第 6 章：LTE 小区搜索和 MIB 恢复

1、内容简介

（1）LTE 帧结构：FDD 和 TDD、帧结构的时频域分析，特别是频域的资源网格；

（2）LTE 信道：逻辑信道、传输信道和物理信道；各个信道的作用，需要知道 MIB 通过哪个信道传输，业务通过哪个信道传输；

（3）小区搜索的基本过程；

（4）MIB 解码的基本过程，以及业务数据解码过程；

（5）如何利用 USRP 进行发射 LTE 基带波形；

（6）如何利用 USRP 进行接收 LTE 基带波形；

（7）作业：从已经录制的波形中恢复图像；

2、4G-LTE 的发展历史

LTE 技术下行链路使用正交频分多址，即 OFDMA，它能够保持 3G 的高移动性以及高频谱效率并提供更高的网络容量以及更大的带宽；上行链路则采用单载波频分多址，即 SC-FDMA，具备和 OFDMA 相似的结构与性能，但能降低移动终端的功耗需要，延长电池续航时间。

2004 年，3GPP（第三代合作伙伴计划）开始进行 LTE 标准化工作，把 LTE 作为全球通用标准。

2005 年，世界各地的五大移动运营商纷纷表示，为了方便部署不同频段，有必要在同时需要 FDD 和 TDD 的情况下把不同规格之间的差异最小化。

2006 年，美国高通公司收购了 Flarion，后者于 2000 年开始移动 OFDM 的研发，并实现全球首个移动 OFDM 系统商用。

2008 年，3GPP 发布了 LTE 的第一个版本——3GPP Release 8，这项规格以单一全球标准支持 LTE 的 FDD 和 TDD，实现了碎片最小化，方便了不同频段的部署，使得绝大多数企业对 LTE 标准做出的贡献可同等用于 FDD 和 TDD 上，促进了全球生态系统的发展。而 LTE 有了 TDD 和 FDD 内在的紧密互操作之后，与 3G 实现了无缝互通。

2009 年，高通发布了全球首个完整的多模 3G/LTE 集成芯片组解决方案，做到了同时支持 FDD 和 TDD。而在那年的早些时候，全球的首个商用 LTE 网络由运营商 TeliaSonera 在北欧的瑞典和挪威开通。

2012 年，中国移动在香港推出了 LTE FDD/TDD 融合网络。

2013 年，韩国的运营商 SK 电信推出了首个 LTE-Advanced 商用服务。LTE-Advance（增强型 LTE）是 LTE 技术上发展的重要里程碑。LTE-Advanced 的第一步是**载波聚合**（CA），第二步是完成两个 10 MHz 载波的聚合。两个载波的聚合峰值数据速率达到 150 Mpbs（Cat 4）。LTE-Advanced 能够聚合最多 5 个载波（最高达 100 MHz），以提高所有用户的用户数据速率，并使得用于突发性应用程序（例如网页浏览）的网络容量增加了一倍。

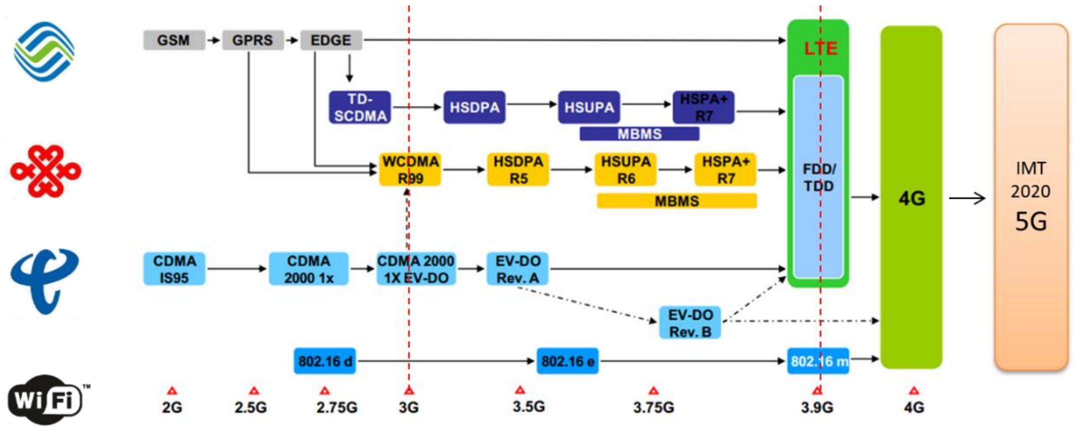
2017 年，预计全球的 3G/4G 连接有可能将达到 45 亿。

3、中国移动通信系统

中国移动的演进路线：GPRS->TD-SCDMA->HSPA->LTE;

2. 中国联通的演进路线：GPRS->WCDMA->HSPA->LTE;

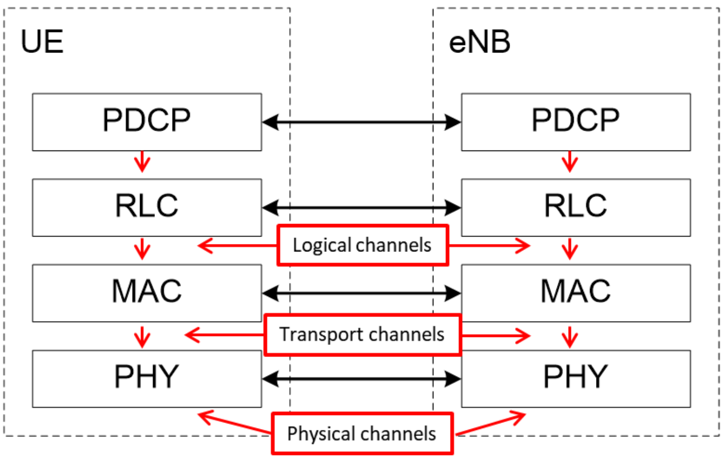
3. 中国电信的演进路线：CDMA->CDMA2000->EVDO->LTE;



具有划时代的两个系统，3G（移动的 TD-SCDMA，联通的 WCDMA，电信的 CDMA2000），4G-LTE（移动的 TDD，联通的 FDD）

4、LTE 的层次结构

物理层信道是学习 LTE 的关键，UE 和 eNodeB 层面上，分层结构如上图所示，每一层的功能是什么？通过 MATLAB 仿真和 USRP 实验，研究物理层的四个要点：（1）帧结构；（2）LTE 信道；（3）LTE 下行信道；（4）LTE 上行信道；重点了解前三个；



PDCP（Packet Data Convergence Protocol）层主要对来自控制面的 RRC

消息和来自数据面的 IP 包进行处理，其功能包括（见 36.323）：

- 头部压缩和解压缩：只使用 ROHC，并只用于用户面数据。
- 加密/解密（Ciphering and deciphering）：用于用户面和控制面数据。
- 完整性保护（Integrity Protection）：只用于控制面数据。
- 传输用户数据和控制面数据。
- 切换（handover）时的重排序和重传处理。
- 由于超时而丢弃用户面数据。

