Simulation of 5G mmWave and 6G THz Links with Beamforming

Objective:

1. Simulate 5G mmWave and 6G THz links to compute beamforming gain and path loss.
2. Calculate received power and SNR to evaluate link quality and performance.
3. Visualize antenna array beam patterns to analyze directional gain and beamforming effectiveness.

Theory:

1. mmWave and THz Propagation

5G mmWave (24–100 GHz) and 6G THz (0.1–10 THz) suffer from high path loss, atmospheric absorption, and blockage, requiring highly directional transmissions for reliable communication.

2. Beamforming Principle

Antenna arrays steer narrow beams toward the receiver using constructive interference, enhancing directional gain and compensating for severe path loss at high frequencies

3. Massive MIMO Arrays

Large antenna arrays at mmWave/THz frequencies enable high beamforming gain, spatial multiplexing, and improved link robustness by focusing power in specific directions.

4. Received Power and SNR Dependence

Received power scales with beamforming gain and decreases with free-space path loss; SNR determines achievable data rate and link quality across mmWave and THz bands..

5. Beam Pattern Characteristics

Beam patterns illustrate main lobe direction, side-lobe levels, and beamwidth, showing how effectively the beamforming system concentrates energy toward the target receiver.

Procedure:

1. Define system parameters: carrier frequency, distance, bandwidth, number ofantennas, transmit power.
2. Compute beamforming gain for the antenna arrays:
3. GdB = 10 log10(Nant)
4. Calculate path loss using Friis equation.
5. Calculate received power at the receiver.
6. Plot beam patterns for 5G mmWave and 6G THz arrays.
7. Optionally, calculate SNR using a predefined noise power.
8. Record all calculated values in the observation table

