MATLAB Example 2: Reproducing Fig. 13 in IEEE TCOM "Efficient PHY Layer Abstraction for Fast Simulations in Complex System Environments"

Sian Jin (sianjin@uw.edu)

Prerequisites: MATLAB 2020b (or later version), MATLAB WLAN toolbox in MATLAB 2020b (or later version)

Goal: Learn how to implement and use the EESM-log-SGN mixture model shown in Algorithm 3 of our IEEE TCOM paper.

Folders:

MU code basic/1 Full PHY (no interferer)
MU code basic/2 EESM parameter optimization
MU code basic/3 Basic EESM-log-SGN method
MU code basic/4 EESM-log-SGN mixture

Step 1 (Full PHY simulation):

Goal: full PHY simulation shown in Fig. 3 of our IEEE TCOM paper.

Open: MU code basic/1 Full PHY (no interferer)/fullPHY.m

PHY layer configuration:

- OFDMA allocation with 106 subcarriers
- 8 × {2, 2} MU-MIMO with 2 streams/user
- TGax channel model-D
- Payload length = 1000 Byte
- MCS4, LDPC coding

In this example, we set the 11ax allocation index to be 97.

Allocation Index	20 MHz Subchannel Resource Unit (RU) Assignment
0	26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26
1	26 | 26 | 26 | 26 | 26 | 26 | 26 | 52
2	26 | 26 | 26 | 26 | 26 | 52 | 26 | 26
3	26 | 26 | 26 | 26 | 26 | 52 | 52
4	26 | 26 | 52 | 26 | 26 | 26 | 26 | 26
5	26 | 26 | 52 | 26 | 26 | 26 | 52
6	26 | 26 | 52 | 26 | 52 | 26 | 26
7	26 | 26 | 52 | 26 | 52 | 52
8	52 | 26 | 26 | 26 | 26 | 26 | 26 | 26
9	52 | 26 | 26 | 26 | 26 | 26 | 52
10	52 | 26 | 26 | 26 | 52 | 26 | 26
11	52 | 26 | 26 | 26 | 52 | 52
12	52 | 52 | 26 | 26 | 26 | 26 | 26
13	52 | 52 | 26 | 26 | 26 | 52
14	52 | 52 | 26 | 52 | 26 | 26
15	52 | 52 | 26 | 52 | 52
16-23 (15 + N)	52 | 52 | - | 106 (N users)
24-31 (23 + N)	106 (N users) | - | 52 | 52
32-39 (31 + N)	26 | 26 | 26 | 26 | 26 | 106 (N users)
40-47 (39 + N)	26 | 26 | 52 | 26 | 106 (N users)
48-55 (47 + N)	52 | 26 | 26 | 26 | 106 (N users)
56-63 (55 + N)	52 | 52 | 26 | 106 (N users)
64-71 (63 + N)	106 (N users) | 26 | 26 | 26 | 26 | 26
72-79 (71 + N)	106 (N users) | 26 | 26 | 26 | 52
80-87 (79 + N)	106 (N users) | 26 | 52 | 26 | 26
88-95 (87 + N)	106 (N users) | 26 | 52 | 52
96-99 (95 + M)	106 | - | 106 (M users)
100-103 (99 + M)	106 (2 users) | - | 106 (M users)
104-107 (103 + M)	106 (3 users) | - | 106 (M users)
108-111 (107 + M)	106 (4 users) | - | 106 (M users)
112	52 | 52 | - | 52 | 52
113	Empty 242-tone RU - No user assigned
116-127	Reserved
128-135 (127 + N)	106 | 26 | 106 (N users)
136-143 (135 + N)	106 (2 users) | 26 | 106 (N users)
144-151 (143 + N)	106 (3 users) | 26 | 106 (N users)
152-159 (151 + N)	106 (4 users) | 26 | 106 (N users)
160-167 (159 + N)	106 (5 users) | 26 | 106 (N users)
168-175 (167 + N)	106 (6 users) | 26 | 106 (N users)
176-183 (175 + N)	106 (7 users) | 26 | 106 (N users)
184-191 (183 + N)	106 (8 users) | 26 | 106 (N users)
192-199 (191 + N)	242 (N users)

26 tone RU assigned to 1 user as part of a 20 MHz subchannel assignment of 9 26-tone RUs

No users assigned to this RU; no data field transmitted on these subcarriers

The number of users (N) assigned to this 106-tone RU depends on the allocation index and must be 1-8.

The number of users (M) assigned to this 106-tone RU depends on the allocation index and must be 1-4.

