Relatório de Simulação

Karen dos Anjos Arcoverde

Introdução

Apresenta-se a simulação de propagação de sinal para a rede privada de Proteção Pública e Assistência a Desastres (PPDR) proposta para o bairro de Icaraí, em Niterói, operando na faixa de 850 MHz em ambiente urbano com edificações de até 20 andares. A análise compara três modelos de perda de percurso — FS (*free-space* – FS), CI (*close-in* – CI) e AB (*floating intercept* – AB) — utilizando medições de potência recebida em distintas distâncias, por meio de gráficos de perda de percurso e histogramas dos erros ajustados a distribuições gaussianas. O objetivo principal é determinar, através da sobreposição das retas teóricas e da caracterização estatística dos desvios, qual dos três modelos melhor descreve a natureza de propagação nessa região.

Resultados Obtidos

Questão 1

O passo a passo para achar os coeficientes da equação *log-distance* para os modelos FS, CI e AB é explicado abaixo:

Equação de Friis (forma linear)

$$P_R(d) = P_T \left(\frac{\lambda}{4\pi\, d}\right)^2 G_t\, G_r \quad [\mathrm{W}]$$

Conversão para dB (potências em dBm e ganhos em dBi)

$$\begin{aligned} P_{R,\mathrm{dBm}} = P_{T,\mathrm{dBm}} &+ 10\log_{10}(G_t\, G_r) + 20\log_{10}(\lambda) \\ &- 20\log_{10}(4\pi) - 20\log_{10}(d) \quad [\mathrm{dBm}] \end{aligned}$$

Definição de Path-Loss (PL)

$$\begin{aligned} PL(d) &= P_{T,\mathrm{dBm}} - P_{R,\mathrm{dBm}} \\ &= -10\log_{10}\left(\frac{G_t\, G_r\, \lambda^2}{(4\pi\, d)^2}\right) \quad [\mathrm{dB}] \end{aligned}$$

Forma log-distance: expandindo o logaritmo

$$\begin{aligned} PL(d) &= -10\log_{10}(G_t\,G_r) - 10\log_{10}(\lambda^2) + 20\log_{10}(4\pi) \\ &\quad + 20\log_{10}(d) \\ &= 20\log_{10}(d) + 20\log_{10}\left(\frac{4\pi}{\lambda}\right) - 10\log_{10}(G_t\,G_r) \end{aligned}$$

Coeficientes FS (antenas isotrópicas: $G_t = G_r = 1$)

$$\begin{aligned} A_{\mathrm{FS}} &= 20\log_{10}\left(\frac{4\pi}{\lambda}\right), \\ B_{\mathrm{FS}} &= 20, \\ PL_{\mathrm{FS,pred}}(d) &= A_{\mathrm{FS}} + B_{\mathrm{FS}}\,\log_{10}(d) \end{aligned}$$

Cálculo da PL medida

Esta expressão calcula a perda de percurso efetivamente medida $\big(PL_{\mathrm{meas}}\big)$: parte-se da potência transmitida em dBm (P_t^{dBm}), soma-se o ganho das antenas transmissora e receptora (G_t^{dBi} e G_r^{dBi}), subtraem-se as perdas nos cabos de transmissão e recepção ($L_{\mathrm{cab,tx}}^{\mathrm{dB}}$ e $L_{\mathrm{cab,rx}}^{\mathrm{dB}}$) e adiciona-se o ganho do LNA ($G_{\mathrm{LNA}}^{\mathrm{dB}}$), e finalmente subtrai-se a potência recebida medida (P_r^{dBm}).

$$\begin{aligned} PL_{\mathrm{meas}} &= P_t^{\mathrm{dBm}} + G_t^{\mathrm{dBi}} - L_{\mathrm{cab,tx}}^{\mathrm{dB}} \\ &\quad + G_r^{\mathrm{dBi}} - L_{\mathrm{cab,rx}}^{\mathrm{dB}} \\ &\quad + G_{\mathrm{LNA}}^{\mathrm{dB}} - P_r^{\mathrm{dBm}} \end{aligned}$$

Modelo Close-In (CI)

O modelo CI mantém o intercepto do espaço livre A_{FS} e ajusta apenas o declive para melhor descrever os dados medidos, utilizando o método dos mínimos quadrados.

