Projeto 1 - Simulação do Canal

Pedro Henrique Dornelas Almeida

Comunicações Móveis

6 de outubro de 2025

Agenda

Agenda

Definições de Variáveis

- Linguagem utilizada: Matlab;
- Cenário utilizado: UMa-NLoS.

```
5   %% Simulation Params
6
7   % Enviroment - UMa-NLoS - number 4 sorted
8   env = 'UMa_NLoS';
9   % Frequency (GHz)
10  freq_ghz = 3;
11  c = 299792458; % m/s
12  lambda = c / (freq_ghz * 1e9);
13  % Multipath components
14  N = 100;
15  % Elevation Angle
16  phi_bar = pi/4;
17  % RX Velocity
18  v = 5; % m/s
```

Figura: Variáveis Globais

Agenda

Espalhamento de Atraso

```
21  %% Delay Spread RMS
22  % Stats
23  [delay_mean, delay_std] = get_delay_stats(freq_ghz, env);
24  % Gaussian - only one sample to represent
25  delay_spread_rms_log = normrnd(delay_mean, delay_std, [1,1]);
26  delay_spread = 10^(delay_spread_rms_log);
27  delay_spread_ns = 1e9*delay_spread
28
29  %% Generate Delay Samples
30  % Proportionality Factor
31  r_tau = get_prop_factor_delay(env);
32  mu_tau = delay_spread * r_tau;
33  % Delay samples
34  tau_n_absolute = exprnd(mu_tau, [N,1]);
35  tau_n_normalized = tau_n_absolute - min(tau_n_absolute);
36  tau_n = sort(tau_n_normalized);
37  % tau_n_ns = tau_n(1:5)*1e9
```

Figura: Espalhamento de Atraso e Amostras de Atraso

Agenda

Potência Multipercurso - Código

```
39  %% Generate Power Samples
40  % Power stats
41  [shadow_std, rice_mean, rice_std] = get_power_stats(env);
42  % Shadowing samples
43  shadow_samples = normrnd(0, shadow_std, [N,1]);
44  % Power samples
45  alpha_n_2 = exp(-tau_n .* (r_tau-1) ./ (r_tau*delay_spread)) .* 10.^(-shadow_samples ./ 10);
46  % alpha_n_2(1:5)
47
48  %% Rice factor
49  kr = 0;
50  kr_db = -inf;
51  if env == "UMi_LoS" || env == "UMa_LoS"
52      kr_db = normrnd(rice_mean, rice_std, [1,1]);
53      kr = 10 ^ (kr_db / 10);
54  end
55  kr
56
57  % Normalizing power
58  hat_omega_c = sum(alpha_n_2(2:N));
59  alpha_n_2 = (1/(kr + 1)) .* (alpha_n_2 ./ hat_omega_c);
60  alpha_n_2(1) = kr / (kr + 1);
61
62  % Verifying power = 1
63  omega_c = sum(alpha_n_2)
64  % Verifying Rice Factor
65  kr_check = alpha_n_2(1) ./ sum(alpha_n_2(2:N))
66  % Verifying Delay Spread
67  tau_bar = (1 / omega_c) * sum(tau_n .* alpha_n_2);
68  delay_spread
69  sigma_tau_check = sqrt((1 / omega_c) * sum(alpha_n_2 .* ((tau_n - tau_bar) .^ 2)))
```