RLC（Radio Link Control）层主要负责（见 36.322）：

- 分段/串联和重组 RLC SDU（concatenation/segmentation/reassembly，只适用于 UM 和 AM 模式）：由于 RLC PDU 的大小是由 MAC 层指定的，其大小通常并不等于 RLC SDU 的大小，所以在发送端需要分段/串联 RLC SDU 以便其适合 MAC 层指定的大小。相应地，在接收端需要对之前分段的 RLC SDU 进行重组，以便恢复出原来的 RLC SDU 并发往上层。
- 通过 ARQ 来进行纠错（只适用于 AM 模式）：MAC 层的 HARQ 机制的目标在于实现非常快速的重传，其反馈出错率大概在 1%左右。对于某些业务，如 TCP 传输（要求丢包率小于 ），HARQ 反馈的出错率就显得过高了。对于这类业务，RLC 层的重传处理能够进一步降低反馈出错率。
- 对 RLC 数据 PDU 进行重排序（reordering，只适用于 UM 和 AM 模式）：MAC 层的 HARQ 可能导致到达 RLC 层的报文是乱序的，所以需要 RLC 层对数据进行重排序。
- 重复包检测（duplicate detection，只适用于 UM 和 AM 模式）：出现重复包的最大可能性为发送端反馈了 HARQ ACK，但接收端错误地将其解释为 NACK，

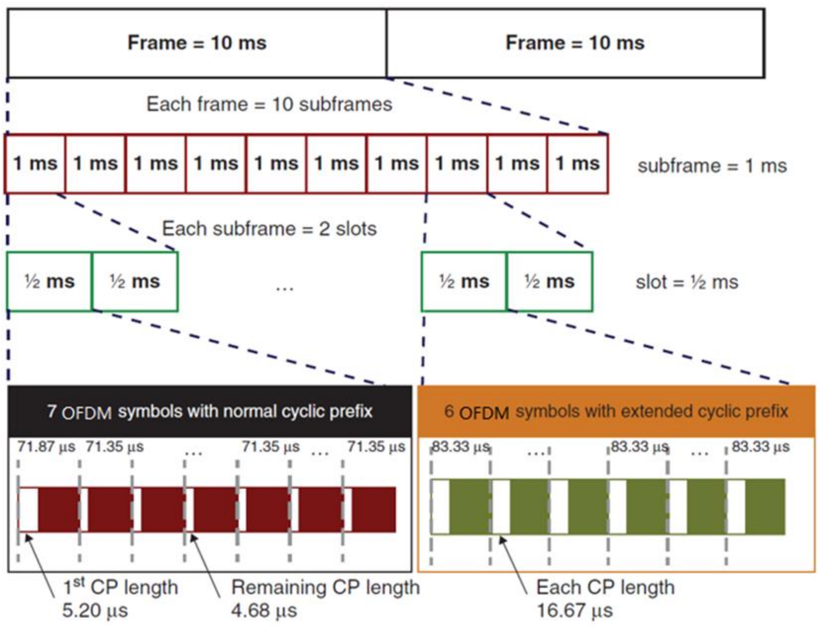
从而导致了不必要的 MAC PDU 重传。

- 对 RLC 数据 PDU 进行重分段（resegmentation，只适用于 AM 模式）：当 RLC 数据 PDU（注意：这里不是 SDU）需要重传时，也可能需要进行重分段。例如：MAC 层指定的大小比原始的 RLC 数据 PDU 还要小时，就需要进行重分段。

MAC 层主要负责（见 36.321）：

- 匹配逻辑信道和传输信道。
- 将属于一个或不同的逻辑信道（无线承载）的多个 MAC SDU 复用到同一个 MAC PDU（Transport Block）上，并发往物理层。反之为解复用。
- 通过 HARQ 来进行纠错。
- 调度处理。
- 逻辑信道优先级处理。
- 调度信息上报（只存在于 UE 侧和上行）。
- 随机接入过程处理。

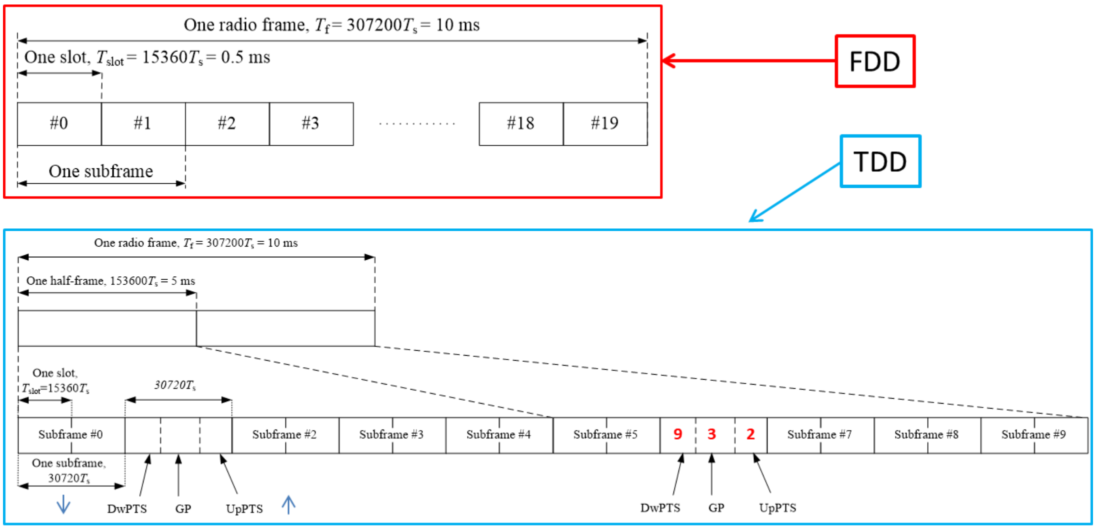
PHY 物理层



LTE 系统帧：一个 LTE 系统帧持续时间为 10ms，由 10 个连续的子帧构成，用系统帧号来区分系统帧；

LTE 子帧：每个子帧持续时间为 1ms，分为两个时隙（slot），每个时隙的时间是 0.5ms；

LTE 时隙：一个 slot 由普通循环前缀 7 个，扩展循环前缀 6 个构成；



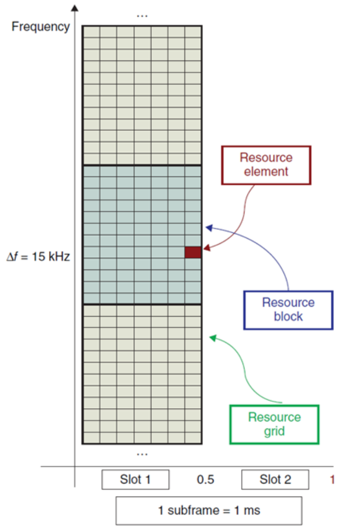
从时域上来看，FDD 和 TDD 的构造大体相同，它们系统帧的时长都是 10ms，都由 10 个子帧构成，但是在子帧时频资源的分配上有些不同，这些不同需要参考协议。

LTE 中最小时间单位 Ts 是多少呢？如何计算的呢？要理解这个问题，首先需要了解 LTE 中 OFDM 子载波间隔为 15KHz，那么，一个 OFDM 时长为 1/15KHz，一个 OFDM 符号由 2048 个采样来表示，因此，采样间隔等于 1/15KHz/2048=1/30720000，约等于 32.6ns。

TDD 中，DwPTS：下行导频时隙（D）；GP：保护间隔（S）；UpPTS 上行导频时隙（U）。

下行转上行需要特殊子帧。所有的用户向基站同步。

DwPTS+GP+UpPTS 总共两个时隙，14 个 OFDM 符号，在现行的 TDD 配置中，一般采用 9+3+2 模式。



A resource block: a group of resource elements corresponding to **12 subcarriers** or 180 kHz in the frequency domain and **one 0.5ms slot** in the time domain.

