General Path Loss Model:

Function for Free Space:

```
function PL=PL_free(fc,d,Gt,Gr)
% Free space path loss model

% Inputs-----> fc: carrier frequency[Hz] (1.5 GHz)
%              d: Distance between base station and mobile
station[m] (1km)
%              Gt/Gr: Transmitter/Receiver gain

% Outputs----> PL: Path loss[dB]

lamda= 3e8/fc;
tmp= lamda./(4*pi*d);
% if nargin>2
%     tmp=tmp*sqrt(Gt);
% end
% if nargin>3
%     tmp=tmp*sqrt(Gr);
% end
%PL=-20*log10(tmp); % Equation(1.3)
PL=-10*log10(Gt*Gr)-
20*log10(lamda)+20*log10(4*pi)+20*log10(d);
```

Function for Log-distance and Log-normal:

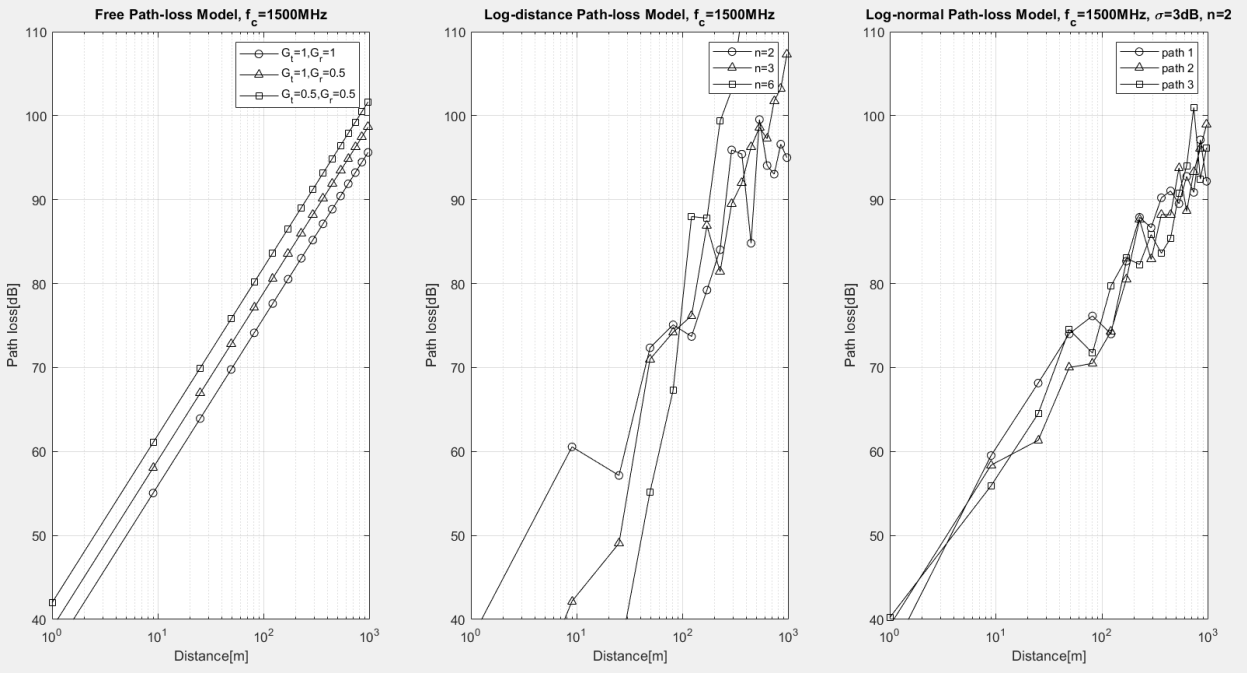
```
function PL=PL_logdist_or_norm(fc,d,d0,n,sigma)
% Log-distance or Log-normal shadowing path loss model


% Inputs-----> fc: carrier frequency[Hz] (1.5 GHz)
%              d: Distance between base station and mobile
station[m] (1km)
%              d0: Reference distance[m] (1km)
%              n: Path loss exponent (n=2 for free space)
%              sigma: Variance[dB]
lamda=3e8/fc;
```

```
PL=-20*log10(lamda/(4*pi*d0))+10*n*log10(d/d0); %
Equation(1.4)
if nargin>4
    PL=PL+sigma*randn(size(d)); % Equation(1.5)
end
```