Code:

```
% mmwave_thz_link_sim.m
% Single-file simulator for 5G mmWave and 6G THz links with beamforming,
path loss,
% received power and SNR calculation, and beam pattern visualization.
% Author: ChatGPT (example educational simulator)
clear; close all; clc;

%% --------------------
%  User parameters (change as needed)
%% --------------------
% General
c = 3e8;                    % speed of light (m/s)
k_B = 1.380649e-23;             % Boltzmann constant

% Choose frequencies (Hz)
f_mm = 28e9;                  % mmWave carrier (28 GHz typical)
f_thz = 300e9;               % THz carrier (300 GHz). Change to 1e12 for 1 THz

% Bandwidths
BW_mm = 400e6;                 % 400 MHz
BW_thz = 1e9;                % 1 GHz (example)
```

```
% TX / RX power and antenna arrays
Pt_dBm = 20;                  % Transmit power in dBm (per TX)
Pt = 10^((Pt_dBm-30)/10);       % W

txArraySize_mm = 16;           % number of tx elements (ULA) for mmWave
rxArraySize_mm = 16;           % number of rx elements for mmWave

txArraySize_thz = 64;          % larger array for THz
rxArraySize_thz = 64;

% Element spacing factor (relative to wavelength)
d_factor = 0.5;                % element spacing = d_factor * lambda

% Steering angles (degrees)
tx_steer_deg = 0;              % transmit steering azimuth (broadside = 0)
rx_steer_deg = 0;

% Distances (m)
dist_mm = 100;                 % distance for mmWave link (meters)
dist_thz = 10;                 % distance for THz link (meters) - THz usually short-
range

% Antenna element gain (per element) in dBi (if element has inherent gain)
elemGain_dBi = 2;              % small patch element gain

% Receiver noise figure (dB)
NF_dB = 7;                     % receiver noise figure

% Additional losses (dB)
miscLoss_dB = 2;               % connector, polarization, implementation loss

% Molecular absorption flag and simple coefficient (dB/m)
apply_absorption_thz = true;
% Example absorption coefficients (dB/m) — very rough placeholders. Replace
with measured data for accuracy.
abs_coeff_thz = 0.01;          % dB/m at 300 GHz (example)
abs_coeff_mm = 1e-6;           % negligible for mmWave

%% --------------------
% Derived parameters & helper functions
```

```
%% --------------------
to_dB = @(x) 10*log10(x);
from_dB = @(x) 10.^(x/10);

% Noise power
T0 = 290;                   % Kelvin

%% --------------------
% --- Simulation for a link function
%% --------------------
simulate_link = @(f, BW, txN, rxN, dist, abs_coeff, label) ...

simulate_one_link(f,BW,txN,rxN,dist,abs_coeff,label,c,k_B,T0,Pt_dBm,elemGain
_dBi,...

NF_dB,miscLoss_dB,d_factor,tx_steer_deg,rx_steer_deg,to_dB,from_dB);

% Run both links
results_mm = simulate_link(f_mm, BW_mm, txArraySize_mm, rxArraySize_mm,
dist_mm, abs_coeff_mm, 'mmWave 28 GHz');
results_thz = simulate_link(f_thz, BW_thz, txArraySize_thz, rxArraySize_thz,
dist_thz, abs_coeff_thz, 'THz 300 GHz');

%% --------------------
% --- Display summary
%% --------------------
fprintf('\n==== SUMMARY ====\n');
print_result(results_mm);
print_result(results_thz);

%% --------------------
% --- Beam patterns visualization (compare)
%% --------------------
figure('Name','Transmit Beam Patterns (Linear scale)');
subplot(2,1,1);
plot(results_mm.theta_deg, results_mm.txPattern_dB, 'LineWidth',1.4);
title('mmWave TX Array Pattern (dB)');
xlabel('Azimuth (deg)'); ylabel('Array pattern (dBi)');
grid on; xlim([-90 90]);
```

```
subplot(2,1,2);
plot(results_thz.theta_deg, results_thz.txPattern_dB, 'LineWidth',1.4);
title('THz TX Array Pattern (dB)');
xlabel('Azimuth (deg)'); ylabel('Array pattern (dBi)');
grid on; xlim([-90 90]);

% Polar plots for main lobes
figure('Name','Polar Beam Patterns (linear magnitude)');
subplot(1,2,1);
polarplot(results_mm.theta_rad, results_mm.txPattern_linear);
title('mmWave TX Array (linear)');
subplot(1,2,2);
polarplot(results_thz.theta_rad, results_thz.txPattern_linear);
title('THz TX Array (linear)');

%% --------------------
% --- Local function definitions
%% --------------------
function res = simulate_one_link(f, BW, txN, rxN, dist, abs_coeff, label,
c,k_B,T0,Pt_dBm,elemGain_dBi,NF_dB,miscLoss_dB,d_factor,tx_steer_deg,rx_st