OFDM parameters for downlink transmission subframe duration (1 ms) subcarrier spacing (15 kHz)						
Bandwidth (MHz)	1.4	3	5	10	15	20
Sampling frequency (MHz)	1.92	3.84	7.68	15.36	23.04	30.72
FFT size	128	256	512	1024	1536	2048
Number of resource blocks	6	15	25	50	75	100
OFDM symbols per slot			14/12		(Normal/extended)	
CP length			4.7/5.6		(Normal/extended)	

从频域上看：20M 带宽，分为 2048 个子载波，但是 1200 个子载波是有用的。

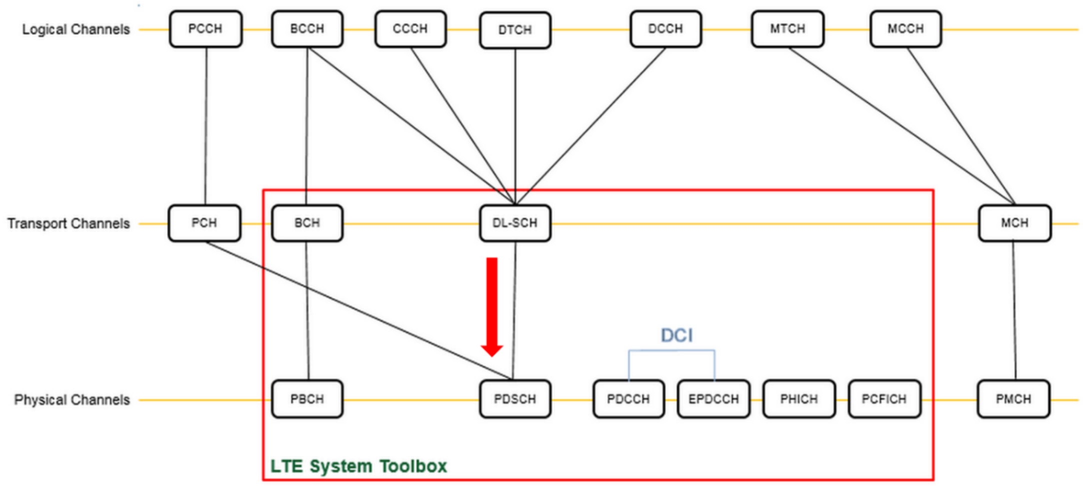
1 个资源块（RB）：由 12 个子载波和 1 个时隙构成，例如普通循环前缀，12X7=84 个资源块。

问题：FFT 大小为什么和资源块个数不对应？例如，资源块数目为 100 时，可以推测出子载波数目为 100X12=1200 个子载波，但是为什么采用 2048 点的 FFT 呢？这是因为 2048 最接近 1200，有 800 多个子载波未使用。

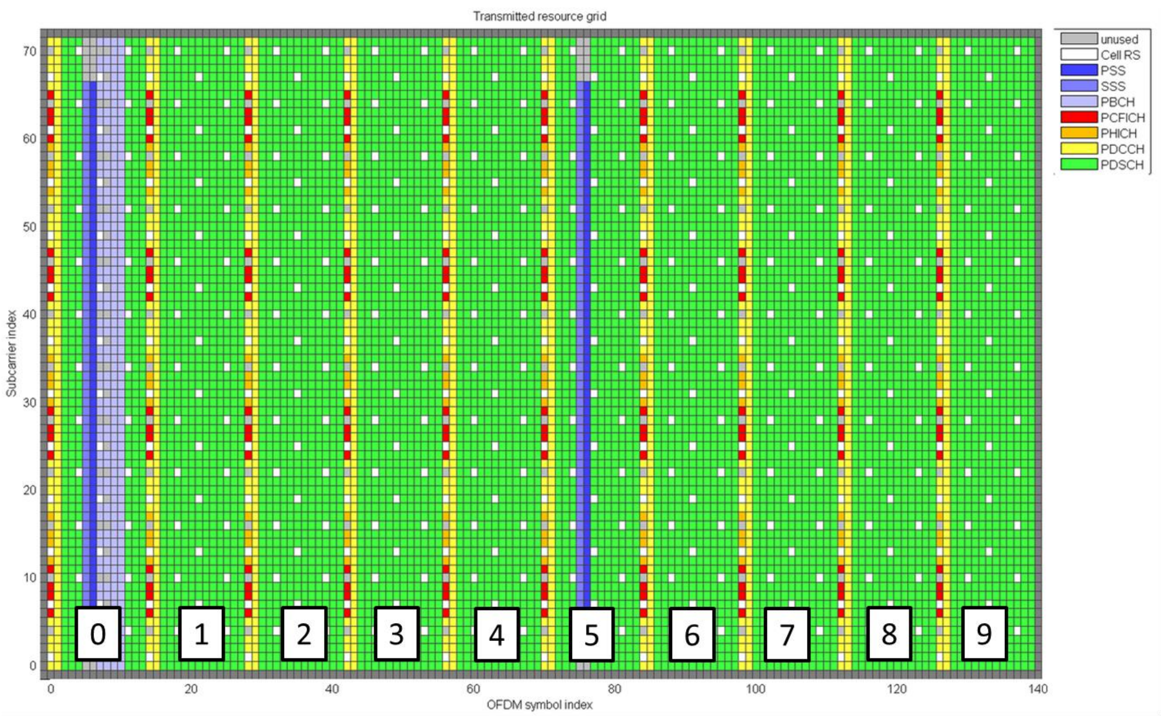
5、LTE 的信道

LTE 下行信道大体分为两种，一种是传输信令，一种是传输业务；传输信令

的通过 BCH->PBCH 信道传输，传输业务的通过 DL-SCH->PDSCH 信道传输。



在 LTE 系统工具箱中,可以调用的传输层函数有 BCH，DL-SCH；可以调用的物理层信道函数有：PBCH，PDSCH，DCI，PHICH 和 PCFICH，其中 DCI 用于格式控制，HI 反馈重传。本次实验的仿真的内容是通过 DL-SCH->PDSCH 传输一幅图像。



