

**Projekt z przedmiotu**  
**PODSTAWY RADIOKOMUNIKACJI**

# **RAY LAUNCHER W MOIM POKOJU**

**PRZYGOTOWAŁ:**  
**MACIEJ NIEDŹWIECKI**

# SPIIS TREŚCI

<b>1. Czym jest Ray Tracing</b>	<b>3</b>
<b>2. Główne założenia projektowe</b>	<b>3</b>
<b>3. Tłumienie ścieżki dla odbiornika</b>	
<b>poruszającego się w osi pokoju</b>	<b>4</b>
<b>4. Tłumienie ścieżki, gdy istnieją ścieżki odbite</b>	<b>7</b>
<b>5. Rozbudowujemy pokój o część o bokach b, a</b>	<b>10</b>
<b>6. Pokój stał się symetryczną podkową</b>	<b>13</b>
<b>a. Sygnał przenika przez dwie ściany i dociera do odbiornika</b>	<b>13</b>
<b>b. Sygnał ulega podwójnej dyfrakcji, modelowanej wg metody Berga</b>	<b>15</b>
<b>c. Sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii</b>	<b>17</b>
<b>7. Wnioski</b>	<b>18</b>
<b>8. Układ budynków według siatki Manhattan</b>	<b>19</b>
<b>9. Kod programu w języku Python</b>	<b>22</b>
<b>10. Bibliografia</b>	<b>22</b>

## 1. CZYM JEST RAY TRACING

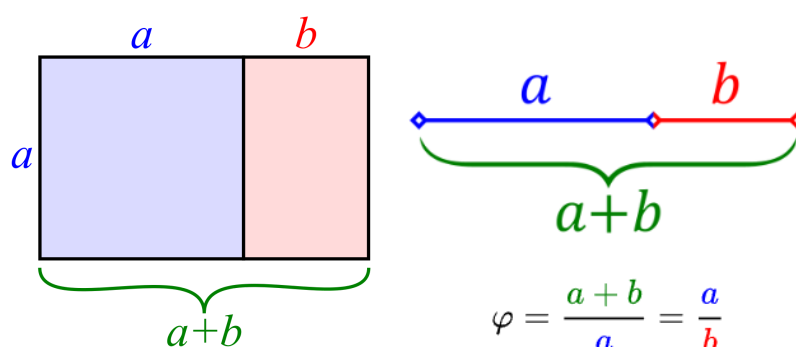
Ray tracing jest wirtualną techniką, umożliwiającą tworzenie realistycznych obrazów poprzez symulowanie przepływu promieni światła w określonym środowisku. Jest on szeroko stosowany w wielu dziedzinach do generowania obrazów np. w grach komputerowych, filmach i innych aplikacjach, w których wymagana jest wysoka jakość grafiki. Może być również wykorzystywany w różnego rodzaju symulatorach, takich jak symulatory lotów czy wirtualne wycieczki po budynkach. Ray tracing działa poprzez symulowanie rozchodzenia światła po scenie, uwzględniając różne elementy takie jak odbicie, rozproszenie i załamanie światła, co pozwala na uzyskanie bardzo realistycznych obrazów.

## 2. GŁÓWNE ZAŁOŻENIA PROJEKTOWE

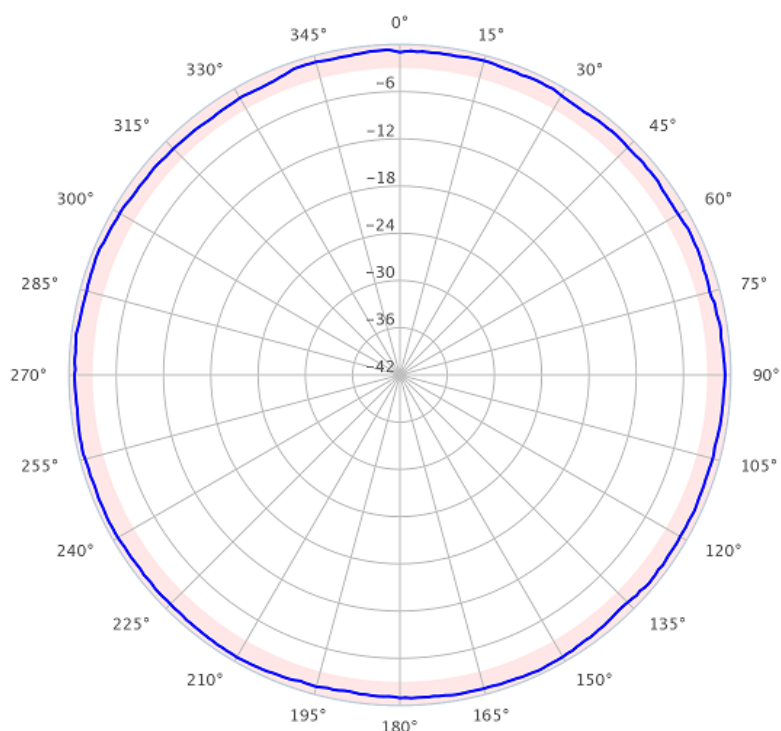
Przyjmujemy rozmiar pokoju według proporcji tzw. złotego podziału. Boki pokoju mają długość,  $a$  i  $b$ , gdzie  $\frac{a+b}{a} = \frac{a}{b}$ . Minimalna przyjęta przez nas wartość boku  $a$  wynosi 12 m.

Na środku krótszej ściany (o długości  $a$ ) montujemy punkt dostępowy transmitujący dla częstotliwości 5 GHz.

Przyjmujemy charakterystykę anteny nadawczej i odbiorczej jako dookólną o zysku 1 (w skali liniowej). Antena w każdym kącie azymutu i elewacji ma taki sam zysk.



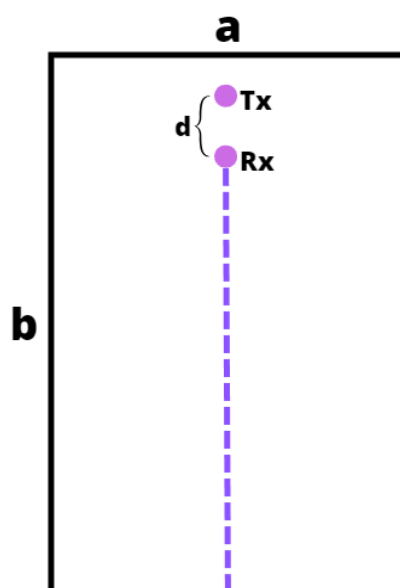
Rys. 1. Zasada złotego podziału.



Rys. 2. Charakterystyka promieniowania anteny dookólnej.

### 3. Tłumienie ścieżki dla odbiornika poruszającego się w osi pokoju

Wyznaczamy wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju (osi równoległej do dłuższej ściany, przecinającej punkt położenia anteny).

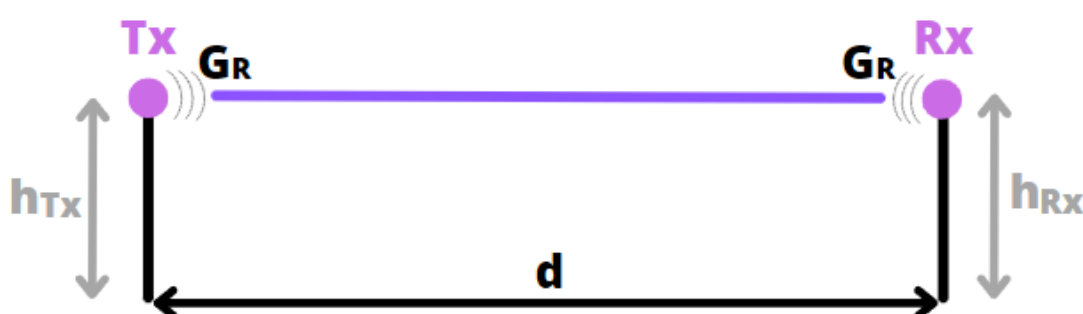


Rys. 3. Schemat pokoju i ścieżka poruszającego się w osi pokoju odbiornika.

