Project: IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)

Submission Title: [**A Modified MATLAB Simulation Program for TSV-channel Model**]

Date Submitted: [November 15, 2006]

Source: [Hiroshi Harada, Ryuhei Funada, Hirokazu Sawada, Chang-soon Choi, Yozo Shoji, Shuzo Kato]
Company [NICT]
Address[3-4 Hikari-no-oka, Yokosuka-shi, Kanagawa 239-0847, Japan]
Voice:[+81-46-847-5074]
FAX:[+81-46-847-5440]
E-Mail:[harada@nict.go.jp, funada@nict.go.jp, sawahiro@nict.go.jp, shoji@nict.go.jp, cschoi@nict.go.jp,shu.kato@nict.go.jp]

Re: []

Abstract: [Proposing a modified MATLAB Simulation Program for TSV-channel model]

Purpose: [To be considered in 15.3c transmission performance by computer simulation]

Notice:

A Modified MATLAB Simulation Program for TSV-channel Model

Hiroshi Harada, Ryuhei Funada,
Hirokazu Sawada,Chang-Soon Choi,
Yozo Shoji, Shuzo Kato (NICT)

Summary of this document

- Finished preparing a MATLAB simulation program for TSV channel model
- Explain the flowchart of the MATLAB program
- Explain the detail of the program
- Show a comparison of experimental and simulated results
- Summarize available LOS / NLOS channel models by the MATLAB-based TSV channel model
- Show recommendations of how to spread programs of the contributors to simulate system requirements

Appendix A: Definition of TSV model (modified)

CIR: $$h(t)=\beta\,\delta(t)+\sum_{l=0}^{L-1}\sum_{m=0}^{M_l-1}\alpha_{l,m}\,\delta\left(t-T_l-\tau_{l,m}\right)\,\delta\left(\varphi-\Psi_l-\psi_{l,m}\right)$$

(Complex impulse response)

$$\overline{\left|\alpha_{l,m}\right|^2}=\Omega_0 e^{-T_l/\Gamma}e^{-\tau_{l,m}/\gamma-k[1-\delta(m)]}\sqrt{G_r\left(0,\Psi_l+\psi_{l,m}\right)},\angle\alpha_{l,m}\propto \text{Uniform}[0,2\pi)$$

PL_d: Path loss of the first impulse response
t: time[ns]
$\delta(\cdot)$: Delta function
l = cluster number,
m = ray number in l-th cluster,
L = total number of clusters;
M_l = total number of rays in the l-th cluster;
T_l = arrival time of the first ray of the l-th cluster;
$\tau_{l,m}$ = delay of the m-th ray within the l-th cluster relative to the firs path arrival time, T_l;
Ω_0 = Average power of the first ray of the first cluster
$\Psi_l \propto$ Uniform[0,2π); arrival angle of the first ray within the l-th cluster
$\psi_{l,m}$ = arrival angle of the m-th ray within the l-th cluster relative to the first path arrival angle, Ψ_l

Two-path response

$$\beta\,[\text{dB}]=20\cdot\log_{10}\left[\left(\frac{\mu_d}{d}\right)\left|\sqrt{G_{t1}G_{r1}}+\sqrt{G_{t2}G_{r2}}\Gamma_0\exp\left[j\frac{2\pi}{\lambda_f}\frac{2h_1h_2}{d}\right]\right|\right]-PL_d(\mu_d)$$

$$PL_d(\mu_d)[\text{dB}]=PL_d(d_0)+10\cdot n_d\cdot\log_{10}\left(\frac{d}{d_0}\right)\qquad PL_d(d_0)[\text{dB}]=20\log_{10}\left(\frac{4\pi d_0}{\lambda_f}\right)+A_{NLOS}$$

A_{NLOS}: Constant attenuation for NLOS

Path number of G_{ti} and G_{ri} (1 : direct, 2 : refrect)

Arrival rate: Poisson process

$$p(T_l\,|\,T_{l-1})=\Lambda\exp\left[-\Lambda\left(T_l-T_{l-1}\right)\right],\quad l>0$$

$$p(\tau_l\,|\,\tau_{l,(m-1)})=\lambda\exp\left[-\lambda\left(\tau_l-\tau_{l,(m-1)}\right)\right],\quad m>0$$

Two-path parameters (4)

$d \propto$ Uniform : Distance between Tx and Rx

$h_1 \propto$ Uniform : Height of Tx

$h_2 \propto$ Uniform : Height of Rx

$\mu_d \propto$ Average of distance between Tx and Rx

$|\Gamma_0|$: Reflection coefficient

- $|\Gamma_0| \cong 1$: LOS Desktop environment (incident angle $\cong \pi/2$)
- $|\Gamma_0| \cong 0$: Other LOS/NLOS environment

S-V parameters (7)

Γ : *cluster* decay factor

$1/\Lambda$: *cluster* arrival rate

γ : *ray* decay factor

$1/\lambda$: *ray* arrival rate

σ_1 : *cluster* lognormal standard deviation

σ_2 : *ray* lognormal standard deviation

σ_ϕ : Angle spread of ray within cluster (Laplace distribution)

Antenna parameters (2)

$Gt(\theta,\phi)$: Antenna gain of Tx

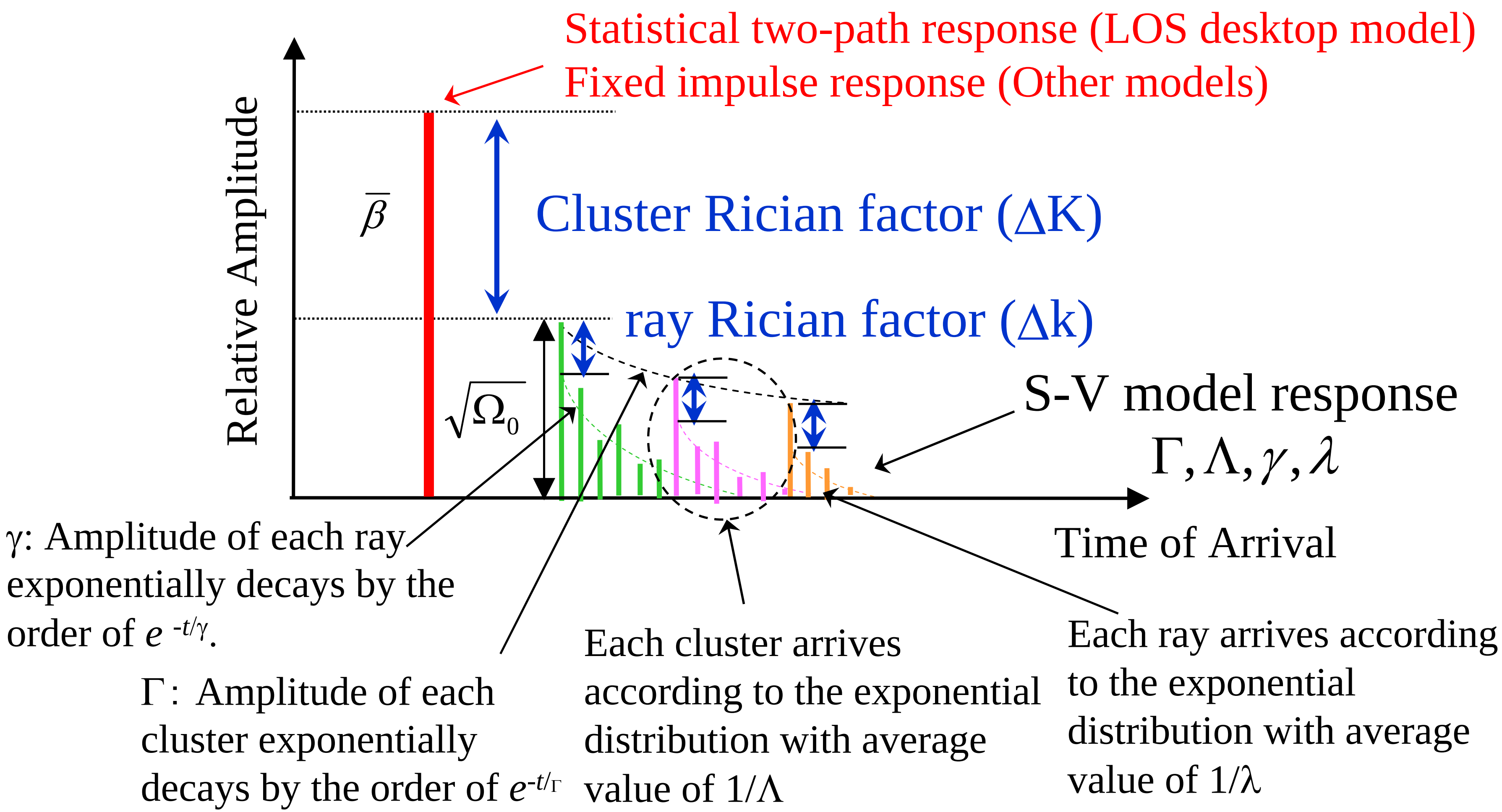
$Gr(\theta,\iota)$: Antenna gain of Rx

Rician factor (2)

k : ray Rician effect in each cluster

$$K=\frac{\beta^2}{\sum_{l=0}^{L-1}\sum_{m=0}^{M_l-1}\left|\alpha_{l,m}^2\right|\delta\left(t-T_l-\tau_{l,m}\right)\delta\left(\varphi-\Psi_l-\psi_{l,m}\right)G_r\left(0,\Psi_l+\psi_{l,m}\right)}$$

Impulse response of TSV model



Examples of parameters for TSV model (LOS desktop channel model (Tx:60, Rx:60))

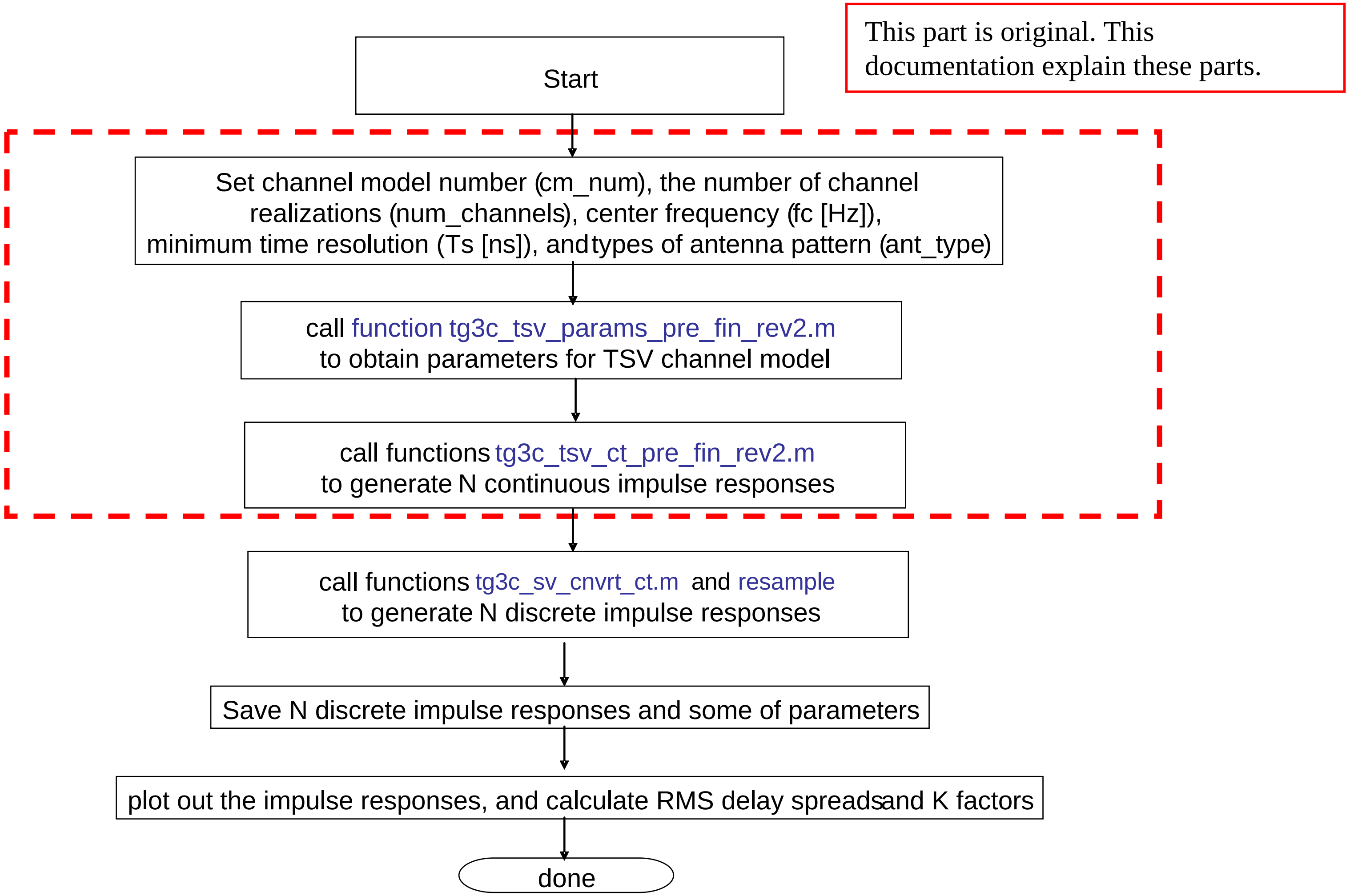
	TSV Model	Small Rician factor	S-V model oriented parameters							Number of cluster
Parameter	$\Omega_0(D)$ [dB]	k (Δk)	Γ [ns]	$1/\Lambda$ [ns]	γ [ns]	$1/\lambda$ [ns]	σ_1 cluster	σ_2 ray	σ_ϕ [deg]	N
Tx:60 Rx:60	3.46 D-98.4	3.97	22.3	21.1	17.2	2.68	7.27	4.42	38.1	3