学习 LTE，一定要了解资源网格，知道不同的信息在哪个时候哪个地方发送。

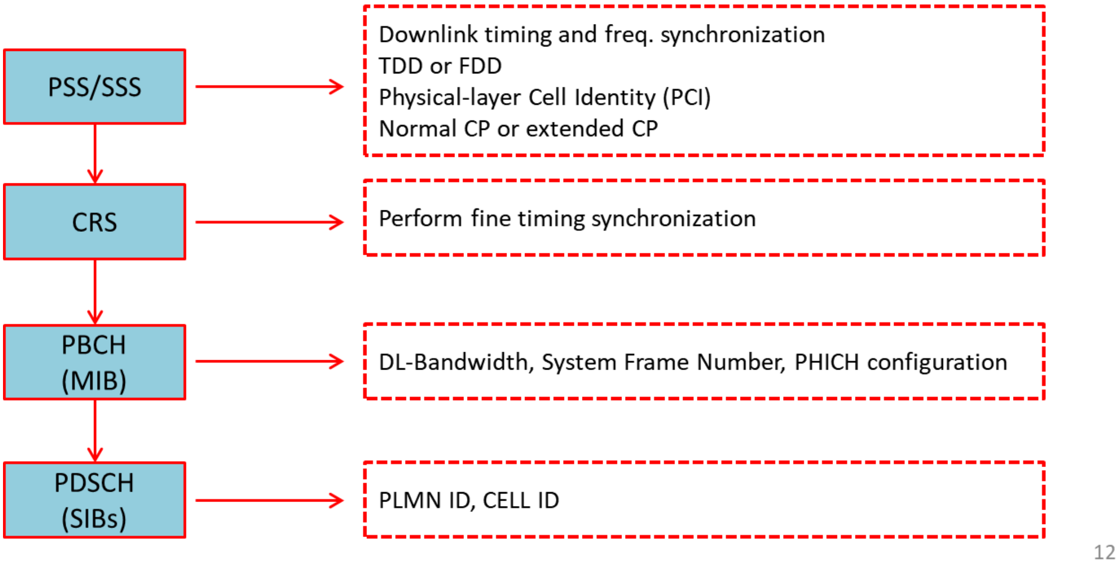
上图是一个系统帧的资源分配，可以看出；

（1）一个系统帧中各个信道的时频资源分布；

（2）主同步、辅同步信号的位置；（2）小区参考信号的位置；（3）广播信道的位置；（4）CFI 信息的位置；（5）HI 的位置；（6）PDCCH 位置；（7）PDSCH 位置；

（3）还需要通过实验了解，这些信道的基本功能，以及它们之间的逻辑关系。

6、LTE 小区搜索的过程



小区搜索的过程分为以下四个步骤：

（1）主同步序列和辅同步序列：位置（0 号子帧和 5 号子帧第一个时隙的最后两个 OFDM 上），功能（PSS 功能，ZC 序列恢复 NID2(0,1,2)，5ms 定时；SSS 功能，m 序列恢复 NID1(0-167),10ms 定时；判断小区是 TDD 还是 FDD，判断小区采用的是普通循环前缀还是扩展循环前缀）。

（2）参考信号：位置（均匀分布在 RB 中），功能（信道估计）

（3）PBCH 信道：位置（在 0 号子帧第二个时隙的前 4 个 OFDM 上）

（4）在资源网格中，还有 3 个信道(1)PCFICH；（2）PHICH；（3）DCI 信道；

（5）由于数据在 PDSCH 上传输，其位置由 CFI 告知

7、基于 LTE 的图像传输实验

发射机程序设计：

```
clc;  clear
```

%（1）设置频谱分析工具

```
hsa = dsp.SpectrumAnalyzer( ...
        'SpectrumType',       'Power density', ...
        'SpectralAverages', 10, ...
        'YLimits',             [-150 -60], ...
        'Title',               'Received Baseband LTE Signal Spectrum', ...
        'YLabel',              'Power spectral density');
```

%（2）设置星座图

```
hcd = comm.ConstellationDiagram('Title','Equalized PDSCH Symbols',...
                                'ShowReferenceConstellation',false);
%% Transmitter Design: System Architecture
```

%（3）DL-SCH 下行链路参数设置

```
txsim.RC = 'R.7';           % Base RMC configuration, 10 MHz bandwidth
txsim.NCellID = 88;         % Cell identity
txsim.NFrame = 700;         % Initial frame number
txsim.TotFrames = 1;        % Number of frames to generate
txsim.DesiredCenterFrequency = 2.45e9; % Center frequency in Hz
txsim.NTxAnts = 1;          % Number of transmit antennas
```

%（4）信源编码：图像->比特流

```
fileTx = 'peppers.png';              % Image file name
fData = imread(fileTx);              % Read image data from file
scale = 0.5;                         % Image scaling factor
origSize = size(fData);              % Original input image size
scaledSize = max(floor(scale.*origSize(1:2)),1); % Calculate new image size
heightIx = min(round(((1:scaledSize(1))-0.5)./scale+0.5),origSize(1));
```

```
widthIx = min(round(((1:scaledSize(2))-0.5)./scale+0.5),origSize(2));
fData = fData(heightIx,widthIx,:); % Resize image
imsize = size(fData);                    % Store new image size
binData = dec2bin(fData(:),8);       % Convert to 8 bit unsigned binary
trData = reshape((binData-'0').',1,[]).'; % Create binary stream
```

%（5）显示图像

```
figure(1);
 imFig.Visible = 'on';
subplot(211);
    imshow(fData);
    title('Transmitted Image');
subplot(212);
    title('Received image will appear here...');
    set(gca,'Visible','off'); % Hide axes
    set(findall(gca, 'type', 'text'), 'visible', 'on'); % Unhide title
pause(1); % Pause to plot Tx image

%%
```

%（6）生成 LTE 基带信号

```
rmc = lteRMCDL(txsim.RC);
```

%（7）计算需要多少个 LTE 帧

```
trBlkSize = rmc.PDSCH.TrBlkSizes;
txsim.TotFrames = ceil(numel(trData)/sum(trBlkSize(:)));
```

%（8）设置 RMC 参数

```
rmc.NCellID = txsim.NCellID;
rmc.NFrame = txsim.NFrame;
rmc.TotSubframes = txsim.TotFrames*10; % 10 subframes per frame
rmc.CellRefP = txsim.NTxAnts; % Configure number of cell reference ports
rmc.PDSCH.RVSeq = 0;

% Fill subframe 5 with dummy data
rmc.OCNGPDSCHEnable = 'On';
rmc.OCNGPDCCHEnable = 'On';

% If transmitting over two channels enable transmit diversity
% if rmc.CellRefP == 2
%     rmc.PDSCH.TxScheme = 'TxDiversity';
%     rmc.PDSCH.NLayers = 2;
%     rmc.OCNGPDSCH.TxScheme = 'TxDiversity';
% end

fprintf('\nGenerating LTE transmit waveform:\n')
```

```
fprintf('    Packing image data into %d frame(s).\n\n', txsim.TotFrames);
```