Na powyższym rysunku został ukazany schemat pokoju wraz ze ścieżką poruszającego się w osi pokoju odbiornika, gdzie:

- Tx - antena nadawcza
- Rx - antena odbiorcza
- d - odległość między antenami [m]
- $h_{Tx} = h_{Rx}$
- $f = 5 \text{ GHz}$

Wyznaczając wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju korzystamy z poniższych wzorów i przekształceń.



Rys. 4. Schemat tłumienia ścieżki.

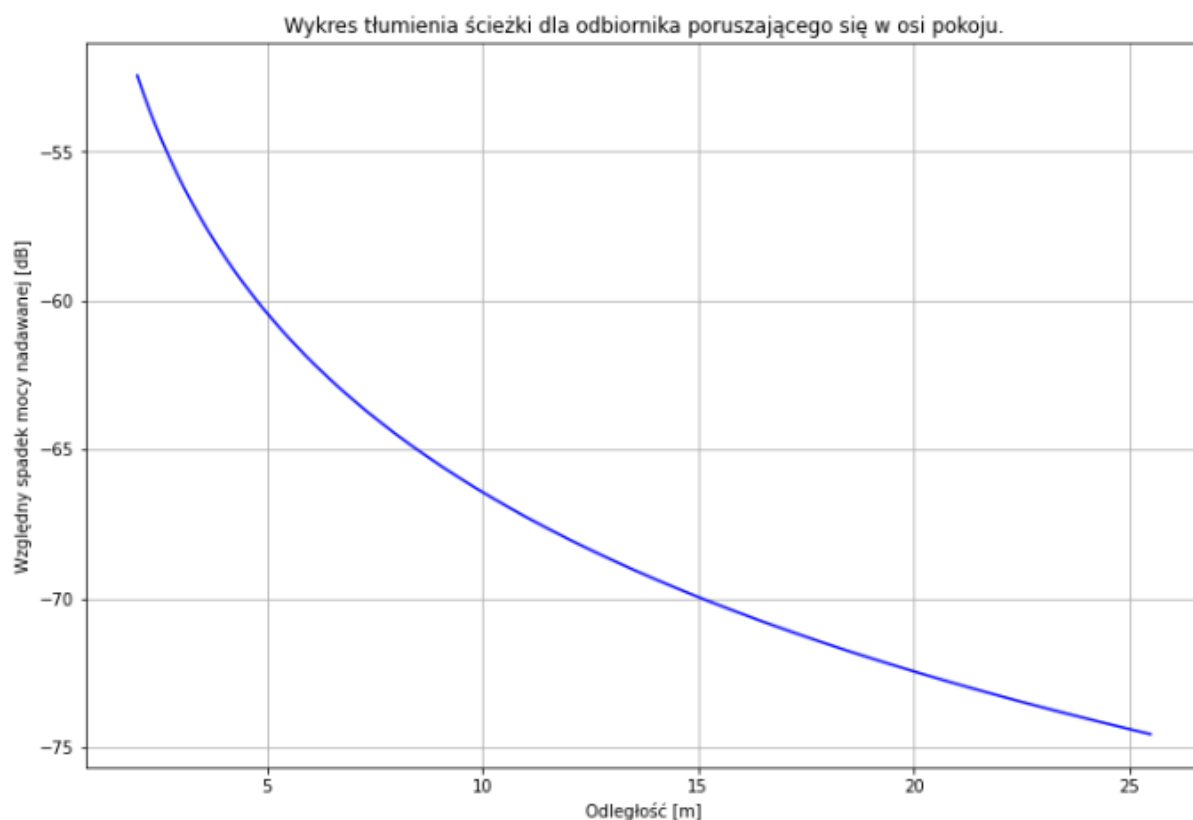
$$\frac{P_R}{P_T} = G_T G_R \left( \frac{\lambda}{4\pi d} \right)^2$$

$$\frac{P_R}{P_T} [dB] = 10 \log \left( \frac{P_R}{P_T} \right)$$


$$\lambda = \frac{c}{f}$$

Zgodnie z założeniami projektu charakterystykę anteny nadawczej i odbiorczej przyjmujemy jako dookólną. Zatem w wyliczeniach parametry  $G_T$  i  $G_R$  wynoszą 1. Ponadto przyjęto wymiary pokoju jako 17 x 27,5 [m].

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres badanego tłumienia.



Poniżej znajduje się fragment kodu odpowiadający za realizację badanego tłumienia.

 RAY LAUNCHER W MOIM POKOJU

## Założenia projektowe

Przyjmij rozmiar pokoju według proporcji tzw. złotego podziału, tzn. boki pokoju mają długość  $a$ ,  $b$ , gdzie  $(a+b)/a = a/b$ . Na środku krótszej ściany (o długości  $a$ ) zamontowałeś punkt dostępowy transmitujący dla częstotliwości 5 GHz. Przyjmij charakterystykę anteny nadawczej i odbiorczej jako dookólną o zysku 1 (w skali liniowej). Minimalna wartość boku  $a$  to 12 m.

```
[1] import numpy as np
    import math
    import matplotlib.pyplot as plt

[3] a = 17 # krótsza ściana
    b = 27.506578 # dłuższa ściana

    k = 1 # odległość anten od krótszej ściany [m]
    hA = a/2 # wysokość anteny nadawczej i odbiorczej od ściany b

    # zakres odległości anten <dmin, dmax>
    dmin = k
    dmax = b-k

    c = 299792458 #m/s
    f = 5 *10**9 # częstotliwość fali radiowej [GHz]

    ni_wall = 5.24 #współczynnik dla drewnianej ściany

    lp = 1000 #częstotliwość pomiarów (co mm)
```

Rys. 5. Część programu, w którym deklarujemy główne zmienne.

## A.1 Dla odbiornika poruszającego się w osi pokoju

A.1 Wyznacz wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju (osi równoległej do dłuższej ściany, przecinającej punkt położenia anteny)

```
[10] lamb = c/f
    print("Długość fali:")
    print(lamb, "[m] =", np.round(lamb*100,3), "[cm]")

    zakres = dmax - dmin
    d = np.arange(dmin, dmax, zakres/(lp-1), dtype=float)

    PrPt_ = 1*1*((lamb/(4*np.pi*d))**2)
    PrPt = np.abs(PrPt_)
    PrPt = 10*np.log10(PrPt)

    Długość fali:
    0.0599584916 [m] = 5.996 [cm]

[13] plt.figure(figsize=(12,8))
    plt.title("Wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju.")
    plt.xlabel("Odległość [m]")
    plt.ylabel("Względny spadek mocy nadawanej [dB]")
    plt.plot(d, PrPt, color="blue")
    plt.grid()
```

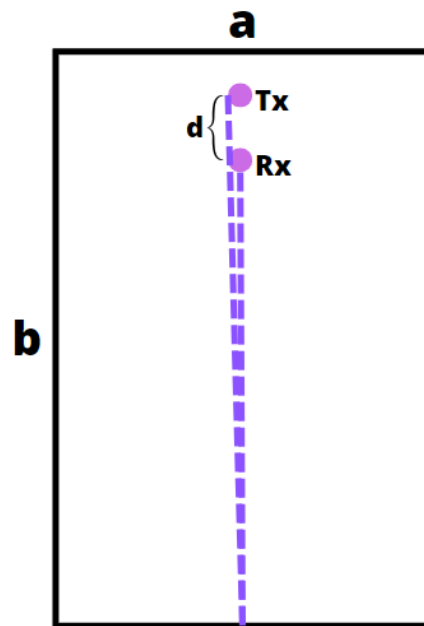
Rys. 6. Część programu, w której wyznaczamy wykres tłumienia dla odbiornika poruszającego się w osi pokoju.

## 4. Tłumienie ścieżki, gdy istnieją ścieżki odbite.

Wyznaczamy wykres tłumienia ścieżki, gdy istnieją także ścieżki odbite:

- 1D – od przeciwległej ściany.