Dependent on the distance between transmitter and receiver

Function calls

- tg3c_tsv_eval_pre_fin_rev2 (Main script M-file)
  - tg3c_tsv_params_pre_fin_rev2
  - tg3c_tsv_ct_pre_fin_rev2
    - tg3c_tsv_beta_calc_pre_fin_rev2
    - laplace_gen
    - calc_ant_gain_pre_fin_rev2
  - tg3c_sv_cnvrt_ct
  - Resample (built-in function)

Modified flowchart of tg3c_tsv_eval_pre_fin_rev2

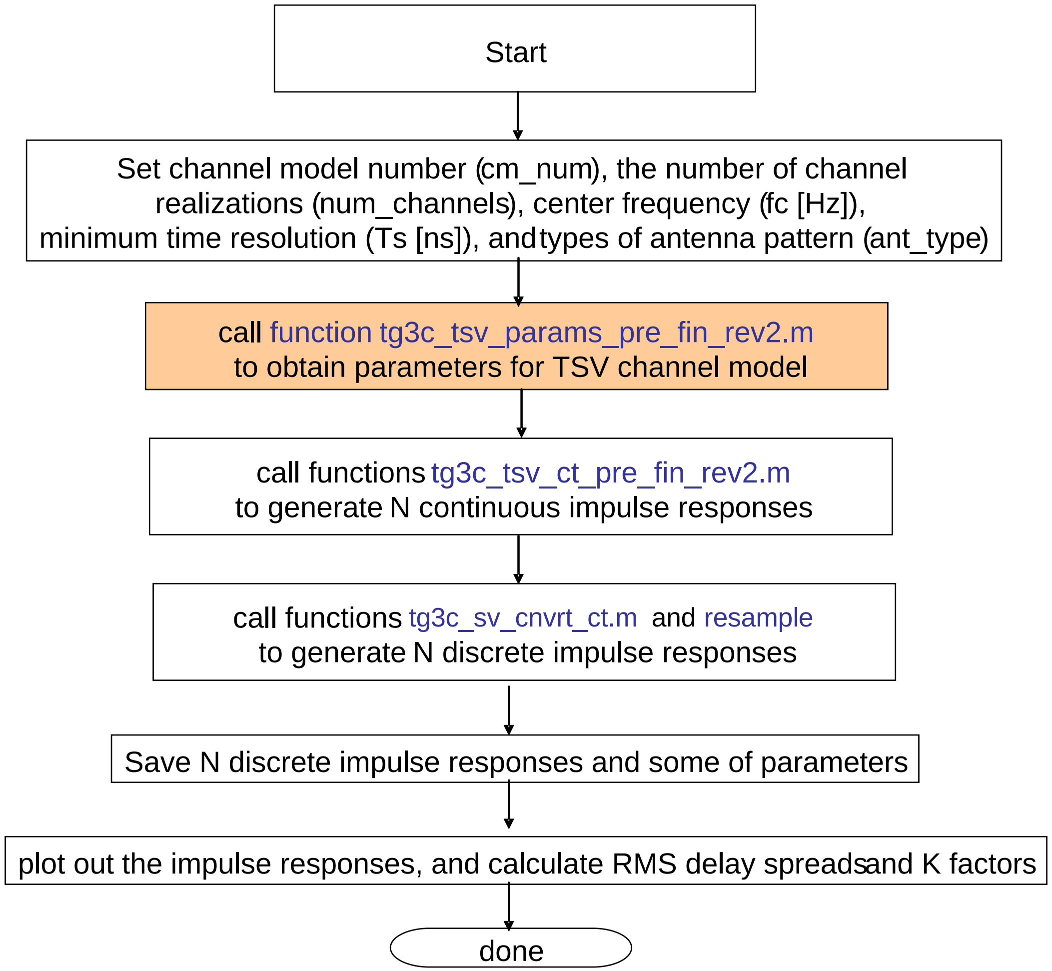


tg3c_tsv_eval_pre_fin_rev2.m

- Main script M-file
- This M-file generates impulse responses on the basis of the TSV model
- Matlab codes distributed in IEEE802.15.4a was modified
- This M-file consists of four sub-functions
  - tg3c_tsv_param_pre_fin_rev1.m
  - tg3c_tsv_ct_pre_fin_rev2.m
  - tg3c_sv_cnvrt_ct.m
  - resample.m (built-in function)

Means parent function

Modified flowchart of tg3c_tsv_eval_pre_fin_rev2



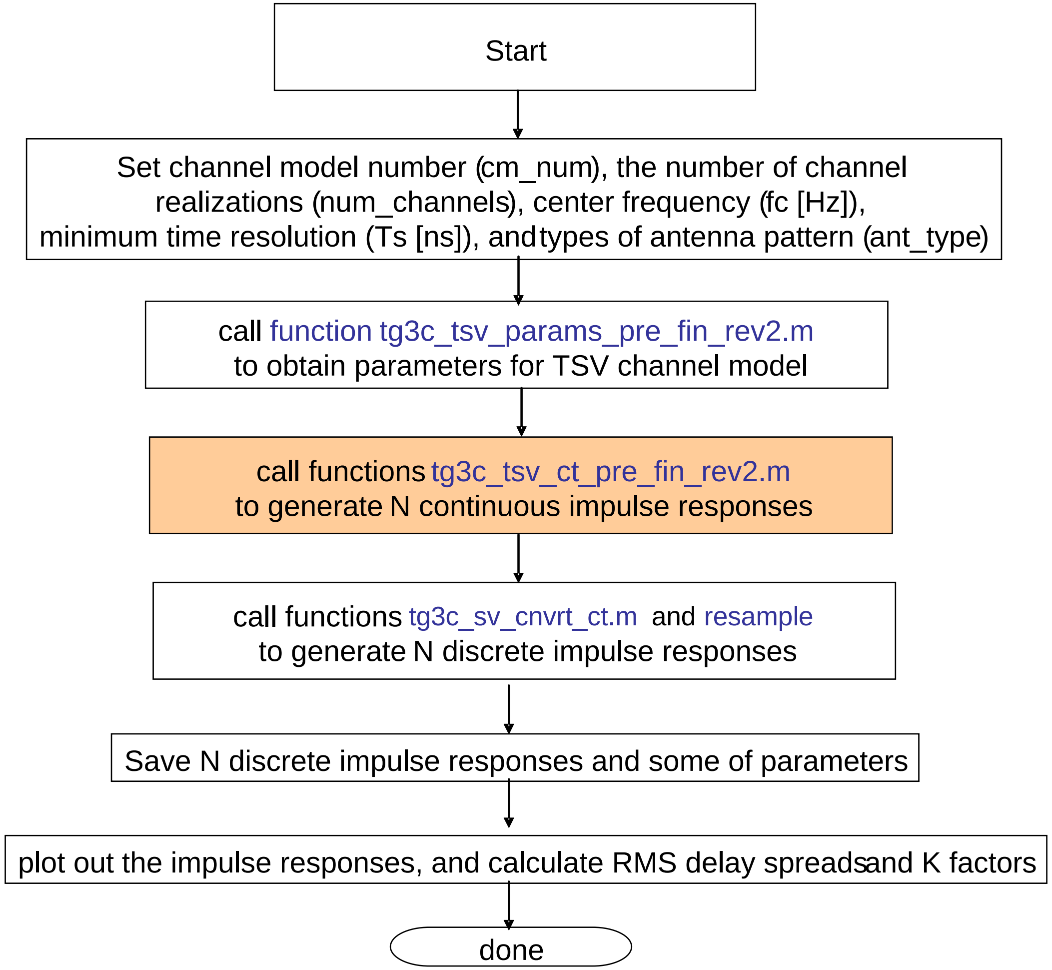
tg3c_tsv_params_pre_fin_rev2.m

- This function M-file outputs channel parameters according to channel model number
- This function consists of a sub-function and related programs
- Tx antenna beam-widths are basically same as those used in the experiments, and Rx antenna beam-widths can be changed for evaluations
- Rx antenna beam-widths can be changed for evaluations
- Power of a LOS component is calculated in this function using carrier frequency and assuming distance

```
function [adist, nlos, LOS_desktp_flg,Omega0, smallk, Lmean, Lam, lambda, Gam, ...
    gamma, std_ln_1, std_ln_2, sigma_fai, L_pl, tx_hpang, rx_hpang] = tg3c_tsv_params_pre_fin_rev2( cm_num, fc)

% Arguments
% cm_num      channel model number
%

% Output parameters
% nlos                    flag of NLOS environment
% Lmean                   number of Average arrival clusters
% Lam                     cluster arrival rate (clusters per nsec)
% lambda      ray arrival rate (rays per nsec)
% Gam                     cluster decay factor (time constant, nsec)
% gamma                   ray decay factor (time constant, nsec)
% std_ln_1    standard deviation of log-normal variable for cluster fading
% std_ln_2    standard deviation of log-normal variable for ray fading
% sigma_fai cluster angle-of-arrival spread in deg

% Parameters added by NICT
% adist                          assuming distance in mappded usage model (meter)
% LOS_desktp_flg                 flag used for beta calculation
% Omega0                         cluster power level
% smallk                         small Rician factor
% L_pl                           pathloss of the LOS component normalized with that of 1m
% tx_hpang                       TX half-power angle in deg
% rx_hpang                       RX half-power angle in deg

% Note: cm_num
% cm_num == 1**: LOS Residential model (mappded to UM1)
% cm_num == 3**: LOS Office model (UM3)
% cm_num == 9**: LOS Desktop model (UM9)
```

```
%************* LOS Residential channel model (UM1) *******************
if cm_num == 11 % Experimental data TX : 360deg, RX : 15deg
    adist        = 5;
    nlos         = 0;
    LOS_desktp_flg  = 0;
    Omega0       = -88.7;
    smallk       = 4.34;
    Lmean        = 9;
    Lam          = 1/5.24;   lambda = 1/0.820;
    Gam          = 4.46;     gamma = 6.25;
    std_ln_1     = 6.28;     std_ln_2    = 13.0;
    sigma_fai    = 49.8;
    tx_hpang     = 360;
    rx_hpang     = 15;
    n_pl         = 2.03;
    L_pl         = -68-10*n_pl*log10(adist);
```