eer_deg,to_dB,from_dB)
    lambda = c/f;
    % Element spacing
    d = d_factor*lambda;

    % Array gains (ideal array factor gain = N for coherent steering =>
10*log10(N))
    % But we'll compute full array pattern and max gain.
    % Element isotropic gain from element dBi
    elemGain_lin = from_dB(elemGain_dBi);

    % Tx & Rx full array max gains
    G_tx_max_dBi = 10*log10(txN*elemGain_lin);
    G_rx_max_dBi = 10*log10(rxN*elemGain_lin);

    % Compute array patterns (steered to tx_steer_deg)
    theta_deg = -90:0.1:90;
    theta_rad = deg2rad(theta_deg);
    % compute TX array factor (steering)
```

```
    [txPattern_dB, txPattern_linear] = array_pattern_ula(txN, d, lambda, theta_rad,
tx_steer_deg, elemGain_dBi);
    [rxPattern_dB, rxPattern_linear] = array_pattern_ula(rxN, d, lambda, theta_rad,
rx_steer_deg, elemGain_dBi);

    % Maximum array gain (use max of pattern + implementation losses)
    G_tx_max_dBi = max(txPattern_dB);
    G_rx_max_dBi = max(rxPattern_dB);

    % Path loss (Friis)
    L_fs_dB = friis_pathloss(f, dist, c);

    % Absorption loss (simple linear: abs_coeff in dB/m)
    L_abs_dB = abs_coeff * dist;

    % Total link loss (dB)
    TotalLoss_dB = L_fs_dB + L_abs_dB;

    % Received power calculation (dBm)
    Pr_dBm = Pt_dBm + G_tx_max_dBi + G_rx_max_dBi - TotalLoss_dB -
miscLoss_dB;

    % Noise floor (dBm)
    noiseP_W = k_B * T0 * BW;
    noiseP_dBm = to_dB(noiseP_W) + 30;  % convert W to dBm
    noiseP_dBm = noiseP_dBm + NF_dB;   % include noise figure

    % SNR (dB)
    SNR_dB = Pr_dBm - noiseP_dBm;
    SNR_linear = from_dB(SNR_dB);

    % Package results
    res.label = label;
    res.f = f;
    res.lambda = lambda;
    res.BW = BW;
    res.txN = txN; res.rxN = rxN;
    res.d = d;
    res.txPattern_dB = txPattern_dB;
    res.txPattern_linear = txPattern_linear;
```

```
    res.rxPattern_dB = rxPattern_dB;
    res.rxPattern_linear = rxPattern_linear;
    res.theta_deg = theta_deg;
    res.theta_rad = theta_rad;
    res.G_tx_max_dBi = G_tx_max_dBi;
    res.G_rx_max_dBi = G_rx_max_dBi;
    res.L_fs_dB = L_fs_dB;
    res.L_abs_dB = L_abs_dB;
    res.TotalLoss_dB = TotalLoss_dB;
    res.Pr_dBm = Pr_dBm;
    res.noiseP_dBm = noiseP_dBm;
    res.SNR_dB = SNR_dB;
    res.SNR_linear = SNR_linear;
    res.txPattern_dB = txPattern_dB;
end

function [pattern_dB, pattern_linear] = array_pattern_ula(N, d, lambda, theta_rad,
steer_deg, elemGain_dBi)
    % Returns array pattern in dBi (element gain included) and linear magnitude
(normalized)
    % Uniform amplitude, progressive phase steering to steer_deg.
    k = 2*pi/lambda;
    steer_rad = deg2rad(steer_deg);
    % Element positions (x axis)
    n = 0:N-1;
    psi = k * d * (sin(theta_rad.') - sin(steer_rad));  % column vector of theta
    AF = abs(sum(exp(1j*(n .* psi)),2));  % array factor magnitude
    AF = AF / max(AF);  % normalize to 1
    % Include element gain
    elemGain_lin = 10^(elemGain_dBi/10);
    pattern_linear = AF * sqrt(elemGain_lin);  % relative linear magnitude
    % Convert to dBi referencing maximum
    pattern_dB = 10*log10((pattern_linear./max(pattern_linear)).^2) +
10*log10(N*elemGain_lin);
    % Explanation: pattern_dB peaks at 10*log10(N*elemGain_lin) i.e. coherent
addition
end

function L_fs_dB = friis_pathloss(f, dist, c)
    lambda = c/f;
```

```
    L_fs = ((4*pi*dist)/lambda).^2;
    L_fs_dB = 10*log10(L_fs);
end

function print_result(r)
    fprintf('\n--- %s ---\n', r.label);
    fprintf('Frequency: %.3g Hz (lambda = %.3g m)\n', r.f, r.lambda);
    fprintf('Tx array elements: %d, Rx array elements: %d\n', r.txN, r.rxN);
    fprintf('Element spacing: %.3g m\n', r.d);
    fprintf('Max Tx gain (dBi): %.2f\n', r.G_tx_max_dBi);
    fprintf('Max Rx gain (dBi): %.2f\n', r.G_rx_max_dBi);
    fprintf('Free-space path loss (dB): %.2f\n', r.L_fs_dB);
    fprintf('Absorption loss (dB): %.4f\n', r.L_abs_dB);
    fprintf('Total loss (dB): %.2f\n', r.TotalLoss_dB);
    fprintf('Received power (dBm): %.2f\n', r.Pr_dBm);
    fprintf('Receiver noise power + NF (dBm): %.2f\n', r.noiseP_dBm);
    fprintf('SNR (dB): %.2f (linear = %.2f)\n', r.SNR_dB, r.SNR_linear);
end
```

