoding PCFICH, PDCCH and PDSCH information respectively.   **Lab results & Analysis：**   1. **Annotate the code sss\_detect.m**   function peak\_out = sss\_detect(peak,capbuf,thresh2\_n\_sigma,fc,sampling\_carrier\_twist,tdd\_flag, varargin)    % Perform maximum likelihood estimation of the SSS.  %初始化变量的赋值  peak\_loc=peak.ind; % 记录PSS xcorr峰的位置  peak\_freq=peak.freq; % 记录PSS xcorr峰对应的频率  n\_id\_2\_est=peak.n\_id\_2; % 取PSS xcorr峰的n\_id\_2信息作为n\_id\_2的起始估计值  %1.81GHz为载波的中心频率，若载波sampling\_carrier\_twist的值为1，则接下来会对k\_factor赋值（k\_factor是莱斯信道的信道k因子）  %反之则将k\_factor赋值为peak的k\_factor属性值;  % % fc\*k\_factor is the receiver's actual RX center frequency.  if sampling\_carrier\_twist==1  k\_factor=(fc-peak.freq)/fc;  % else  % k\_factor=1;  else  k\_factor = peak.k\_factor;  end  %确定我们采用的传输模式:时分复用或还是频分复用(TDD or FDD)  %即通过检测SSS，我们就可以知道小区是工作在FDD模式还是TDD模式  if tdd\_flag == 1 %TDD模式下的赋值  min\_idx = 3\*(128+32)+32;%假如系统工作在TDD模式下,设置盲检测的最小位置  sss\_ext\_offset = 3\*(128+32);  sss\_nrm\_offset = 412;  else %FDD模式下的赋值  min\_idx = 163-9;%假如系统工作在FDD模式下,设置盲检测的最小位置  sss\_ext\_offset = 128+32;  sss\_nrm\_offset = 128+9;  end  %TDD模式下寻找峰值  if (peak\_loc<min\_idx) % 对peakloc的值进行一定的改进，得到序列中初始位置的序号  peak\_loc=peak\_loc+9600\*k\_factor;  end    pss\_loc\_set=peak\_loc:9600\*k\_factor:length(capbuf)-125-9;  %在这里获得长度为16，间隔为9600的序列标识loc\_set  % pss\_loc\_set=peak\_loc + (0:9600:7\*9600);  % pss\_loc\_set=peak\_loc + (8\*9600:9600:15\*9600);  n\_pss=length(pss\_loc\_set); %pss\_loc\_set的总长度为16  pss\_np=NaN(1,n\_pss);  %创建16\*62空矩阵，并未赋值  h\_raw=NaN(n\_pss,62);  h\_sm=NaN(n\_pss,62);  %定义了普通循环前缀模式与拓展循环前缀模式(normal and extend)  sss\_nrm\_raw=NaN(n\_pss,62);  sss\_ext\_raw=NaN(n\_pss,62);    % % fo correction and ce by my method  % [~, td\_pss] = pss\_gen;  % tmp\_store = zeros(n\_pss, 128);  % pss\_local = td\_pss(:, n\_id\_2\_est+1);  % pss\_local = pss\_local(10:end);  % pss\_local\_fft = fft(pss\_local);  % for k=1:n\_pss  % pss\_loc=round(pss\_loc\_set(k));  % pss\_dft\_location=pss\_loc + 9;  % dft\_in=fshift(capbuf(pss\_dft\_location:pss\_dft\_location+127),-peak\_freq,fs\_lte/16);  % late = pss\_loc - pss\_loc\_set(k);  % fd\_data = fft(dft\_in);  % fd\_data = [fd\_data(65:end) fd\_data(1:64)];  % fd\_data = fd\_data.\*exp(1i.\*2.\*pi.