Definição do erro e critério MMSE

Neste caso, o modelo de perda de percurso é dado por

$$PL_{\mathrm{model}}(d) = A_{FS} + B_{CI}\,\log_{10}(d).$$

O erro em cada medida i é

$$\varepsilon_i = PL_{\mathrm{medido},i} - PL_{\mathrm{model},i} = PL_{\mathrm{medido},i} - \big(A_{FS} + B_{CI}\,\log_{10}(d_i)\big),$$

e o erro quadrático total sobre N pontos é

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \big(PL_{\mathrm{medido},i} - (A_{FS} + B_{CI}\,\log_{10}(d_i))\big)^2.$$

Como queremos minimizar esse somatório (MMSE), definimos

$$\mathrm{MMSE} = \min_{B_{CI}} \sum_{i=1}^{N} \big(PL_{\mathrm{medido},i} - (A_{FS} + B_{CI}\,\log_{10}(d_i))\big)^2.$$

Neste modelo, como A é fixo:

$$B_{CI} = \frac{PL_{\text{medido},i} - A_{FS}}{\log_{10}(d_i)}.$$

Reorganizando os termos:

$$y_i = PL_{\text{medido},i} - A_{FS}, \quad x_i = \log_{10}(d_i),$$

de modo que a equação do modelo simplifica-se para:

$$y_i = B_{CI}\, x_i.$$

de modo que

$$\text{MMSE} = \min_{B_{CI}} \sum_{i=1}^{N} \big(y_i - B_{CI}\, x_i\big)^2,$$

onde $x_i = \log_{10}(d_i)$.

Ajuste por mínimos-quadrados

O declive B_{CI} minimiza o erro quadrático $\varepsilon_i = y_i - B_{\text{CI}}\, x_i$. Fazendo a derivada em relação a B_{CI}:

$$\frac{\partial}{\partial B_{\text{CI}}} \sum_{i=1}^{N} (y_i - B_{\text{CI}} x_i)^2 = -2 \sum_{i=1}^{N} x_i\, (y_i - B_{\text{CI}} x_i) = 0$$

Dividindo por -2 e rearranjando,

$$-\sum_{i=1}^{N} x_i\, y_i + B_{\text{CI}} \sum_{i=1}^{N} x_i^2 = 0 \quad \Longrightarrow \quad B_{\text{CI}} = \frac{\sum_{i=1}^{N} x_i\, y_i}{\sum_{i=1}^{N} x_i^2}.$$

Predição do modelo CI

Por fim, a predição de perda de percurso é:

$$PL_{\text{CI}}(d) = A_{\text{FS}} + B_{\text{CI}}\, \log_{10}(d).$$

Modelo AB

O modelo AB ajusta tanto o intercepto quanto o declive por mínimos quadrados, e é definido por

$$PL_{model}(d_i) = A_{\text{AB}} + B_{\text{AB}}\, \log_{10}(d_i).$$

Preparação dos dados

Definimos

$$x_i = \log_{10}(d_i), \qquad y_i = PL_{\text{medido},i}.$$

Função de custo (MMSE)

O erro em cada ponto é

$$\varepsilon_i = y_i - (A_{\mathrm{AB}} + B_{\mathrm{AB}}\, x_i),$$

e o critério de mínimos-quadrados é

$$J(A_{\mathrm{AB}}, B_{\mathrm{AB}}) = \sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \big(y_i - A_{\mathrm{AB}} - B_{\mathrm{AB}}\, x_i\big)^2.$$

Derivadas parciais

Para encontrar o mínimo, igualamos a zero as derivadas em relação a A_{AB} e B_{AB}:

$$\frac{\partial J}{\partial A_{\mathrm{AB}}} = -2 \sum_{i=1}^{N} \big(y_i - A_{\mathrm{AB}} - B_{\mathrm{AB}}\, x_i\big) = 0,$$

$$\frac{\partial J}{\partial B_{\mathrm{AB}}} = -2 \sum_{i=1}^{N} x_i \big(y_i - A_{\mathrm{AB}} - B_{\mathrm{AB}}\, x_i\big) = 0.$$