% （9）LTE 帧封装

```
[eNodeBOutput,txGrid,rmc] = lteRMCDLTool(rmc,trData);
```

%% 接收机设计程序

```
% User defined parameters --- configure the same as transmitter
rxsim = struct;
rxsim.RadioFrontEndSampleRate = rmc.SamplingRate; % Configure for same sample rate
                                                  % as transmitter
rxsim.RadioCenterFrequency = txsim.DesiredCenterFrequency;
rxsim.NRxAnts = txsim.NTxAnts;
rxsim.FramesPerBurst = txsim.TotFrames+1; % Number of LTE frames to capture in each burst.
                                          % Capture 1 more LTE frame than transmitted to
                                          % allow for timing offset wraparound...
rxsim.numBurstCaptures = 1; % Number of bursts to capture

% Derived parameters
samplesPerFrame = 10e-3*rxsim.RadioFrontEndSampleRate; % LTE frames period is 10 ms
```

%%（1）参数设置

```
rx.BasebandSampleRate = rxsim.RadioFrontEndSampleRate;
rx.CenterFrequency = rxsim.RadioCenterFrequency;
rx.SamplesPerFrame = samplesPerFrame;
rx.OutputDataType = 'double';
rx.EnableBurstMode = true;
rx.NumFramesInBurst = rxsim.FramesPerBurst;
rx.ChannelMapping = 1;

% burstCaptures holds rx.FramesPerBurst number of consecutive frames worth
% of baseband LTE samples. Each column holds one LTE frame worth of data.
burstCaptures = zeros(samplesPerFrame,rxsim.NRxAnts,rxsim.FramesPerBurst);

%%
% *LTE Receiver Setup*
enb.PDSCH = rmc.PDSCH;
enb.DuplexMode = 'FDD';
enb.CyclicPrefix = 'Normal';
enb.CellRefP = 4;

%%
% Bandwidth: {1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 20 MHz}
SampleRateLUT = [1.92 3.84 7.68 15.36 30.72]*1e6;
NDLRBLUT = [6 15 25 50 100];
```

```
enb.NDLRB = NDLRBLUT(SampleRateLUT==rxsim.RadioFrontEndSampleRate);
if isempty(enb.NDLRB)
    error('Sampling rate not supported. Supported rates are %s.',...
            '1.92 MHz, 3.84 MHz, 7.68 MHz, 15.36 MHz, 30.72 MHz');
end
fprintf('\nSDR hardware sampling rate configured to capture %d LTE RBs.\n',enb.NDLRB);

%%
% Channel estimation configuration structure
cec.PilotAverage = 'UserDefined';   % Type of pilot symbol averaging
cec.FreqWindow = 9;                 % Frequency window size in REs
cec.TimeWindow = 9;                 % Time window size in REs
cec.InterpType = 'Cubic';           % 2D interpolation type
cec.InterpWindow = 'Centered';      % Interpolation window type
cec.InterpWinSize = 3;              % Interpolation window size

% （2）信号捕获
enbDefault = enb;

while rxsim.numBurstCaptures
    % Set default LTE parameters
    enb = enbDefault;

%     % SDR Capture
%     fprintf('\nStarting a new RF capture.\n\n')
%     len = 0;
%     for frame = 1:rxsim.FramesPerBurst
%         while len == 0
%             % Store one LTE frame worth of samples
%             [data,len,lostSamples] = step(rx);
%             burstCaptures(:,:,frame) = data;
%         end
%         if lostSamples
%             warning('Dropped samples');
%         end
%         len = 0;
%     end
%     if rxsim.NRxAnts == 2
%         rxWaveform = reshape(permute(burstCaptures,[1 3 2]), ...
%                         rxsim.FramesPerBurst*samplesPerFrame,rxsim.NRxAnts);
%         hsa.ShowLegend = true; % Turn on legend for spectrum analyzer
%         hsa.ChannelNames = {'SDR Channel 1','SDR Channel 2'};
%     else
%     rxWaveform = burstCaptures(:);
```

```
        rxWaveform    = eNodeBOutput;
%       end
    % Show power spectral density of captured burst
    hsa.SampleRate = rxsim.RadioFrontEndSampleRate;
    step(hsa,rxWaveform);
```

