Modeling a 4G-LTE System in MATLAB

Houman Zarrinkoub PhD.
Product Manager
Signal Processing & Communications
MathWorks
Agenda

- 4G LTE and LTE Advanced standards
- MATLAB and communications system design
- Case study: A 4G LTE system model in MATLAB
  - Modeling & simulation
  - Simulation acceleration
  - Path to implementation
- Summary
4G LTE and LTE Advanced
Distinguishing Features

- **Motivation**
  - Very high capacity & throughput
  - Support for video streaming, web browsing, VoIP, mobile apps

- **A true global standard**
  - Contributions from all across globe
  - Deployed in AMER, EMEA, APLA

- **A true broadband mobile standard**
  - From 2 Mbps (UMTS)
  - To 100 Mbps (LTE)
  - To 1 Gbps (LTE Advanced)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Low Mobility</th>
<th>High Mobility</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE</td>
<td>~250 kbps</td>
<td></td>
</tr>
<tr>
<td>WCDMA / UMTS</td>
<td>2 Mbps</td>
<td>384 kbps</td>
</tr>
<tr>
<td>HSDPA</td>
<td>14 Mbps</td>
<td></td>
</tr>
<tr>
<td>HSPA+</td>
<td>42 Mbps</td>
<td></td>
</tr>
<tr>
<td>LTE (R8 or R9)</td>
<td>100 Mbps</td>
<td></td>
</tr>
<tr>
<td><strong>4G Requirement</strong></td>
<td>1 Gbps</td>
<td>100 Mbps</td>
</tr>
<tr>
<td>LTE Advanced (*)</td>
<td>&gt; 1 Gbps</td>
<td>&gt; 100 Mbps</td>
</tr>
</tbody>
</table>
How is this remarkable advance possible?

- Integration of enabling technologies with sophisticated mathematical algorithms
  - OFDM (Multi-carrier transmission, large bandwidth requires attention to multipath fading)
  - MIMO (multiple antennas)
  - Turbo Coding (near Shannon limit, e.g., near capacity channel coding)

- Smart usage of resources and bandwidth (smart engineering techniques)
  - Adaptive modulation
  - Adaptive coding
  - Adaptive MIMO
  - Adaptive bandwidth
What MATLAB users care about LTE standard?

- Academics
  - Researchers contributing to future standards
  - Professors
  - Students

- Practitioners
  - System Engineers
  - Software designers
  - Implementers of wireless systems

- Challenge in interaction and cooperation between these two groups
- MATLAB is their common language
Challenges:
From specification to implementation

- Simplify translation from specification to a model as blue-print for implementation
- Introduce innovative proprietary algorithms
- Dynamic system-level performance evaluation
- Accelerate simulation of large data sets
- Address gaps in the implementation workflow
Where does MATLAB fit in addressing these challenges?

- MATLAB and Communications System Toolbox are ideal for LTE algorithm and system design
- MATLAB and Simulink provide an environment for dynamic & large scale simulations
- Accelerate simulation with a variety of options in MATLAB
- Connect system design to implementation with
  - C and HDL code generation
  - Hardware-in-the-loop verification
Case Study: Downlink physical layer of LTE (Release 10)
Modeling and simulation
LTE Physical layer model in standard

- Downlink Shared Channel
  - Transport channel
  - Physical channel

Reference: 3GPP TS 36 211 v10 (2010-12)
LTE Physical layer model in MATLAB

Adaptation of everything

Turbo Channel Coding

MIMO

OFDMA

Open LTE system model
Overview of Turbo Coding

- Error correction & coding technology of LTE standard
- Performance: Approach the channel capacity (Shannon bound)
- Represents an evolution of convolutional coding
- Based on an iterative decoding scheme
MATLAB Demo

Modeling and Simulation of a transceiver system, showcasing:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding
- Turbo Encoding & Decoding

```matlab
function [ber bits]=fcn_04PSK_Viterbi(EbNo,MaxNumErrs,MaxNumBits)

%% Initialization
% Components
persistent hModulator hAWGN hDeModulator hBitError
persistent hConvEncoder hViterbi
if isempty(hModulator)
hModulator = comm.PSKModulator(...
    'ModulationOrder',4,...
    'PhaseOffset',0,...
    'BitInput',true);
hAWGN = comm.AWGNChannel;
hDeModulator = comm.PSKDemodulator(...
    'DecisionMethod','Hard decision');
hBitError = comm.ErrorRate;
end

= comm.ConvolutionalEncoder(...
    'DecisionMethod','Hard decision');
= comm.ViterbiDecoder(...
    'InputFormat', 'Hard',...
MATLAB Demo

Modeling and Simulation of a transceiver system, showing Coding Gain:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding

Convolutional Coding + Viterbi Decoding

BER vs. $E_b/N_0$ (dB)

function [ber bits] = fcn_04PSK_Viterbi_Soft(EBNo, MaxNumErrs, MaxNumBits)

%% Initialization
persistent hModulator hAWGN hDemodulator hBitError
persistent hConvEncoder hViterbi hQuantizer
if isempty(hModulator)
    hModulator = comm.PSKModulator(...
        'ModulationOrder', 4, ...)
        'PhaseOffset', 0, ...)
        'BitInput', true);
    hAWGN = comm.AWGNChannel(...
    hDemodulator = comm.QPSKDemodulator(...
        'DecisionMethod', 'Log-Likelihood ratio'
    hQuantizer = dsp.ScalarQuantizerEncoder(...
        'Partitioning', 'Unbounded', ...)
        'BoundaryPoints', QuantizerBoundarys,...
        'InputWordLength', 4,...
    hConvEncoder = comm.ConvolutionalEncoder(...
        'InputFormat', 'Soft', ...)
        'SoftInputWordLength', 4,...
        'OutputDataType', 'double', ...
        'TerminationMethod', 'Terminate');

end
MATLAB Demo

Modeling and Simulation of a transceiver system, showcasing key concepts:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding
- Turbo Encoding & Decoding

Turbo Coding