PSEUDO CODE

```
BEGIN

1. Initialize constants
    c = speed of light
    k = Boltzmann constant

2. Set system parameters
    Define carrier frequencies for mmWave and THz
    Define bandwidths
    Define transmit power (dBm)
    Define antenna array sizes (Tx, Rx)
    Define steering angles
    Define element spacing factor
    Define link distances
    Define element gain (dBi), noise figure, misc losses
    Define absorption coefficients

3. Define helper conversions
    to_dB(x)   → convert linear to dB
```

from_dB(x) → convert dB to linear

4. Define function SIMULATE_LINK(frequency, bandwidth, TxN, RxN, distance, absorption_coeff):

a. Compute wavelength λ = c / frequency
b. Compute element spacing d = λ * spacing_factor

c. Compute TX and RX array patterns:
- Generate angle range θ from -90 to +90 degrees
- Compute array factor for each angle
- Normalize array magnitude
- Convert to dB including element gain
- Find maximum array gain

d. Compute path losses:
- Free-space loss using Friis formula
- Absorption loss = absorption_coeff × distance
- Total loss = free_space_loss + absorption_loss

e. Compute received power:
ReceivedPower = TxPower + GainTx + GainRx − TotalLoss − misc_losses

f. Compute noise:
NoisePower = k * T * bandwidth
NoisePower_dBm = convert to dBm + NoiseFigure

g. Compute SNR:
SNR_dB = ReceivedPower − NoisePower_dBm

h. Return all metrics and beam patterns

END FUNCTION

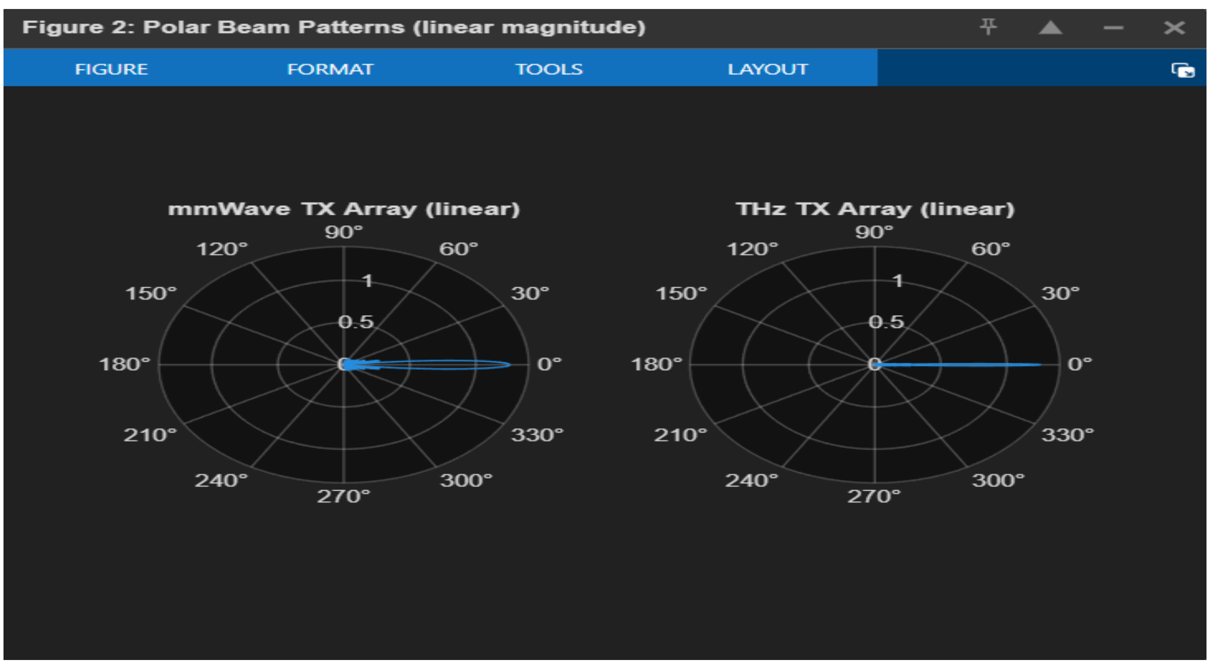
5. Run simulations:
mmwave_results = SIMULATE_LINK(mmWave parameters)
thz_results = SIMULATE_LINK(THz parameters)

6. Display summary:
Print received power, noise power, path loss, array gains, SNR

7. Plot beam patterns:
   Plot TX array pattern for mmWave
   Plot TX array pattern for THz
   Plot polar patterns for both

END

OUTPUT:



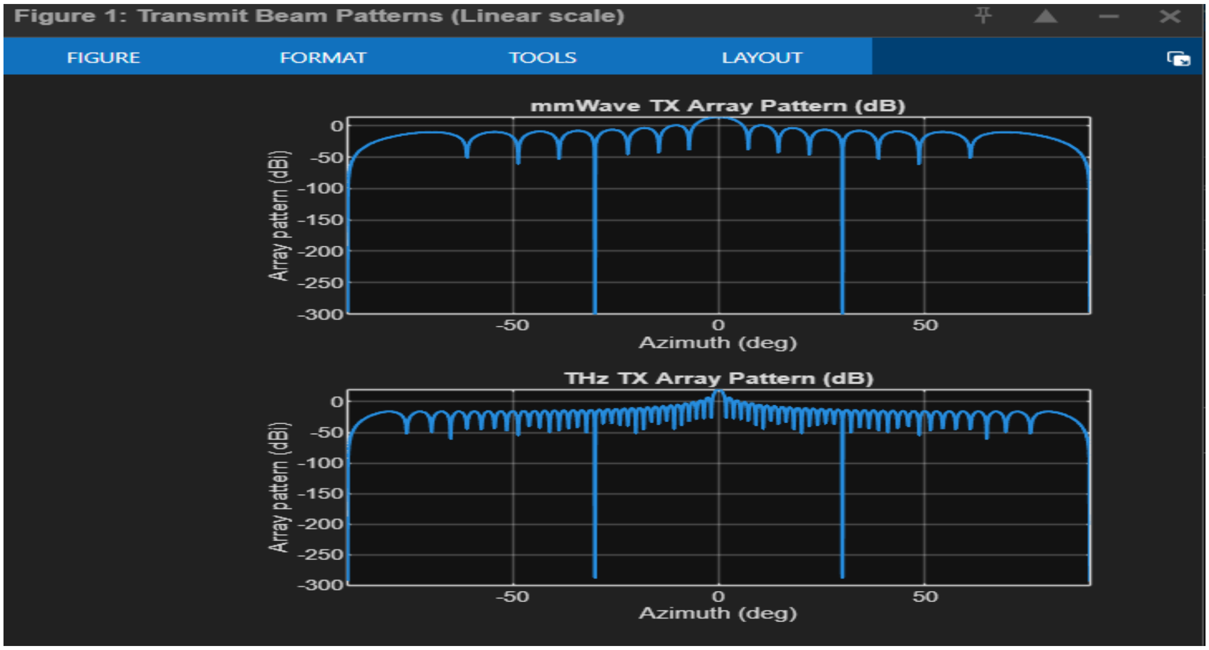
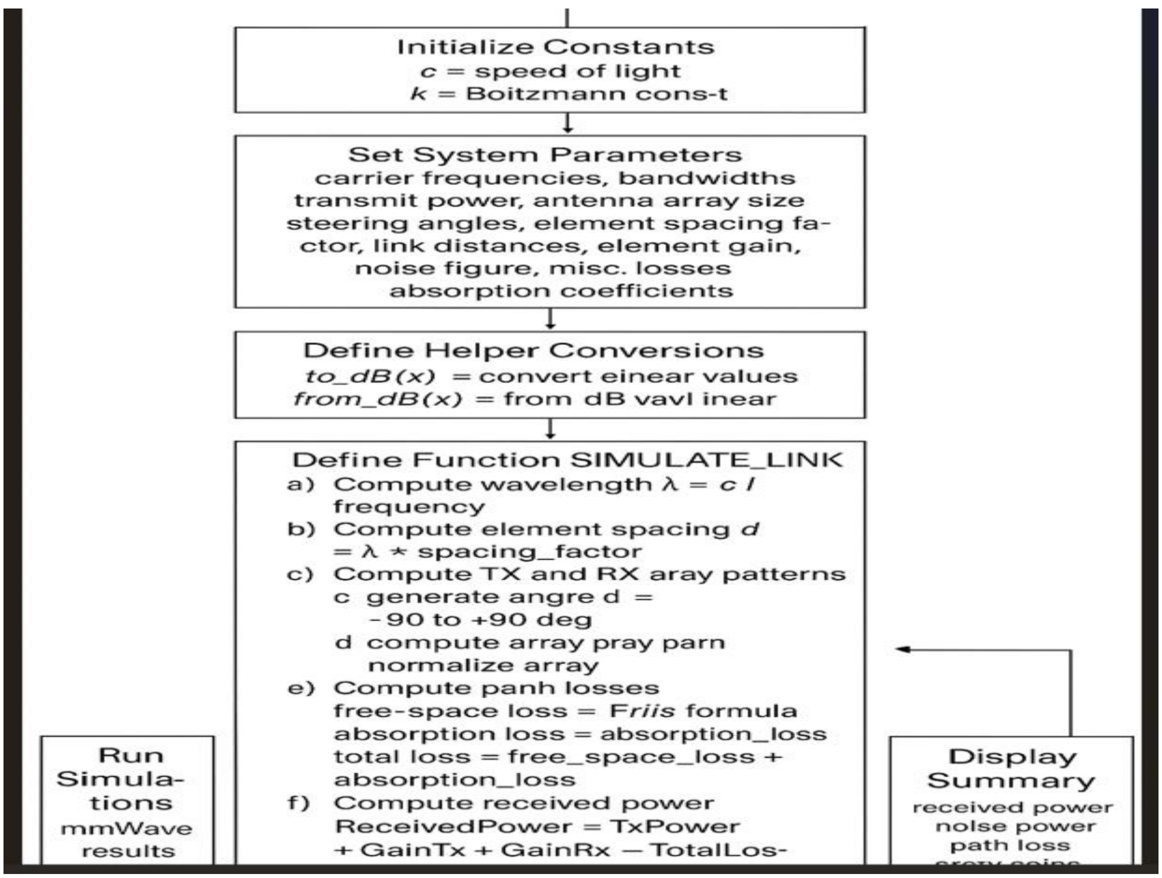