Figura: Variáveis para Potência Multipercurso

Fator de Rice

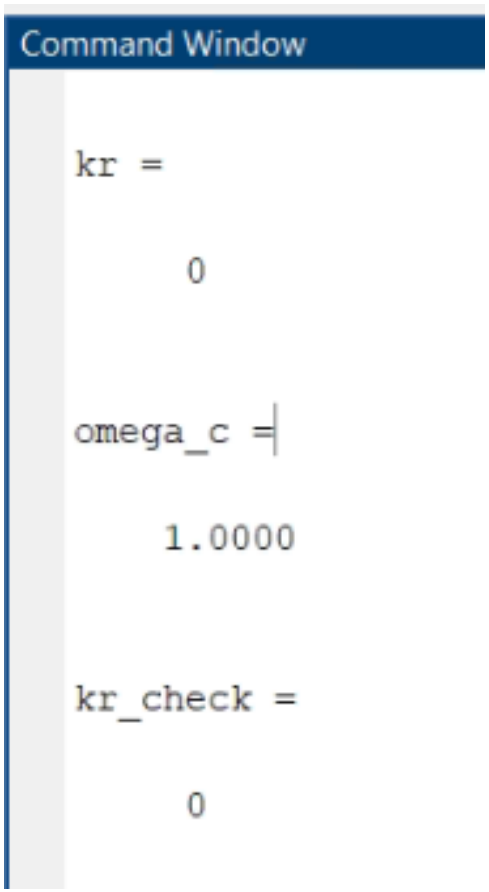

Figura: UMa-NLoS

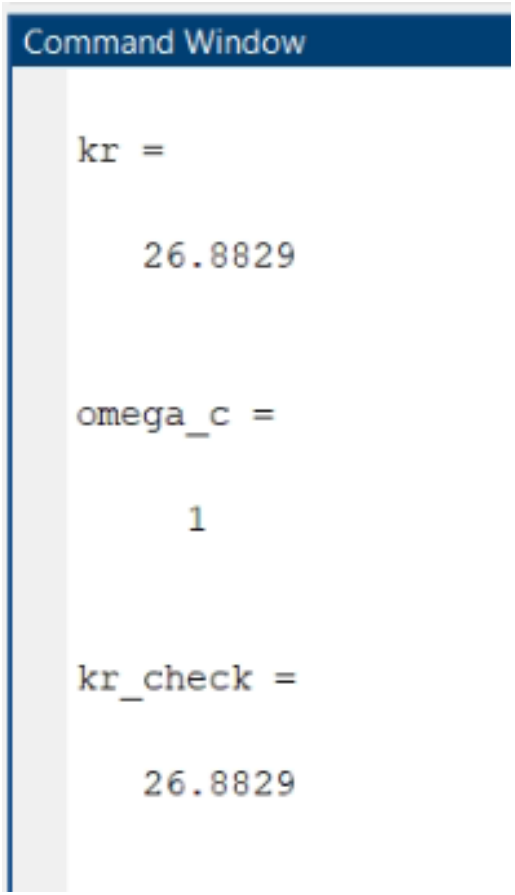

Figura: UMa-LoS

Curva do Perfil de Atraso de Potência

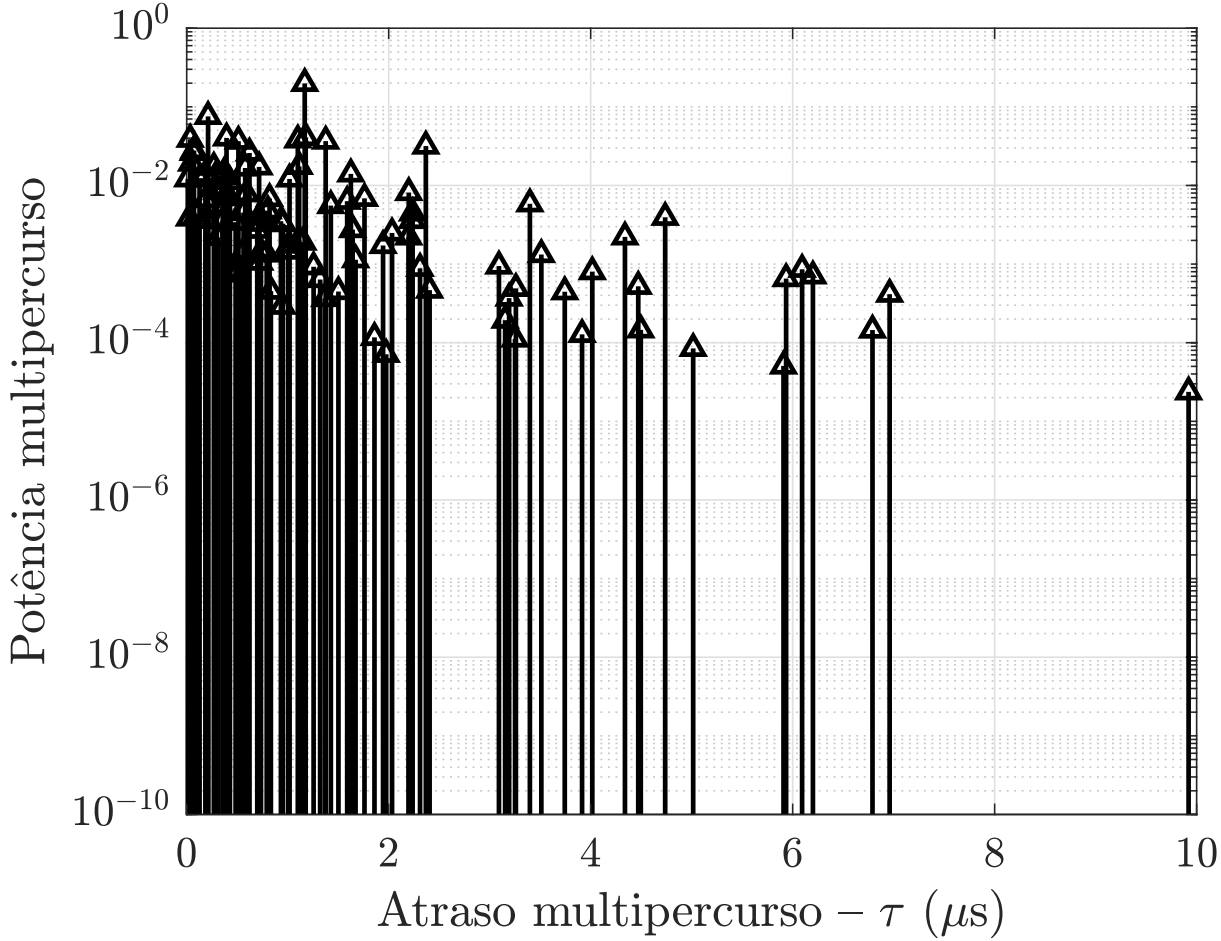

Figura: Perfil de Atraso de Potência

Curva do Perfil de Atraso de Potência

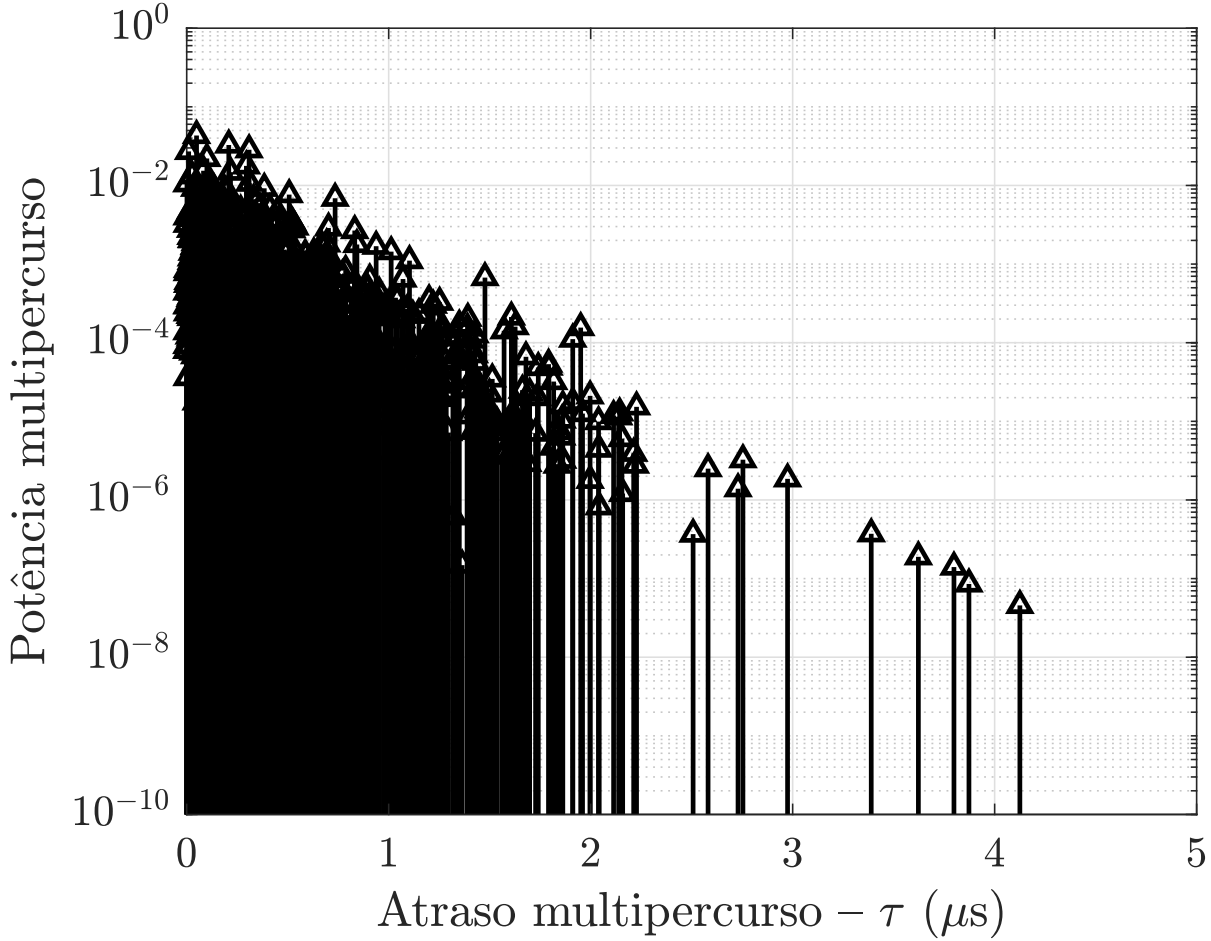

Figura: Perfil de Atraso de Potência - 1000 componentes

Espalhamento de Atraso Gerado e Medido

```
delay_spread =

   8.8316e-07


sigma_tau_check =

   8.3889e-07
```

Figura: Espalhamento de Atraso Gerado e Medido

Agenda

Ângulos em Azimute - Geração

```
94  % -------------------------------------
95  %% Angles Stats
96  % -------------------------------------
97  [azimuth_mean, azimuth_std, elevation_mean, elevation_std] = get_angle_stats(freq_ghz, env);
98  % Azimuth Angles
99  sigma_theta_degree = 10 ^ (normrnd(azimuth_mean, azimuth_std, [1,1]));
100 sigma_theta_rad = sigma_theta_degree * (pi / 180);
101 max_alpha = max(alpha_n_2);
102 theta_n = 1.42 * sigma_theta_rad * sqrt(-log(alpha_n_2 ./ max_alpha));
103 % Adjusts
104 un = randsample([-1,1], N, true)';
105 % un(1:5)
106 yn = normrnd(0, sigma_theta_rad/7, [N, 1]);
107 % yn(1:5)
108 % Finally
109 theta_n = (un .* theta_n) + yn;
110 if env == "UMi_LoS" || env == "UMa_LoS"
111     theta_n = theta_n - theta_n(1);
    You, 2 hours ago | 1 author (You)
112 else
113     theta_n(1) = 0;
114 end
```