Współczynniki odbicia przyjmujemy według normy P.2040, Przenikalność elektryczną materiałów budowlanych przyjmujemy także według normy (rodzaj materiałów dobieramy samodzielnie).



Rys. 7. Schemat pokoju i ścieżka poruszającego się w osi pokoju odbiornika (rysunek pomocniczy, w celu zobrazowania badanego problemu).

Wyznaczając wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju korzystamy z poniższych wzorów i przekształceń.

Wzory zostały wybrane z rekomendacji ITU-R P.2040

$$\frac{P_R}{P_o} = \left| \frac{1}{d} e^{-j\varphi_1} + \frac{a_{wall}}{2b-d} e^{-j\varphi_2} \right|^2$$

Współczynnik odbicia (wzór z rekomendacji nr 37a, str. 13):

$$R_{ETM} = \frac{\eta \cos \theta - \sqrt{\eta - \sin^2 \theta}}{\eta \cos \theta + \sqrt{\eta - \sin^2 \theta}}$$

Polaryzacja równoległa:

Skorzystaliśmy z tabeli zawartej rekomendacji (tabela nr 3, str. 23) i wybraliśmy interesujący nas współczynnik. W przypadku rozważanego przypadku został dobrany współczynnik betonowej ściany.



Material properties

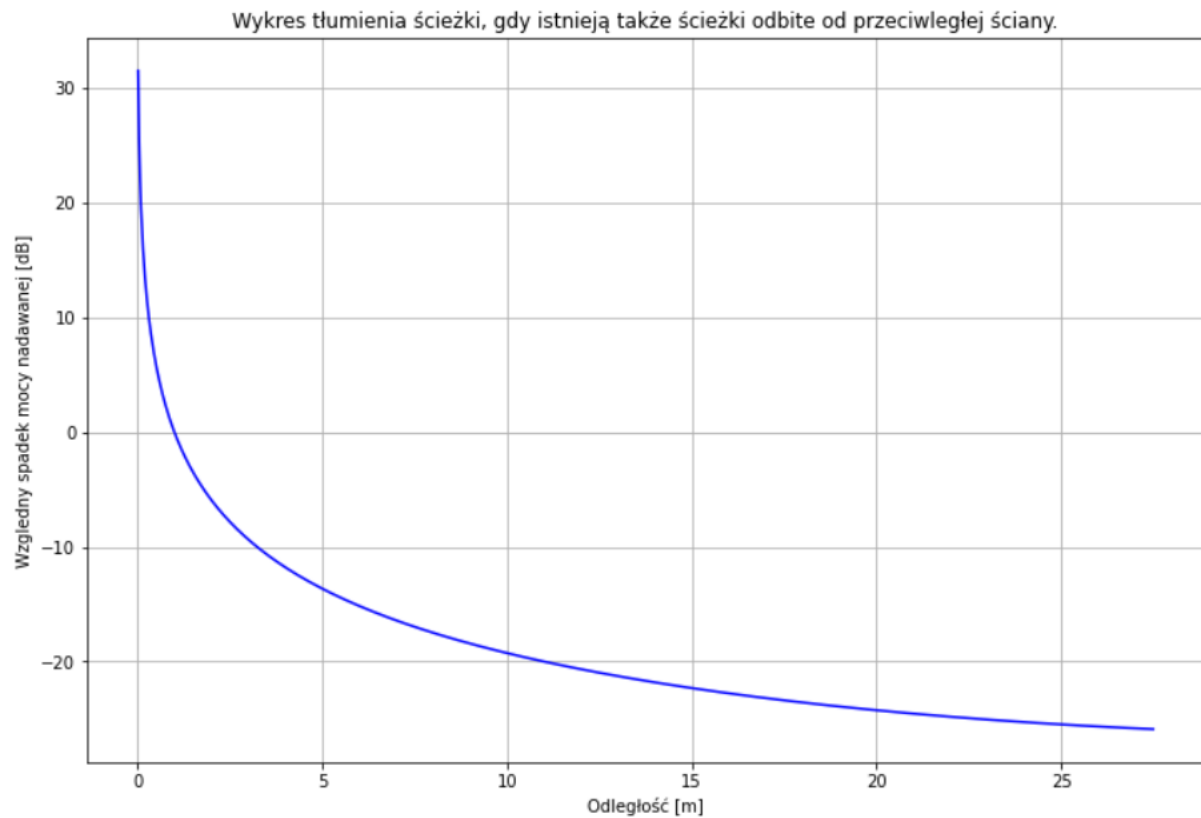
Material class	Real part of relative permittivity		Conductivity S/m		Frequency range
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	
Vacuum ( $\approx$ air)	1	0	0	0	0.001-100
Concrete	5.24	0	0.0462	0.7822	1-100
Brick	3.91	0	0.0238	0.16	1-40
Plasterboard	2.73	0	0.0085	0.9395	1-100
Wood	1.99	0	0.0047	1.0718	0.001-100
Glass	6.31	0	0.0036	1.3394	0.1-100
Glass	5.79	0	0.0004	1.658	220-450
Ceiling board	1.48	0	0.0011	1.0750	1-100
Ceiling board	1.52	0	0.0029	1.029	220-450
Chipboard	2.58	0	0.0217	0.7800	1-100
Plywood	2.71	0	0.33	0	1-40
Marble	7.074	0	0.0055	0.9262	1-60
Floorboard	3.66	0	0.0044	1.3515	50-100
Metal	1	0	$10^7$	0	1-100
Very dry ground	3	0	0.00015	2.52	1-10 only
Medium dry ground	15	-0.1	0.035	1.63	1-10 only
Wet ground	30	-0.4	0.15	1.30	1-10 only

Stąd:

$$\eta = 5,24$$

$$\theta = 0^\circ$$

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres badanego tłumienia.



Poniżej znajduje się fragment kodu odpowiadający za realizację badanego tłumienia.

## A.2 Gdy istnieją także ścieżki odbite

A.2 Wyznacz wykres tłumienia ścieżki, gdy istnieją także ścieżki odbite:

- 1D – od przeciwległej ściany

Współczynniki odbicia przyjmij według normy P.2040, Przenikalność elektryczną materiałów budowlanych przyjmij także według normy (rodzaj materiałów dobierz samodzielnie)

```
[25] a_wall = (ni_wall - np.sqrt(ni_wall))/(ni_wall + np.sqrt(ni_wall)) #obliczenie współczynnika odbicia od ściany
      print("Współczynnik a_wall:")
      print(a_wall)

      dmin = 0
      dmax = b-k

      zakres = dmax - dmin
      d0 = np.arange(dmin, dmax+k, zakres/(lp-1), dtype=float) #nowe d ze względu na odbicie od ściany

      fi1 = np.pi*(2)*d0/c #faza bez odbicia
      fi2 = np.pi*(2)*(2*b-d0)/c #faza z odbiciem

      e1 = np.exp(-1j*fi1)
      e2 = np.exp(-1j*fi2)

      PrPt1 = np.abs((1/d0*e1 + a_wall/(2*b-d0)*e2))
      PrPt1 = 20*np.log10(PrPt1)

      Współczynnik a_wall:
      0.39193177903211346
```

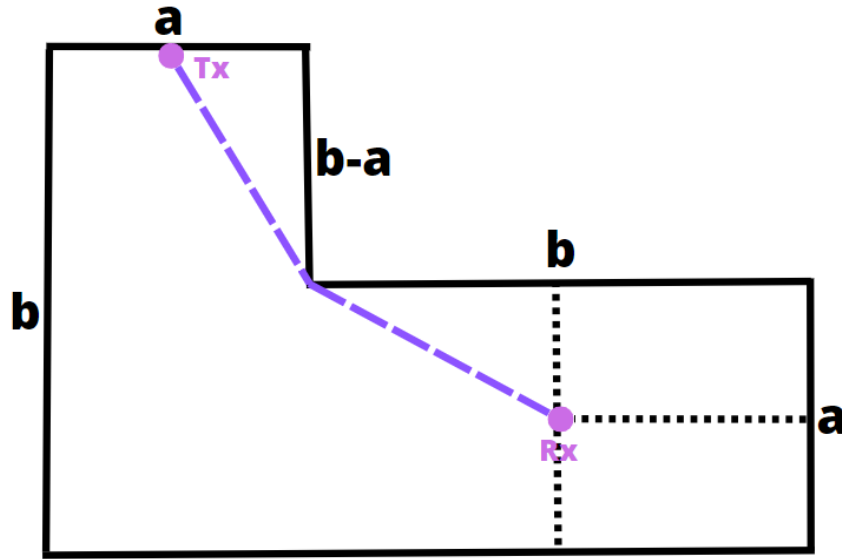
```
[24] plt.figure(figsize=(12,8))
      plt.title("Wykres tłumienia ścieżki, gdy istnieją także ścieżki odbite od przeciwległej ściany.")
      plt.xlabel("Odległość [m]")
      plt.ylabel("Względny spadek mocy nadawanej [dB]")
      plt.plot(d0,PrPt1, color="blue")
      plt.grid()
```

Rys. 8. Część programu, w której wyznaczamy wykres tłumienia dla tłumienia, gdy istnieją ścieżki odbite.