Equações normais

Distribuindo e reorganizando cada condição:

$$\sum_{i=1}^{N} y_i - N\, A_{\mathrm{AB}} - B_{\mathrm{AB}} \sum_{i=1}^{N} x_i = 0 \quad \Longrightarrow \quad N\, A_{\mathrm{AB}} + B_{\mathrm{AB}} \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} y_i,$$

$$\sum_{i=1}^{N} x_i y_i - A_{\mathrm{AB}} \sum_{i=1}^{N} x_i - B_{\mathrm{AB}} \sum_{i=1}^{N} x_i^2 = 0 \quad \Longrightarrow \quad A_{\mathrm{AB}} \sum_{i=1}^{N} x_i + B_{\mathrm{AB}} \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} x_i y_i.$$

Forma matricial

As duas equações podem ser escritas como

$$\begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} A_{\mathrm{AB}} \\ B_{\mathrm{AB}} \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}.$$

Solução explícita

Invertendo a matriz, obtemos

$$\begin{pmatrix} A_{\mathrm{AB}} \\ B_{\mathrm{AB}} \end{pmatrix} = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}.$$

Em forma explícita:

$$A_{\mathrm{AB}} = \frac{\big(\sum_i y_i\big)\big(\sum_i x_i^2\big) - \big(\sum_i x_i\big)\big(\sum_i x_i y_i\big)}{N \sum_i x_i^2 - \big(\sum_i x_i\big)^2}, \quad B_{\mathrm{AB}} = \frac{N\big(\sum_i x_i y_i\big) - \big(\sum_i x_i\big)\big(\sum_i y_i\big)}{N \sum_i x_i^2 - \big(\sum_i x_i\big)^2}.$$

Predição do modelo AB

Por fim, a predição de perda de percurso é:

$$PL_{\text{AB}}(d) = A_{\text{AB}} + B_{\text{AB}} \log_{10}(d).$$

Com essas informações e programando a lógica em Python (código em Anexo), os valores de A (Intercepto - dB) e B (Declive) foram encontrados para os três modelos FS, CI e AB. Como pode ser vista na Tabela 1:

Tabela 1: Parâmetros dos modelos de perda de percurso.

Modelo	**Intercepto (dB)**	**Declive**
FS	31,030151	20,0
CI	31,030151	28,11339
AB	44,634566	23,078437

Um gráfico comparativo com os valores de PL (dB) versus distância (km), contendo os pontos medidos, sobreposto às retas dos modelos FS, CI e AB:

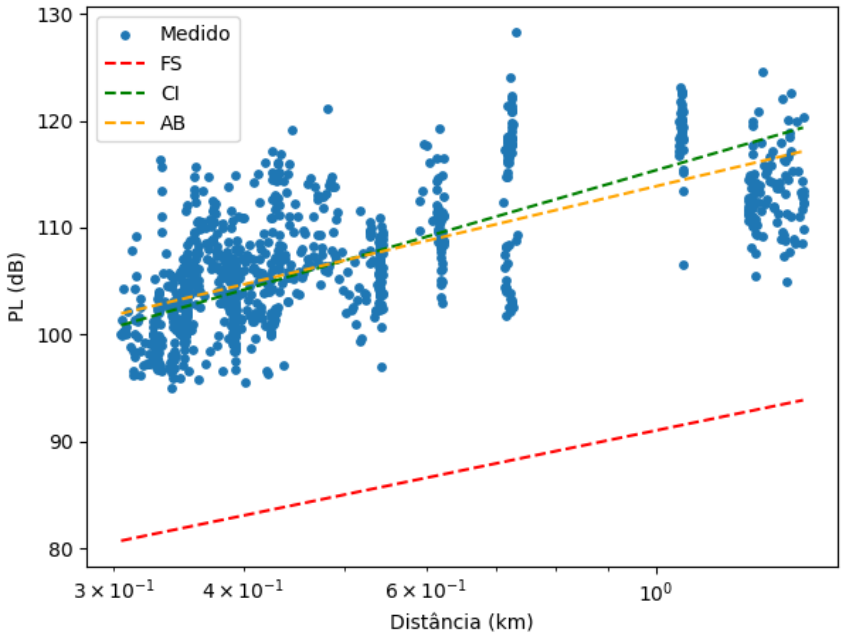

Figura 1: Perda de percurso medida e ajuste pelos modelos FS, CI e AB em função da distância.