Figura: Ângulos Azimutais

Ângulos em Azimute

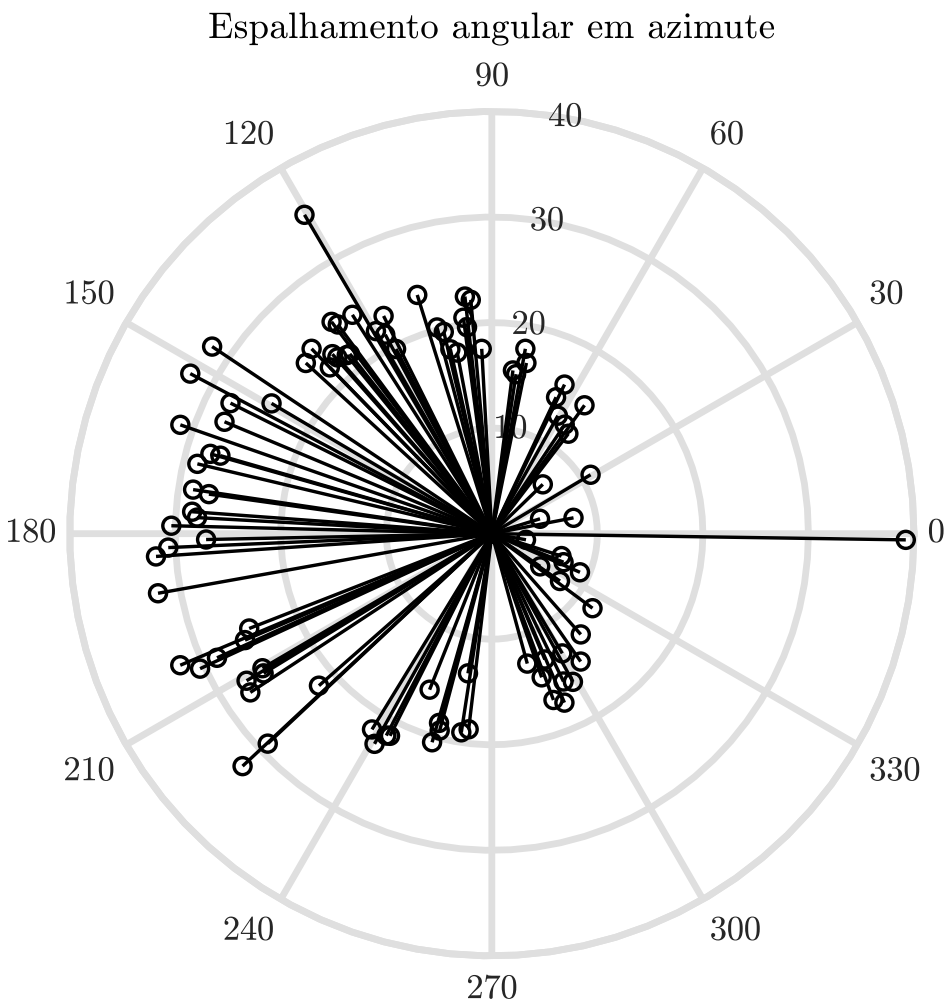

Figura: Ângulos Azimutais

Ângulos em Azimute

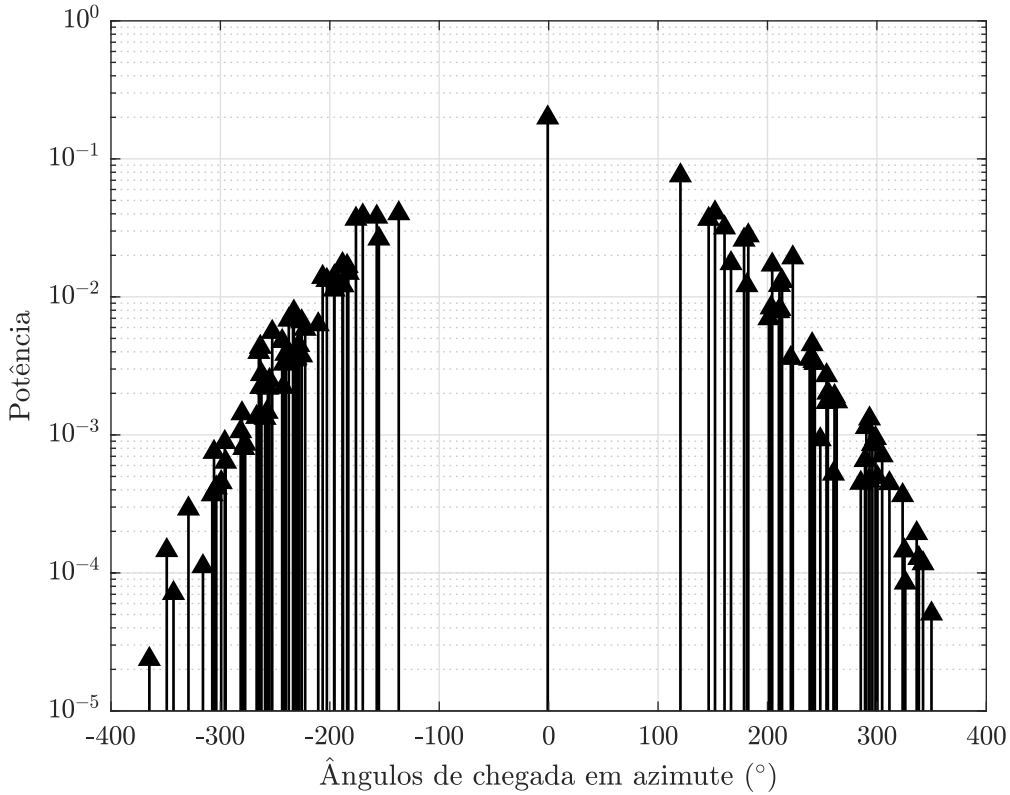

Figura: Espalhamento de Potência com Ângulos Azimutais

Ângulos em Elevação - Geração

```
161  %% Elevation Angles
162  sigma_phi_degree = 10 ^ (normrnd(elevation_mean, elevation_std, [1,1]));
163  sigma_phi_rad = sigma_phi_degree * (pi / 180);
164  max_alpha = max(alpha_n_2);
165  phi_n = -sigma_phi_rad * log(alpha_n_2 ./ max_alpha);
166  % Adjusts
167  un = randsample([-1,1], N, true)';
168  % un(1:5)
169  yn = normrnd(0, sigma_phi_rad/7, [N, 1]);
170  % yn(1:5)
171  % Finally
172  phi_n = (un .* phi_n) + yn;
173  if env == "UMi_LoS" || env == "UMa_LoS"
174      phi_n = phi_n - phi_n(1) + phi_bar;
     You, 2 hours ago | 1 author (You)
175  else
176      phi_n(1) = 0;
177  end
```