## 5. Rozbudowujemy pokój o część o bokach b, a

Rozbudowujemy pokój o część o bokach b, a. Odbiornik porusza się w osi tej części. Wyznaczamy stratę dyfrakcji według metody Deygout. Przyjmujemy występowanie dyfrakcji wyłącznie, gdy odbiornik nie ma bezpośredniej widoczności z nadajnikiem.

- 1D – Wyznaczamy wartość straty dyfrakcji w osi dobudowanej części.



Rys. 9. Schemat powiększonego pokoju oraz ścieżka poruszającego się w osi pokoju odbiornika.

Wyznaczając wykres tłumienia ścieżki dla odbiornika poruszającego się w osi pokoju korzystamy z poniższych wzorów, przekształceń i funkcji.

$$S_1 + S_2 = \sqrt{\left(b - \frac{1}{2}a\right)^2 + \left(d - \frac{1}{2}a\right)^2}$$

$$r_1 = \text{const.} = \sqrt{\left(\frac{1}{2}a\right)^2 + (b - a)^2}$$

$$r_2 = \text{const.} = \sqrt{\left(\frac{1}{2}a\right)^2 + (d - a)^2}$$

Z twierdzenia cosinusów:

$$r_2^2 = r_1^2 + (s_1 + s_2)^2 - 2r_1(s_1 + s_2)\cos\alpha$$

$$\alpha = \arccos\left(\frac{r_2^2 - r_1^2 - (s_1 + s_2)^2}{-2r_1(s_1 + s_2)}\right)$$

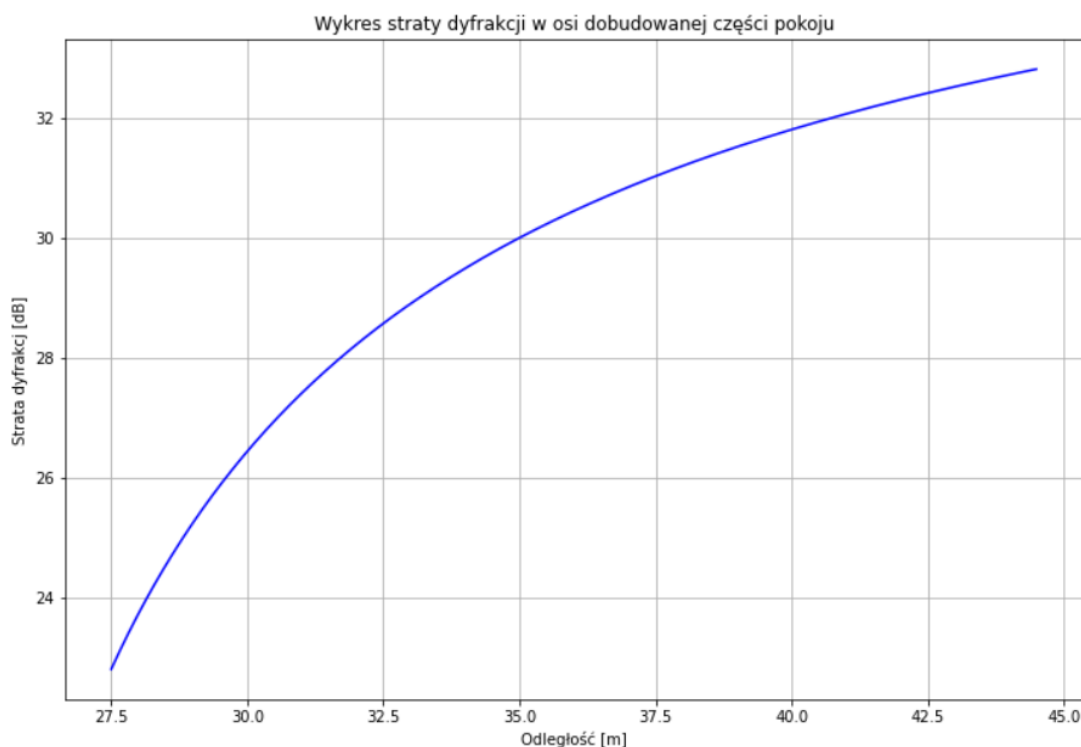
$$h = r_1 \frac{n}{r_1} = r_1 \sin\alpha$$

Obliczenie straty dyfrakcji C:

$$v = h \sqrt{\frac{2}{\lambda} \frac{s_1 + s_2}{r_1 r_2}}$$

$$C = 6,9 + 20 \log(\sqrt{(v - 0,1)^2 + 1} + v - 0,1)$$

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres straty dyfrakcji w osi dobudowanej części pokoju.



Poniżej znajduje się fragment kodu odpowiadający za realizację badanej straty dyfrakcji.

### A.3 Rozbudowujemy pokój o część o bokach $b$ , $a$ . Odbiornik porusza się w osi tej części

A.3 Wyznacz stratę dyfrakcji według metody Deygout. Przyjmij występowanie dyfrakcji wyłącznie, gdy odbiornik nie ma bezpośredniej widoczności z nadajnikiem.

- 1D – wyznaczyć wartość straty dyfrakcji w osi dobudowanej części

```
[30] dmin = b
      dmax = a+b
      d= np.arange(dmin, dmax, zakres/(lp-1),dtype=float)

      S = np.sqrt((b-1/2*a)**2+(d-1/2*a)**2) # wektor przechodzący przez ścianę

      r1 = np.sqrt((b-a)**2 + (1/2*a)**2) # przeciwprostokątna trójkąta r1
      r2 = np.sqrt((1/2*a)**2 + (a-d)**2) # przeciwprostokątna trójkąta r2

      alpha = np.arccos((r2**2 - r1**2 - S**2)/(-2*r1*S))

      # Wyznaczenie straty dyfrakcji w osi dobudowanej pokoju (na rogu ściany)
      h = r1*np.sin(alpha)
      v = h*np.sqrt(2/lamb * S/(r1*r2))

      C = 6.9 + 20*np.log10(np.sqrt((-0.1)**2 + 1) + v-0.1)

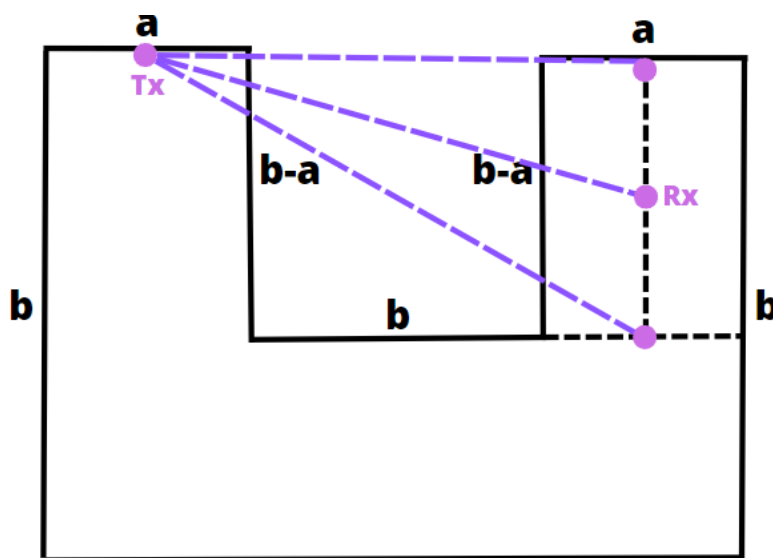
[31] # plot
      plt.figure(figsize=(12,8))
      plt.title("Wykres straty dyfrakcji w osi dobudowanej części pokoju")
      plt.xlabel("Odległość [m]")
      plt.ylabel("Strata dyfrakcji [dB]")
      plt.plot(d, C, color="blue")
      plt.grid()
```

Rys. 10. Część programu, w której wyznaczamy stratę dyfrakcji.