Os histogramas com os valores de erro sobrepostos a uma variável aleatória (VA) Gaussiana que modela os valores de erro, para cada um dos modelos FS, CI e AB, onde a VA tem média e variância iguais ao conjunto de valores de erro para cada modelo:

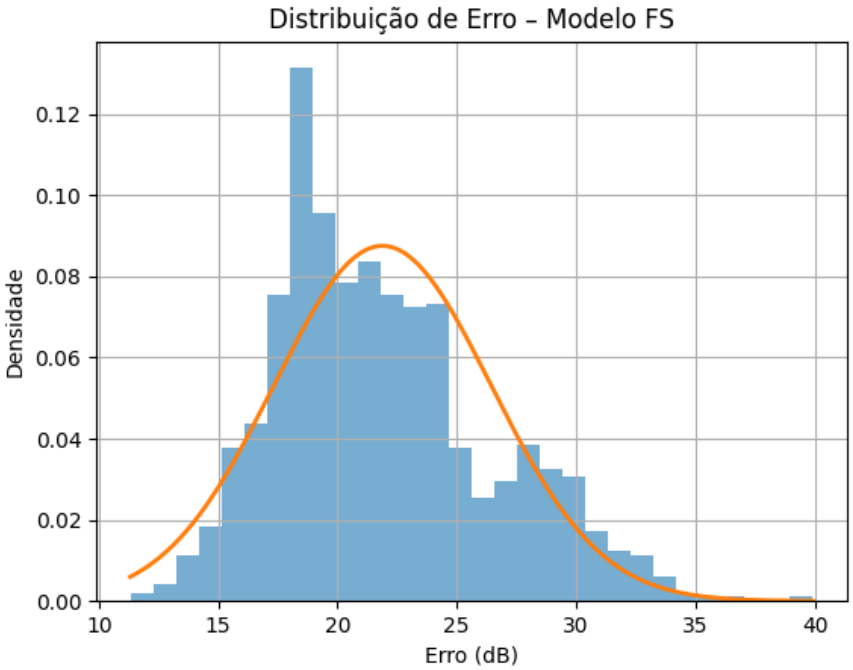

Figura 2: Histograma com os valores de erro sobrepostos a uma variável aleatória (VA) Gaussiana – Modelo FS.

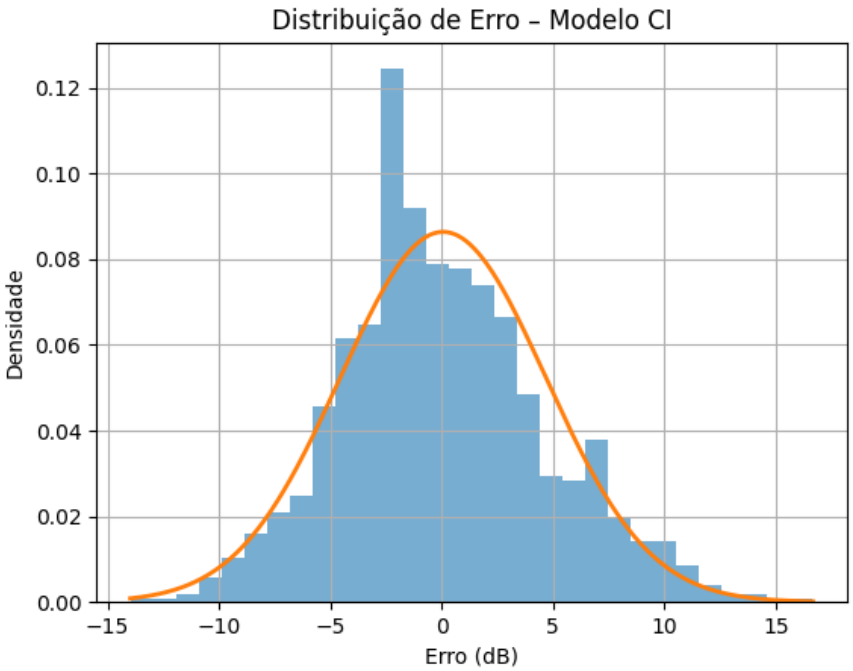

Figura 3: Histograma com os valores de erro sobrepostos a uma variável aleatória (VA) Gaussiana – Modelo CI.