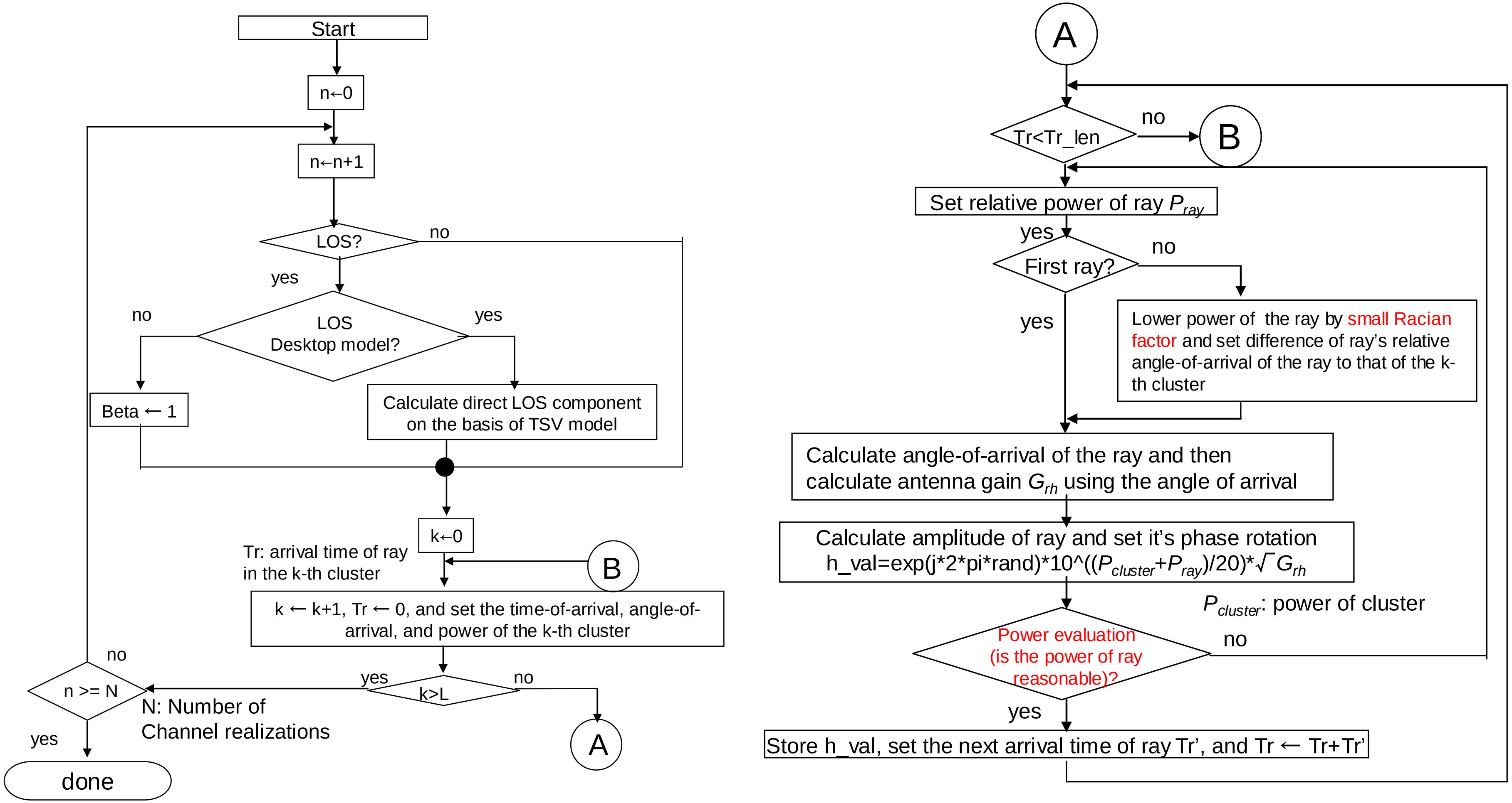
Modified flowchart of tg3c_tsv_eval_pre_fin_rev2



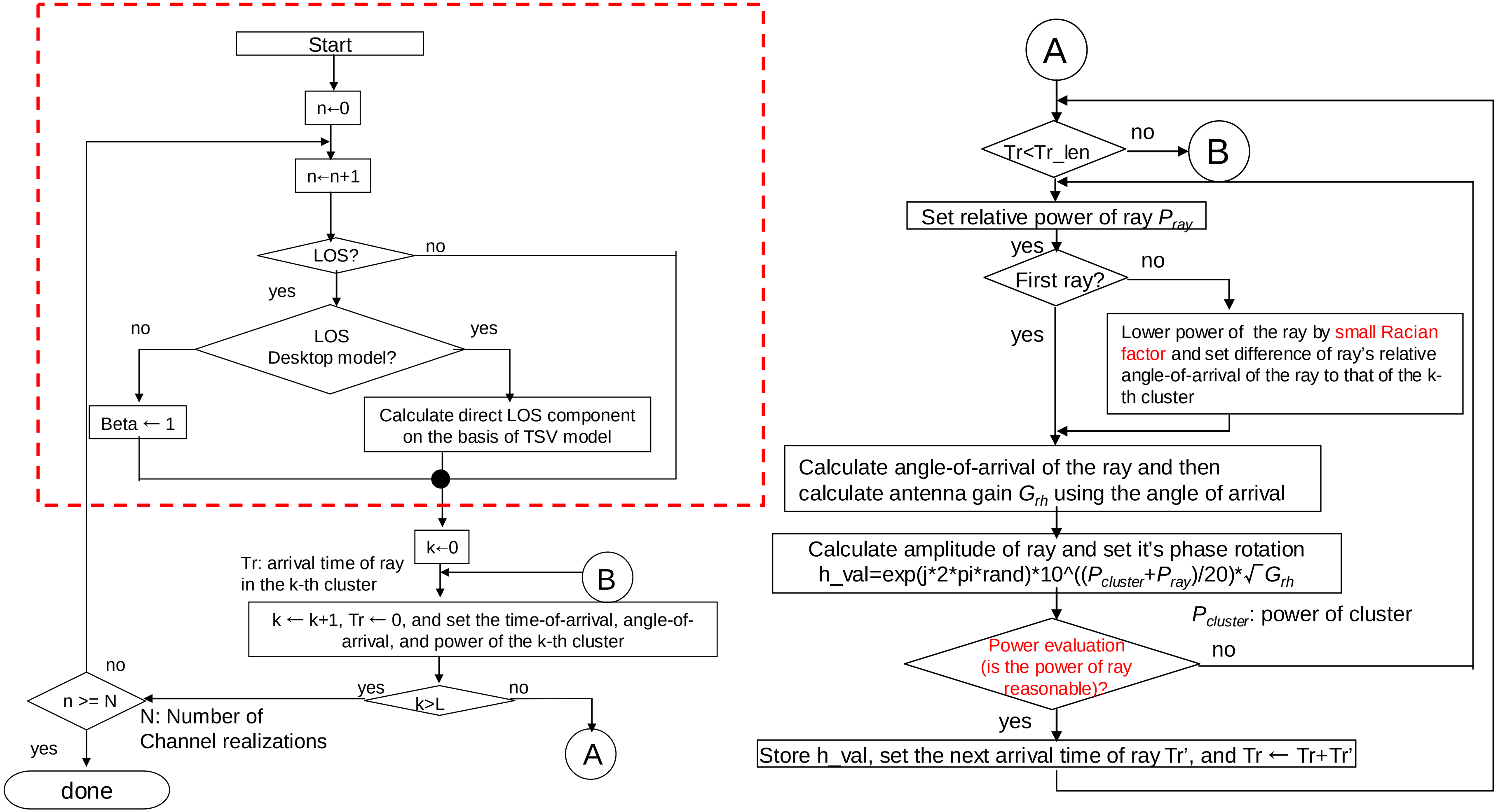
tg3c_tsv_ct_pre_fin_rev2.m

- This function M-file generates continuous impulse responses on the basis of TSV model
- This function consists of three sub-functions
  - tg3c_tsv_beta_calc_pre_fin_rev2.m
  - laplace_gen.m
  - calc_ant_gain_pre_fin_rev2.m

Modified flowchart of tg3c_tsv_ct_pre_fin_rev2.m



Modified flowchart of tg3c_tsv_ct_pre_fin_rev2.m



```
function [beta,sv_h,sv_aoa,t,t0,np] = tg3c_tsv_ct_pre_fin_rev2(...
   nlos, num_channels,...                         % Channel params
   adist, fc, LOS_desktp_flg, L_pl,...            % T-S-V model params
   Lam, lambda, Gam, gamma, std_ln_1, std_ln_2, ...    % SV model params
   Lmean, Omega0, smallk, sigma_fai,...
   tx_hpang, rx_hpang, ant_type)                  % Antenna gain params

% Arguments:
% nlos          : Flag of NLOS environment                        % num_channels     : Number of channel realizations
% Lam           : Cluster arrival rate (clusters per nsec)        % lambda           : Ray arrival rate (rays per nsec)
% Gam           : Cluster decay factor (time constant, nsec)      % gamma            : Ray decay factor (time constant, nsec)
% std_ln_1      : Standard deviation of log-normal variable for cluster fading
% std_ln_2      : Standard deviation of log-normal variable for ray fading
% Lmean         : Number of Average arrival clusters

% New paraemters added by NICT
% fc            : Carrier frequency [Hz]     % LOS_desktp_flg   : Flag used for beta calculation
% L_pl          : path loss regarding LOS component     % Omega0     : Cluster attenuation power level
% smallk        : Small Rician effect     % sigma_fai     : Cluster arrival angle spread in deg
% tx_hpang      : TX half-power angle in deg     % rx_hpang     : RX half-power angle in deg
% ant_type      : Antenna pattern     %   1: Simple Gaussian distribution

% Output values:
%   sv_h    : Continuous impulse responses of SV clusters (h in 154a_chmodel_v9)
%   t_ct    : Time of arrival of sv_h_ct
%   t0      : Arrival time of the first ray of the first SV cluster
%   np      : Number of paths of SV clusters
% New output values
%   beta    : Impulse response of the LOS component
%   sv_aoa  : Angle of arrival of each impulse response of SV clusters
```

```
%****************** Initialize and precompute some things ******************
std_L   = 1/sqrt(2*Lam);    % std dev (nsec) of cluster arrival spacing
std_lam = 1/sqrt(2*lambda); % std dev (nsec) of ray arrival spacing

%************************** Simulation preparation **************
h_len = 1000;  % there must be a better estimate of # of paths than this
ngrow = 1000; % amount to grow data structure if more paths are needed

%Output variables
beta    = zeros(1,num_channels);
sv_h    = zeros(h_len,num_channels); % renamed from h
sv_aoa  = zeros(h_len,num_channels);  % add by this document
t       = zeros(h_len,num_channels);
t0      = zeros(1,num_channels);
np      = zeros(1,num_channels);
```

Constant value to calculate time of arrival of cluster and ray in each cluster

Initial number of array to store results is 1000 and the number will increased with the unit of 1000.