Figura: Ângulos em Elevação

Ângulos em Elevação

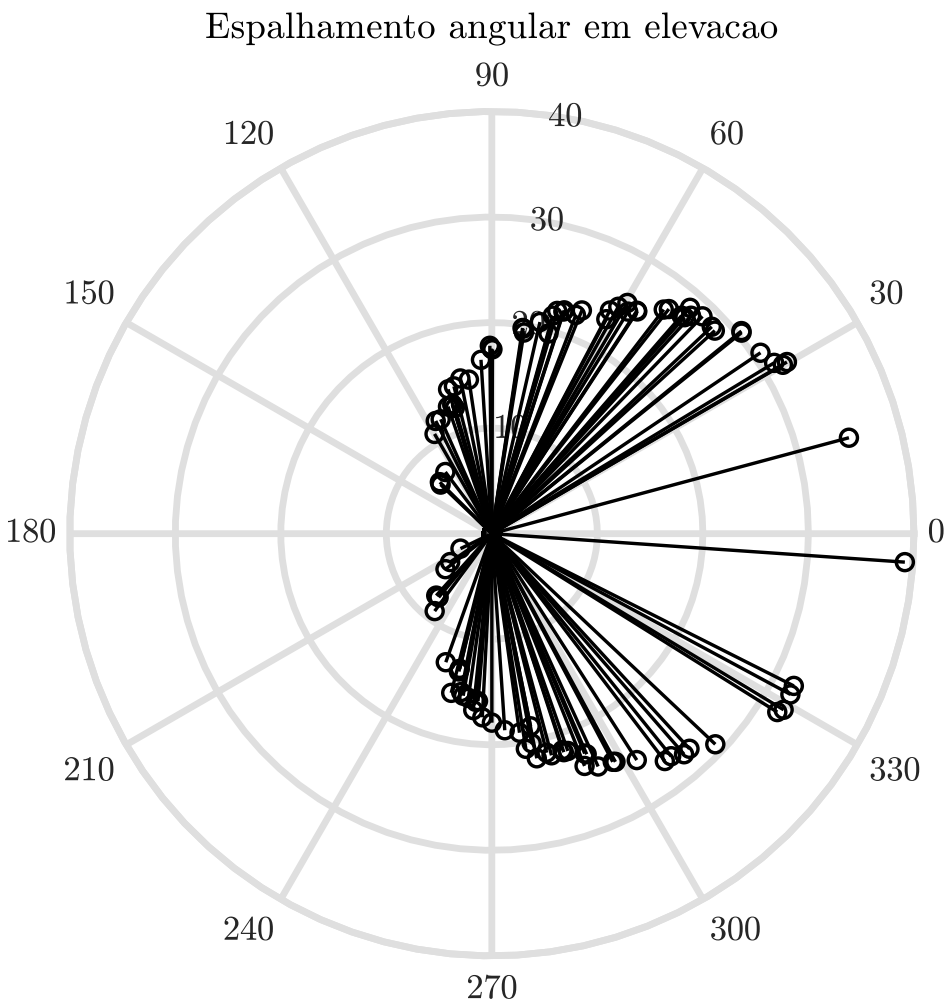

Figura: Ângulos em Elevação

Ângulos em Elevação

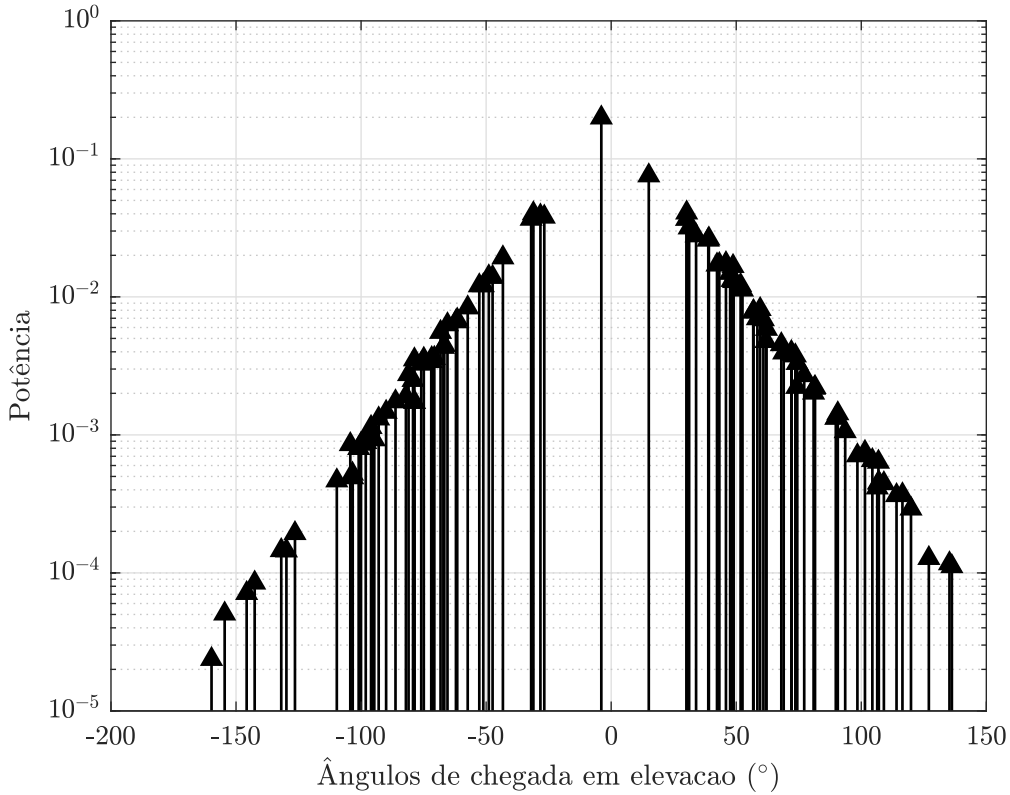

Figura: Espalhamento de Potência com Ângulos em Elevação

Vetores de Direção de Chegada

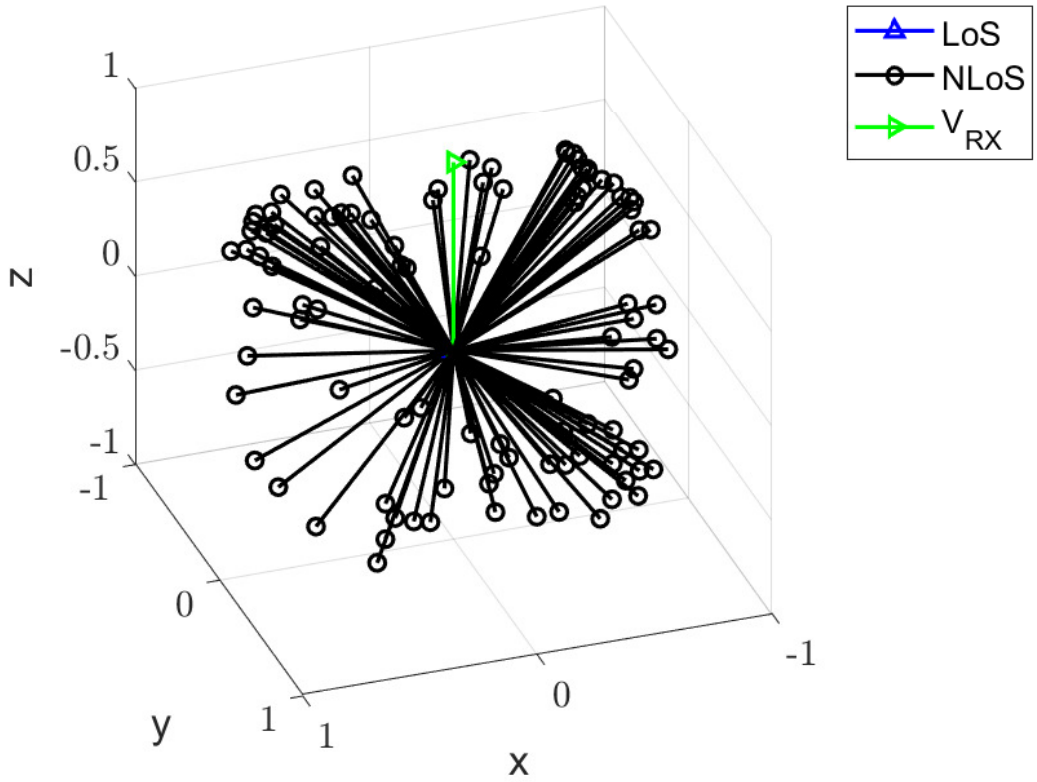

Figura: Vetores de Direção de Chegada

Agenda

Desvios Doppler - Geração

```
259    % ---------------------------------
260    %% Doppler Effects
261    % ---------------------------------
262    vn = (v / lambda) * (rn * vrx');
263    vn = sum(vn, 2);
```

Figura: Desvio Doppler

Desvios Doppler - $v = 5m/s$

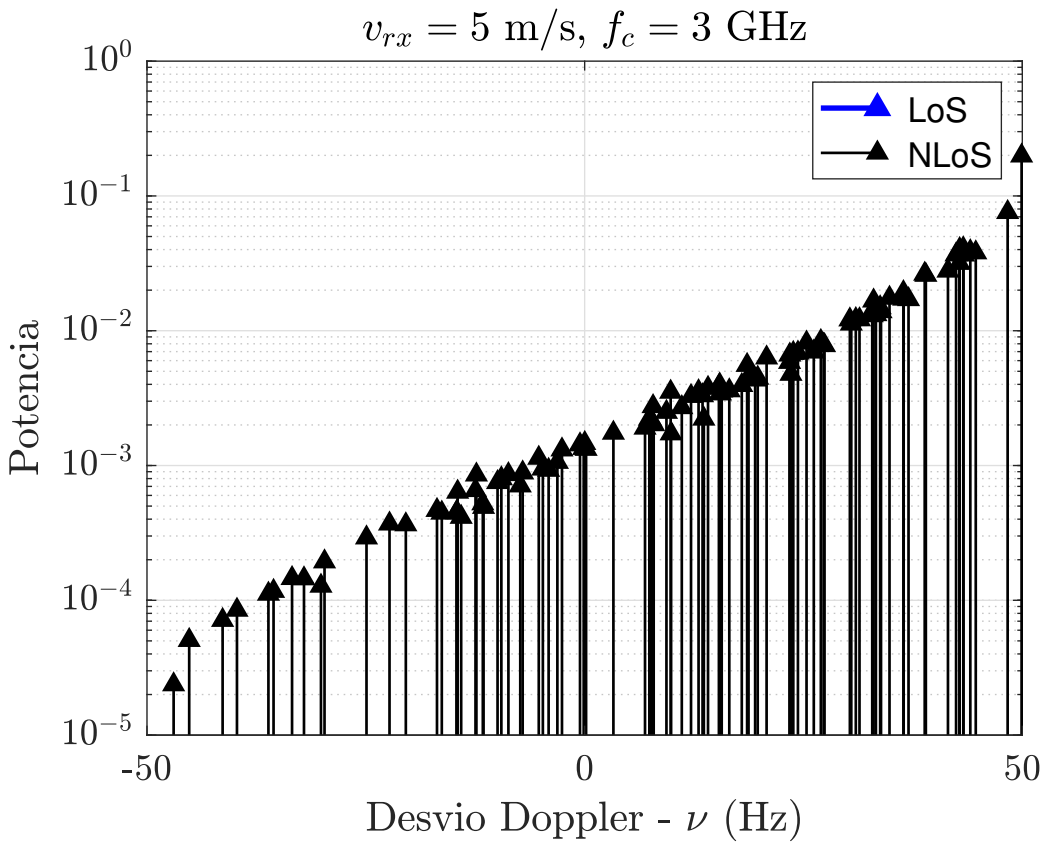

Figura: Espalhamento de Potência - Desvios Doppler $v = 5m/s$