（3）信号处理（重点阅读以下程序）

```
    %（1）首先进行频偏纠正
    frequencyOffset = lteFrequencyOffset(enb,rxWaveform);
    rxWaveform = lteFrequencyCorrect(enb,rxWaveform,frequencyOffset);
    fprintf('\nCorrected a frequency offset of %i Hz.\n',frequencyOffset)

    %（2）小区搜索 PCI
    cellSearch.SSSDetection = 'PostFFT'; cellSearch.MaxCellCount = 1;
    [NCellID,frameOffset] = lteCellSearch(enb,rxWaveform,cellSearch);
    fprintf('Detected a cell identity of %i.\n', NCellID);
    enb.NCellID = NCellID; % From lteCellSearch

    %（3）10ms 同步
    rxWaveform = rxWaveform(frameOffset+1:end,:);
    tailSamples = mod(length(rxWaveform),samplesPerFrame);
    rxWaveform = rxWaveform(1:end-tailSamples,:);
    enb.NSubframe = 0;
    fprintf('Corrected a timing offset of %i samples.\n',frameOffset)

    %（4）OFDM  解调
    rxGrid = lteOFDMDemodulate(enb,rxWaveform);

    % Perform channel estimation for 4 CellRefP as currently we do not
    % know the CellRefP for the eNodeB.
    [hest,nest] = lteDLChannelEstimate(enb,cec,rxGrid);

    sfDims = lteResourceGridSize(enb);
    Lsf = sfDims(2); % OFDM symbols per subframe
    LFrame = 10*Lsf; % OFDM symbols per frame
    numFullFrames = length(rxWaveform)/samplesPerFrame;

    rxDataFrame = zeros(sum(enb.PDSCH.TrBlkSizes(:)),numFullFrames);
    recFrames = zeros(numFullFrames,1);
    rxSymbols = []; txSymbols = [];

    %（5）MIB 解码, PDSCH and DL-SCH
    for frame = 0:(numFullFrames-1)
        fprintf('\nPerforming DL-SCH Decode for frame %i of %i in burst:\n', ...
            frame+1,numFullFrames)
```

```
% 抽取 0 号子帧和信道估计结果
enb.NSubframe = 0;
rxsf = rxGrid(:,frame*LFrame+(1:Lsf),:);
hestsf = hest(:,frame*LFrame+(1:Lsf),:,:);

% PBCH 解调  PBCH demodulation. Extract resource elements (REs)
% corresponding to the PBCH from the received grid and channel
% estimate grid for demodulation.
enb.CellRefP = 4;
pbchIndices = ltePBCHIndices(enb);
[pbchRx,pbchHest] = lteExtractResources(pbchIndices,rxsf,hestsf);
[~,~,nfmod4,mib,CellRefP] = ltePBCHDecode(enb,pbchRx,pbchHest,nest);

% If PBCH decoding successful CellRefP~=0 then update info
if ~CellRefP
    fprintf('    No PBCH detected for frame.\n');
    continue;
end
enb.CellRefP = CellRefP; % From ltePBCHDecode

% 有了正确的 enb 值，就可以进行 MIB 解码，
enb = lteMIB(mib,enb);

% Incorporate the nfmod4 value output from the function
% ltePBCHDecode, as the NFrame value established from the MIB
% is the system frame number modulo 4. 获得帧号
enb.NFrame = enb.NFrame+nfmod4;
fprintf('    Successful MIB Decode.\n')
fprintf('    Frame number: %d.\n',enb.NFrame);

% The eNodeB transmission bandwidth may be greater than the
% captured bandwidth, so limit the bandwidth for processing  （下行带宽）
enb.NDLRB = min(enbDefault.NDLRB,enb.NDLRB);

% Store received frame number
recFrames(frame+1) = enb.NFrame;

% Process subframes within frame (ignoring subframe 5)
for sf = 0:9
    if sf~=5 % Ignore subframe 5
        % Extract subframe
        enb.NSubframe = sf;
        rxsf = rxGrid(:,frame*LFrame+sf*Lsf+(1:Lsf),:);
```

```
% Perform channel estimation with the correct number of CellRefP
[hestsf,nestsf] = lteDLChannelEstimate(enb,cec,rxsf);

% PCFICH demodulation. Extract REs corresponding to the PCFICH
% from the received grid and channel estimate for demodulation.
pcfichIndices = ltePCFICHIndices(enb);
[pcfichRx,pcfichHest] = lteExtractResources(pcfichIndices,rxsf,hestsf);
[cfiBits,recsym] = ltePCFICHDecode(enb,pcfichRx,pcfichHest,nestsf);

% CFI 解码
enb.CFI = lteCFIDecode(cfiBits);

% Get PDSCH indices
[pdschIndices,pdschIndicesInfo] = ltePDSCHIndices(enb, enb.PDSCH,
enb.PDSCH.PRBSet);
[pdschRx, pdschHest] = lteExtractResources(pdschIndices, rxsf, hestsf);

% PDSCH 解码 Perform deprecoding, layer demapping, demodulation and
% descrambling on the received data using the estimate of
% the channel
[rxEncodedBits, rxEncodedSymb] =
ltePDSCHDecode(enb,enb.PDSCH,pdschRx,...
                              pdschHest,nestsf);

% Append decoded symbol to stream
rxSymbols = [rxSymbols; rxEncodedSymb{:}]; %#ok<AGROW>

% Transport block sizes
outLen = enb.PDSCH.TrBlkSizes(enb.NSubframe+1);

% Decode DownLink Shared Channel (DL-SCH)
[decbits{sf+1}, blkcrc(sf+1)] = lteDLSCHDecode(enb,enb.PDSCH,...
                              outLen,
rxEncodedBits);   %#ok<SAGROW>

% Recode transmitted PDSCH symbols for EVM calculation
%    Encode transmitted DLSCH
txRecode = lteDLSCH(enb,enb.PDSCH,pdschIndicesInfo.G,decbits{sf+1});
%    Modulate transmitted PDSCH
txRemod = ltePDSCH(enb, enb.PDSCH, txRecode);
%    Decode transmitted PDSCH
[~,refSymbols] = ltePDSCHDecode(enb, enb.PDSCH, txRemod);
%    Add encoded symbol to stream
```

```
                txSymbols = [txSymbols; refSymbols{:}]; %#ok<AGROW>

                release(hcd); % Release previous constellation plot
                step(hcd,rxEncodedSymb{:}); % Plot current constellation
            end
        end

        % Reassemble decoded bits
        fprintf('    Retrieving decoded transport block data.\n');
        rxdata = [];
        for i = 1:length(decbits)
            if i~=6 % Ignore subframe 5
                rxdata = [rxdata; decbits{i}{:}]; %#ok<AGROW>
            end
        end

        % Store data from receive frame
        rxDataFrame(:,frame+1) = rxdata;

        % Plot channel estimate between CellRefP 0 and the receive antennae
        focalFrameIdx = frame*LFrame+(1:LFrame);
%         figure(hhest);
%         hhest.Visible = 'On';
%         surf(abs(hest(:,focalFrameIdx,1,1)));
%           shading flat;
%           xlabel('OFDM symbol index');
%           ylabel('Subcarrier index');
%           zlabel('Magnitude');
%           title('Estimate of Channel Magnitude Frequency Repsonse');
    end
    rxsim.numBurstCaptures = rxsim.numBurstCaptures-1;
end
% Release both transmit and receive objects once reception is complete

%%
%（6）恢复结果质量统计
% Determine index of first transmitted frame (lowest received frame number)
[~,frameIdx] = min(recFrames);

fprintf('\nRecombining received data blocks:\n');

decodedRxDataStream = zeros(length(rxDataFrame(:)),1);
frameLen = size(rxDataFrame,1);
% Recombine received data blocks (in correct order) into continuous stream
```

```
for n=1:numFullFrames
    currFrame = mod(frameIdx-1,numFullFrames)+1; % Get current frame index
    decodedRxDataStream((n-1)*frameLen+1:n*frameLen) = rxDataFrame(:,currFrame);
    frameIdx = frameIdx+1; % Increment frame index
end
% Perform EVM calculation
if ~isempty(rxSymbols)
    hEVM = comm.EVM();
    hEVM.MaximumEVMOutputPort = true;
    [evm.RMS,evm.Peak] = step(hEVM,txSymbols, rxSymbols);
    fprintf('    EVM peak = %0.3f%%\n',evm.Peak);
    fprintf('    EVM RMS     = %0.3f%%\n',evm.RMS);
else
    fprintf('    No transport blocks decoded.\n');
end

% Perform bit error rate (BER) calculation
hBER = comm.ErrorRate;
err = step(hBER, decodedRxDataStream(1:length(trData)), trData);
fprintf('    Bit Error Rate (BER) = %0.5f.\n', err(1));
fprintf('    Number of bit errors = %d.\n', err(2));
fprintf('    Number of transmitted bits = %d.\n',length(trData));

% Recreate image from received data
fprintf('\nConstructing image from received data.\n');
str = reshape(sprintf('%d',decodedRxDataStream(1:length(trData))), 8, []).';
decdata = uint8(bin2dec(str));
receivedImage = reshape(decdata,imsize);

%（7）显示接收的图像
figure(1);
subplot(212);
imshow(receivedImage);
title(sprintf('Received Image: %dx%d Antenna Configuration',txsim.NTxAnts, rxsim.NRxAnts));
```