## 6. Pokój stał się symetryczną podkową

Kolejna rozbudowa, tak, aby pokój stał się symetryczną podkową. Wyznaczamy wartość tłumienia ścieżki dla trzech wariantów radiowych.

### a. Sygnał przenika przez dwie ściany i dociera do odbiornika

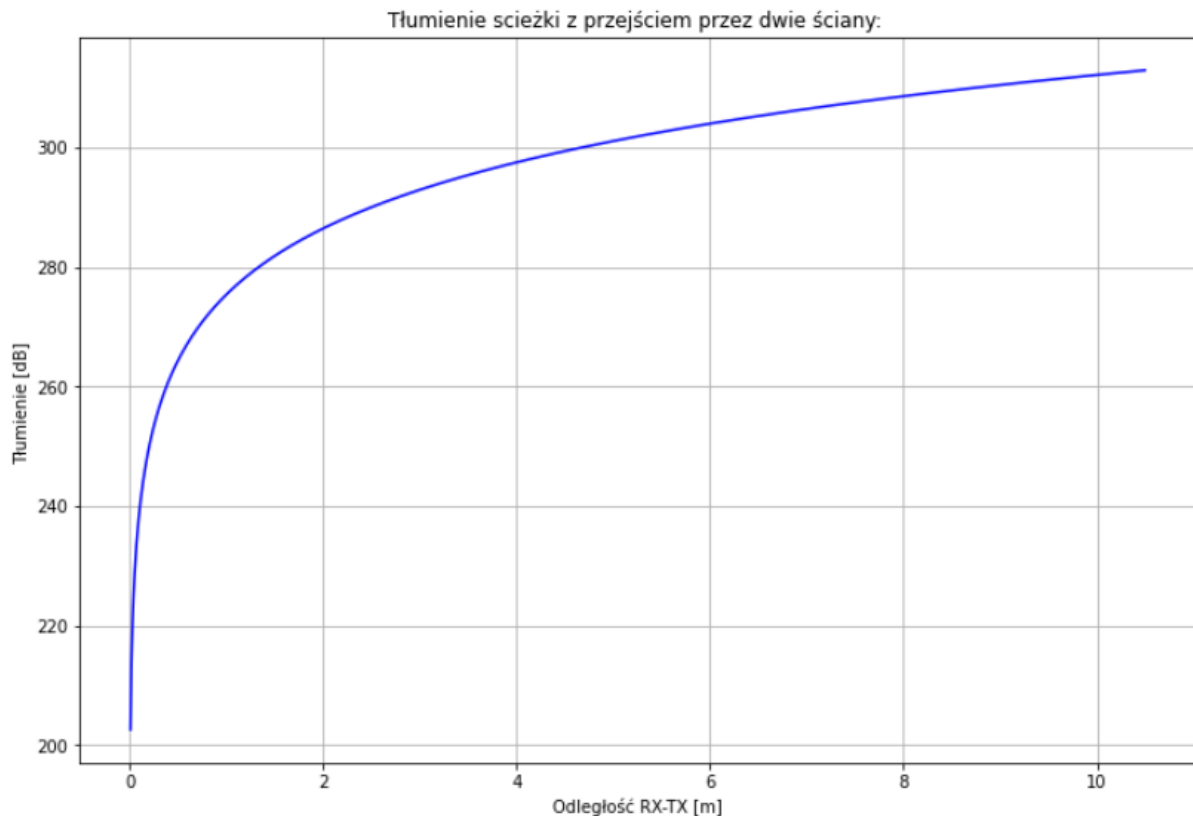


Rys. 11. Schemat powiększonego pokoju oraz ścieżka poruszającego się w osi pokoju odbiornika, gdy sygnał przenika przez dwie ściany.

W badanym przypadku wykorzystujemy model propagacji wewnątrzbudynkowej.

Scenariusz	Path Loss (dB)	Zaniki wolne (dB)	Uwagi
<b>indoor office</b>			
LOS	$PL = 18.7\lg(d) + 46.8 + 20\lg(f_c)$	$\sigma = 3$	$3\text{ m} < d < 100$
NLOS	$PL = 36.8\lg(d) + 43.8 + 20\lg(f_c)$	$\sigma = 4$	$3\text{ m} < d < 100$
wall-penetration	light wall: $A_{wall} = 5(n_w - 1)$ $n_w$ is the number of walls heavy wall: $A_{wall} = 12(n_w - 1)$ $n_w$ is the number of walls	–	$PL_{base}$ should be considered
floor penetration	$A_{floor} = 17(n_f - 1)$ $n_f$ is the number of floor	–	$PL_{base}$ should be considered
<b>indoor hotspot</b>			
LOS	$PL = 13.9\lg(d) + 64.4 + 20\lg(f_c)$	$\sigma = 3$	$5\text{ m} < d < 100$
NLOS	$PL = 37.8\lg(d) + 36.5 + 23\lg(f_c)$	$\sigma = 4$	$5\text{ m} < d < 100$

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres badanego tłumienia.



Poniżej znajduje się fragment kodu odpowiadający za realizację badanego tłumienia.

#### A.4.1 Sygnał przenika przez dwie ściany i dociera do odbiornika

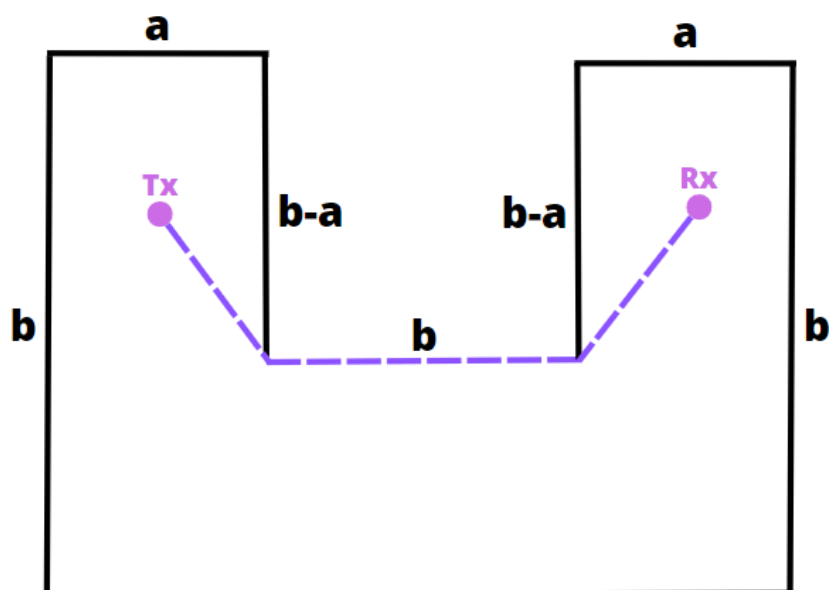
```
[39] d_min = 0
      d_max = b-a
      zakres = d_max - d_min
      d=np.arange(d_min, d_max, zakres/(lp-1),dtype=float)

      PL = np.ones(len(d))
      odl = np.ones(len(d))
      odl = d_min + d*d_max
      PL = 36.8 * np.log10(odl) + 43.8 + 20 * np.log10(f)

[42] plt.figure(figsize=(12,8))
      plt.title('Tłumienie ścieżki z przejściem przez dwie ściany: ')
      plt.xlabel('Odległość RX-TX [m]')
      plt.ylabel('Tłumienie [dB] ')
      plt.plot(d, PL, color = "blue")
      plt.grid()
      plt.show()
```

Rys. 12. Część programu, w której wyznaczamy badane tłumienie.