Desvios Doppler - $v = 50m/s$

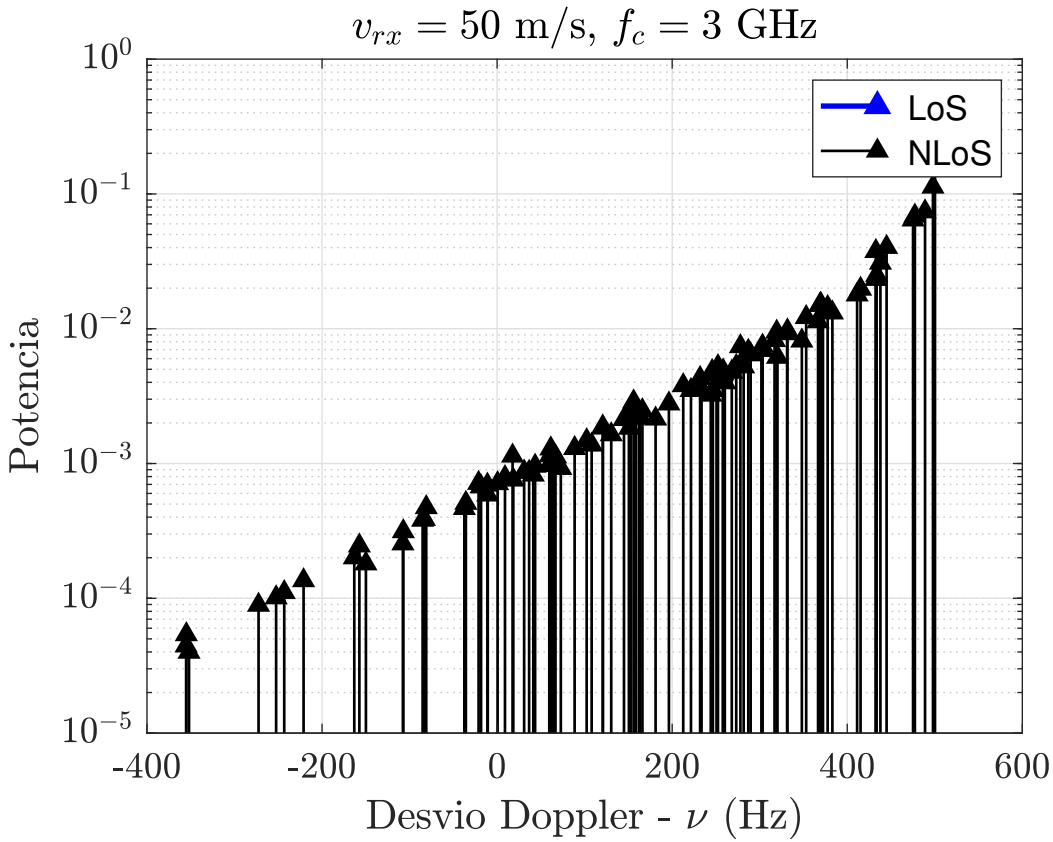

Figura: Espalhamento de Potência - Desvios Doppler $v = 50m/s$

Agenda

Sinal Transmitido - Geração

Sinais a serem transmitidos:

```
283  % -------------------------------------
284  %% Signal Analysis
285  % -------------------------------------
286  deltas_t = [1e-7, 1e-5, 1e-3];
287  n_samples = 1e5;
288
289
290  % Transmited Signal
291  t = zeros(n_samples, length(deltas_t));
292  signal_tx = zeros(n_samples, length(deltas_t));
293  for d = 1 : length(deltas_t)
294      t(: , d) = linspace(0, 5*deltas_t(d), n_samples);
295      [signal_tx(:, d)] = generate_pulse(0, t(: , d), deltas_t(d));
296  end
```