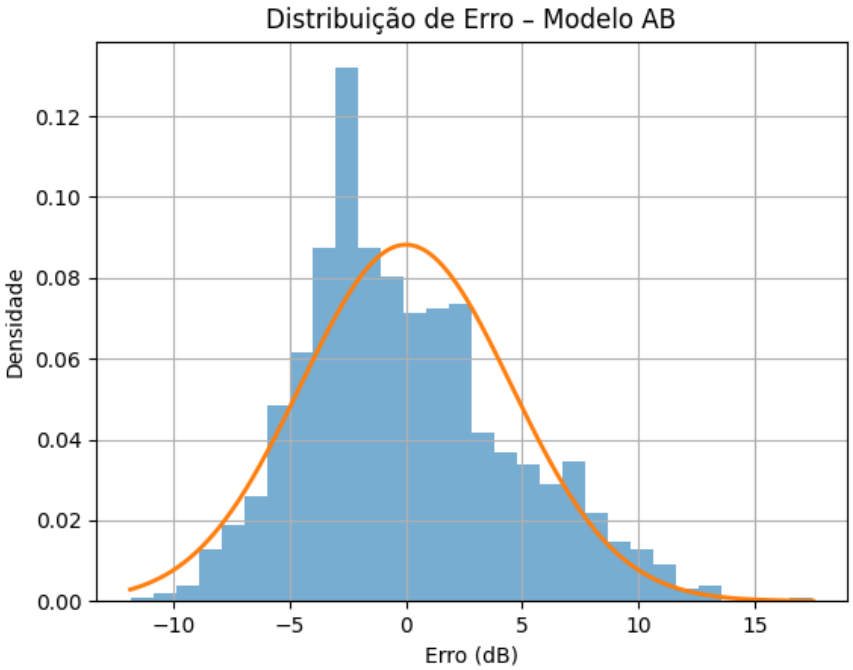

Figura 4: Histograma com os valores de erro sobrepostos a uma variável aleatória (VA) Gaussiana – Modelo AB.

Os valores encontrados para média e desvio-padrão dos erros para cada modelo é apresentado abaixo na Tabela 2:

Modelo	Média do Erro (dB)	Desvio Padrão (dB)
FS	21.88309	4.558410
CI	0.06418937	4.617720
AB	1.379695×10^{-14}	4.522632

Tabela 2: Média e desvio-padrão dos erros para cada modelo.

Dessa forma, podemos concluir que o modelo FS supõe propagação em espaço livre, ou seja, linha de visada sem obstáculos, por isso acaba subestimando as perdas em áreas urbanas repletas de prédios e interferências.

No modelo CI, mantém-se o valor inicial de atenuação (intercepto) calculado pelo modelo de espaço livre e ajusta-se apenas o expoente de distância (declive) para reduzir o desvio em relação aos dados reais, ajustado de acordo com o ambiente.

Já o modelo AB reconfigura tanto o intercepto quanto o declive a partir das medições, oferecendo maior flexibilidade para capturar características particulares do ambiente urbano, como efeitos de sombreamento e múltiplos caminhos de propagação.

O modelo AB é o mais apropriado para o cenário urbano de Icaraí, Niterói, pois seus resíduos formam uma VA Gaussiana centrada em zero e com menor variância e acompanhando mais à curva gaussiana, indicando mínimo erro sistemático e menor dispersão. O modelo CI surge como segunda opção — por manter quase zero viés mas com desvio-padrão ligeiramente maior — enquanto o FS não se aproxima dos valores reais devido ao alto viés, tendo os erros mais espalhados e não centralizados em zero, representando um desvio maior em relação à curva gaussiana.

Além disso, conforme observado no gráfico de PL em função da distância (Figura 1), o modelo AB (linha laranja) acompanha de forma os pontos medidos ao longo de toda a faixa de distâncias, representando o modelo mais adequado. Já modelo CI (linha verde) ajusta bem a inclinação mas apresenta pequenos desvios sistemáticos nas distâncias extremas, e o modelo FS (linha vermelha) é o que mais se distancia dos pontos medidos, sendo o modelo menos apropriado.

Códigos Utilizados

A seguir, apresentam-se os códigos em Python utilizados no relatório.