The number of users assigned to the upper 106-tone RU depends on the allocation index, but 2 users are always assigned to the lower 106-tone RU

If selected, this 20 MHz subchannel is unused; the subchannel is punctured

RU assigned to 1 user

RU assigned to 1-4/8 users, depending on the allocation index

RU assigned to specified number of users, irrespective of the allocation index

Checking the above RU assignment figure, for allocation index 97, the 2nd RU contains 106 subcarriers; the 2nd user and the 3rd user occupy this RU. So, we set userIdx = 2, and ruIdx = 2.

8×2 MIMO with 2 streams corresponds to numTxRx = [8 2] and Nsts = 2.
MCS4 corresponds to mcs = [4].
TGax channel model-D corresponds to chan = “Model-D”.
Payload length = 1000 Byte corresponds to cfgHE.User{userIdxIter}.APEPLength = 1000.
LDPC coding is the default coding for 11ax. There is no need to setup LDPC. Please check cfgHE.User for the LDPC configuration.

The configuration code is shown as follows:

```
%% Full PHY simulation setup
clear all
tStart = tic;
mcs = [4]; % Vector of MCS to simulate between 0 and 11
numTxRx = [8 2]; % Matrix of MIMO schemes, each row is [numTx numRx]
chan = "Model-D"; % String array of delay profiles to simulate
userIdx = 2; % User of investigation
ruIdx = 2; % RU of investigation
Nsts = 2; % Number of space-time streams
maxnumberrors = 40*1e3;  % The maximum number of packet errors at an SNR
point
maxNumPackets = 40*1e3; % The maximum number of packets at an SNR point
% maxnumberrors = 5*1e1;  % The maximum number of packet errors at an SNR
point
% maxNumPackets = 5*1e2; % The maximum number of packets at an SNR point

% Fixed PHY configuration for all simulations
cfgHE = wlanHEMUConfig(97); % Input 11ax allocation index
% The full RU assignment and allocation index lookup table is shown
% in the quick start guide
% Example: when allocation index = 24,
% then 1st RU has size 106, 2nd/3rd RU size = 52
for userIdxIter = 1:numel(cfgHE.User)
    cfgHE.User{userIdxIter}.APEPLength = 1000; % Payload length in bytes
end
```

Open: MU code basic/1 Full PHY (no interferer)/getBox0SimParams.m

Consider simulating the PHY layer under RX SNR = 15, 16, 17, 18, 19, 20, 21, 22 dBs.
In the variable snr: Model-D, 8x2, MCS4 part, set the snr to be [15,16,17,18,19,20,21,22] (notice the bold part in the following code):

```
snr = {
    % Model-B
    [ ...
        {... % 1x1
    [-3:4:9,11], ...    % MCS 0
    [1:4:13], ...      % MCS 1
    [2:4:18], ...      % MCS 2
```

```
    [7:4:19,21], ...      % MCS 3
    [9:4:25], ...      % MCS 4
    [13:4:29], ...     % MCS 5
    [14:4:30], ...     % MCS 6
    [16:4:32,34], ...     % MCS 7
    [18:4:34,36] ... % MCS 8
    [18:4:38] ...      % MCS 9
    }; ...
        {... % 4x1
    1:3:17, ...   % MCS 0
    5:3:23, ...   % MCS 1
    9:4:30, ...   % MCS 2
    12:4:36, ... % MCS 3
    16:4:40, ... % MCS 4
    20:4:43, ... % MCS 5
    22:4:46, ... % MCS 6
    24:4:48, ... % MCS 7
    26:4:50 ...   % MCS 8
    29:4:54 ...   % MCS 9
    }; ...
        {... % 4x2
    1:3:17, ...   % MCS 0
    5:3:23, ...   % MCS 1
    9:4:30, ...   % MCS 2
    12:4:36, ... % MCS 3
    16:4:40, ... % MCS 4
    20:4:43, ... % MCS 5
    22:4:46, ... % MCS 6
    24:4:48, ... % MCS 7
    26:4:50 ...   % MCS 8
    29:4:54 ...   % MCS 9
    }; ...
         {... % 8x2
    1:3:17, ...   % MCS 0
    5:3:23, ...   % MCS 1
    9:4:30, ...   % MCS 2
    12:4:36, ... % MCS 3
    16:4:40, ... % MCS 4
    20:4:43, ... % MCS 5
    22:4:46, ... % MCS 6
    24:4:48, ... % MCS 7
    26:4:50 ...   % MCS 8
    29:4:54 ...   % MCS 9
    }; ...
    ];

% Model-D
    [ ...
        {... % 1x1
    -3:3:9, ...      % MCS 0
    0:4:12, ...      % MCS 1
    2:4:18, ...      % MCS 2
    8:4:20, ...      % MCS 3
    11:4:23, ...     % MCS 4
```