\*late./128);  % fd\_data = [fd\_data(65:end) fd\_data(1:64)];  % fd\_data(2:32) = fd\_data(2:32)./(pss\_local\_fft(2:32).');  % fd\_data(98:end) = fd\_data(98:end)./(pss\_local\_fft(98:end).');  % % dft\_in = ifft(fd\_data);  % tmp\_store(k, :) = fd\_data;  % tmp\_store(k, 1) = 0;  % tmp\_store(k, 33:97) = 0;  % %  % % figure;  % % scatterplot(exp(1i.\* angle(dft\_in.\*(pss\_local'))) );  % end  % figure;  % subplot(2,1,1); plot(abs(tmp\_store).');  % subplot(2,1,2); plot(angle(tmp\_store).');  % return;  %在确定功能模式后，尝试检测SSS的确切位置。由于循环前缀的长度是未知的，所以我们会进行盲检测，首先扫描SSS的一些可能的位置。 %随后，UE将使用最大似然估计和找到SSS可能出现的位置。  for k=1:n\_pss % for循环的范围是1-16  pss\_loc=round(pss\_loc\_set(k));  %在这个for循环中判断每次循环所查找到的PSS序列的位置  %找到PSS进行傅里叶变换的位置  pss\_dft\_location=pss\_loc+9-2;  %if (pss\_dft\_location+127>length(capbuf))  % break;  %end  %计算信道相应h  %取出找到PSS进行傅里叶变换的位置后的128个数并进行移动  （也就是把信号下变频到基带信号）  % 移动方法:将频率为0的部分，将其插入到频率谱的中心位置  dft\_in=fshift(capbuf(pss\_dft\_location:pss\_dft\_location+127),-peak\_freq,fs\_lte/16);  % TOC  %将PSS的子帧移到末尾两位  dft\_in=[dft\_in(3:end) dft\_in(1:2)];  %通过DFT得到结果  dft\_out=dft(dft\_in);  %去除相应的循环前缀，并取出其中62个元素放置到h\_raw中第k行  h\_raw(k,:)=[dft\_out(end-30:end) dft\_out(2:32)];  %共轭相乘，计算每一个子载波信号的信道响应  h\_raw(k,:)=h\_raw(k,:).\*conj(pss(n\_id\_2\_est));  %plot(angle(h\_raw(k,:)));  %ylim([-pi pi]);  %drawnow;  %pause    % Smoothening... Basic...  for t=1:62  %在这个for循环中定义了lt和rt  %通过查阅资料这是滑动平均窗口的起始点与终止点  %arm\_length=min([6 t-1 62-t]);  lt=max([1 t-6]);  rt=min([62 t+6]);  % Growing matrix...  %h\_sm(k,t)=mean(h\_raw(k,t-arm\_length:t+arm\_length));  %计算hraw中从lt:rt总共7个元素的平均值，即平滑信道矩阵h  h\_sm(k,t)=mean(h\_raw(k,lt:rt));  end  %估计噪声功率，由刚刚计算出来的均值减去各项的初始值即可得到噪声  % Estimate noise power.  pss\_np(k)=sigpower(h\_sm(k,:)-h\_raw(k,:));    % 在频域计算SSS  %1.计算SSS的拓展循环前缀的位置  sss\_ext\_dft\_location=pss\_dft\_location-sss\_ext\_offset;  %这里和上面第85-92行的代码注释是一样的，主要目的就是为了个DFT变换而做准备  dft\_in=fshift(capbuf(sss\_ext\_dft\_location:sss\_ext\_dft\_location+127),-peak\_freq,fs\_lte/16);  % TOC  dft\_in=[dft\_in(3:end) dft\_in(1:2)];  %进行DFT变换得到结果  dft\_out=dft(dft\_in);  %去除相应的循环前缀，并取出其中62个元素放置到h\_raw中第k行  sss\_ext\_raw(k,1:62)=[dft\_out(end-30:end) dft\_out(2:32)];      % Calculate the SSS in the frequency domain (nrm)  %2.