Red parts are originally added by this document

```
for k = 1:num_channels     % loop over number of channels

    tmp_h = zeros(size(sv_h,1),1);
    tmp_t = zeros(size(sv_h,1),1);
    tmp_aoa = zeros(size(sv_h,1),1);    %added by this document

    %Set the number of generated clusters
    %L is the total number of clusters (TBD, see pps 9 and 10 in IEEE 15-06-0195-03-003c)
    %L = max(1, poissrnd(Lmean));       % number of clusters
    L = Lmean;          % added by this document

    %Initialize counter regarding the number of rays in SV clusters
    path_ix = 0;
```

Temporary array to store AOA

Number of clusters are decided by the modeling of TSV

Counter to calculate the number of paths

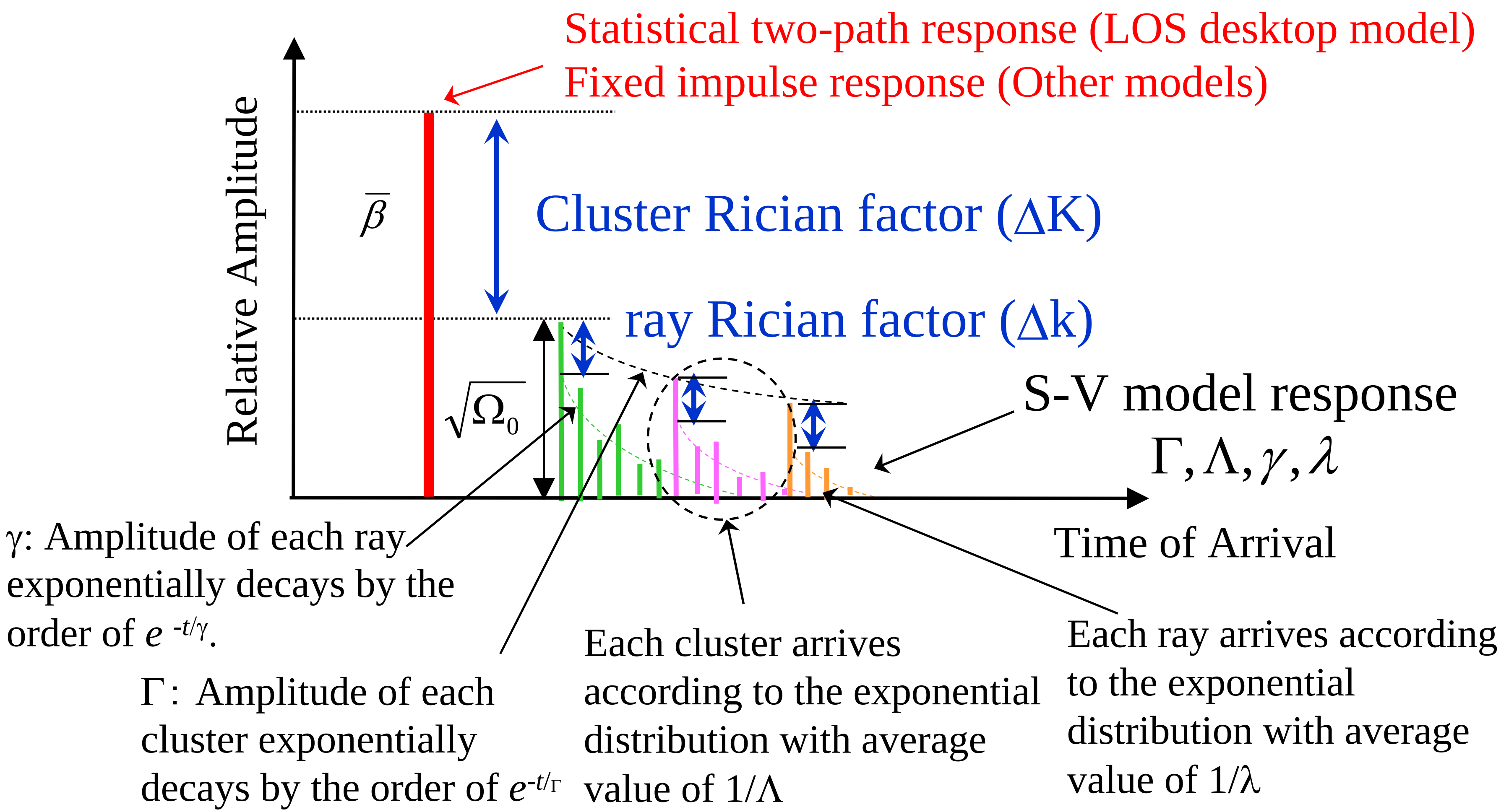
When LOS_desktp_flg =1, beta will be calculated.

```
%The following lines are added by this document
  if nlos==0  % LOS TSV model
     if  LOS_desktp_flg == 1 % Desktop model
        % Compute LOS component on the basis of TSV model
        [beta0] = tg3c_tsv_beta_calc_pre_fin_rev2(fc, adist, tx_hpang, rx_hpang, ant_type);
        % beta0 is multiplied by LOS path loss
        beta(k)=beta0;
     else  % The other LOS models
        % LOS path loss
        beta(k)=1;
     end
  else    % NLOS SV model
     % LOS component is set to zero (TBD)
     beta(k)=0;
  end
```

In the case of LOS residential and LOS office, $\beta = 1$.

In the case of NLOS, beta must be 0.

Impulse response of TSV model

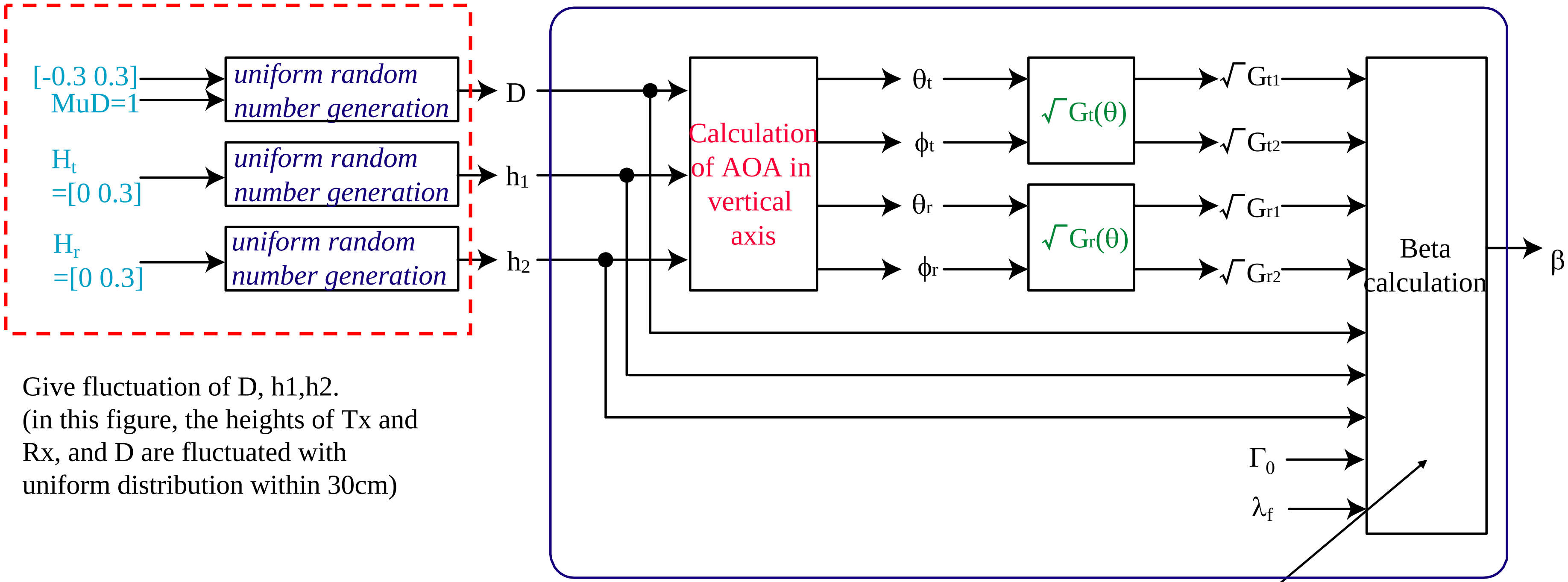


tg3c_tsv_beta_calc_pre_fin_rev1.m

- This function M-file computes beta on the accordance with the two-path theory of TSV model.

```
function [beta] = tg3c_tsv_beta_calc_pre_fin_rev1(fc, muD, tx_hpang, rx_hpang, ant_type)

%  Arguments:
% Parameters used for beta calculation
%   fs              : Center carrier frequency
%   muD             : Average distnace between TX and RX (TBD)
%   tx_hpang        : TX half-power angle in deg (horizontal and vartical gain are same)
%   rx_hpang        : RX half-power angle in deg (horizontal and vartical gain are same)
%   ant_type        : Types of antenna pattern
```

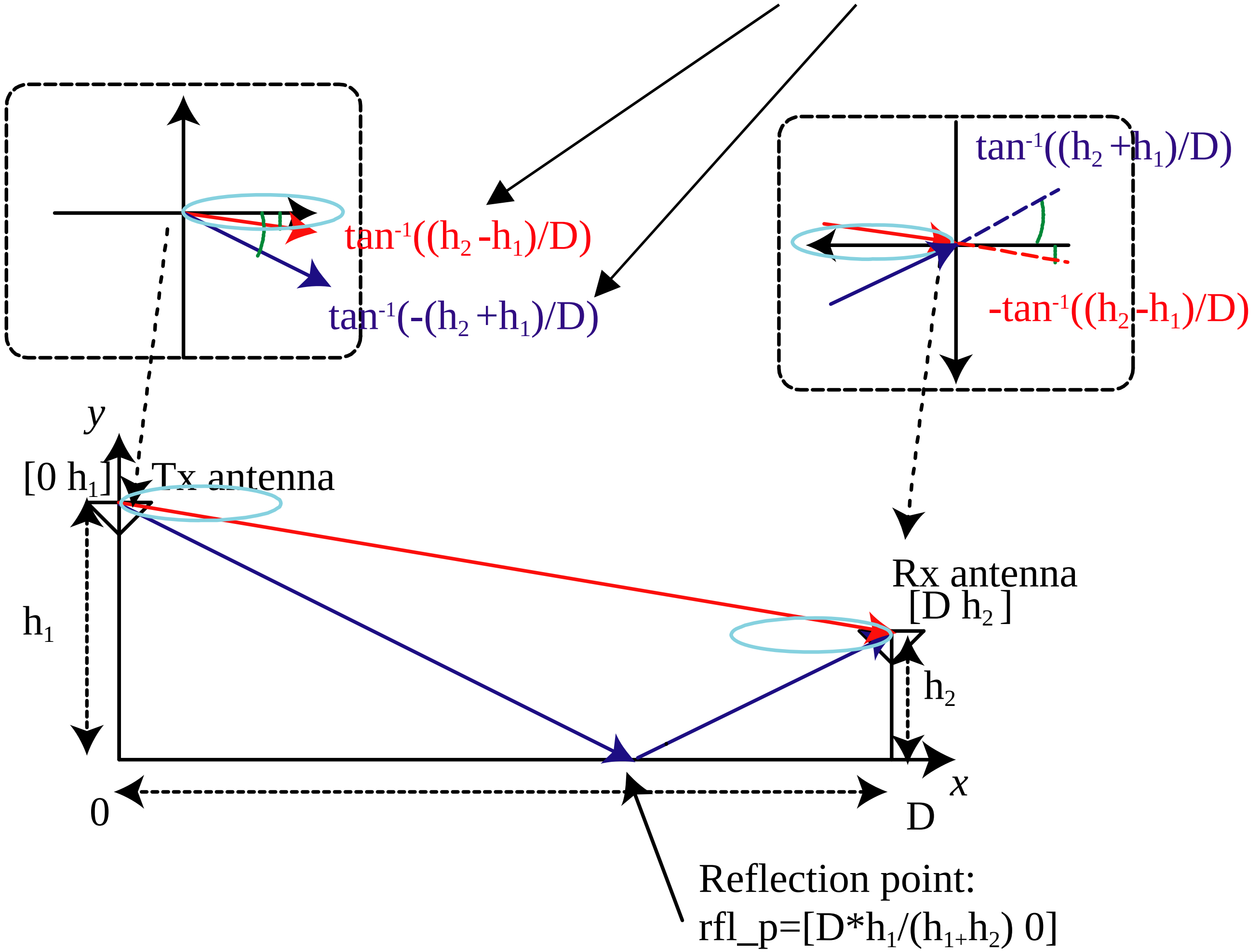
Block diagram to calculate β



Give fluctuation of D, h1,h2.
(in this figure, the heights of Tx and Rx, and D are fluctuated with uniform distribution within 30cm)

$$\beta = \left(\frac{\upsilon_D}{D}\right)\left|\sqrt{G_{t1}G_{r1}} + \sqrt{G_{t2}G_{r2}}\Gamma_0 \exp\left[j\frac{2\pi}{\lambda_f}\frac{2h_1h_2}{D}\right]\right|$$