```
matlab
% MATLAB code snippet
function [ber_bits]=fcn_04PSK_zTurbo_Soft(EbNo,MaxNumErrs,MaxNumBits)
% Initialization
persistent hModulator hAWGN hDeModulator hBitError
persistent hTurboEncoder hTurboDecoder
if isempty(hModulator)
hModulator = comm.QPSKModulator('PhaseOffset',0,'BitInput',true);
end
hAWGN = comm.AWGNChannel(...)
%NoiseMethod: Viscous

hTurboEncoder = comm.TurboEncoder(...
'TrellisStructure','Trellis'

hTurboDecoder = comm.TurboDecoder(...
'NumIterations',6,...
'DecisionMethod','Log-Likelihood ratio'

end
```

Plot of BER vs. E_b/N_0 (dB) for Turbo Coding.
OFDM Overview

- **Orthogonal Frequency Division Multiplexing**
  - Multicarrier modulation scheme (FFT-based)

- Sample the spectrum at uniform intervals called sub-carriers
  - Transmit data independently at each sub-carrier

- Most important feature
  - Robust against multi-path fading
  - Using low-complexity frequency-domain equalizers
OFDM & Multi-path Fading

- Multi-path propagation leads to frequency selective fading
- Frequency-domain equalization is less complex and perfectly matches OFDM
- We need to know channel response at each sub-carrier – We need pilots

\[
y(n) = \sum_{n=0}^{N} h_n x(n - d_n)
\]

\[
Y(\omega) = H(\omega)X(\omega)
\]

If \( G(\omega_k) \approx H^{-1}(\omega_k) \)
\( G(\omega_k) Y(\omega_k) \approx X(\omega_k) \)
How Does LTE Implement OFDM?

- Frequency
  - Nmax = 2048

- Resource block
  - Resource element
  - Resource grid

- Time (msec)
  - Δf = 15 kHz
  - 1 sub-frame = 1 ms

- Pilots
  - Interpolate vertically (Frequency)
  - Interpolate horizontally (Time)
How to Implement LTE OFDM in MATLAB

switch prmLTEPDSCH.Nrb

Case 25, N=512;

Depending on Channel Bandwidth

Set Frequency-domain FFT size

Case 100, N=2048;

= 6*numRb + mod(v+vsh, 6) + 1;

Transmitter:
Place pilots in regular intervals

= mean([hp(:,1,1,n) hp(:,3,1,n)])

Receiver:
Estimate channel by interpolating in time & frequency

= mean([hp(k,2,1,:) hp(k,4,1,:)]);

X=ifft(tmp,N,1);
MIMO Overview

- Multiple Input Multiple Output
- Using multiple transmit and receive antennas

\[
Y = H \times X + n
\]
Where is MIMO being used?

- Several wireless standards
  - 802.11n: MIMO extensions of WiFi as of 2008
  - 802.16e: As of 2005 in WiMax Standard
  - 3G Cellular: 3GPP Release 6 specifies transmit diversity mode
  - 4G LTE

- Two main types of MIMO
  - Spatial multiplexing
  - Space-Time Block Coding (STBC)
Spatial Multiplexing

- MIMO technique used in LTE standard

- Divide the data stream into independent sub-streams and use multiple transmit antennas

- MIMO is one of the main reasons for boost in data rates
  - More transmit antennas leads to higher capacity

- MIMO Receiver essentially solves this system of linear equations

\[ Y = HX + n \]
MIMO-OFDM overview

\[ Y = H \ast X + n \]

What if 2 rows are linearly dependent?

\[ H = \begin{bmatrix}
    h_1 & h_2 & h_3 & h_4 \\
    h_1 & h_2 & h_3 & h_4 \\
    h_{31} & h_{32} & h_{33} & h_{34} \\
    h_{41} & h_{42} & h_{43} & h_{44}
\end{bmatrix} \]

Dimension = 4; Rank = 3; H = singular (not invertible)

\[ H = UDV^H \]

Singular Value Decomposition

To avoid singularity:
1. Precode input with pre-selected V
2. Transmit over antennas based on Rank
Adaptive MIMO: Closed-loop Pre-coding and Layer Mapping

Layer Mapping

Precoding

OFDM

Base station

Channel Rank Estimation

Precoder Matrix Estimation

mobile

Rank Indicator

Precoder Matrix Indicator
Adaptive MIMO in MATLAB

- In Receiver:
  - Detect $V = \text{Rank of the H Matrix}$
  - $V =$ Number of layers

- In Transmitter: (next frame)
  - Based on number of layers
  - Fill up transmit antennas with available rank