Figure 1: Transmit Beam Patterns (Linear scale)
FIGURE
FORMAT
TOOLS
LAYOUT
mmWave TX Array Pattern (dB)
Array pattern (dB)
0
-50
-100
-150
-200
-250
-300
-50
0
50
Azimuth (deg)
THz TX Array Pattern (dB)
Array pattern (dB)
0
-50
-100
-150
-200
-250
-300
-50
0
50
Azimuth (deg)

FLOW CHART



Initialize Constants
c = speed of light
k = Boitzmann cons-t
Set System Parameters
carrier frequencies, bandwidths
transmit power, antenna array size
steering angles, element spacing fa-
ctor, link distances, element gain,
noise figure, misc. losses
absorption coefficients
Define Helper Conversions
to_dB(x) = convert einear values
from_dB(x) = from dB vavl inear
Define Function SIMULATE_LINK
a) Compute wavelength λ = c / frequency
b) Compute element spacing d = λ * spacing_factor
c) Compute TX and RX aray patterns
c generate angre d = -90 to +90 deg
d compute array pray parn
normalize array
e) Compute panh losses
free-space loss = Friis formula
absorption loss = absorption_loss
total loss = free_space_loss + absorption_loss
f) Compute received power
ReceivedPower = TxPower
+ GainTx + GainRx — TotalLos-
Run Simula-tions
mmWave results
Display Summary
received power
nolse power
path loss

CONSOLE OUTPUT:

```
==== SUMMARY ====

--- mm Wave 28 GHz ---
Frequency: 2.8e+10 Hz (lambda = 0.0107 m)
Tx array elements: 16, Rx array elements: 16
Element spacing: 0.00536 m
Max Tx gain (dBi): 14.04
Max Rx gain (dBi): 14.04
Free-space path loss (dB): 101.38
Absorption loss (dB): 0.0001
Total loss (dB): 101.39
Received power (dBm): -55.30
Receiver noise power + NF (dBm): -80.95
SNR (dB): 25.65 (linear = 367.45)

--- THz 300 GHz ---
Frequency: 3e+11 Hz (lambda = 0.001 m)
Tx array elements: 64, Rx array elements: 64
Element spacing: 0.0005 m
Max Tx gain (dBi): 20.06
Max Rx gain (dBi): 20.06
Free-space path loss (dB): 101.98
Absorption loss (dB): 0.1000
Total loss (dB): 102.08
Received power (dBm): -43.96
Receiver noise power + NF (dBm): -76.98
SNR (dB): 33.01 (linear = 2001.98)
```

Conclusion:

1. Beamforming effectively boosts link strength in mmWave and THz systems.
2. Received power and SNR improve significantly with antenna array gains.
3. Beam patterns confirm precise directional signal steering.
4. THz links face higher losses, requiring stronger beamforming.
5. Massive MIMO is essential for reliable high-speed 5G/6G communication.