计算SSS的普通循环前缀的位置  sss\_nrm\_dft\_location=pss\_dft\_location-sss\_nrm\_offset;  dft\_in=fshift(capbuf(sss\_nrm\_dft\_location:sss\_nrm\_dft\_location+127),-peak\_freq,fs\_lte/16);  % TOC  dft\_in=[dft\_in(3:end) dft\_in(1:2)];  dft\_out=dft(dft\_in);  sss\_nrm\_raw(k,1:62)=[dft\_out(end-30:end) dft\_out(2:32)];  %从132-137行的操作和计算拓展循环前缀时使用的操作是一样的，这里就不再解释了  end    if nargin == 6  figure(4);  %绘制平滑前和平滑后信道矩阵的模长与辐角  subplot(2,2,1); pcolor(abs(h\_raw)); shading flat; drawnow;  subplot(2,2,2); pcolor(angle(h\_raw)); shading flat; drawnow;  subplot(2,2,3); pcolor(abs(h\_sm)); shading flat; drawnow;  subplot(2,2,4); pcolor(angle(h\_sm)); shading flat; drawnow;    figure(5);  %取信道矩阵的前三行，绘制平滑前和平滑后信道矩阵的模长与辐角  subplot(2,2,1); plot(abs(h\_raw(1:3,:).'));drawnow;  subplot(2,2,2); plot(angle(h\_raw(1:3,:).')); drawnow;  subplot(2,2,3); plot(abs(h\_sm(1:3,:).')); drawnow;  subplot(2,2,4); plot(angle(h\_sm(1:3,:).')); drawnow;  end    % % interpolation along time to get accurate response at sss.  % h\_sm\_ext\_interp = zeros(n\_pss, 62);  % h\_sm\_nrm\_interp = zeros(n\_pss, 62);  % for t=1:62  % h\_sm\_ext\_interp(:,t) = interp1(pss\_loc\_set, h\_sm(:,t), pss\_loc\_set-sss\_ext\_offset, 'linear','extrap');  % h\_sm\_nrm\_interp(:,t) = interp1(pss\_loc\_set, h\_sm(:,t), pss\_loc\_set-sss\_nrm\_offset, 'linear','extrap');  % end  %  % h\_raw\_ext\_interp = zeros(n\_pss, 62);  % h\_raw\_nrm\_interp = zeros(n\_pss, 62);  % for t=1:62  % h\_raw\_ext\_interp(:,t) = interp1(pss\_loc\_set, h\_raw(:,t), pss\_loc\_set-sss\_ext\_offset, 'linear','extrap');  % h\_raw\_nrm\_interp(:,t) = interp1(pss\_loc\_set, h\_raw(:,t), pss\_loc\_set-sss\_nrm\_offset, 'linear','extrap');  % end  %  % pss\_np\_ext=zeros(1,n\_pss);  % pss\_np\_nrm=zeros(1,n\_pss);  % for k=1:n\_pss  % pss\_np\_ext(k)=sigpower(h\_sm\_ext\_interp(k,:)-h\_raw\_ext\_interp(k,:));  % pss\_np\_nrm(k)=sigpower(h\_sm\_nrm\_interp(k,:)-h\_raw\_nrm\_interp(k,:));  % end    % ----recover original algorithm by using following 4 lines  %记录上述计算结果的值，直接复制一遍信道矩阵H和PSS的值?  h\_sm\_ext\_interp = h\_sm;  h\_sm\_nrm\_interp = h\_sm;  pss\_np\_ext = pss\_np;  pss\_np\_nrm = pss\_np;    % Combine results from different slots  %创建8个1\*62的空向量，分别代表不同的slot里面的普通循环前缀模式和拓展循环前缀模式,目的是为了分配内存。  