Calculation method of θ_t, ϕ_t, θ_r, ϕ_r



```
%   gamma0      : Reflection coefficient
gamma0  = 1;    % Assuming angle of incidence is large

% these parameters will be discussed
D0      = [-0.3 0.3]+muD;                    % Range of D  (m)
Ht      = [0 0.3];                           % Range of Ht (m)
Hr      = [0 0.3];                           % Range of Hr (m)

% Determine TX and RX heights by the Monte-carlo method
h1  = (Ht(2)-Ht(1))*rand+Ht(1);
h2  = (Hr(2)-Hr(1))*rand+Hr(1);

% Determine distance between TX and RX by the Monte-carlo method
D   = (D0(2)-D0(1))*rand+D0(1);

% Wave length
ramda       = 3e8/fc;
```

Decide the fluctuation value of D and heights of Tx and Rx antennas

Heights are fluctuated by the order of uniform distribution

```
%********** Calculate the reflection point of the reflection path **********
tx_p    = i.*h1;
rx_p    = D+i.*h2;
rfl_p   = D*h1/(h1+h2);
```

Describe the position of tx and rx antennas and reflection point in vector

```
%************ Determine direction of direct and reflection paths ***********
tp      = angle([rx_p-tx_p (tx_p-rx_p) rfl_p-tx_p (rfl_p-rx_p)]);
tp      = tp./pi*180;

dr_theta    = tp(1);
dr_fai      = dr_theta;
rfl_theta   = tp(3);
rfl_fai     = -rfl_theta;
```

Decide angle of departure and angle of arrival in the horizontal axis.

Calculate θ_t, ϕ_t, θ_r, ϕ_r

```
%******************** Determine Antenna horizontal gain ********************
% TX-------------------
% Direct path
gt1 = calc_ant_gain_pre_fin_rev2(ant_type, tx_hpang, dr_theta);
% Reflection path
gt2 = calc_ant_gain_pre_fin_rev2(ant_type, tx_hpang, rfl_theta);

% RX-------------------
% Direct path
gr1 = calc_ant_gain_pre_fin_rev2(ant_type, rx_hpang, dr_fai);
% Reflection path
gr2 = calc_ant_gain_pre_fin_rev2(ant_type, rx_hpang, rfl_fai);

% calculation of sqrt is included in calc_ant_gain_pre_fin_rev2.m
beta    = (muD/D).*abs(gt1.*gr1+gt2.*gr2...
    .*gamma0.*exp(j.*(2*pi./ramda).*(2.*h1.*h2./D)
```

Decide square roots of gain of Tx and Rx antennas (in slide 23)

See the equation of beta calculation expressed in slide 23

calc_ant_gain_pre_fin_rev2.m

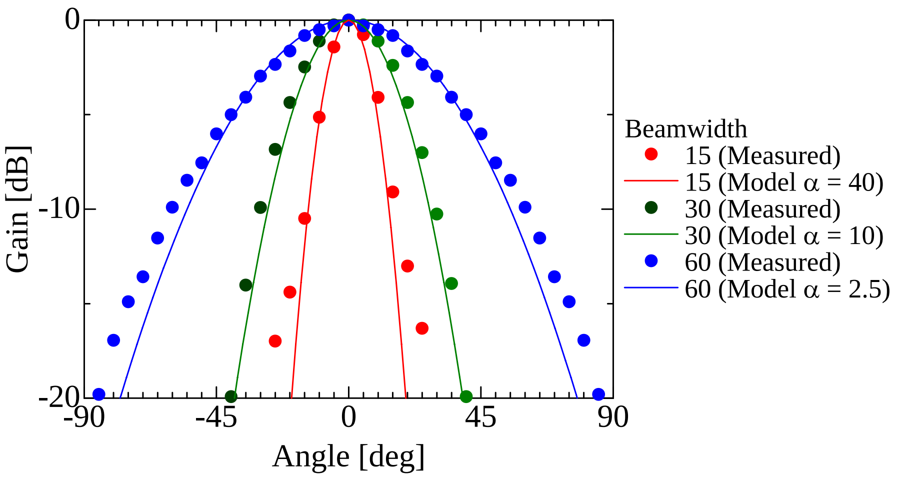
- This function outputs electric strength according to angle of arrival (AOA). The antenna pattern is determined according to antenna type.

```
function g = calc_ant_gain_pre_fin_rev2(ant_type, hpang, fai)
% Arguments
% ant_type: Antenna pattern  1: Simple Gaussian distribution
% hpang : Half-power angle in deg
% theta : Angle of arrival in deg

% Output value
% g     : Electric strength (True value)
```

```
switch hpang
    case 15
        alfa = 40;
    case 30
        alfa = 10;
    case 60
        alfa = 2.5;
end

switch ant_type
    case 1
        g   = sqrt(exp(-
      alfa*abs(fai/180*pi).^2));
    otherwise
        error('Antenna type error')
end
```

Antenna gain: $G_r(\theta,\phi) = G\,D(\theta,\phi)$

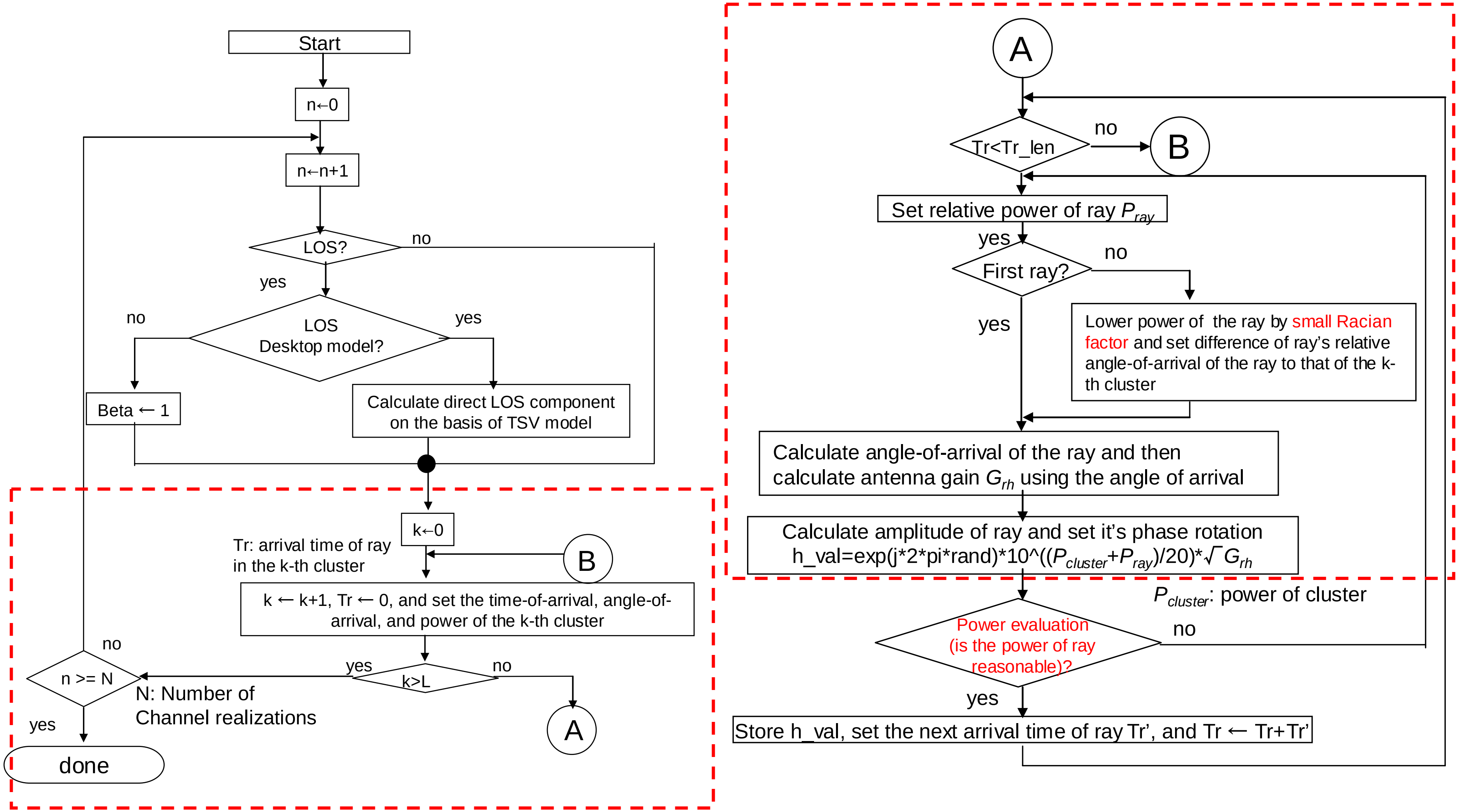
- Omni directional antenna: $D(0,\phi) = 1$
- Directional antenna: $D(0,\phi) = \exp(-\alpha\,\phi^2)$



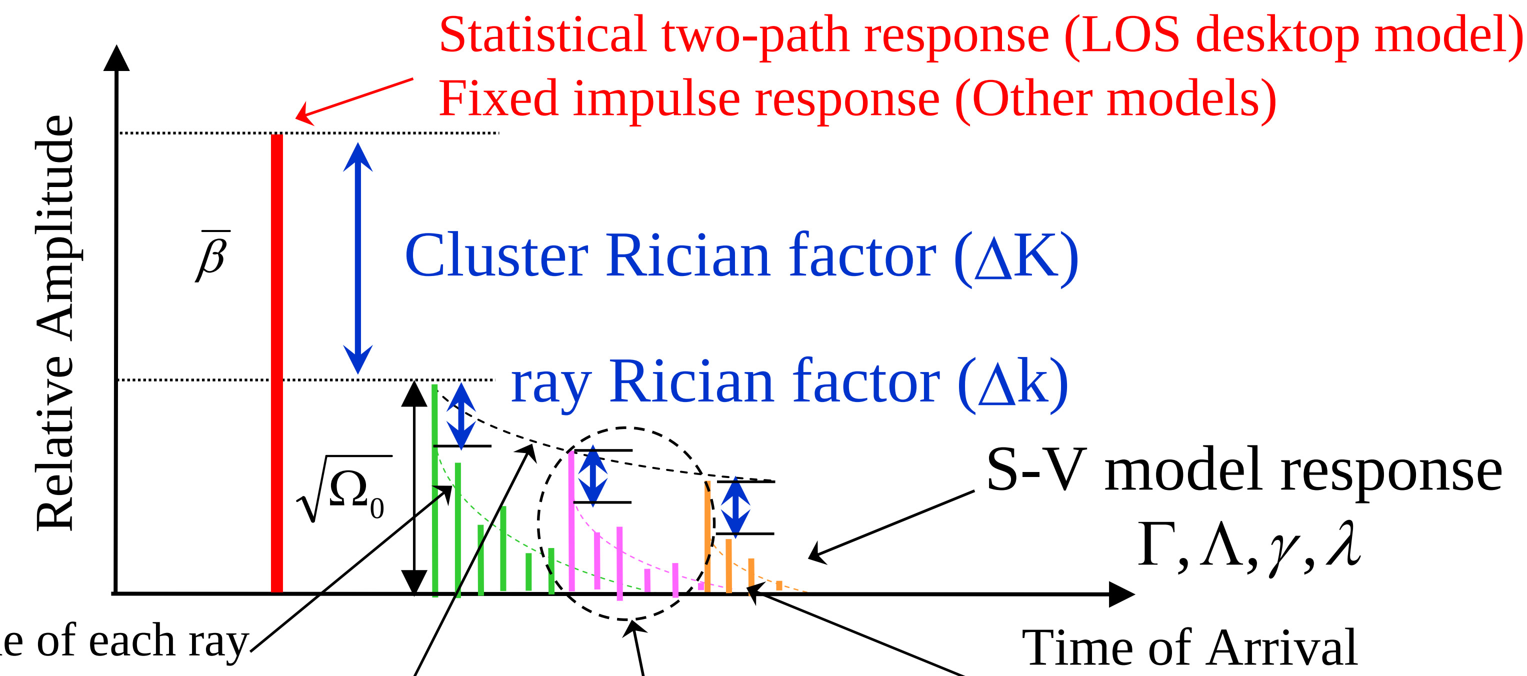
IEEE 802. 15-06-0427-01-003c

You can input your own antenna pattern!!