## b. Sygnał ulega podwójnej dyfrakcji, modelowanej wg metody Berga



Rys. 13. Schemat, gdy sygnał ulega podwójnej dyfrakcji, modelowanej według metody Berga.

Wyznaczając wykres tłumienia ścieżki, gdy sygnał ulega podwójnej dyfrakcji, modelowanej według metody Berga korzystamy z poniższych wzorów, przekształceń i funkcji.

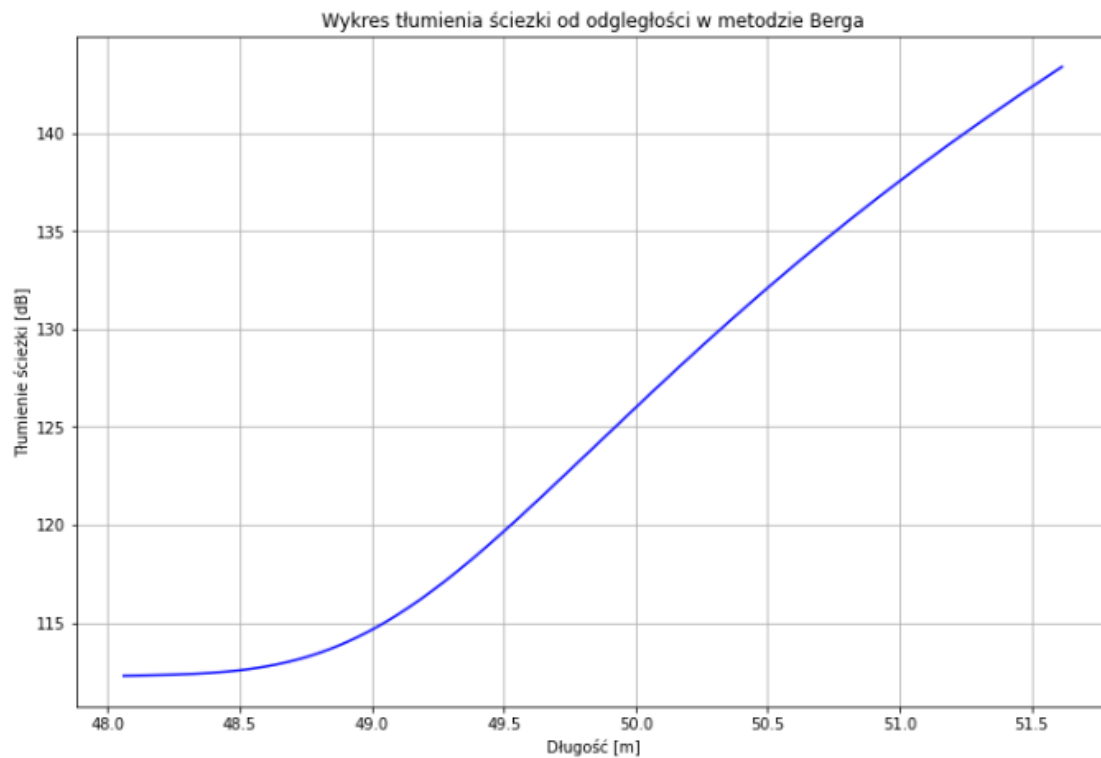
$$d_j = k_j s_{j-1} + d_{j-1}$$

$$k_j = k_{j-1} + d_{j-1} q_{j-1}$$

$$q_j = q_{90} \frac{(\deg \theta_j)}{(\deg 90)}$$

$$L_j = 20 \log \left( \frac{4\pi d}{\lambda} \right)$$

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres badanego tłumienia.



Poniżej znajduje się fragment kodu odpowiadający za realizację badanego tłumienia.

#### A.4.2 Sygnał ulega podwójnej dyfrakcji, modelowanej tym razem według metody Berga

```
[57] s0 = ((b/2)**2+(a-b)**2)**(1/2)
      s1 = a
      s2 = np.linspace(b/2, s0, 400)
      s = s0+s1+s2
      k0 = 1
      d0 = 0
      k1 = 1
      d1 = s0
      qlambda = 0.031
      alfa = 90-math.atan((b/2)/(a-b))*(180/math.pi)
      q90 = (qlambda/lamb)**(1/2)
      q1 = q90*(alfa/90)**v
      k2 = k1+d1*q1
      d2 = k2*s1+d1
      beta = np.linspace(0, alfa, 400)
      q2 = [0]*400
      k3 = [0]*400
      d3 = [0]*400

      for i in range(400):
          q2[i] = q90*(beta[i]/90)**v
          k3[i] = k2+d2*q2[i]

      for i in range(400):
          d3[i] = k3[i]*s2[i]+d2

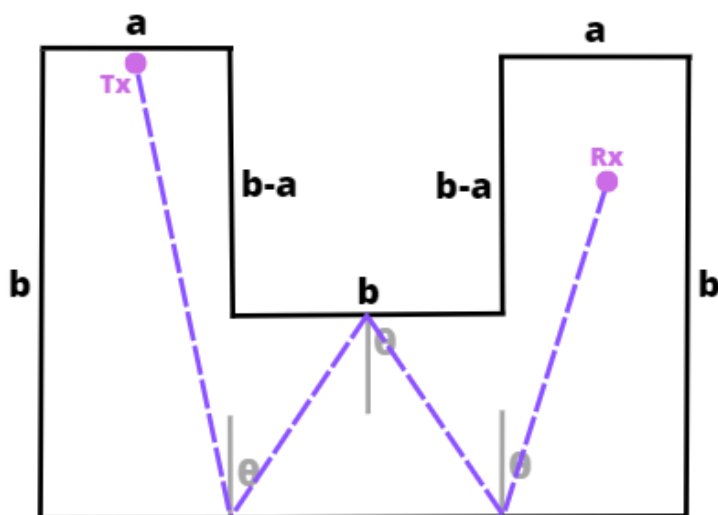
      A = [0]*400
      for i in range(400):
          A[i] = 20*math.log10(4*math.pi*d3[i]/lamb)

[58] plt.figure(figsize=(12,8))
      plt.title("Wykres tłumienia ścieżki od odległości w metodzie Berga")
      plt.xlabel("Długość [m]")
      plt.ylabel("Tłumienie ścieżki [dB]")
      plt.plot(s, A, color = "blue")
      plt.grid()
```

Rys. 14. Część programu, w której wyznaczamy badane tłumienie.



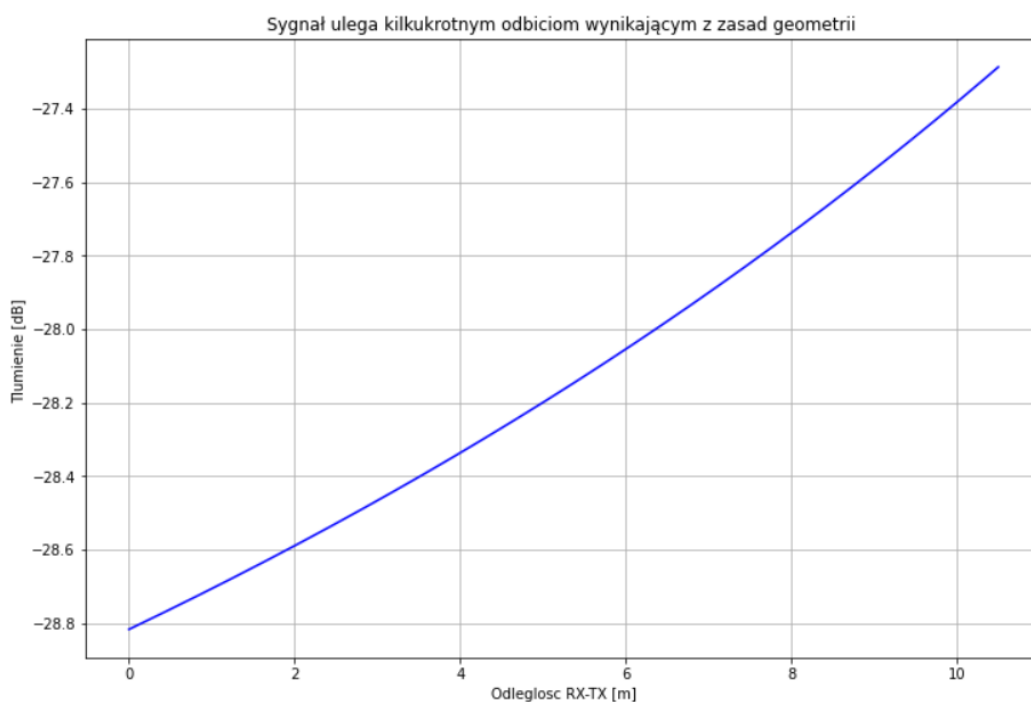
### c. Sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii



Rys. 15. Schemat, gdy sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii.