sss\_h1\_nrm\_np\_est=NaN(1,62);  sss\_h2\_nrm\_np\_est=NaN(1,62);  sss\_h1\_ext\_np\_est=NaN(1,62);  sss\_h2\_ext\_np\_est=NaN(1,62);    sss\_h1\_nrm\_est=NaN(1,62);  sss\_h2\_nrm\_est=NaN(1,62);  sss\_h1\_ext\_est=NaN(1,62);  sss\_h2\_ext\_est=NaN(1,62);  for t=1:62  %利用for循环进行互相关计算  %普通循环前缀模式  sss\_h1\_nrm\_np\_est(t)=real((1+ctranspose(h\_sm\_nrm\_interp(1:2:end,t))\*diag(1./pss\_np\_nrm(1:2:end))\*h\_sm\_nrm\_interp(1:2:end,t))^-1);  sss\_h2\_nrm\_np\_est(t)=real((1+ctranspose(h\_sm\_nrm\_interp(2:2:end,t))\*diag(1./pss\_np\_nrm(2:2:end))\*h\_sm\_nrm\_interp(2:2:end,t))^-1);    %拓展循环前缀模式sss\_h1\_ext\_np\_est(t)=real((1+ctranspose(h\_sm\_ext\_interp(1:2:end,t))\*diag(1./pss\_np\_ext(1:2:end))\*h\_sm\_ext\_interp(1:2:end,t))^-1);  sss\_h2\_ext\_np\_est(t)=real((1+ctranspose(h\_sm\_ext\_interp(2:2:end,t))\*diag(1./pss\_np\_ext(2:2:end))\*h\_sm\_ext\_interp(2:2:end,t))^-1);    % 计算通过信道矩阵H后的信号S  sss\_h1\_nrm\_est(t)=sss\_h1\_nrm\_np\_est(t)\*ctranspose(h\_sm\_nrm\_interp(1:2:end,t))\*diag(1./pss\_np\_nrm(1:2:end))\*sss\_nrm\_raw(1:2:end,t);  sss\_h2\_nrm\_est(t)=sss\_h2\_nrm\_np\_est(t)\*ctranspose(h\_sm\_nrm\_interp(2:2:end,t))\*diag(1./pss\_np\_nrm(2:2:end))\*sss\_nrm\_raw(2:2:end,t);    sss\_h1\_ext\_est(t)=sss\_h1\_ext\_np\_est(t)\*ctranspose(h\_sm\_ext\_interp(1:2:end,t))\*diag(1./pss\_np\_ext(1:2:end))\*sss\_ext\_raw(1:2:end,t);  sss\_h2\_ext\_est(t)=sss\_h2\_ext\_np\_est(t)\*ctranspose(h\_sm\_ext\_interp(2:2:end,t))\*diag(1./pss\_np\_ext(2:2:end))\*sss\_ext\_raw(2:2:end,t);  end    % Maximum likelihood detection of SSS  %用最大似然估计算法检测SSS  %创建两个18=68\*2的空矩阵，以便进行后面的处理  log\_lik\_nrm=NaN(168,2);  log\_lik\_ext=NaN(168,2);  for t=0:167  %创建两个候选序列  sss\_h1\_try=sss(t,n\_id\_2\_est,0);  sss\_h2\_try=sss(t,n\_id\_2\_est,10);    %对普通循环前缀旋转候选序列以匹配接收序列。  %计算相位偏差量  ang=angle(sum(conj([sss\_h1\_nrm\_est sss\_h2\_nrm\_est]).\*[sss\_h1\_try sss\_h2\_try]));  sss\_h1\_try=sss\_h1\_try\*exp(j\*-ang);  sss\_h2\_try=sss\_h2\_try\*exp(j\*-ang); %纠正相位偏差  df=[sss\_h1\_try sss\_h2\_try]-[sss\_h1\_nrm\_est sss\_h2\_nrm\_est];  log\_lik\_nrm(t+1,1)=sum(-[real(df) imag(df)].^2./repmat([sss\_h1\_nrm\_np\_est sss\_h2\_nrm\_np\_est],1,2));    %使用最大似然法判定SSS  %将h1与h2交换(通过临时变量temp)，重复上述操作来进行旋转  %对普通循环前缀和拓展循环前缀是一样的代码块  temp=sss\_h1\_try;  sss\_h1\_try=sss\_h2\_try;  sss\_h2\_try=temp;  ang=angle(sum(conj([sss\_h1\_nrm\_est sss\_h2\_nrm\_est]).\*[sss\_h1\_try sss\_h2\_try]));  sss\_h1\_try=sss\_h1\_try\*exp(j\*-ang);  sss\_h2\_try=sss\_h2\_try\*exp(j\*-ang);  df=[sss\_h1\_try sss\_h2\_try]-[sss\_h1\_nrm\_est sss\_h2\_nrm\_est];  log\_lik\_nrm(t+1,2)=sum(-[real(df) imag(df)].^2./repmat([sss\_h1\_nrm\_np\_est sss\_h2\_nrm\_np\_est],1,2));    %对拓展循环前缀旋转候选序列以匹配接收序列。