Figura: Geração dos Sinais Transmitidos

Função auxiliar para gerar o pulso:

```
506  function [signal] = generate_pulse( delay, t, pulse_width )
507      signal = zeros(length(t), 1);
         You, 3 hours ago | 1 author (You)
508      for i = 1 : length(t)
             You, 3 hours ago | 1 author (You)
509          if t(i) >= delay && t(i) <= pulse_width + delay
510              signal(i) = 1;
511          end
512      end
513  end
```

Figura: Função Auxiliar para Geração dos Pulsos

Sinais Recebidos - Geração

```
317  % -------------------------------------
318  %% Received Signal
319  % -------------------------------------
320
321
322  signal_rx = zeros(n_samples, N, length(deltas_t));
323  % signal_rx = zeros(N, length(deltas_t));
324  fig_init = 10;
325
326  for d = 1 : length(deltas_t)
         You, 3 hours ago | 1 author (You)
327      for i = 1 : N
328          delayed_signal = generate_pulse(tau_n(i), t(:, d), deltas_t(d));
329          phase_n = 2*pi * (((freq_ghz * 1e9 + vn(n)) * tau_n(n)) - (vn(n) * t(:, d)));
330          signal_rx(:, i, d) = alpha_n_2(i) * exp(-phase_n * 1j) .* delayed_signal;
331      end
332
333      scattered_signal_rx(:, d) = sum(signal_rx(:, :, d), 2);
334      abs_scat_signal_rx(:, d) = abs(scattered_signal_rx(:, d));
335      normalized_signal_rx(:, d) = (abs_scat_signal_rx(:, d) - min(abs_scat_signal_rx(:, d))) / (max(abs_scat_signal_rx(:, d)) - min(abs_scat_signal_rx(:, d)));
336
337      figure(fig_init)
338      plot(t(:, d), abs(signal_tx(:, d)), 'Color', 'b', 'Linewidth', 1.5)
339      hold on
340      plot(t(:, d), normalized_signal_rx(:, d), 'Color', 'r', 'Linewidth', 1.2)
341      xlim([0, 5*deltas_t(d)])
342      % xticklabels({'$-2\delta_t$', '$-\delta_t$', '$0$', '$\delta_t$', '$2\delta_t$', '$3\delta_t$', '$4\delta_t$', '$5\delta_t$', '$6\delta_t$'})
343      ylabel('$s(t)$', 'Interpreter', 'Latex')
344      xlabel('$t$ [s]', 'Interpreter', 'Latex')
345      title("$\delta t="+num2str(deltas_t(d)*1e6)+" \mu s$, $\sigma_{\tau}="+num2str(delay_spread_ns)+"\eta s$, Rice$="+num2str(kr_db)+"$dB", 'Interpreter', 'Late
346      legend('TX', 'RX')
347      ax = gca;
348      ax.TickLabelInterpreter = 'latex';
349      ax.FontSize = 14;
350      grid on
351
352      fig_init = fig_init + 1;
353  end
```