Questão 1

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from astropy.coordinates import Angle
import astropy.units as u
from pyproj import Geod

def dms_to_dd(deg, minutes, seconds, sign=1):
    """
    Converte DMS para decimal usando Astropy Angle.
    deg, minutes, seconds: componentes DMS
    sign: +1 para N/E, -1 para S/W
    """
    ang = Angle(f"{abs(deg)}d{minutes}m{seconds}s")
    return sign * ang.to(u.deg).value

# --- 2) coordenadas do transmissor (O) e boresight (B) --- (O) e
    boresight (B) ---
lat_O = dms_to_dd(22, 54, 9.83, sign=-1)
lon_O = dms_to_dd(43, 6, 57.63, sign=-1)
lat_B = dms_to_dd(22, 54, 16.03, sign=-1)
lon_B = dms_to_dd(43, 6, 46.11, sign=-1)

# pega o azimute de O para B usando uma lib pronta do python
geod = Geod(ellps='WGS84')
bearing_OB, back_az, dist = geod.inv(lon_O, lat_O, lon_B, lat_B)

# --- 3) leitura do CSV sem header ---
df = pd.read_csv(
    './Lat_long_dist_Pr (1).csv',
    header=None,
    names=['latitude','longitude','dist_km','Pr_dBm']
)

# --- 4) cálculo de teta, R(teta) e Gt(teta) ---
def calc_theta(row):
```

```
    # pega o azimute de O para A
    bOA, back_az, dist = geod.inv(lon_O, lat_O, row['longitude'],
        row['latitude'])
    #a diferenca do angulo para um ponto arbitrario para
       boresight para vai dar o angulo teta da imagem
    return abs(bOA - bearing_OB)

#aplica a funcao para achar o angulo teta
df['theta_deg'] = df.apply(calc_theta, axis=1)

#coeficientes do polinomio p R(teta)
coefs = np.array([
    6.68119099491645e-10,
   -1.10210602759622e-07,
    6.97748729121278e-06,
   -0.000211522819954194,
    0.00309474470699973,
   -0.0206164470242935,
   -0.00685676523907350,
   -0.000491338726848875
])
#dado no enunciado - pega os coeficientes e o angulo teta para
   cada ponto arbitrario e forma o polinomio de ordem 7
df['R_theta'] = np.polyval(coefs, df['theta_deg'])
#Ganho da antena transmissora Gt
df['Gt_dBi'] = 14.1 + df['R_theta']   # ganho Tx corrigido


# --- 5) parâmetros do enunciado ---
Pt_dBm, Gr_dBi, f_MHz = -7.0, 2.0, 850.0

# --- 6) modelos FS (Friis), CI e AB ---
c = 3e8                       # velocidade da luz (m/s)
lam = c / (f_MHz * 1e6)       # comprimento de onda (m)

df['d_m'] = df['dist_km'] * 1000

# Modelo FS usando fórmula correta do slide
# PL(dB) = -10 log10( (Gt*Gr*lambda^2) / (4pid)^2 )
# 1) coeficientes teoricos:

B_FS = 20.0
A_FS = (
    20 * np.log10(4 * math.pi / lam)
)

# 2) predição via log-distance:
df['PL_FS_pred'] = A_FS + B_FS * np.log10(df['d_m'])

# --- 7) path-loss medido para CI e AB ---
# defina antes as perdas e ganhos do seu sistema:
```

```
Lcab_tx_dB = 1.5     # perda no cabo TX (dB)
Lcab_rx_dB = 2*1.5     # perda no cabo RX (dB)
GLNA_dB    = 26.0    # ganho do LNA (dB)

df['PL_meas'] = (
    Pt_dBm
    + df['Gt_dBi']        # ganho da antena Tx
    - Lcab_tx_dB          # perda cabo Tx
    + Gr_dBi              # ganho da antena Rx
    - Lcab_rx_dB          # perda cabo Rx
    + GLNA_dB             # ganho do LNA
    - df['Pr_dBm']        # potência recebida medida
)

results = pd.DataFrame({
    'Modelo':           ['FS'],
    'Intercepto (dB)': [A_FS],
    'Declive':          [B_FS]
})
print(results.to_string(index=False))