(重复和上面完全相同的操作，因此这里不再解释)  % Re-do for extended prefix  % Rotate the candiate sequence to match the received sequence.  temp=sss\_h1\_try;  sss\_h1\_try=sss\_h2\_try;  sss\_h2\_try=temp;  ang=angle(sum(conj([sss\_h1\_ext\_est sss\_h2\_ext\_est]).\*[sss\_h1\_try sss\_h2\_try]));  sss\_h1\_try=sss\_h1\_try\*exp(j\*-ang);  sss\_h2\_try=sss\_h2\_try\*exp(j\*-ang);  df=[sss\_h1\_try sss\_h2\_try]-[sss\_h1\_ext\_est sss\_h2\_ext\_est];  log\_lik\_ext(t+1,1)=sum(-[real(df) imag(df)].^2./repmat([sss\_h1\_ext\_np\_est sss\_h2\_ext\_np\_est],1,2));    % Exchange h1 and h2 and re-do  temp=sss\_h1\_try;  sss\_h1\_try=sss\_h2\_try;  sss\_h2\_try=temp;  ang=angle(sum(conj([sss\_h1\_ext\_est sss\_h2\_ext\_est]).\*[sss\_h1\_try sss\_h2\_try]));  sss\_h1\_try=sss\_h1\_try\*exp(j\*-ang);  sss\_h2\_try=sss\_h2\_try\*exp(j\*-ang);  df=[sss\_h1\_try sss\_h2\_try]-[sss\_h1\_ext\_est sss\_h2\_ext\_est];  log\_lik\_ext(t+1,2)=sum(-[real(df) imag(df)].^2./repmat([sss\_h1\_ext\_np\_est sss\_h2\_ext\_np\_est],1,2));  end    %判断所得SSS的循环前缀的类型，确定其模式是普通循环前缀还是拓展循环前缀  %warning('Check code here!!!!');  % cp\_type\_flag = 0;  if (max(log\_lik\_nrm(:))>max(log\_lik\_ext(:)))  %如果普通循环前缀的对数似然函数更大,则判定为normal CP SSS  cp\_type\_est='normal';  cp\_type\_flag = 0;  log\_lik=log\_lik\_nrm;  else  %如果拓展循环前缀的对数似然函数更大,则判定为normal CP SSS  cp\_type\_est='extended';  cp\_type\_flag = 1;  log\_lik=log\_lik\_ext;  end    % frame\_start是帧循环前缀的起始位置。  %帧的第一个DFT应该位于frame\_start+cp\_length  if tdd\_flag==1  if cp\_type\_flag == 0  frame\_start=peak\_loc+(-(2\*(128+9)+1)-1920-2)\*k\_factor; % TDD的普通循环前缀对应的起始位置  else  frame\_start=peak\_loc+(-(2\*(128+32))-1920-2)\*k\_factor; % TDD的拓展循环前缀对应的起始位置  end  else  frame\_start=peak\_loc+(128+9-960-2)\*k\_factor; % FDD循环前缀对应的起始位置  end  %这块代码也是在寻找frame\_start的起始位置  if (max(log\_lik(:,1))>max(log\_lik(:,2)))  ll=log\_lik(:,1);  else  frame\_start=frame\_start+9600\*k\_factor;  ll=log\_lik(:,2);  end  frame\_start=wrap(frame\_start,0.5,2\*9600+.5);  [lik\_final n\_id\_1\_est]=max(ll);  n\_id\_1\_est=n\_id\_1\_est-1; %获得n\_id\_1（SSS）的估计值    % 进行第二次阈值检查并更新输出的峰值peak\_out的相关信息:比如重要的n\_id\_1（SSS）的估计值  L=[log\_lik\_nrm log\_lik\_ext];  L\_mean=mean(L(:));%获得均值  L\_var=var(L(:));  if nargin == 6  figure(6);  plot(0:167,[log\_lik\_nrm log\_lik\_ext],[0 167],repmat(L\_mean,1,2),  [0 167],repmat(L\_mean+sqrt(L\_var)\*thresh2\_n\_sigma,1,2));%获得阈值  zgo;  drawnow;  