Figura: Cálculo dos Sinais Recebidos

Sinal Recebido - $\delta t = 10^{-7} s$

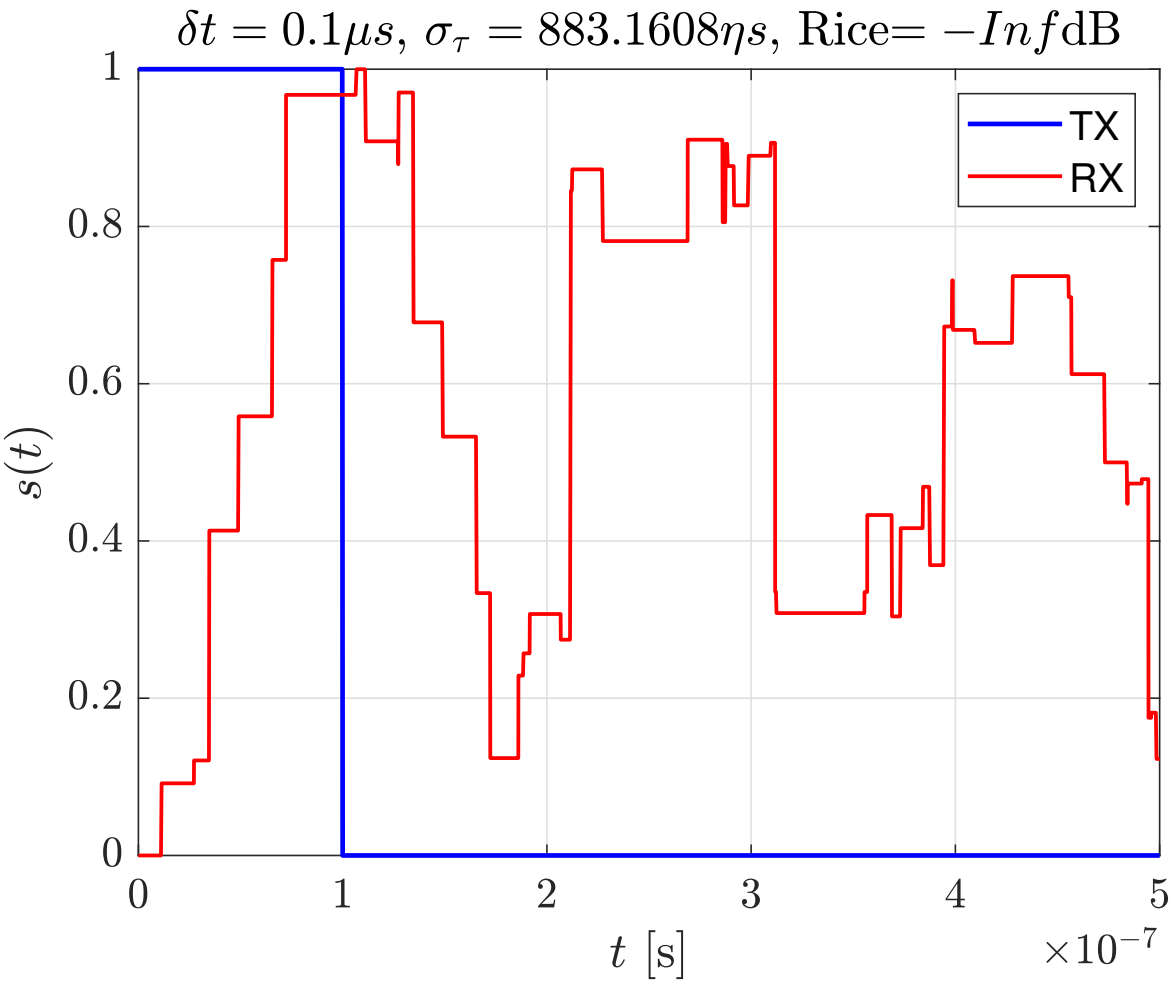

Figura: Sinal Recebido $\delta t = 10^{-7} s$

Sinal Recebido - $\delta t = 10^{-5} s$

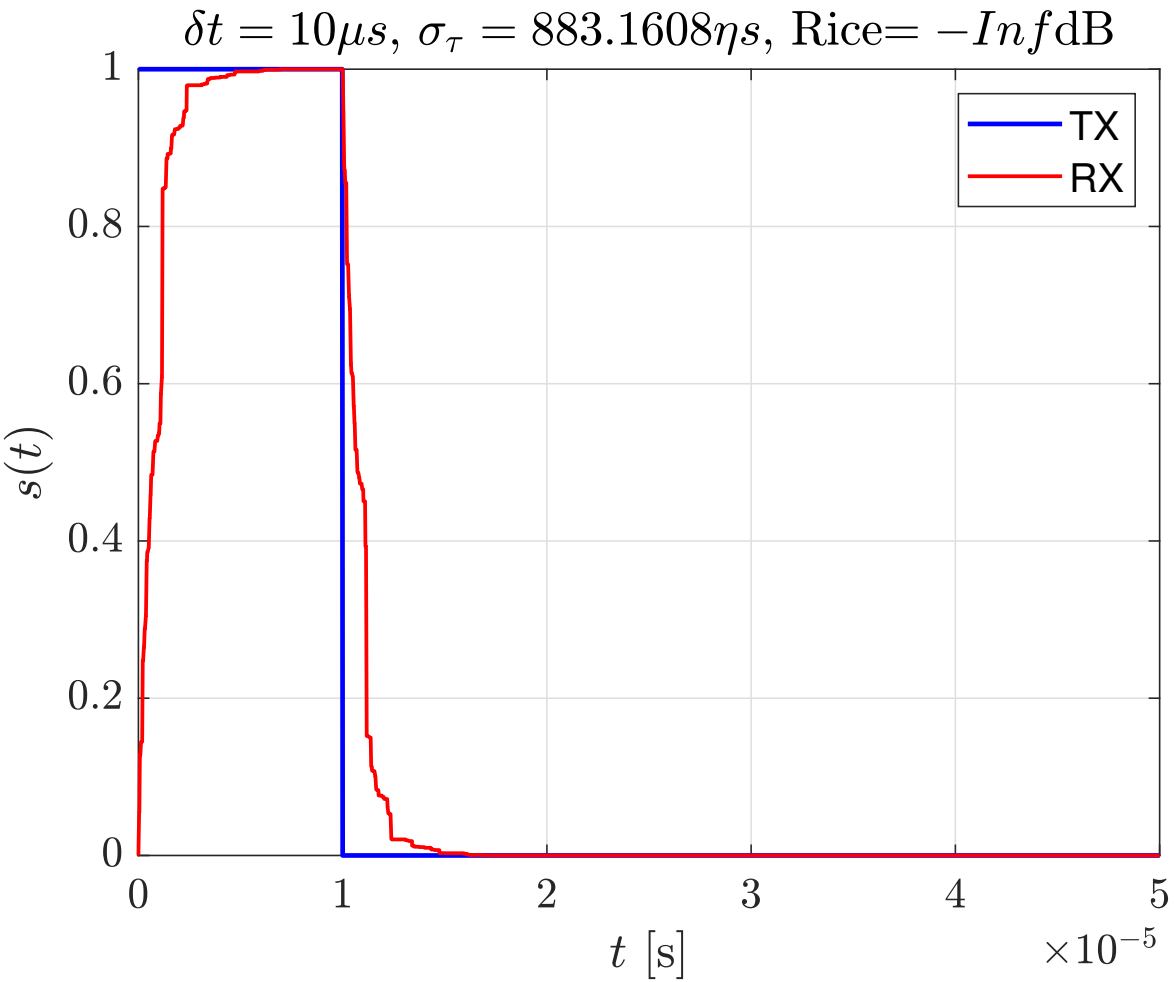

Figura: Sinal Recebido $\delta t = 10^{-5} s$

Sinal Recebido - $\delta t = 10^{-3}s$

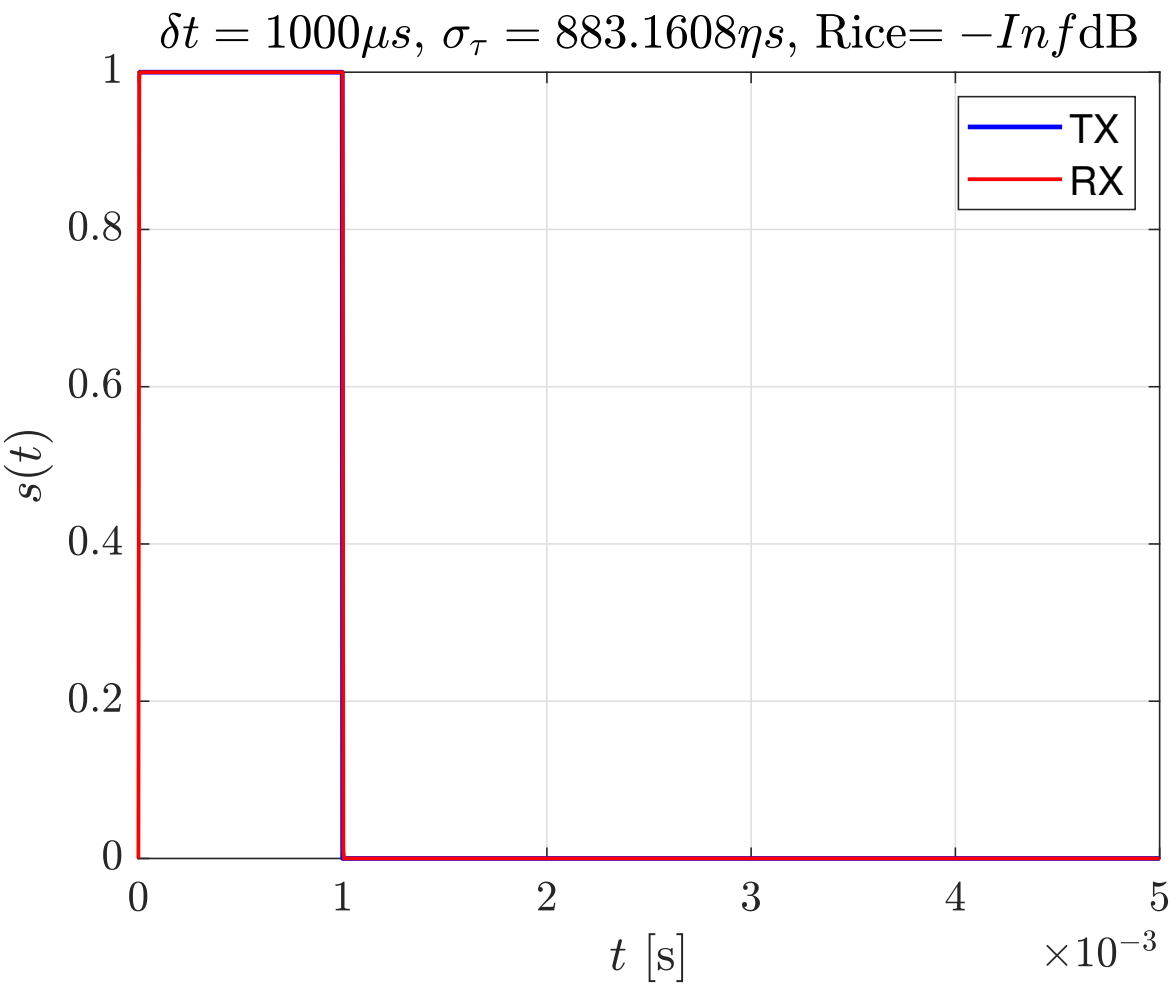

Figura: Sinal Recebido $\delta t = 10^{-3}s$

Agenda

Autocorrelação - Geração

```
355   %% Coherency bandwidth and coherency time
356   n_samples = 1e4;
357   k = logspace(-3, 10, n_samples);
358   t = logspace(-6, 0, n_samples);
359
360   omega_c = sum( alpha_n_2 );