Wyznaczając wykres tłumienia ścieżki, gdy sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii korzystamy z poniższych wzorów, przekształceń i funkcji.

$$\frac{P_R}{P_O} = \left| \frac{(a_{wall})^3}{d_1} \right|^2$$



#### A.4.3 Sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii

```
[22] d_min = 0
    d_max = b-a
    zakres = d_max - d_min

    d=np.arange(d_min, d_max, zakres/(lp-1),dtype=float)

    h1 = b*np.ones(len(d))
    h2 = a*np.ones(len(d))
    h3 = a*np.ones(len(d))
    h4 = (b-d)*np.ones(len(d))

    d1 = np.sqrt((1/2*a)**2 + h1**2)
    d2 = np.sqrt((1/2*b)**2 + h2**2)
    d3 = np.sqrt((1/2*b)**2 + h2**2)
    d4 = np.sqrt((1/2*a)**2 + h4**2)

    a_wall = (ni_wall*np.cos(theta) - np.sqrt(ni_wall**2 - (np.sin(theta)**2)))/(ni_wall*np.cos(theta) + np.sqrt(ni_wall**2 - (np.sin(theta)**2))) # współczynnik odbicia od ściany

    fi1 = np.pi*(-2)*f*d1/c #faza bez odbicia
    fi2 = np.pi*(-2)*f*d2/c # faza z odbiciem od przeciwległej ściany
    fi3 = np.pi*(-2)*f*d3/c # faza odbicia od ściany bocznej

    e1=np.exp(1j*fi1)
    e2=np.exp(1j*fi2)
    e3=np.exp(1j*fi3)

    PrPt4 = np.abs(a_wall/d1*e1 + a_wall/d2*e2 + a_wall/d3*e2 + a_wall/d4*e3)
    PrPt4 = 20*np.log10(PrPt4)

[23] plt.figure(figsize=(12,8))
    plt.title('Sygnał ulega kilkukrotnym odbiciom wynikającym z zasad geometrii ')
    plt.xlabel('Odleglosc RX-TX [m]')
    plt.ylabel('Tlumienie [dB] ')
    plt.plot(d, PrPt4, color = "blue")
    plt.grid()
```

Rys. 16. Część programu, w której wyznaczamy badane tłumienie.

## 7. Wnioski

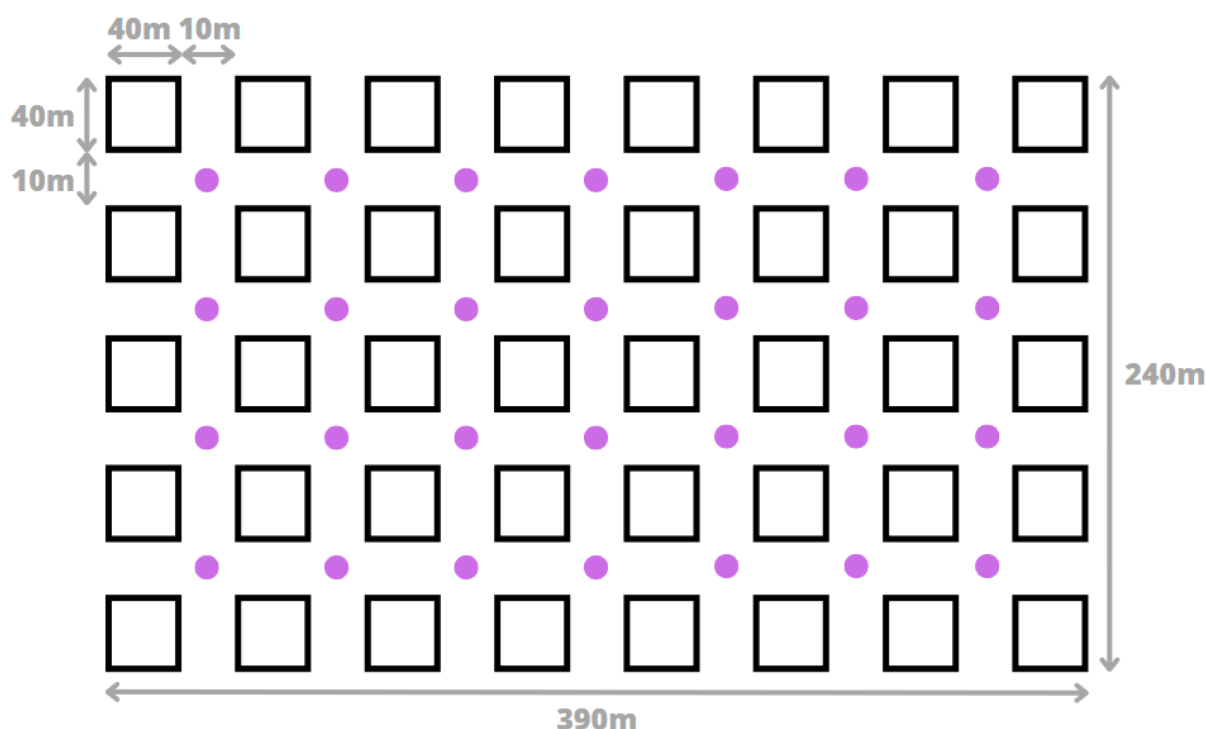
Obliczanie wartości tłumienia sygnału jest procesem opartym na analizie geometrii. Im większa jest odległość między nadajnikiem a odbiornikiem, tym większe otrzymujemy wartości tłumienia. Kształt wykresu strat dyfrakcji jest uzależniony od kąta załamania fal radiowych.

Symulowanie pełnej charakterystyki środowiska jest praktycznie niemożliwe, ponieważ powinniśmy uwzględnić nieskończoną liczbę sygnałów. Dlatego wszystkie przeprowadzone symulacje uwzględniają tylko podstawowe zjawiska propagacyjne w warunkach wyidealizowanych i nie oddają pełni rzeczywistych warunków.

## 8. Układ budynków według siatki Manhattan

Przyjmujemy układ budynków według siatki Manhattan. Dla wylosowanego rozkładem jednostajnym położenia nadajnika, wyznaczamy wartość tłumienia ścieżki (w wariancie 2D, dla stałej wysokości położenia odbiornika - według wybranego modelu propagacyjnego O2O:

- mmMagic



Rys. 17. Schemat zastosowanej siatki Manhattan wraz z zaznaczonymi możliwościami położenia nadajnika.

W celu realizacji modelu propagacyjnego skorzystaliśmy z poniższych wzorów:

Wzór na  $PL_{LOS}$  zostanie użyty w przypadku, gdy nadajnik "widzi się z odbiornikiem".