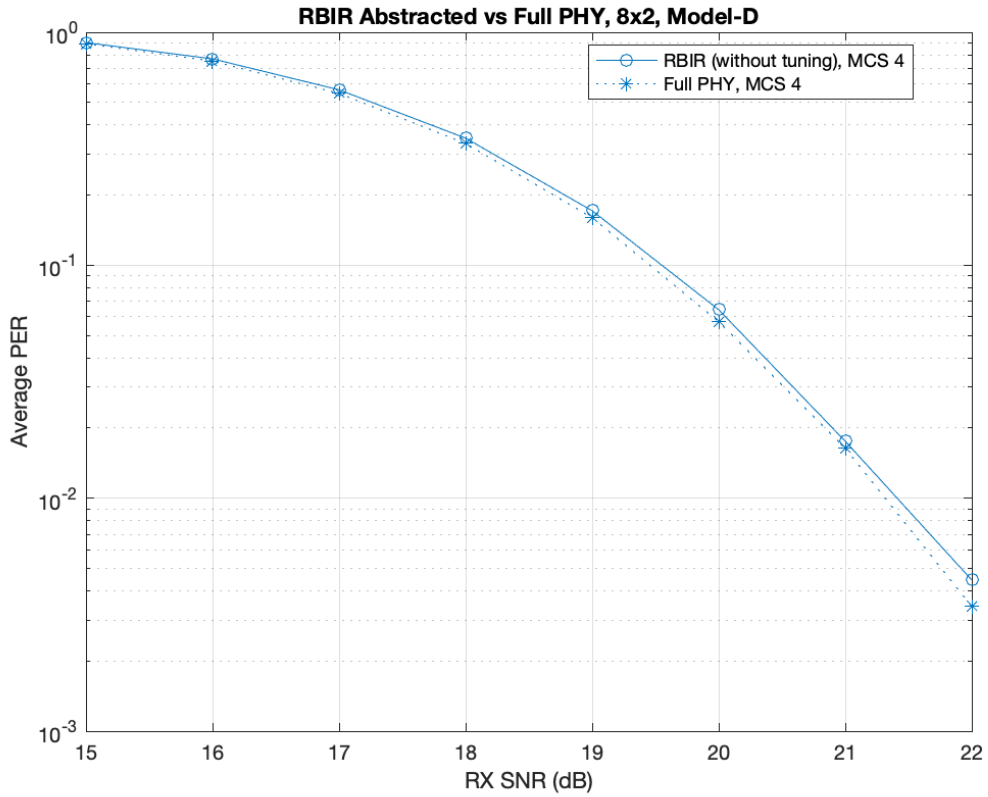
```
[14:4:26,28], ...    % MCS 5
16:4:28, ...    % MCS 6
18:4:30, ...    % MCS 7
21.5:4:37.5 ... % MCS 8
20:4:36 ...     % MCS 9
}; ...
    {... % 4x1
-4:2:2, ...   % MCS 0
5:2:15, ...   % MCS 1
9:2:21, ...   % MCS 2
12:2:24, ...  % MCS 3
12:2:16, ...  % MCS 4
20:3:32, ...  % MCS 5
22:3:34, ...  % MCS 6
24:3:39, ...  % MCS 7
27:3:40 ...   % MCS 8
29:3:44 ...   % MCS 9
}; ...
    {... % 4x2
[11,13,14,15], ...   % MCS 0
5:2:15, ...   % MCS 1
9:2:21, ...   % MCS 2
12:2:24, ...  % MCS 3
[10 13 15 16], ...  % MCS 4
20:3:32, ...  % MCS 5
22:3:34, ...  % MCS 6
24:3:39, ...  % MCS 7
27:3:40 ...   % MCS 8
29:3:44 ...   % MCS 9
}; ...
    {... % 8x2
1:2:10, ...   % MCS 0
5:2:15, ...   % MCS 1
9:2:21, ...   % MCS 2
12:2:24, ...  % MCS 3
[15,16,17,18,19,20,21,22], ...  % MCS 4
20:2:30, ...  % MCS 5
22:3:34, ...  % MCS 6
24:3:39, ...  % MCS 7
27:3:40 ...   % MCS 8
29:3:44 ...   % MCS 9
}; ...
] ...
};
```

After the above setup, go back to MU code basic/1 Full PHY (no interferer)/fullPHY.m
This is the main script of full PHY simulation. click on 'Run' in MATLAB.
The main script will call MU code basic/1 Full PHY (no interferer)/box0Simulation.m, the key function to realize full PHY simulation. This key function assumes ZF MIMO precoding and MMSE MIMO decoding. Users can comment out the ZF MMO precoding calculation part in box0Simulation.m to enable the default Fourier MIMO precoding.