361   autocorr_freq_n = (1 / omega_c) .* alpha_n_2 .* exp( -2i * pi * tau_n .* k );
362   autocorr_freq = sum(autocorr_freq_n, 1);
363
364   autocorr_time_n = (1 / omega_c) .* alpha_n_2 .* exp( 2i * pi * vn .* t);
365   autocorr_time = sum(autocorr_time_n, 1);
```

Figura: Autocorrelação

Banda de Coerência

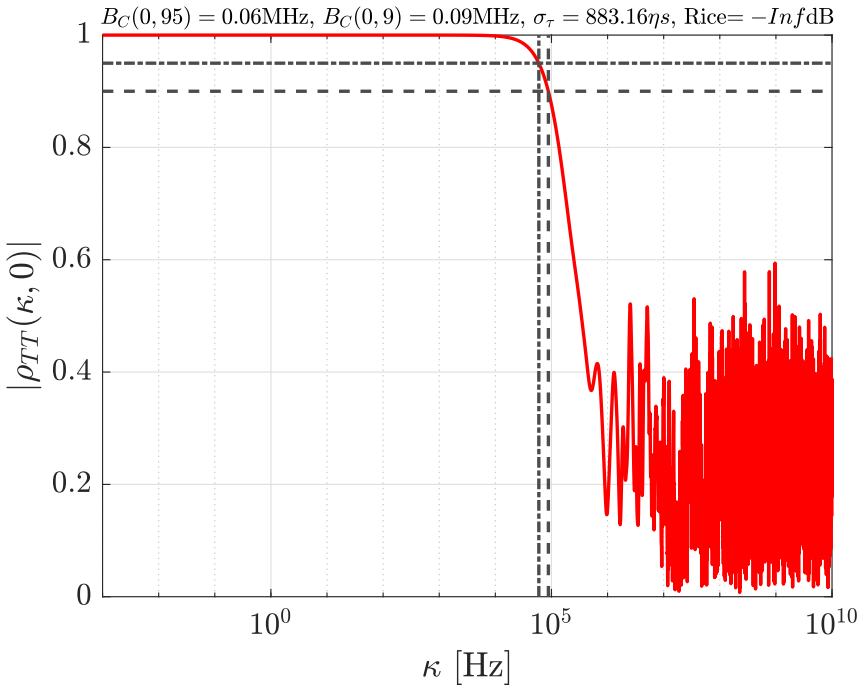

Figura: Banda de Coerência

Tempo de Coerência - $v = 5m/s$

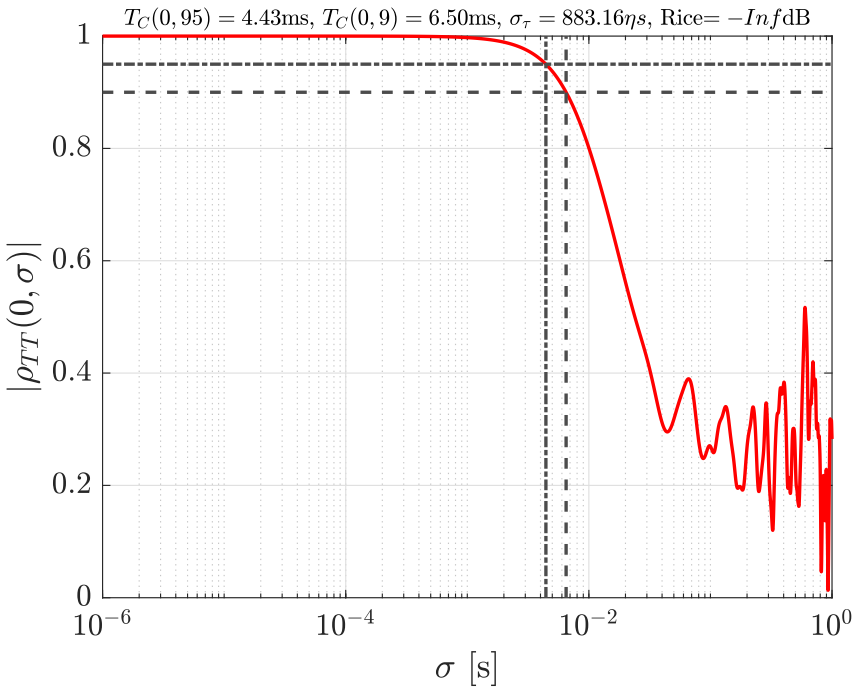

Figura: Tempo de Coerência $v = 5m/s$

Tempo de Coerência - $v = 50m/s$

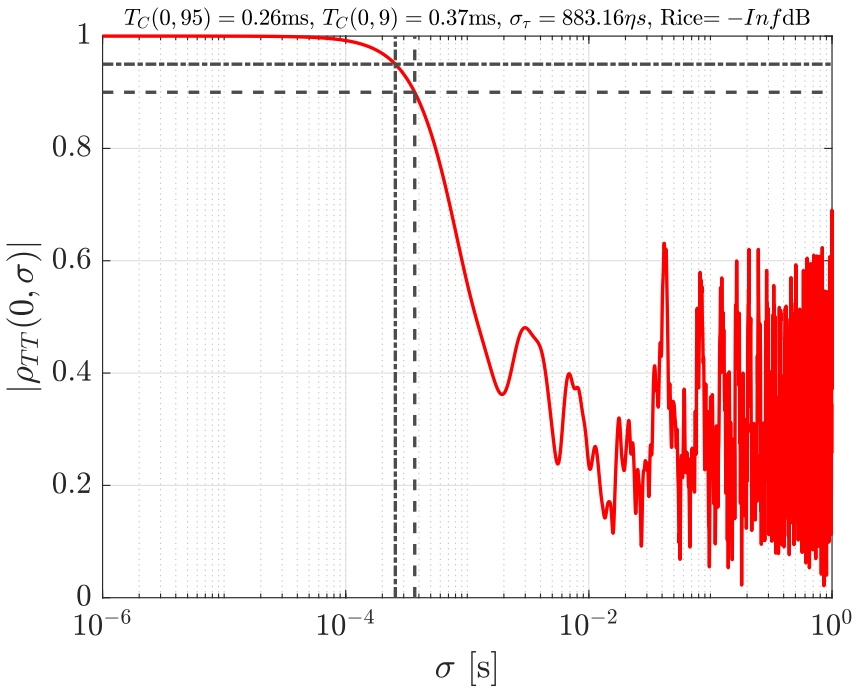

Figura: Tempo de Coerência $v = 50m/s$

Obrigado pela atenção!

Pedro Henrique Dornelas Almeida
2421100048@aluno.unb.br