Modified flowchart of tg3c_tsv_ct_pre_fin_rev2.m



Impulse response of TSV model



γ: Amplitude of each ray exponentially decays by the order of $e^{-t/\gamma}$.

Γ: Amplitude of each cluster exponentially decays by the order of $e^{-t/\Gamma}$

Each cluster arrives according to the exponential distribution with average value of 1/Λ

Each ray arrives according to the exponential distribution with average value of 1/λ

```
%************************** SV cluster computation **************************
   % Determine time-of-arrival and angle-of-arrival of the fisrt SV cluster
   Tc             = (std_L*randn)^2 + (std_L*randn)^2;

   %added by this document
   %Angle-of-arrival of clusters is distributed according to uniform distribution
   cl_ang_deg      = 360*rand-180;

   if  nlos == 1
      t0(k)       = Tc;
   end

   % delta K factor
   dK  = Omega0-L_pl;  %added by this document
   Tc0 = Tc;
```

Decide time of arrival of cluster by using Poisson distribution same as 15.3a and 4a

Calculate AOA of the first cluster. The angle is uniformly distributed from -180 to 180 degree.

In the case of NLOS, the first arrival time of ray is stored for the display.

Calculate Rician factor(dK)

To generate rays is processed cluster by cluster

```
for ncluster = 1:L
    % First ray arrival defined to be time 0 relative to cluster
    Tr        = 0;

    %added by this document
    %Flag: fray = 1 when it is the first ray in a cluster
    fray        = 1;

    Mcluster = std_ln_1*randn;

    %Pcluster = 10*log10(exp(-1*Tc/Gam))+Mcluster; % total cluster power

    %added by this document
    %The first ray of the first cluster are related to delta K factor
    Pcluster = (dK-10*(Tc-Tc0)/Gam./log(10))+Mcluster;
```

In each cluster, the TOA of the first lay is 0.

In the case of first ray, flag is used.

The power of cluster is distributed by log-normal distribution with variance of std_ln_1^2 and mean of (dK-10*(Tc-Tc0)/Gam./log(10)). The first ray is dK because Tc=Tc0.

```
Tr_len = 10*gamma;
    while (Tr < Tr_len),
      t_val = (Tc+Tr);  % time of arrival of this ray

      %------------------------------------------------------------------------------------------
      %The following lines are added by this document
      % Momo: first ray of the first cluster is set to the mean of the cluster.
      if fray == 1    % AOA = cluster arival angle (first ray in each cluster)
        ray_ang_deg = cl_ang_deg;
      else            % AOA = cluster arival angle + ray arival angle
        % Recalculate if angle of arrival is more than 180 or less than -180
        while 1
          % Determine AOA of the ray employing Laplace distribution in deg
          ray_ang_deg0 = laplace_gen(sigma_fai); % average is 0 deg
          if abs(ray_ang_deg0) <= 180
            break;
          end
        end
        ray_ang_deg = cl_ang_deg+ray_ang_deg0;
      end

      aoa_val = exp(j.*ray_ang_deg./180*pi);
      ray_ang_deg2 = angle(aoa_val)/pi*180;
```

The TOA of ray is calculated until Tr is larger than Tr_len(10*gamma)

Calculated TOA of ray

The angle of first ray is same as the angle of cluster

The angles of the other lay is Laplace distributed and the center is AOA of cluster.

If the angle is larger than +- 180 degree, the angle is re-generated.

Calculated AOA of ray

If AOA is larger than +-180 degree, the AOA is converted to within +- 180 degree.

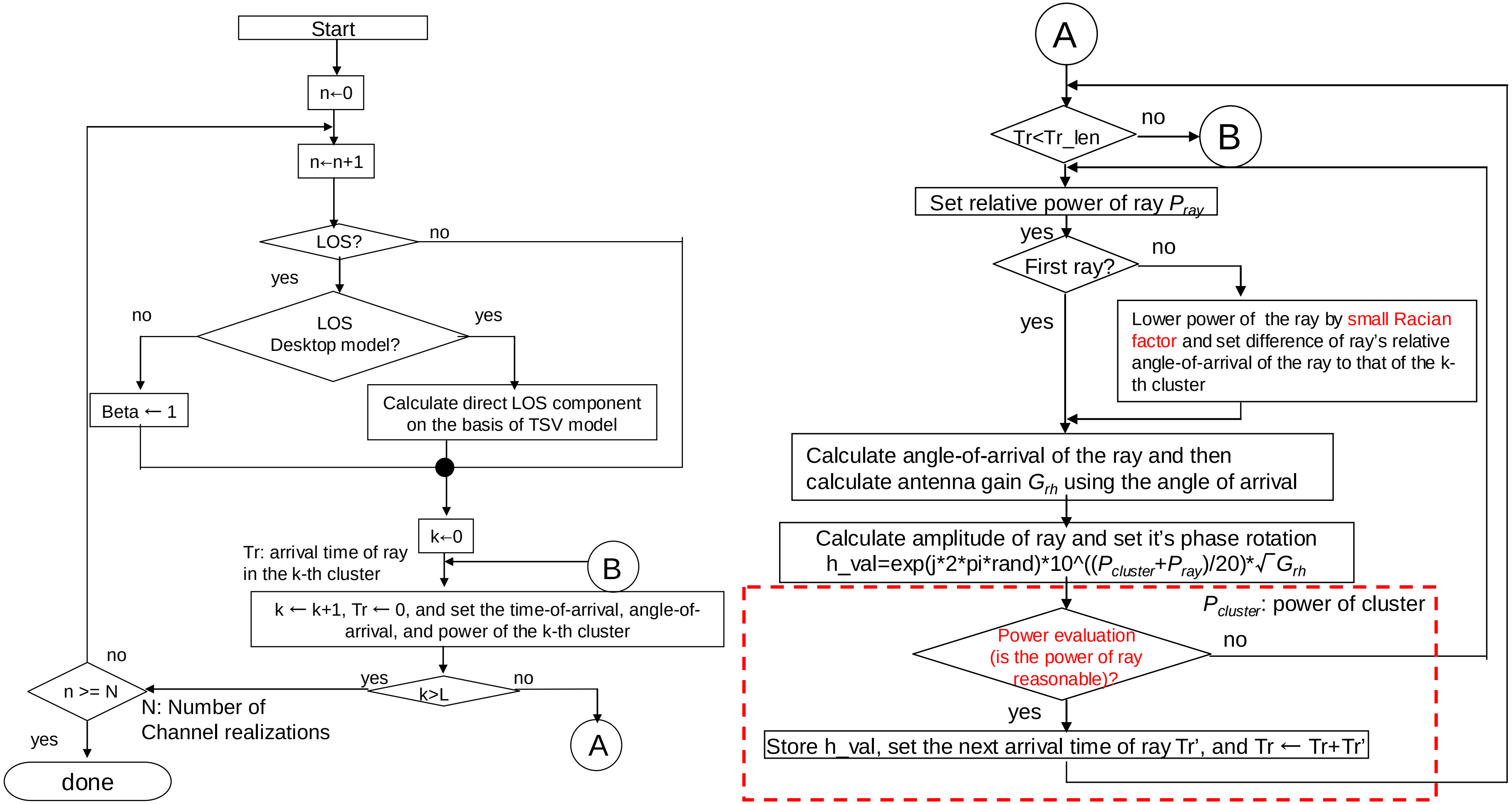
```
% Determine antenna horizontal gain
        tGrh    = calc_ant_gain_pre_fin_rev2(ant_type,rx_hpang, ray_ang_deg2);

        Mray = std_ln_2*randn;
        if fray == 1 %First ray of a cluster
           %Pray = 10*log10(exp(-Tr/gamma))+Mray;
           Pray = Mray;       %Tr = 0 if small_dk = 0
           % Set flag to be 0 after the first-ray's power calculation
           fray=0;
        else
           % Convert the base of small Racian facter
           small_dk = smallk.*10*log10(exp(1));
           Pray = -10*Tr/gamma./log(10)-small_dk+Mray;
           %Pray=10*log10(exp(-Tr/gamma))-small_dk+Mray;
        end

        pk = exp(j*2*pi*rand);
        % h_val = phase rotation x amplitude x electric strength of antenna
        h_val = pk *10^((Pcluster+Pray)/20)*tGrh;
```

By using AOA of ray, the square root of antenna gain is calculated.

Pray: power of lay, fray:flag (1:first lay)

The power of ray is distributed by log-normal distribution with variance of std_ln_2^2 and mean of 10*log10(exp(-Tr/gamma))-small_dk.