After running the above full PHY simulation (takes about 16~18 hours), a file named "sinrPer_Config97_Model-D_8-by-2_MCS4.mat" will be created.

This file includes 40000 {post-MIMO processing SINR matrix, packet error state} pairs for each RX SNR. The post-MIMO processing SINR matrices are in results{snrIdx}.sinrStore, and the packet error states are in results{snrIdx}.perStore, where snrIdx =1,2,3,4,5,6,7,8.

A figure named “RBIR Abstracted vs Full PHY, 8x2, Model-D” will also pop up.
This figure shows the the average PER vs RX SNR(dB) of the full PHY simulation and RBIR prediction (without tuning). If the two curves approximately match, then we can move on the next step.



Step 2 (EESM parameter estimation):
Goal: optimizing EESM parameter (beta) using 40000 {post-MIMO processing SINR matrix, packet error state} pairs generated from full PHY simulation. This step is shown in Fig. 3 of our IEEE TCOM paper. The principle follows the traditional PHY layer abstraction flow plotted in Fig. 6 of our IEEE TCOM paper.

Copy “sinrPer_Config97_Model-D_8-by-2_MCS4.mat” generated in Step 1 into MU code basic/2 EESM parameter optimization.

Open: MU code basic/2 EESM parameter optimization/eesmAbstractionPerVsEffSinr.m
This is the main script in MU code basic/2 EESM parameter optimization.

Load “sinrPer_Config97_Model-D_8-by-2_MCS4.mat”:

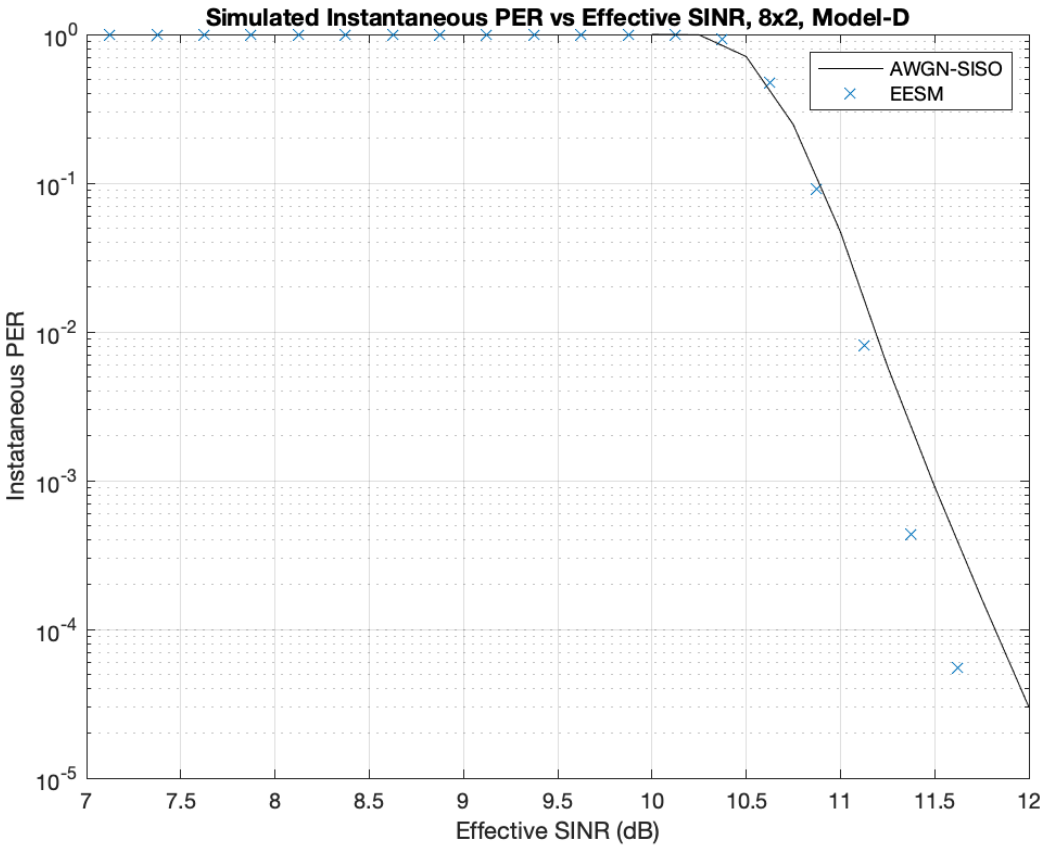
```
load('sinrPer_Config97_Model-D_8-by-2_MCS4.mat');
```