end  peak\_out=peak;  %判定阈值  %如果我们的似然函数比阈值小，则失败  if (lik\_final<L\_mean+sqrt(L\_var)\*thresh2\_n\_sigma)  %disp('Thresh2 fail');  %更新输出的峰值peak\_out的相关信息  peak\_out.n\_id\_1=NaN;  peak\_out.cp\_type='';  peak\_out.cp\_type\_val=-1;  peak\_out.frame\_start=NaN;  peak\_out.duplex\_mode=NaN;  else  peak\_out.n\_id\_1=n\_id\_1\_est;%N\_id\_1的估计值  peak\_out.cp\_type=cp\_type\_est;% 得到了循环前缀的类型  peak\_out.cp\_type\_val=cp\_type\_flag;  peak\_out.frame\_start=frame\_start;% 得到帧起始时刻  peak\_out.duplex\_mode=tdd\_flag;% 判断结果是TDD 还是 FDD  end   1. **Explain the results, such as the sub figures**   **2.1 sub-figure 1**  1  The figure above shows the waveform after sampling from the original signal. The waveform is shown by the real and imaginary parts respectively.  **2.2 sub-figure 2**  2  The figure above shows the original waveform sample of PBCH by the real and imaginary parts respectively.  **2.3 sub-figure 3**  3  The left side of the figure above shows the columns corresponding to the sorted index in in capbuf pbch signal after the FFT. And the right side of the above figure shows the value of peak index.  **2.4 sub-figure 4**  **4**  The figure above shows the length and argument of the channel matrix before and after smoothing process in both time and frequency range.  **2.5 sub-figure 5**  **5**  The figure above shows the mode length and argument of the first three rows of the channel matrix before and after smoothing.  **2.6 sub-figure 6**  6  The figure above stands for the threshold detection to determine which autocorrelation is SSS, and it shows the comparison of the likelihood waveform, mean waveform, and the combination of the mean and variance waveform.   1. **Draw the flow-chart of sss\_detect.m**   wps  Figure. The flow-chart of sss\_detect.m | |
| **Experience**  In this experiment, through reading and understanding the specific program of LTE -cell search and the detection of SSS information, I have further understood how the specific program is realized. Of course, some codes may not be very clear about its specific role, but the program framework of the whole LTE retrieval program has basically formed in my mind.  Finally, I paste the screenshot of the class exercise in the lab class:  2  Figure. Class exercise 1  3  Figure. Class exercise 2  4  Figure. Class exercise 3 | |
| **Score** | 自评分数：99 |