Main code:

```
clear;
clf;

fc = 1.5e9;
d0 = 100;
sigma = 3;
distance = (1:2:31).^2;
Gt = [1 1 0.5];
Gr = [1 0.5 0.5];
Exp = [2 3 6];

for k = 1:3
    y_Free(k,:) = PL_free(fc, distance, Gt(k), Gr(k));
    y_logdist(k,:) = PL_logdist_or_norm(fc, distance, d0,
Exp(k),sigma);
    y_lognorm(k,:) = PL_logdist_or_norm(fc, distance, d0,
Exp(1), sigma);
end

subplot(131);
semilogx(distance, y_Free(1,:), 'k-o', distance,
y_Free(2,:), 'k-^', distance, y_Free(3,:), 'k-s');
grid on;
axis([1 1000 40 110]);
title(['Free Path-loss Model, f_c=', num2str(fc/1e6),
'MHz']);
xlabel('Distance[m]');
ylabel('Path loss[dB]');
legend('G_t=1,G_r=1', 'G_t=1,G_r=0.5', 'G_t=0.5,G_r=0.5');

subplot(132);
```

```
semilogx(distance, y_logdist(1,:), 'k-o', distance,
y_logdist(2,:), 'k-^', distance, y_logdist(3,:), 'k-s');
grid on;
axis([1 1000 40 110]);
title(['Log-distance Path-loss Model, f_c=',
num2str(fc/1e6), 'MHz']);
xlabel('Distance[m]');
ylabel('Path loss[dB]');
legend('n=2', 'n=3', 'n=6');

subplot(133);
semilogx(distance, y_lognorm(1,:), 'k-o', distance,
y_lognorm(2,:), 'k-^', distance, y_lognorm(3,:), 'k-s');
grid on;
axis([1 1000 40 110]);
legend('path 1', 'path 2', 'path 3');
title(['Log-normal Path-loss Model, f_c=', num2str(fc/1e6),
'MHz, \sigma=', num2str(sigma), 'dB, n=2']);
xlabel('Distance[m]');
ylabel('Path loss[dB]');
```



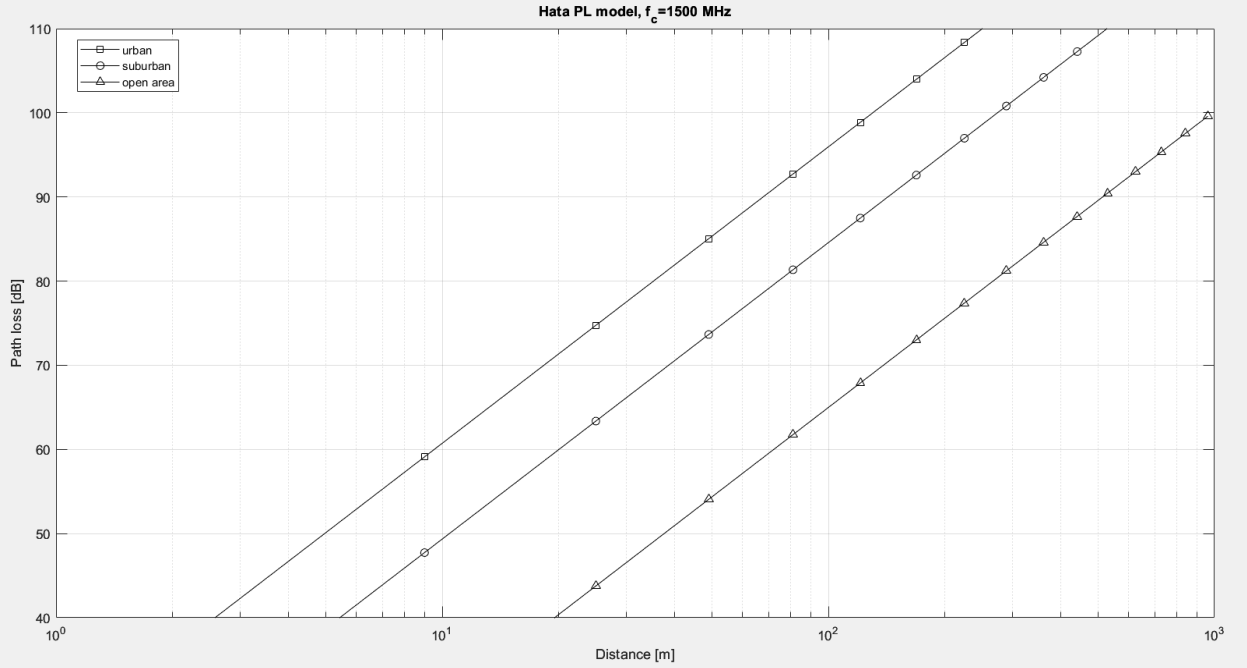
Okumura/Hata Model:

Function for PL_Hata:

```
function PL=PL_Hata(fc,d,htx,hrx,EType)

% input: fc--> Carrier frequency
%        d--> Distance between base station and mobile
station
%        htx--> Height of transmitter
%        hrx--> Height of receiver
%        EType--> Environment type (urban, suburban, open)
% Output: PL--> Path loss [dB]

if nargin<5
    EType='URBAN';
end
fc=fc/(1e6);

if fc>=150 && fc<=200
    C_Rx= 8.29*(log10(1.54*hrx))^2-1.1;
elseif fc>200
        C_Rx= 3.2*(log10(11.75*hrx))^2-4.97; % Equation
(1.9)
else
    C_Rx= 0.8+(1.11*log10(fc)-0.7)*hrx-1.56*log10(fc); %
Equation (1.8)
end

PL= 69.55+26.16*log10(fc)-13.82*log10(htx)-C_Rx+(44.9-
6.55*log10(htx))*log10(d/1000); % Equation (1.7)
EType= upper(EType);

if EType(1)=='S'
    PL=PL-2*(log10(fc/28))^2-5.4; % Equation (1.10)
elseif EType(1)=='O'
    PL=PL+(18.33-4.78*log10(fc))*log10(fc)-40.97; % Equation
(1.11)
end
```

Main Code:

```
% plot_PL_Hata.m
clc;
clear;
fc=1.5e9;
htx=30;
hrx=2;
distance=(1:2:31).^2;
y_urban= PL_Hata(fc,distance,htx,hrx,'urban');
y_suburban= PL_Hata(fc,distance,htx,hrx,'suburban');
y_open= PL_Hata(fc,distance,htx,hrx,'open');

% semilogx(distance,y_urban,'k-s',distance,y_suburban,'k-
o',distance,y_open,'k-^')
% title(['Hata PL model, f_c=',num2str(fc/1e6),'MHz'])
% xlabel('Distance[m]'),ylabel('Path loss[dB]')
% legend('urban','suburban','open area',2), grid on, axis([1
1000 40 110])

semilogx(distance, y_urban, 'k-s', distance, y_suburban, 'k-
o', distance, y_open, 'k-^')
title(['Hata PL model, f_c=', num2str(fc/1e6), ' MHz'])
xlabel('Distance [m]')
ylabel('Path loss [dB]')
legend('urban', 'suburban', 'open area', 'Location', 'best')
grid on
axis([1 1000 40 110])
```



IEEE 802.16d Model:

Function:

```
% IEEE 802.16

function PL=PL_IEEE80216d(fc,d,type,htx,hrx,corr_fact,mod)
    % IEEE 802.16d model
    % inputs---
    % fc --> carrier frequency
    % d ---> distance between base and terminal
    % type-> A,B,C
    % htx -> height of transmitter
    % hrx -> height of receiver
    % corr_fact: if shadowing exists, set to "ATnT' or
'Okumura', or 'no
    % mod: set to 'mod' to obtain modified IEEE 802.16d
model
      % this says if a modified version of this model is
used or not

    % outputs---
```

```
% PL: path loss [dB]

Mod='UNMOD';

if nargin>6
    Mod=upper(mod);
end

if nargin==6 && corr_fact(1)=='m'
    Mod='MOD';
    corr_fact='NO';
elseif nargin<6
    corr_fact='NO';
    if nargin==5 && hrx(1)=='m'
        Mod='MOD';
        hrx=2;
    elseif nargin<5
        hrx=2;
        if nargin==4 && htx(1)=='m'
            Mod='MOD';
            htx=30;
        elseif nargin<4
            htx=30;
            if nargin==3 && type(1)=='m'
                Mod='MOD';
                type='A';
            elseif nargin<3
                type='A';
            end
        end
    end
end

d0=100;
Type=upper(type);

if Type~='A' && Type~='B' && Type~='C'
    disp('Error: The selected type is not supported');
    return;
```

```
end
switch upper(corr_fact)
    case 'ATNT'
        PLf=6*log10(fc/2e9); % Cf
        PLh=-10.8*log10(hrx/2); % Crx
    case 'OKUMURA'
        PLf=6*log10(fc/2e9);
        if hrx<=3
            PLh=-10*log10(hrx/3);
        else
            PLh=-20*log10(hrx/3);
        end
    case 'NO'
        PLf=0;
        PLh=0;
end

if Type=='A'
    a=4.6;
    b=0.0075;
    c=12.6;
elseif Type=='B'
    a=4;
    b=0.0065;
    c=17.1;
else
    a=3.6;
    b=0.005;
    c=20
end

lamda=3e8/fc;
gamma=a-b*htx+c/htx;
d0_pr=d0;

if Mod(1)=='M'
    d0_pr=d0*10^(-(PLf+PLh)/(10*gamma));
end
```

```
    A=20*log10(4*pi*d0_pr/lamda)+PLf+PLh;

    for k=1:length(d)
        if d(k)>d0_pr
            PL(k)=A+10*gamma*log10(d(k)/d0);
        else
            PL(k)=20*log10(4*pi*d(k)/lamda);
        end
    end
```

Main Code:

```
% plot_PL_IEEE80216d.m
clear, clf, clc
fc=2e9; htx=[30 30]; hrx=[2 10]; distance=[1:1000];
for k=1:2

y_IEEE16d(k,:)=PL_IEEE80216d(fc,distance,'A',htx(k),hrx(k),'
atnt');

y_MIEEE16d(k,:)=PL_IEEE80216d(fc,distance,'A',htx(k),hrx(k),
'atnt', 'mod');
end
subplot(121),
semilogx(distance,y_IEEE16d(1,:),'k:','linewidth',1.5)
hold on, semilogx(distance,y_IEEE16d(2,:),'k-
','linewidth',1.5)
grid on, axis([1 1000 10 150])
title(['IEEE 802.16d Path-loss Model,
f_c=',num2str(fc/1e6),'MHz'])
xlabel('Distance[m]'), ylabel('Pathloss[dB]')
legend('h_{Tx}=30m, h_{Rx}=2m','h_{Tx}=30m,
h_{Rx}=10m','Location','southeast')
subplot(122),
semilogx(distance,y_MIEEE16d(1,:),'k:','linewidth',1.5)
hold on, semilogx(distance,y_MIEEE16d(2,:),'k-
','linewidth',1.5)
grid on, axis([1 1000 10 150])
```

```
title(['Modified IEEE 802.16d Path-loss Model, f_c=',
num2str(fc/1e6), 'MHz'])
xlabel('Distance[m]'), ylabel('Pathloss[dB]')
legend('h_{Tx}=30m, h_{Rx}=2m','h_{Tx}=30m,
h_{Rx}=10m','Location','southeast')
```