**IMT-Advanced Channel Simulator**

**Version 0.991 (Last Update 2009, May 8)**

**Source : TTA PG 707**

**Contact : Kyung-Won Kim (kyungwon@korea.ac.kr)**

**Wireless Network Lab at Korea Univ.**

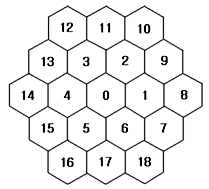
**Abstract**

The C-based source code implements the channel model with the network deployment environment according to the ITU-R report M.2135 [1] – the IMT-Advanced Channel Model. The program can be compiled and executed for any platform that supports ANSI C/C++ compilation. It has been tested on the MS Visual C++ environment as well. The program includes some random number generators from [2], but as it consistently operates according to the input seed, the results are the same regardless of the platforms or the time of execution. The codes are verified against an independent M.2135 implementation [3] in a bit-exact fashion for the deterministic functions. These source codes can be used for the system-level simulation of IMT-Advanced system.

**Reference**

1. ITU-R Report M.2135, *Guidelines for evaluation of radio transmission technologies for IMT-Advanced*, 2008.
2. William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery, *Numerical recipes: the art of scientific computing,Third Edition*, Cambridge University Press, 2007.
3. Finland and China (People’s Republic of), *Software Implementation of IMT.EVAL Channel Model*, Contribution 5D/313, presented in ITU-R WP 5D #3 in Seoul, October 2008.

**1. Introduction**

 The Channel model package takes the user-defined simulation environment defined in the configuration file. It is implemented with default values of M.2135. The channel model has five scenarios. Indoor hotspot scenario consists of 16 rooms and long corridor. In this scenario, there are two base stations (only two sectors). The other scenarios have 19 cells with wraparound. One cell consists of three sectors with antenna sectoring.

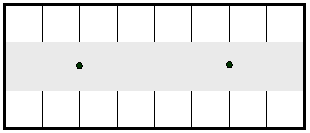


Figure 1. Layout of indoor hotspot

Figure 2. Layout of outdoor scenarios

**2. cpp files**

The ITU-R Channel simulator Package includes the following source files.

|  |  |
| --- | --- |
| main\_channel.cpp | Main function of Channel simulator. |
| initialization.cpp | Set up the initial state. Load the configure file and drop UEs. |
| channel.cpp | Generate UE distribution. Generate static gain and channel coefficients. |
| ue.cpp | Determine the sectors to be computed channel coefficients |
| logging\_point.cpp | Print the simulation results. |

Source files refer to public headers and each private header file. Public header files are <const.h> and <common.h>. <const.h> includes constant numbers and <common.h> includes public variables.

**3. Simulation Environment**

**① Scenario**

There are five Scenarios in M.2135. The channel model generates outputs in the scenarios and use default values in M.2135.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scenarios** | **Indoor Hotspot** | **Urban Macro** | **Urban**  **Micro**  **(O-to-I)** | **Rural**  **Macro** | **Sub-urban**  **Macro** |
| **site-to-site**  **distance**  **[m]** | 60 | 500 | 200 | 1731 | 1299 |
| **speed of UE**  **[km/h]** | 3 | 30 | 3 | 120 | 90 |
| **Carrier freq**  **[㎓]** | 3.4 | 2.0 | 2.5 | 0.8 | 2.0 |
| **Number of sectors** | 2 | 57 | | | |

**② Antenna pattern**

The channel model implements antenna pattern of down link. The channel model uses reasonable antenna distances. We assume that receiver antenna is omni-antenna and transmit antenna is separated antenna.

|  |  |
| --- | --- |
| **Link Types** | **Value** |
| **d\_rx** | 0.5ㆍλ |
| **d\_tx** | 10ㆍλ |
| **F\_rx** | 1 |
| **F\_tx** | 120 sectoring (except Indoor Hotspot)  1 (only case by Indoor Hotspot) |

(d\_rx and d\_tx are antenna distance terms. F\_rx and F\_tx are antenna gains.)

**4. Input Parameters**

Input parameters are given by the configure file. It can setup parameters as below.