For different MCS, set different initial EESM parameter for optimization.
The suggestions for choosing initial EESM parameter are shown in the following code.

```
% Tuning parameter in this function: EESM parameter - beta
% Suggestion: the higher MCS, the larger initialized beta
% Initial values for reference:
%   MCS: 0  -> beta : 1
%   MCS: 1  -> beta : 2
%   MCS: 2  -> beta : 1.5
%   MCS: 3  -> beta : 5
%   MCS: 4  -> beta : 7
%   MCS: 5  -> beta : 26
%   MCS: 6  -> beta : 33
%   MCS: 7  -> beta : 43
%   MCS: 8  -> beta : 111
%   MCS: 9  -> beta : 170
%   MCS: 10  -> beta : 410
%   MCS: 11  -> beta : 650
beta = 7;
```

As we use MCS4, set the initial EESM parameter value to be 7 according the above suggestion.

Opening eesmAbstractionPerVsEffSinr.m, click on 'Run' in MATLAB.
This main script minimizes MSE between 1) instantaneous PER VS EESM based effective SNR under the PHY layer configuration in step 1, and 2) instantaneous PER VS SNR under AWGN-SISO channel. The output is the optimized EESM parameter (beta): betaOpt. This is realized in the awgnPerSnrFittingMse.m function and MATLAB fminsearch function.



After running eesmAbstractionPerVsEffSinr.m (takes about a few mins), a file named "eesmEffSinr_Config97_Model-D_8-by-2_MCS4.mat" will be created.
This file includes the optimized beta value stored in betaOpt.

A figure named "Simulated Instantaneous PER vs Effective SINR, 8x2, Model-D" will also pop up. This figure shows 1) instantaneous PER VS EESM based effective SNR under the PHY layer configuration in step 1, and 2) instantaneous PER VS SNR under AWGN-SISO channel. If the two curves approximately match, then the traditional EESM L2S mapping works well and we can move on the next step.

Step 3 (Log-SGN parameters optimization):
Goal: optimizing the four log-SGN parameters using 40000 EESM based effective SINR histogram generated from the optimized beta and 40000 post-MIMO processing SINR matrices. This step is shown in Fig. 3 of our IEEE TCOM paper. The principle of log-SGN parameters optimization is shown in Algorithm 1 of our IEEE TCOM paper.

Open MU code basic/3 Basic EESM-log-SGN method/eesmLogSGNBasic.m. This is the main script in MU code basic/3 Basic EESM-log-SGN method.

First, choosing the RX SNR snrs(snrIdx) that we want to simulate. Now, we choose snrIdx = 6; corresponding to RX SNR of 20dB.

Copy "sinrPer_Config97_Model-D_8-by-2_MCS4.mat" generated in Step 1 and "eesmEffSinr_Config97_Model-D_8-by-2_MCS4.mat" generated in Step 2 into MU code basic/3 Basic EESM-log-SGN method. Load these two files.

```
% Load Post-MIMO processing SINR matrix
load('sinrPer_Config97_Model-D_8-by-2_MCS4.mat')
sinrStore = results{snrIdx}.sinrStore;
% Load optimized EESM parameter beta
load('eesmEffSinr_Config97_Model-D_8-by-2_MCS4.mat')
```

snrPer_Config24_Model-D_8-by-2_MCS4.mat provides the 40000 post-MIMO processing SINR matrices and eesmEffSnr_Config24_Model-D_8-by-2_MCS4.mat provides the optimized EESM parameter beta. Click on 'Run' in MATLAB, using these two information, we can obtain effective SINR histogram gammaEffdB using effectiveSinrVec.m in tgaxEESMLinkPerformanceModel.m. This step is shown in Fig. 3 of our TCOM paper, and the principle of generating gammaEffdB follows equation (3) in our TCOM paper.