Modified flowchart of tg3c_tsv_ct_pre_fin_rev2.m



Problems of generated rays in Matlab code

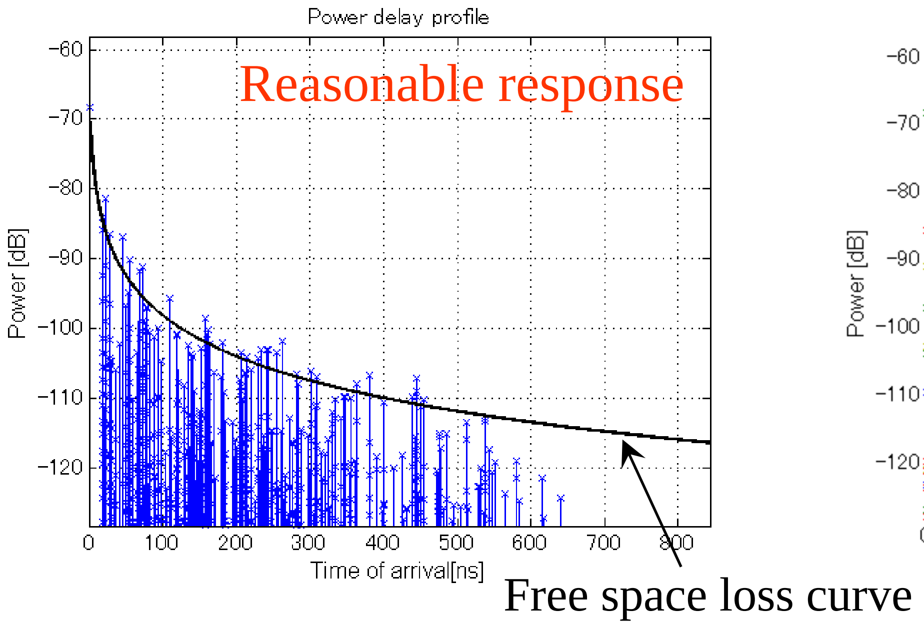


Fig.1 Snapshot of generated delay profile using LOS office parameters

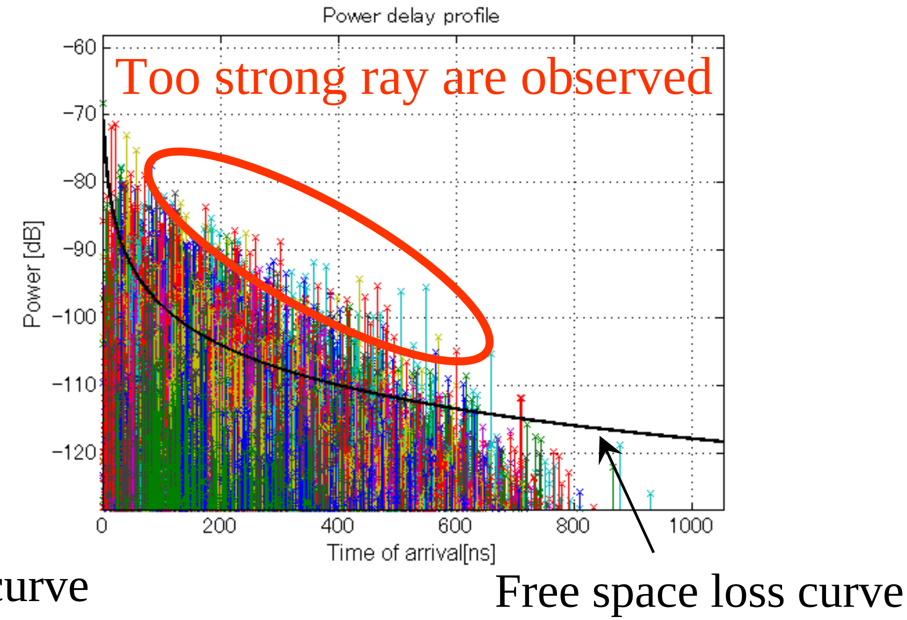


Fig.2 Realization of 200 generated delay profile using LOS office parameters

- Current version Matlab code generates too strong rays due to Log-normal distribution assumption for its amplitude model
- Any limitation should be processed for the ray amplitude by according to measurement results
- =>Regenerate if power of ray before the multiplication by antenna gain is higher than free space loss + 6dB

```
% Compute free space path loss of the ray from its time of arrival
        ramda   = 3e8/fc;
        Dt      = adist+0.3*t_val; %Travel distance of the ray in SV clusters (meter)

        % Free space path loss at Dt [m]
        Pdt     = 20*n_pl*log10(ramda/(4*pi*Dt));

% In this section, in some case, the situation that four waves are coherently added is  envisaged.
        pl_th = 10*log10(4);
```

Calculate free space loss at Dt m. In normal case, the level of reflected waves is less than the free space loss. But in some cases, several waves are coherently added. The situation must be considered. In this program, the situation that four waves are coherently added is envisaged.

```
 % The following lines are performed if power of the ray are less than free space path loss + 6 dB
  if Pcluster+Pray+L_pl<(pl_th+Pdt),
      % The following lines are the same as that of 15.4a Matlab code except for some notes
      % Increment the number of paths
      path_ix = path_ix + 1;

      if path_ix > h_len,
         % grow the output structures to handle more paths as needed
         tmp_h = [tmp_h; zeros(ngrow,1)];
         tmp_t = [tmp_t; zeros(ngrow,1)];
         sv_h = [sv_h; zeros(ngrow,num_channels)];
         t = [t; zeros(ngrow,num_channels)];

         %Added by this document
         tmp_aoa = [tmp_aoa; zeros(ngrow,1)];
         sv_aoa = [sv_aoa; zeros(ngrow,num_channels)];
         h_len = h_len + ngrow;
      end
      tmp_h(path_ix) = h_val;      tmp_t(path_ix) = t_val;

      %Added by this document
      tmp_aoa(path_ix) = aoa_val;
        Tr = Tr + (std_lam*randn)^2 + (std_lam*randn)^2;
       end
 end
 % Set the time-of-arrivaltime-of-arrival and angle-of-arrival of the next cluster to be generated
 Tc = Tc + (std_L*randn)^2 + (std_L*randn)^2;
 cl_ang_deg = 360*rand-180;
end
```

Store if power of ray before the multiplication of antenna gain is lower than Free space loss + 6dB

The number of rays are counted up

If the number of arrays are fully occupied, 1000 of arrays are added to the old arrays.

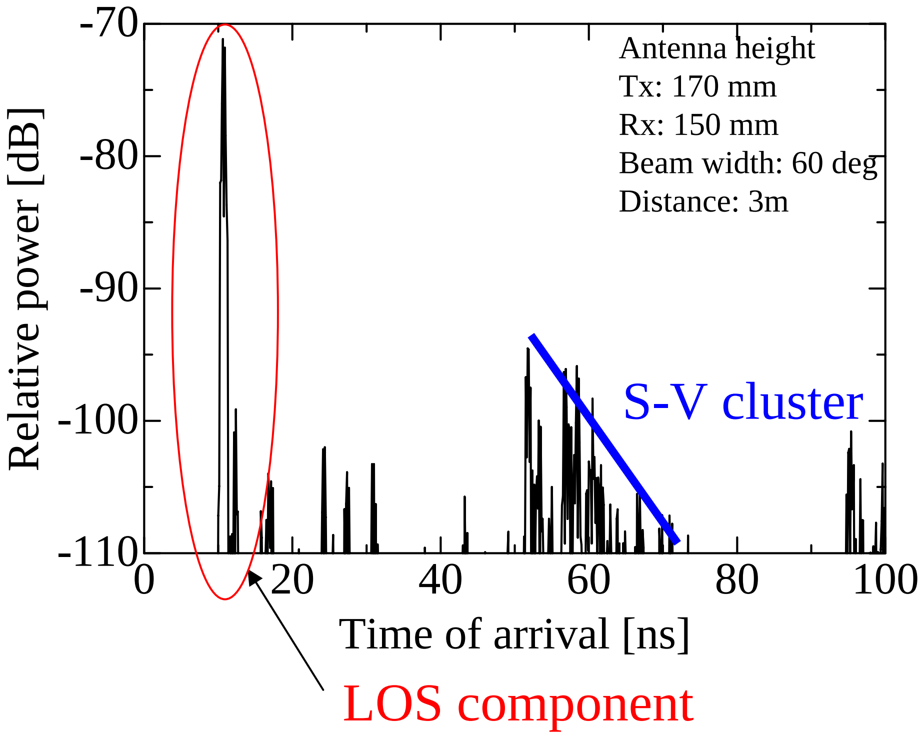
Save the impulse response of ray, TOA, AOA.

TOA of the next ray is set.

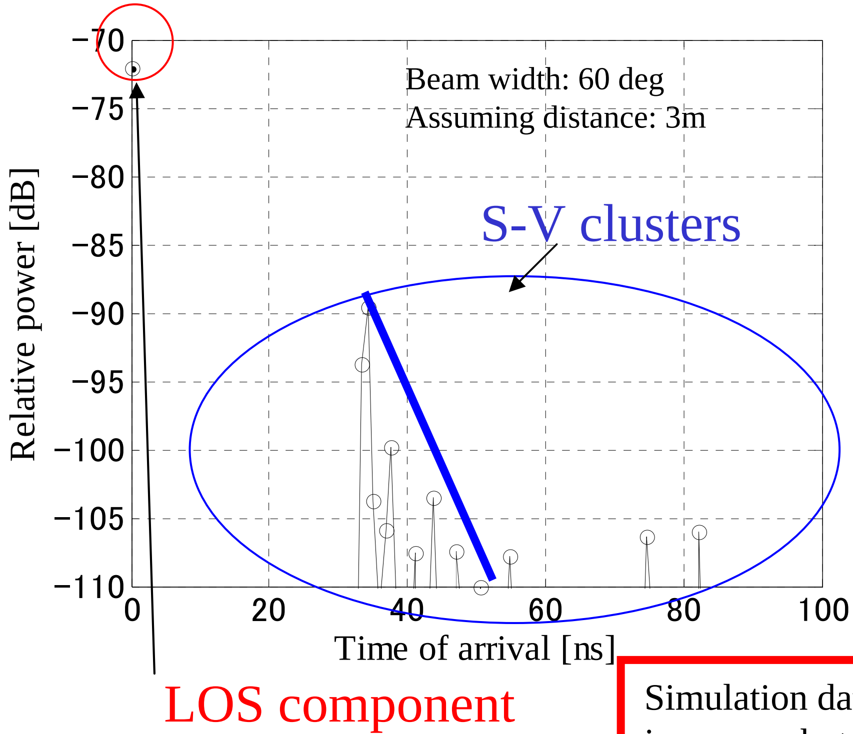
TOA and AOA of next cluter is set.

```
% The following lines are the same as that of 15.4a Matlab code except for some notes
   %******************************** Sorting ********************************
   np(k) = path_ix;                              % Number of rays (or paths) for this realization
   [sort_tmp_t,sort_ix] = sort(tmp_t(1:np(k)));   % sort in ascending time order
   t(1:np(k),k) = sort_tmp_t;
   sv_h(1:np(k),k) = tmp_h(sort_ix(1:np(k)));
   sv_aoa(1:np(k),k) = tmp_aoa(sort_ix(1:np(k)));    %Added by this document
   %Store the index of SV clusters attached to each ray
end
```

Comparison of experimental and simulated results



(a) Experimental result



(b) Simulation result

Simulation data is a snap-shot.

	Experimental results	Simulated results
Average RMS delay spread	10.6[ns]	7.9 [ns] (Dependent on the distribution of β and antenna pattern)

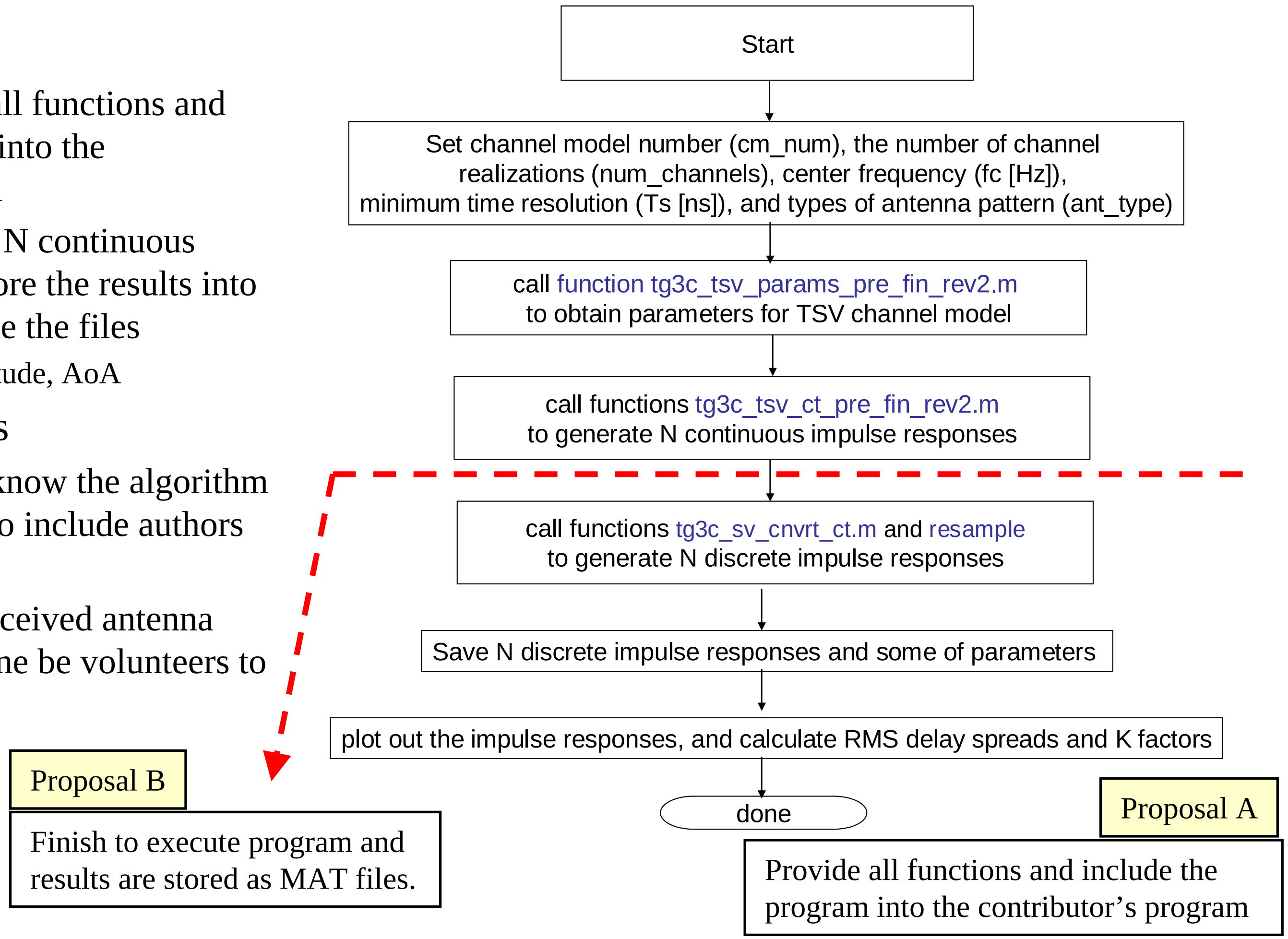
Summary of available LOS / NLOS channel models by MATLAB based TSV-channel model