# --- 8) Modelo CI (Close-In) ---
# Monta sistema y = Bx, com y = PL_meas - A_FS e x = log10(d/d0)
x_ci = np.log10(df['d_m']).values.reshape(-1, 1)
y_ci = (df['PL_meas'] - A_FS).values

# Resolve B pelo método dos mínimos-quadrados
B_CI, *_ = np.linalg.lstsq(x_ci, y_ci, rcond=None)

# Predição do modelo CI
df['PL_CI_pred'] = A_FS + B_CI[0] * np.log10(df['d_m'])

results = pd.DataFrame({
    'Modelo':           ['CI'],
    'Intercepto (dB)': [A_FS],
    'Declive':          [B_CI[0]]
})
print(results.to_string(index=False))


# --- Modelo AB (Floating Intercept) ---
# Monta a matriz X_ab com coluna de 1's (intercepto) e log10(d_m)
X_ab = np.column_stack([
    np.ones(len(df)),
    np.log10(df['d_m'])
])
# y_ab é o vetor de PL medido
y_ab = df['PL_meas'].values

# Resolve [A_AB, B_AB] pelo método dos mínimos-quadrados
```

```
(A_AB, B_AB), *_ = np.linalg.lstsq(X_ab, y_ab, rcond=None)

# Predição do modelo AB
df['PL_AB_pred'] = A_AB + B_AB * np.log10(df['d_m'])

# Exibe tabela de coeficientes
results_ab = pd.DataFrame({
    'Modelo':           ['AB'],
    'Intercepto (dB)': [A_AB],
    'Declive':          [B_AB]
})
print(results_ab.to_string(index=False))

# --- 8) plot comparativo de path-loss ---
plt.figure()
plt.scatter(df['dist_km'], df['PL_meas'], label='Medido', s=15)
plt.plot(df['dist_km'], df['PL_FS_pred'], '--', label='FS', color
   ='red')
plt.plot(df['dist_km'], df['PL_CI_pred'], '--', label='CI', color
   ='green')
plt.plot(df['dist_km'], df['PL_AB_pred'], '--', label='AB', color
   ='orange')
plt.xscale('log')
plt.xlabel('Distância (km)')
plt.ylabel('PL (dB)')
plt.legend()
plt.show()

# --- 1) calcular erros ---
error_FS = df['PL_meas'] - df['PL_FS_pred']
error_CI = df['PL_meas'] - df['PL_CI_pred']
error_AB = df['PL_meas'] - df['PL_AB_pred']

# --- 2) função de plotagem ---
def plot_error_with_gaussian(error, title):
    mu    = error.mean() #Calcula a media aritmetica do vetor
       error
    sigma = error.std(ddof=0)      #Calcula o desvio-padrao
       populacional de error
    # O parametro ddof=0 garante que dividimos por N, nao por N-1
    x     = np.linspace(error.min(), error.max(), 1000) #Gera um
    #array x de 1000 pontos igualmente espaçados entre o valor
       minimo e o valor maximo de error.
    # Esses pontos servem de eixo x para avaliar a curva normal.
    pdf   = norm.pdf(x, mu, sigma) # usa a pdf com mu e sigma
       para construcao

    plt.figure()
    plt.hist(error, bins=30, density=True, alpha=0.6)
    plt.plot(x, pdf, linewidth=2)
    plt.title(title)
```

```
    plt.xlabel('Erro (dB)')
    plt.ylabel('Densidade')
    plt.grid(True)
    plt.show()

# --- 3) gerar os três gráficos ---
plot_error_with_gaussian(error_FS, 'Distribuicao de Erro - Modelo
    FS')
plot_error_with_gaussian(error_CI, 'Distribuicao de Erro - Modelo
    CI')
plot_error_with_gaussian(error_AB, 'Distribuicao de Erro - Modelo
    AB')

# calcula média e desvio-padrão para cada modelo
stats = pd.DataFrame({
    'Modelo': ['FS', 'CI', 'AB'],
    'Média do Erro (dB)': [
        error_FS.mean(),
        error_CI.mean(),
        error_AB.mean()
    ],
    'Desvio Padrão (dB)': [
        error_FS.std(ddof=0),    # ddof=0 para desvio padrão
           populacional
        error_CI.std(ddof=0),
        error_AB.std(ddof=0)
    ]
})

print(stats)
```