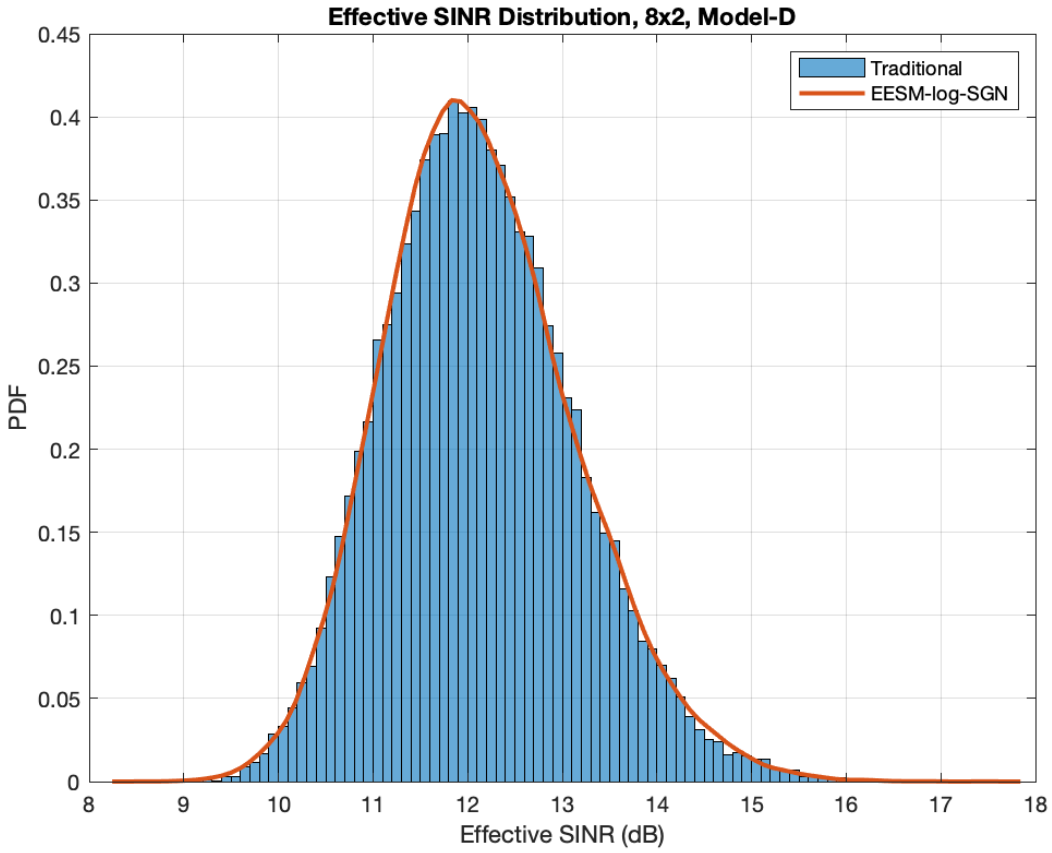
Using the EESM based effective SINR histogram gammaEffdB, fit gammaEffdB using maximum likelihood estimation to approximate gammaEffdB as the log-SGN PDF. This can be achieved by the function logSGNFitting.m. The output of logSGNFitting.m is the optimized vector of the 4 log-SGN parameters logSGNParamBest.

After running eesmLogSGNBasic.m, the best logSGNParamBest is
logSGNParamBest =

2.5712 0.3219 1.7736 0.0000

We record this value as logSGNParamBest1 in MU code basic/4 EESM-log-SGN mixture/eesmlogSGNMixture.m.

The plot showing the EESM-log-SGN predicted effective SINR histogram and the effective SINR histogram gammaEffdB predicted by the traditional EESM L2S mapping will also pop up after running eesmLogSGNBasic.m.