```matlab
V = prmLTEPDSCH.numLayers;

switch V
  case 4
    out = complex(zeros(inLen1/2, v));
    out(:,1:2) = reshape(in1, 2, inLen1/2).';
    out(:,3:4) = reshape(in2, 2, inLen2/2).';
```

Link Adaptation Overview

- Examples of link adaptations
  - Adaptive modulation
    - QPSK, 16QAM, 64QAM
  - Adaptive coding
    - Coding rates from (1/13) to (12/13)
  - Adaptive MIMO
    - 2x1, 2x2, ..., 4x2, ..., 4x4, 8x8
  - Adaptive bandwidth
    - Up to 100 MHz (LTE-A)
LTE Physical layer model in MATLAB

Turbo Channel Coding

MIMO

OFDMA

Adaptation of everything
Simulation acceleration options in MATLAB

- System Objects
- Parallel Computing
- MATLAB to C
- GPU processing
GPU Processing with Communications System Toolbox

- Alternative implementation for many System objects take advantage of GPU processing
- Use Parallel Computing Toolbox to execute many communications algorithms directly on the GPU
- Easy-to-use syntax
- Dramatically accelerate simulations

<table>
<thead>
<tr>
<th>GPU System objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>comm.gpu.TurboDecoder</td>
</tr>
<tr>
<td>comm.gpu.ViterbiDecoder</td>
</tr>
<tr>
<td>comm.gpu.LDPCDecoder</td>
</tr>
<tr>
<td>comm.gpu.ViterbiDecoder</td>
</tr>
<tr>
<td>comm.gpu.PSKDemodulator</td>
</tr>
<tr>
<td>comm.gpu.AWGNChannel</td>
</tr>
</tbody>
</table>
Example: Turbo Coding

- Impressive coding gain
- High computational complexity
- Bit-error rate performance as a function of number of iterations

```matlab
function [ber bits]=Fcn4_04PSK_Turbo_Soft(EbNo,MaxNumErrs,MaxNumBits)
    %Initialization
    FRM=65536;
    Trellis=poly2trellis(5, [37 21], 37);
    Indices=randperm(FRM);
    M=4;K=log2(M);
    R= FRM
    snr = EbNo/Nois
    % Com
    persistent hModulator hAWGN hDeModulator hBitError
    persistent hTurboEncoder hTurboDecoder
    if isempty(hModulator)
        hModulator = comm.QUASKModulator('PhaseOffset',0,'BitInput',t
```
Simulation acceleration with GPU System objects

<table>
<thead>
<tr>
<th>Version</th>
<th>Elapsed time</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>8 hours</td>
<td>1.0</td>
</tr>
<tr>
<td>1 GPU</td>
<td>40 minutes</td>
<td>12.0</td>
</tr>
<tr>
<td>Cluster of 4 GPUs</td>
<td>11 minutes</td>
<td>43.0</td>
</tr>
</tbody>
</table>

- Same numerical results

```python
= comm.TurboDecoder(...
    'NumlIterations', N,...
)

= comm.gpu.TurboDecoder(...
    'NumlIterations', N,...
)
Path to implementation
C/C++ and HDL Code Generation
Path to implementation

**Design**
- Algorithm Development
- Fixed-Point

**Implementation**
- C, C++
- VHDL, Verilog
- MCU, DSP, FPGA, ASIC

**Verification**
- Mathematical modeling and algorithm development
- Model elaboration including fixed-point numerical representation
- Automatic code generation for rapid on-target prototyping of Hardware/Software
- Co-simulation for verification in Hardware / Software

**Integration**
Automatic Translation of MATLAB to C

With MATLAB Coder, design engineers can

- Maintain one design in MATLAB
- Design faster and get to C/C++ quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB
HDL Workflow

- **Floating Point Model**
  - Satisfies System Requirements
    - Executable Specification
  - MATLAB and/or Simulink Model

- **Model Elaboration**
  - Develop Hardware Friendly Architecture
  - Convert to Fixed-Point
    - Determine Word Length
    - Determine Binary Point Location

- **Implement Design**
  - Generate HDL code using HDL Coder
  - Import Custom and Vendor IP

- **Verification**
  - Software co-simulation with HDL simulator
  - Hardware co-simulation
Key Points

- MATLAB is an ideal language for LTE modeling and simulation
- Communications System Toolbox extends MATLAB capabilities with algorithms for communications system design
- You can accelerate simulation with a variety of options in MATLAB
  - Parallel computing, GPU processing, MATLAB to C
- Address implementation workflow gaps with
  - Automatic MATLAB to C/C++ and HDL code generation
  - Hardware-in-the-loop verification