8、基于 USRP 的图像恢复

```
%%	导入 IQ 数据
load eNodeBOutput.mat              % Load I/Q capture of eNodeB output
eNodeBOutput = double(eNodeBOutput)/32768; % Scale samples
sr = 15.36e6;                      % Sampling rate for loaded samples

%% 1. 显示接收信号频谱（eNodeBOutput 一共 107520 个数据,持续时间为 0.007s，7ms）
spectrumAnalyzer = dsp.SpectrumAnalyzer();
spectrumAnalyzer.Name = 'Received signal spectrum';
fprintf('\nPlotting received signal spectrum...\n');
step(spectrumAnalyzer, eNodeBOutput);

%% 2. 显示 PSS/SSS 相关波形，自相关和卷积只有一点区别是，其中一个序列不需要反褶
synchCorrPlot = dsp.ArrayPlot();
synchCorrPlot.Name = 'PSS/SSS correlation';

%% 3. 显示 PDCCH 信道 OFDM 解调后符号的星座图
pdcchConstDiagram = comm.ConstellationDiagram();
pdcchConstDiagram.Name = 'PDCCH constellation';

%% 4. 统计 EVM
pdschEVM = comm.EVM();

%% 6. eNodeB 对象初始化，设置资源块（RB）6 个，也就意味着 72 个子载波
%	设采用普通循环前缀，1 个 RB 包含 12 个子载波（子载波间隔为 15KHz）和 1 个时隙（0.5ms，7 个 OFDM 符号）
enb = struct;                      % eNodeB config structure
enb.NDLRB = 6;                     % Number of resource blocks
ofdmInfo = lteOFDMInfo(setfield(enb,'CyclicPrefix','Normal')); %#ok<SFLD>

%% 7. 下采样信号，使用 resample 函数，将信号从 15.36MS/s->1.92Ms，采样后信号长度为 13440。
%	由于小区信息分布在距离 DC 最近的 6 个 RB 上，占用带宽 15KHz*12*6=1.08MHz，所以采用基本 1.92MS/s 可以恢复。
nSamples = ceil(ofdmInfo.SamplingRate/round(sr)*size(eNodeBOutput,1));
nRxAnts = 1;
downsampled = zeros(nSamples, nRxAnts);
downsampled(:,1) = resample(eNodeBOutput(:,1), ofdmInfo.SamplingRate, round(sr));

%% 8. 小区搜索：盲检，FDD 和 TDD；Normal 和 Extended 去匹配；
%	PSS 和 SSS 盲检可以得出小区 PCI（enb.NCellID）：17，反过来计算 N1 和 N2
```

```
fprintf('\nPerforming cell search...\n');
duplexModes = {'TDD' 'FDD'};
cyclicPrefixes = {'Normal' 'Extended'};

searchalg.MaxCellCount = 1;
searchalg.SSSDetection = 'PostFFT';
peakMax = -Inf;

for duplexMode = duplexModes
    for cyclicPrefix = cyclicPrefixes
        enb.DuplexMode = duplexMode{1};
        enb.CyclicPrefix = cyclicPrefix{1};
        [enb.NCellID, offset, peak] = lteCellSearch(enb, downsampled, searchalg);
        enb.NCellID = enb.NCellID(1);
        offset = offset(1);
        peak = peak(1);
        if (peak>peakMax)
            enbMax = enb;
            offsetMax = offset;
            peakMax = peak;
        end
    end
end

% Use the cell identity, cyclic prefix length, duplex mode and timing
% offset which gave the maximum correlation during cell search
enb = enbMax;
offset = offsetMax;

corr = cell(1,3);
idGroup = floor(enbMax.NCellID/3);
for i = 0:2
    enb.NCellID = idGroup*3 + mod(enbMax.NCellID + i,3);
    [~,corr{i+1}] = lteDLFrameOffset(enb, downsampled);
    corr{i+1} = sum(corr{i+1},2);
end
threshold = 1.3 * max([corr{2}; corr{3}]); % multiplier of 1.3 empirically obtained
if (max(corr{1})<threshold)
    warning('Synchronization signal correlation was weak; detected cell identity may be
incorrect.');
end
% Return to originally detected cell identity
enb.NCellID = enbMax.NCellID;
```

```
% Plot PSS/SSS correlation and threshold
synchCorrPlot.YLimits = [0 max([corr{1}; threshold])*1.1];
step(synchCorrPlot, [corr{1}]);

%% 9 符号同步 Perform timing synchronisation
fprintf('Timing offset to frame start: %d samples\n',offset);
downsampled = downsampled(1+offset:end,:);
enb.NSubframe = 0;

% Show cell-wide settings
fprintf('Cell-wide settings after cell search:\n');
disp(enb);

%% 10 系统信息频偏纠正 Frequency offset estimation and correction
fprintf('\nPerforming frequency offset estimation...\n');
delta_f = lteFrequencyOffset(enb, downsampled);
fprintf('Frequency offset: %0.3fHz\n',delta_f);
downsampled = lteFrequencyCorrect(enb, downsampled, delta_f);

%% 11. 系统信息信道估计 OFDM Demodulation and Channel Estimation
% Channel estimator configuration
cec.PilotAverage = 'UserDefined';         % Type of pilot averaging
cec.FreqWindow = 9;                       % Frequency window size
cec.TimeWindow = 9;                       % Time window size
cec.InterpType = 'cubic';                 % 2D interpolation type
cec.InterpWindow = 'Centered';            % Interpolation window type
cec.InterpWinSize = 1;                    % Interpolation window size

% Assume 4 cell-specific reference signals for initial decoding attempt;
% ensures channel estimates are available for all cell-specific reference signals
enb.CellRefP = 4;

%OFDM 解调
fprintf('Performing OFDM demodulation...\n\n');
griddims = lteResourceGridSize(enb); % Resource grid dimensions
%一个子帧中有 14 个 OFDM 符号；假设 6 个子帧
L = griddims(2);                          % Number of OFDM symbols in a subframe
%rxgrid 的每一列表示一个 OFDM 解调结果 OFDM demodulate signal
rxgrid = lteOFDMDemodulate(enb, downsampled);

% 取第一个子帧做信道估计
[hest, nest] = lteDLChannelEstimate(enb, cec, rxgrid(:,1:L,:));

%% 12 PBCH 信道解码 PBCH Demodulation, BCH Decoding, MIB parsing
```

```
fprintf('Performing MIB decoding...\n');
pbchIndices = ltePBCHIndices(enb);
[pbchRx, pbchHest] = lteExtractResources(pbchIndices, rxgrid(:,1:L,:), hest(:,1:L,:,:));

% pbchSymbols 查看星座图，Decode PBCH
[bchBits, pbchSymbols, nfmod4, mib, enb.CellRefP] = ltePBCHDecode(enb, pbchRx, pbchHest, nest);

% Parse MIB bits
enb = lteMIB(mib, enb);

enb.NFrame = enb.NFrame+nfmod4;

% Display cell wide settings after MIB decoding
fprintf('Cell-wide settings after MIB decoding:\n');
disp(enb);


%% 13. SIB1 解码

fprintf('Restarting reception now that bandwidth (NDLRB=%d) is known...\n',enb.NDLRB);

%% （1） 重采样
ofdmInfo = lteOFDMInfo(enb);

fprintf('\nResampling not required; received signal is at desired sampling rate for NDLRB=%d (%0.3fMs/s).\n',enb.NDLRB,sr/1e6);

nSamples = ceil(ofdmInfo.SamplingRate/round(sr)*size(eNodeBOutput,1));
resampled = zeros(nSamples, nRxAnts);

resampled(:,1) = resample(eNodeBOutput(:,1), ofdmInfo.SamplingRate, round(sr));

%% （2） 频偏估计和纠正
fprintf('\nPerforming frequency offset estimation...\n');