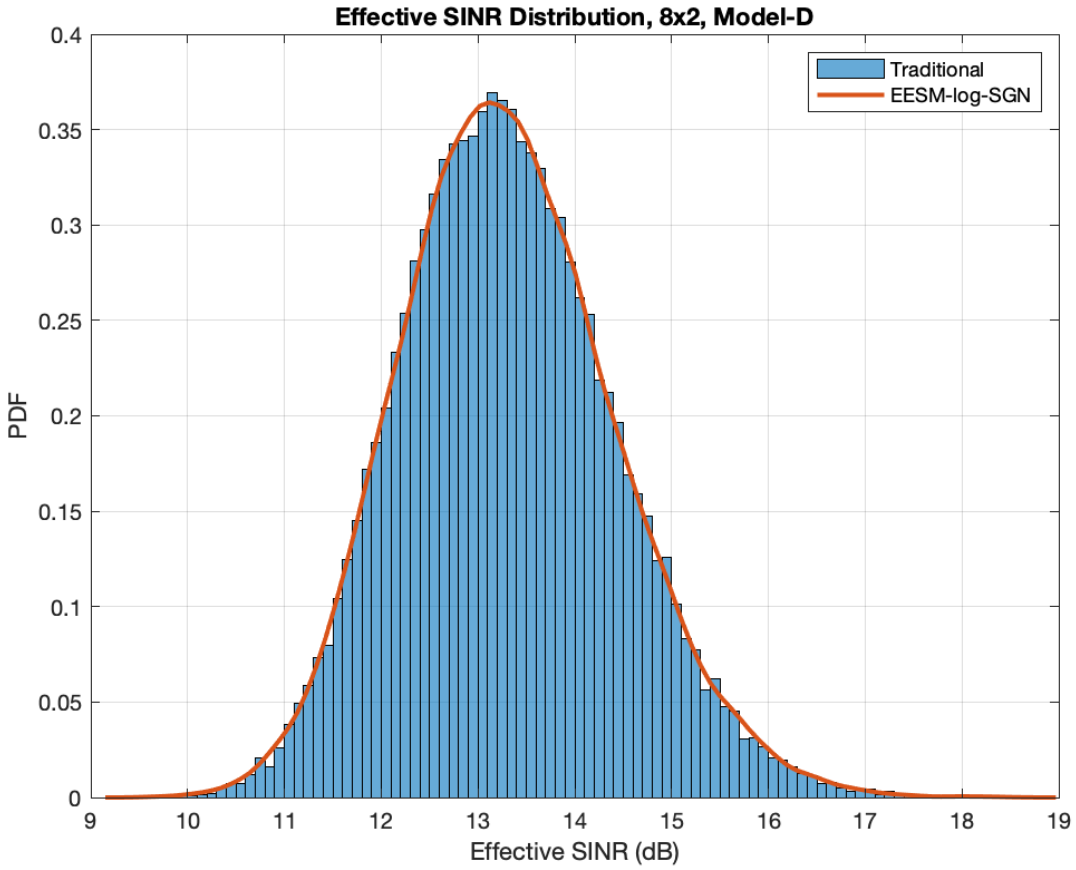


This figure shows the goodness of the effective SINR PDF fitting predicted by the EESM-log-SGN method.
If you find the fitting result in the above figure is not so desirable, this is because the function logSGNFitting.m does not produce the a good optimized log-SGN parameter estimation result due to the improper random initial value x0 shown in logSGNFitting.m. You can run the current script eesmLogSGNBasic.m multiple times until you find a good enough fitting result.

Next, we choose snrIdx = 8; corresponding to RX SNR of 22dB. Rerun eesmLogSGNBasic.m.
After running eesmLogSGNBasic.m, the best logSGNParamBest is
logSGNParamBest =

2.8356 0.3418 1.5308 0.0000

We record this value as logSGNParamBest2 in MU code basic/4 EESM-log-SGN mixture/ eesmlogSGNMixture.m.
The plot showing the EESM-log-SGN predicted effective SINR histogram and the effective SINR histogram gammaEffdB predicted by the traditional EESM L2S mapping will also pop up after running eesmLogSGNBasic.m.



This figure shows the goodness of the effective SINR PDF fitting predicted by the EESM-log-SGN method.

Step 4 (EESM-Log-SGN mixture model implementation):
Goal: using two sets of log-SGN parameters under two close RX SNRs (gamma1 and gamma2, gamma1<gamma2) to predict log-SGN random variables under the RX SNR in between (gamma1~gamma2). This step is shown in Algorithm 3 of our IEEE TCOM paper.

Open MU code basic/4 EESM-log-SGN mixture/eesmlogSGNMixture.m.
This is the main script in MU code basic/4 EESM-log-SGN mixture.

In step 3, we have already load the two set of log-SGN parameters in eesmlogSGNMixture.m. Now, set snr1 (snr2) to be the RX SNR corresponding to the 1st (2nd) set of log-SGN parameters. In this example, snr1 = 20dB, and snr2 = 22dB.

We would like to predict the log-SGN random variables under 21dB RX SNR. We set the RX SNR under investigation snr = 21.

Data length, channel coding, and MCS are the same as that in step 1~3.