	LOS	NLOS
Office	Available (NICT)	Available (NICT)
Residential	Available (NICT)	Available (NICT)
Desktop	Available (NICT)	N/A
Library	Available (IMST/Intel)	N/A

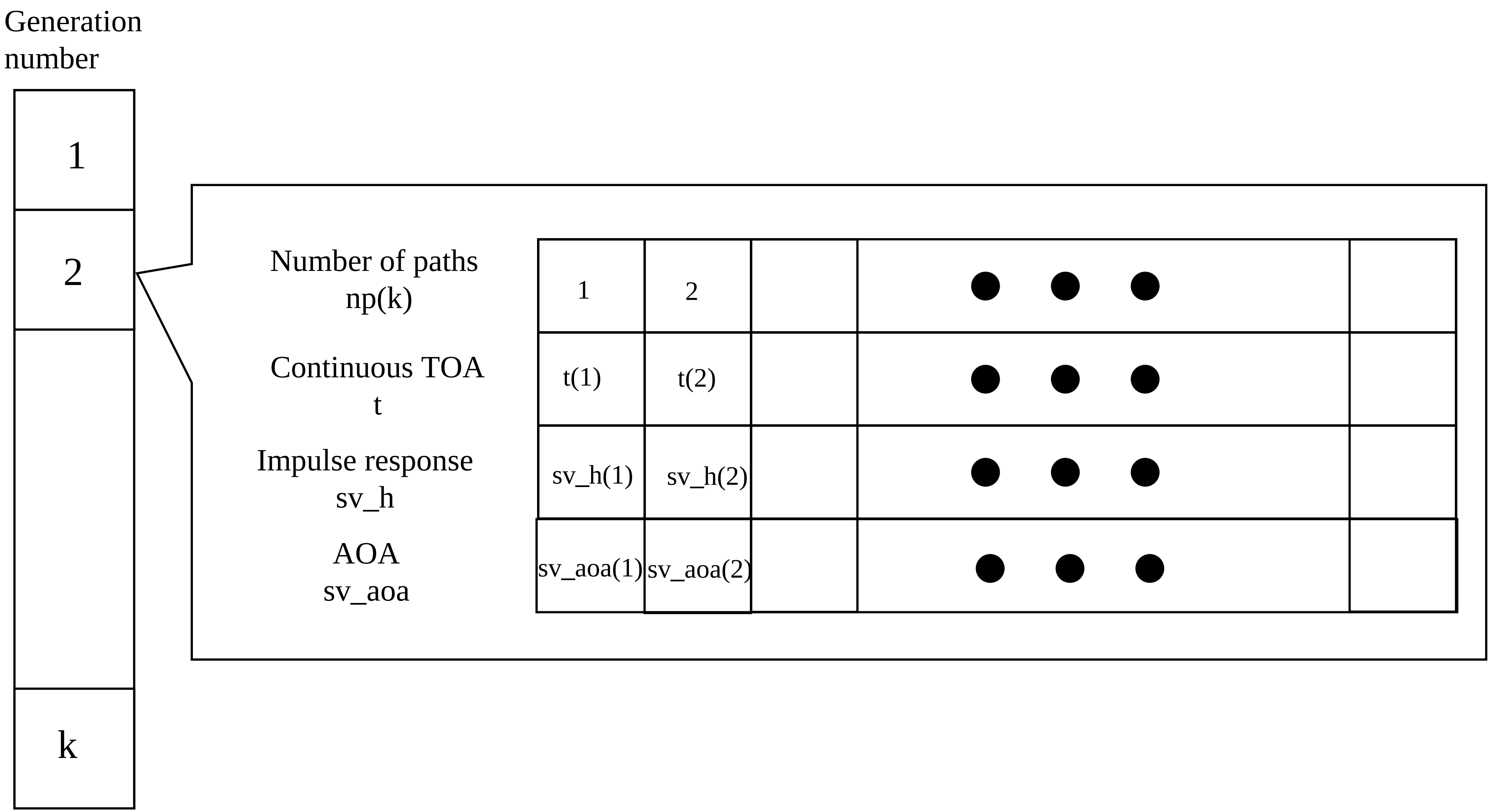
Measurement and analysis to get TSV parameters are finished by NICT. MATLAB program is now available by using analyzed parameters.

Recommendation of how to spread programs to the contributors to simulate system requirement

- Prepare two choices
  - Proposal A: Provide all functions and include the functions into the contributor's program
  - Proposal B: Generate N continuous impulse responses, store the results into MAT files and provide the files
    - arrival time, amplitude, AoA
- Problems in two choices
  - Proposal A: Have to know the algorithm completely and how to include authors antenna pattern
  - Proposal B: Define received antenna pattern and let someone be volunteers to make the MAT files



File format of MAT file



Summary

- Finished to prepare MATLAB simulation program for TSV-channel model
- Explained the flowchart of the MATLAB program
- Explained the detail of the program
- Showed comparison of experimental and simulated results
- Summarized available LOS / NLOS channel models by the MATLAB-based TSV channel model
- Showed recommendations of how to spread programs to the contributors to simulate system requirement

Appendix A: laplace_gen.m

- This function generates random values according to Laplace distribution as

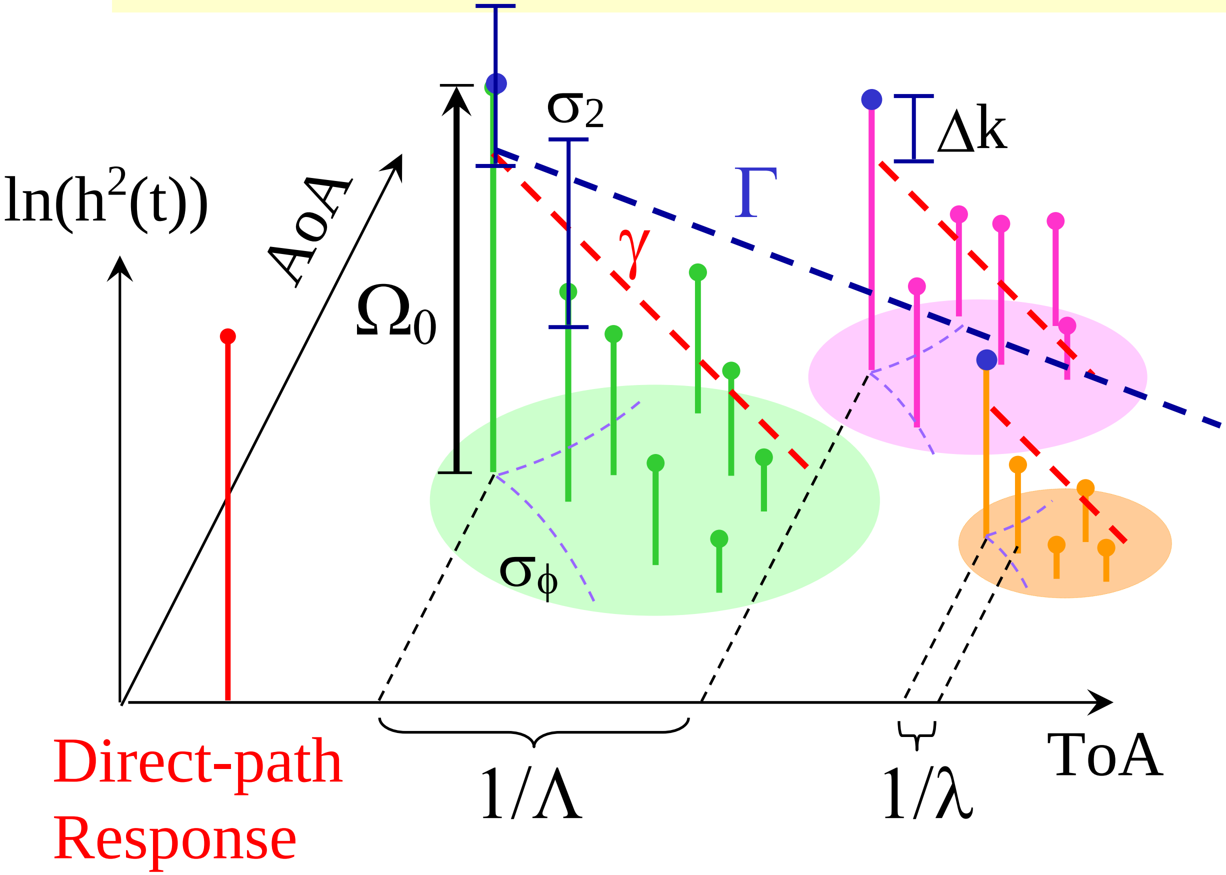
$$p(\theta) = \frac{1}{\sqrt{2}\sigma_\phi} e^{-\left|\sqrt{2}\theta/\sigma_\phi\right|}$$

```
function [out]=laplace_gen(a);

U1=rand;
U2=rand;
out=(2.*(U1>=0.5)-1).*(a./sqrt(2)).*log(U2);
```

Appendix B: tg3c_sv_cnvrt_ct.m

- The function converts continuous-time channel model h_ct to N-times over-sampled discrete-time samples convert continuous-time channel model h_ct to N-times oversampled discrete-time samples h_ct, t, np, and num_channels are as specified in uwb_sv_model ts is the desired time resolution hN will be produced with time resolution ts / N.
- It is up to the user to then apply any filtering and/or complex down-conversion and then decimate by N to finally obtain an impulse response at time resolution ts.

Appendix C: TSV model



Γ : *cluster* decay factor

$1/\Lambda$: *cluster* arrival rate

γ : *ray* decay factor

$1/\lambda$: *ray* arrival rate

σ_1 : *cluster* lognormal standard deviation

σ_2 : *ray* lognormal standard deviation

σ_ϕ : Angle spread of ray within cluster (Laplace distribution)

Ω_0 : Average power of the first ray of the first cluster

Small Rican factor Δk and Ω_0 are necessary for TSV model