1. **Simulation setting parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Definition** | **Default value** |
| \_seed | Random seed | 100000 |
| run\_times | Number of time samples. one time sample is 1㎳. | 100 |
| scenario | Select simulation environment.  # Scenario types are INDOOR\_HOTSPOT, URBAN\_MACRO, URBAN\_MICRO, RURAL\_MACRO, SUBURBAN\_MACRO. | URBAN\_MACRO |
| los\_type | Select LOS type.  # LOS types are  LOS or NLOS or RANDOM. | RANDOM |
| num\_drops | Number of drops | 2 |
| num\_user\_cell | Number of users per one cell | 10 |
| num\_received\_antenna | Number of received antennas | 1 |
| num\_transmit\_antenna | Number of transmit antennas | 1 |
| num\_compute\_coef | Number of computing channel coefficients.  (per 1UE.)  (= number of adj\_sector. It is explained in 4-**①**) | 1 |

**② Logging point parameters**

Logging point parameters are intended to print simulation results with data files. It is used to verify the channel model. You can see results at ‘../Output/FILENAME.dat'.

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Definition** | **Default value** |
| ue\_distribution | Output is distribution of UEs.  Output data is '../Output/UE\_location.dat'. | 0(off) |
| PathLoss | Output is path loss of UEs.  Path loss is in ㏈  Output data is '../Output/PathLoss.dat'. | 0(off) |
| ChannelCoef | Outputs are channel coefficients.  Output data of channel coefficients is '../Output/Channel\_Coef.dat'. | 0(off) |
| DelaySpread | Output is delay spread of sample UE.  Delay time is in s and cluster power is in ㏈  Output data is '../Output/DelaySpread.dat'. | 0(off) |
| AngleSpread | Output is angle spread of sample UE.  Angle is in degree and cluster power is in ㏈  Output data is '../Output/AoASpread.dat' and ‘../Output/AoDSpread.dat’. | 0(off) |
| PDF | Output is probability density function of random variables.  Random variables are defined in [2]. | 0(off) |
| sample\_ue | Sample UE of channel coefficients and delay spread | 0(off) |

**③ Examples of configure file**

|  |  |
| --- | --- |
| # simulation environment setting  \_seed  run\_times  scenario  los\_type  num\_drops  num\_user\_cell  num\_received\_antenna  num\_transmit\_antenna  num\_compute\_coef  # logging point  ue\_distribution  PathLoss  ChannelCoef  DelaySpread  AngleSpread  PDF  sample\_ue | 456235234  100  URBAN\_MICRO  RANDOM  2  10  2  2  5  1 # 1(on) or 0 (off)  1 # 1(on) or 0 (off)  1 # 1(on) or 0 (off) need sample\_ue  1 # 1(on) or 0 (off)  1 # 1(on) or 0 (off)  1 # 1(on) or 0 (off)  1 # UE index |

**5. Output Parameters**

You can use output parameters in the main function without output data files. See below that outputs and main function.

1. **Output parameters**

|  |  |
| --- | --- |
| **Parameter name** | **Definition** |
| ue[k].static\_gain[ sector ] | Sum of path loss, shadowing between UEs and sectors.  Static gain is in dB |
| ue[k].sector\_in\_control | Sector number which has the largest static gain of each UEs. |
| ue[k].adj\_sector[i] | Sector numbers when static gains are listed in descending order. Small scale parameters are generated between UEs and adj\_sector.  (ue[k].adj\_sector[0] = ue[k].sector\_in\_control) |
| Channel\_coef [k][i][n][u][s] | Channel coefficients between UEs.  Channel coefficients are in linear scale. |
| num\_path [k][i] | Number of clusters. |
| delay [k][i][n][u][s] | Delay time is in sec. |
| cluster\_power[k][i][n][u][s] | Cluster powers. |

( k, i, n, u, s and sector are indices of UEs, sectors, clusters, receiver antennas and transmit antennas)

**② Output files**

In output folder, there are output files printed by logging point parameters.

**③ Main Function**

|  |
| --- |
| int main( int argc,char \*argv[] )  {  SimulationConfiguration( argc, argv );  InitializeSystem();  Size\_Channel\_Parameters();  Size\_Logging\_point\_Parameters() ;    for(drop\_idx = 0; drop\_idx < num\_drops; drop\_idx ++)  {  InitializeADrop();  for(t = 0; t < run\_times; t ++)  {  for(int ue\_idx = 0; ue\_idx < num\_ues; ue\_idx ++)  {  for(int adj\_sec\_idx = 0; adj\_sec\_idx < num\_compute\_coef; adj\_sec\_idx++)  {  for(int u = 0; u < num\_received\_antenna; u++)  {  for(int s = 0; s < num\_transmit\_antenna; s++)  {  ChannelSampleGeneration( ue\_idx, u, s, adj\_sec\_idx ) ;  }  }  }  }  **// Simulator Here**  Logging\_Point() ;  loading () ;  }  }  } |