delta_f = lteFrequencyOffset(enb, resampled);

fprintf('Frequency offset: %0.3fHz\n',delta_f);
resampled = lteFrequencyCorrect(enb, resampled, delta_f);

%% （3） 找到帧的起始位置
fprintf('\nPerforming timing offset estimation...\n');
offset = lteDLFrameOffset(enb, resampled);
fprintf('Timing offset to frame start: %d samples\n',offset);
```

```
% Aligning signal with the start of the frame
resampled = resampled(1+offset:end,:);

%% （4） OFDM 解调
fprintf('\nPerforming OFDM demodulation...\n\n');
rxgrid = lteOFDMDemodulate(enb, resampled);

%% （5） SIB1 解码
if (mod(enb.NFrame,2)~=0)
    if (size(rxgrid,2)>=(L*10))
        rxgrid(:,1:(L*10),:) = [];
        fprintf('Skipping frame %d (odd frame number does not contain SIB1).\n\n',enb.NFrame);
    else
        rxgrid = [];
    end
    enb.NFrame = enb.NFrame + 1;
end

% Advance to subframe 5, or terminate if we have less than 5 subframes
if (size(rxgrid,2)>=(L*5))
    rxgrid(:,1:(L*5),:) = [];    % Remove subframes 0 to 4
else
    rxgrid = [];
end
enb.NSubframe = 5;

if (isempty(rxgrid))
    fprintf('Received signal does not contain a subframe carrying SIB1.\n\n');
end

% Reset the HARQ buffers
decState = [];

% While we have more data left, attempt to decode SIB1
while (size(rxgrid,2) > 0)

    fprintf('SIB1 decoding for frame %d\n',mod(enb.NFrame,1024));

    % Reset the HARQ buffer with each new set of 8 frames as the SIB1
    % info may be different
    % 重新设置 HARQ 缓存
    if (mod(enb.NFrame,8)==0)
        fprintf('Resetting HARQ buffers.\n\n');
```

```
    decState = [];
end

% 抽取当前子帧
rxsubframe = rxgrid(:,1:L,:);

% 信道估计
[hest,nest] = lteDLChannelEstimate(enb, cec, rxsubframe);

% PCFICH 和 CFI 解调
fprintf('Decoding CFI...\n\n');
pcfichIndices = ltePCFICHIndices(enb);   % Get PCFICH indices
[pcfichRx, pcfichHest] = lteExtractResources(pcfichIndices, rxsubframe, hest);

% PCFICH 解调
cfiBits = ltePCFICHDecode(enb, pcfichRx, pcfichHest, nest);
cfi = lteCFIDecode(cfiBits); % Get CFI
if (isfield(enb,'CFI') && cfi~=enb.CFI)
    release(pdcchConstDiagram);
end
enb.CFI = cfi;
fprintf('Decoded CFI value: %d\n\n', enb.CFI);

tddConfigs = 0; % not used for FDD, only used to control while loop

dci = {};

while (isempty(dci) && ~isempty(tddConfigs))

    tddConfigs(1) = [];

    % PDCCH 解调
    pdcchIndices = ltePDCCHIndices(enb); % Get PDCCH indices
    [pdcchRx, pdcchHest] = lteExtractResources(pdcchIndices, rxsubframe, hest);

    % PDCCH 解码，星座图
    [dciBits, pdcchSymbols] = ltePDCCHDecode(enb, pdcchRx, pdcchHest, nest);
    step(pdcchConstDiagram, pdcchSymbols);

    % PDCCH blind search for System Information (SI) and DCI decoding.
    % The LTE System Toolbox provides full blind search of the PDCCH to
    % find any DCI messages with a specified RNTI, in this case the
    % SI-RNTI.
    fprintf('PDCCH search for SI-RNTI...\n\n');
```

```
    pdcch = struct('RNTI', 65535);
    pdcch.ControlChannelType = 'PDCCH';
    pdcch.EnableCarrierIndication = 'Off';
    pdcch.SearchSpace = 'Common';
    pdcch.EnableMultipleCSIRequest = 'Off';
    pdcch.EnableSRSRequest = 'Off';
    pdcch.NTxAnts = 1;
    dci = ltePDCCHSearch(enb, pdcch, dciBits); % Search PDCCH for DCI
end

% If DCI was decoded, proceed with decoding PDSCH / DL-SCH
if ~isempty(dci)

    dci = dci{1};
    fprintf('DCI message with SI-RNTI:\n');
    disp(dci);
    % Get the PDSCH configuration from the DCI
    [pdsch, trblklen] = hPDSCHConfiguration(enb, dci, pdcch.RNTI);
    pdsch.NTurboDecIts = 5;
    fprintf('PDSCH settings after DCI decoding:\n');
    disp(pdsch);

    % PDSCH demodulation and DL-SCH decoding to recover SIB bits.
    % The DCI message is now parsed to give the configuration of the
    % corresponding PDSCH carrying SIB1, the PDSCH is demodulated and
    % finally the received bits are DL-SCH decoded to yield the SIB1
    % bits.

    fprintf('Decoding SIB1...\n\n');
    % Get PDSCH indices
    [pdschIndices,pdschIndicesInfo] = ltePDSCHIndices(enb, pdsch, pdsch.PRBSet);
    [pdschRx, pdschHest] = lteExtractResources(pdschIndices, rxsubframe, hest);
    % Decode PDSCH
    [dlschBits,pdschSymbols] = ltePDSCHDecode(enb, pdsch, pdschRx, pdschHest, nest);
    % Decode DL-SCH with soft buffer input/output for HARQ combining
    if ~isempty(decState)
        fprintf('Recombining with previous transmission.\n\n');
    end
    [sib1, crc, decState] = lteDLSCHDecode(enb, pdsch, trblklen, dlschBits, decState);

    % Compute PDSCH EVM
    recoded = lteDLSCH(enb, pdsch, pdschIndicesInfo.G, sib1);
    remod = ltePDSCH(enb, pdsch, recoded);
    [~,refSymbols] = ltePDSCHDecode(enb, pdsch, remod);
```

```
%            [rmsevm,peakevm] = step(pdschEVM, refSymbols{1}, pdschSymbols{1});
%            fprintf('PDSCH RMS EVM: %0.3f%%\n',rmsevm);
%            fprintf('PDSCH Peak EVM: %0.3f%%\n\n',peakevm);

            fprintf('SIB1 CRC: %d\n',crc);
            if crc == 0
                fprintf('Successful SIB1 recovery.\n\n');
            else
                fprintf('SIB1 decoding failed.\n\n');
            end

        else
            % Indicate that DCI decoding failed
            fprintf('DCI decoding failed.\n\n');
        end

        % Update channel estimate plot
%        figure(channelFigure);
        surf(abs(hest(:,:,1,1)));
%        hSIB1RecoveryExamplePlots(channelFigure);
%        channelFigure.CurrentAxes.XLim = [0 size(hest,2)+1];
%        channelFigure.CurrentAxes.YLim = [0 size(hest,1)+1];

        % Skip 2 frames and try SIB1 decoding again, or terminate if we
        % have less than 2 frames left.
        if (size(rxgrid,2)>=(L*20))
            rxgrid(:,1:(L*20),:) = [];   % Remove 2 more frames
        else
            rxgrid = []; % Less than 2 frames left
        end
        enb.NFrame = enb.NFrame+2;

end
```