The overall setup in step 4 is provided as follows:

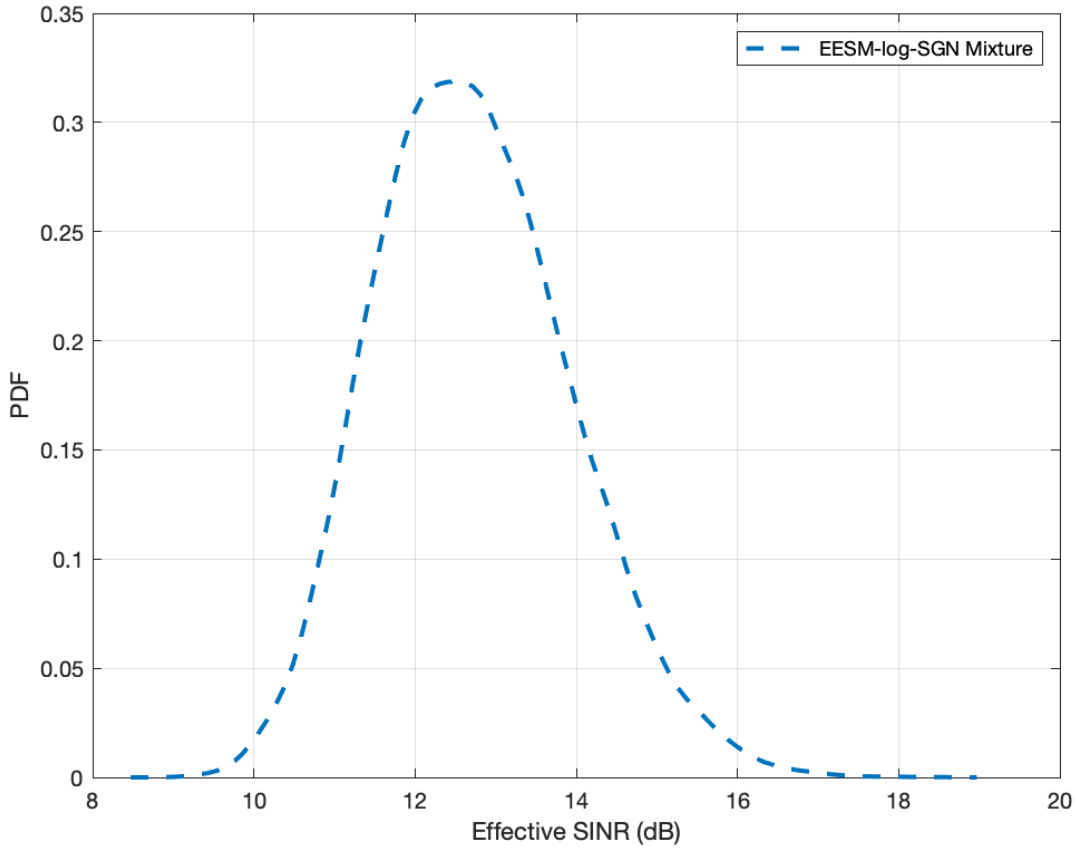
```
coding = 'LDPC'; % Channel coding
dataLength = 1000; % APEP length
format = 'HE_MU'; % OFDM/OFDMA MIMO/MU-MIMO setup
```

```
mcs = 4; % MCS
eesmAbstraction = tgaxEESMLinkPerformanceModel;
snr1 = 20; % RX SNR 1 in dB
snr2 = 22; % RX SNR 2 in dB
snr = 21; % RX SNR under investigation in dB
% Optimized log-SGN parameters under RX SNR 1
logSGNParamBest1 = [
2.5712    0.3219    1.7736    0.0000
];
% Optimized log-SGN parameters under RX SNR 2
logSGNParamBest2 = [
2.8356    0.3418    1.5308    0.0000
];
```

Data length, channel coding, and MCS are the same as that in step 1~3.

Click on 'Run' in MATLAB. eesmlogSGNMixture.m will generate effective SNR histogram effSnrdB predicted by the EESM-log-SGN mixture model in Algorithm 3 of our IEEE TCOM paper.

After running eesmlogSGNMixture.m, the effective SNR histogram effSnrdB will pop up.



We can compare the average PER predicted by EESM-log-SGN mixture model logSGNMixtureAvgPer and the average PER generated by the full PHY simulation results{snrIdx}.packetErrorRate under the RX SNR under investigation (21dB, corresponding to the 7th RX SNR index).

```
logSGNMixtureAvgPer =

    0.0249

results{7}.packetErrorRate

ans =

    0.0164
```

Final result:
Putting the effective SNR histogram generated in step 3 and step 4 together, we have the following figure (Fig. 13 in our IEEE TCOM paper).