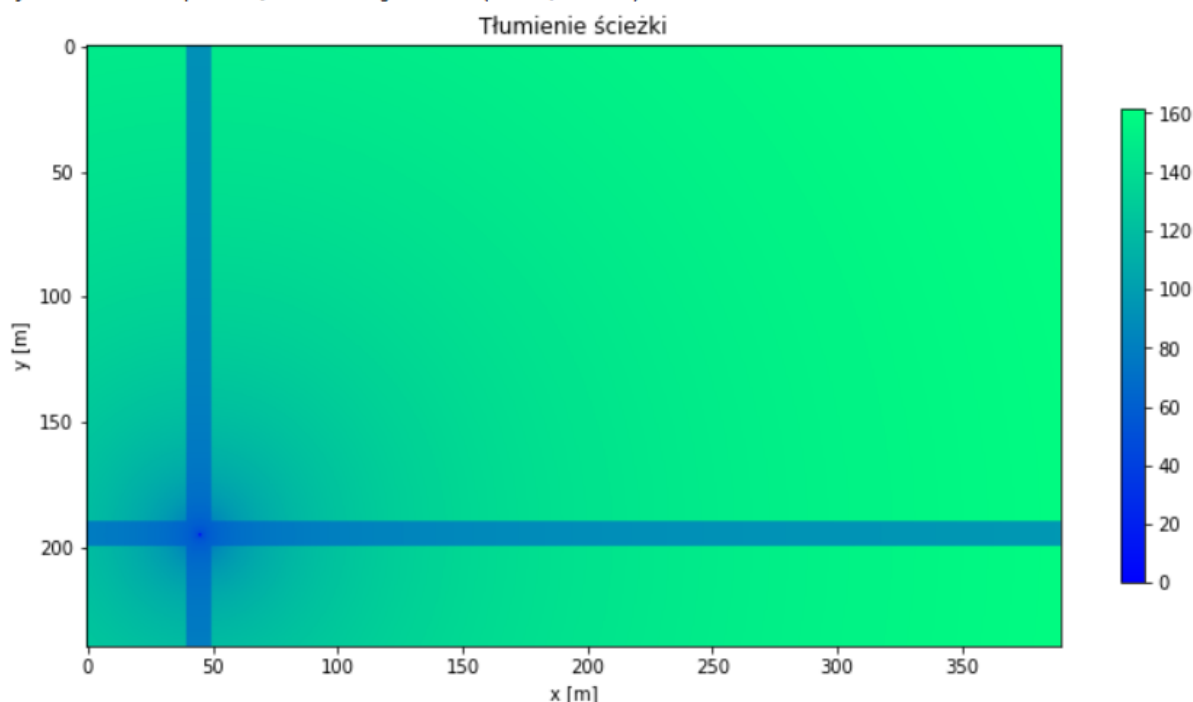
$$PL_{LOS} = 19,2 * \log_{10}(d) + 32,9 + 20,8 * \log_{10}(f)$$

Natomiast tłumienie  $PL_{NLOS}$  jest wyliczane w przypadku, gdy nadajnik i odbiornik "się nie widzą".

$$PL_{NLOS} = 45 * \log_{10}(d) + 31 + 20 * \log_{10}(f)$$

Umieszczając wzory i dane do napisanego programu otrzymujemy wykres badanego tłumienia.

wylosowane współrzędne nadajnika: ( 45 , 195 )



Powyższy wykres przedstawia tłumienie na zastosowanej siatce. Zgodnie z oczekiwaniami, wartości tłumienia są najniższe na ścieżkach, gdzie nadajnik ma swobodny dostęp. Natomiast w miejscach, gdzie znajdują się budynki, zauważamy wysokie wartości tłumienia. Brak tłumienia, zgodnie z oczekiwaniami, występuje w miejscu położenia nadajnika.

Poniżej znajduje się fragment kodu odpowiadający za realizację badanego tłumienia.

## A.6 Układ budynków według siatki Manhattan

Przyjmij układ budynków według siatki Manhattan. Dla wylosowanego rozkładem jednostajnym położenia nadajnika, wyznacz wartość tłumienia ścieżki (w wariancie 2D, dla stałej wysokości położenia odbiornika - według wybranego modelu propagacyjnego O20:

- mmMagic

```
[92] import numpy as np
import matplotlib.pyplot as plt
import math
import random

# losowanie współrzędnych
wsp[0] = random.choice([45, 95, 145, 195, 245, 295, 345])
wsp[1] = random.choice([45, 95, 145, 195])
print("Wylosowane współrzędne nadajnika: (" + wsp[0], ", ", wsp[1], ")")

widocznosc_x = np.arange(wsp[0]-5, wsp[0]+5, 1)
widocznosc_y = np.arange(wsp[1]-5, wsp[1]+5, 1)

zakres_x = np.arange(0,390,1)
zakres_y = np.arange(0,240,1)

# pusty wektor wartości tłumienia
wartosci_tlumienia = []

for y in zakres_y:
    for x in zakres_x:
        distance = math.sqrt((x - wsp[0])**2 + (y - wsp[1])**2)

        if(distance == 0):
            wartosci_tlumienia.append(0)
        else:
            if (x in widocznosc_x or y in widocznosc_y):
                PL = 19.2 * math.log10(distance) + 32.9 + 20.8 * math.log10(5)
                wartosci_tlumienia.append(PL)
            else:
                PL = 45 * math.log10(distance) + 31 + 20 * math.log10(5)
                wartosci_tlumienia.append(PL)

tlumienie_map = np.array(wartosci_tlumienia).reshape((len(zakres_y), len(zakres_x)))

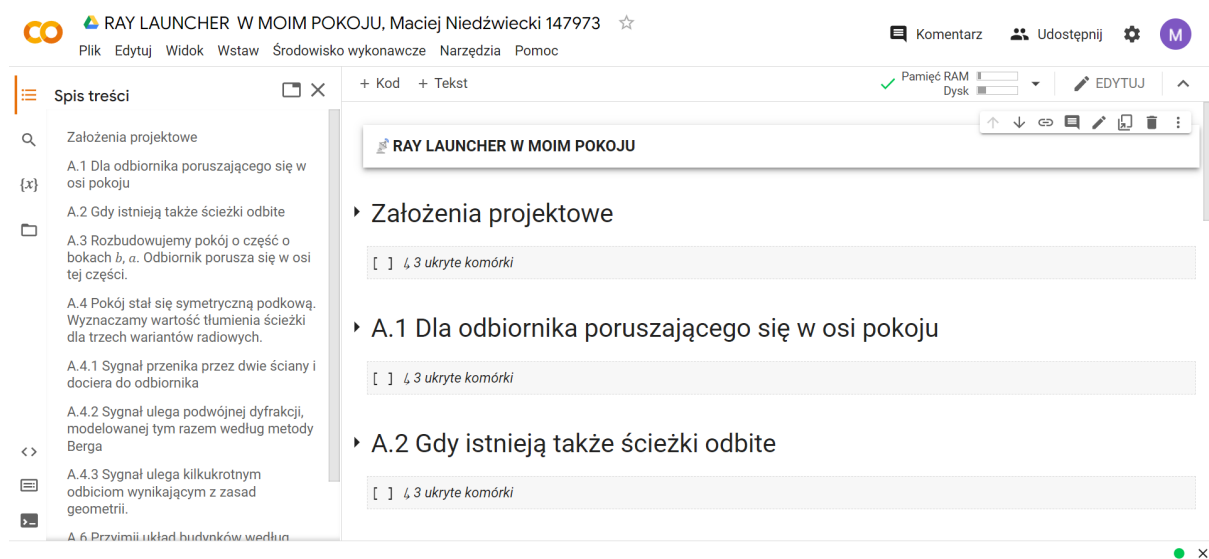
# wykres tłumienia
fig, ax = plt.subplots(figsize=(12,8))
plot = ax.imshow(tlumienie_map, cmap='winter')
fig.colorbar(plot, shrink = 0.6, aspect = 20)
plt.title('Tłumienie ścieżki')
plt.xlabel('x [m]')
plt.ylabel('y [m]')
plt.show()
```

Rys. 18. Część programu, w której wyznaczamy badane tłumienie.

## 9. Kod Programu w języku Python

Wyliczenia niezbędne do realizacji projektu zostały wykonane przy użyciu interpretera języka Python - Google Colab. Poszczególne części programu zostały umieszczone w odpowiadających im rozwijanych sekcjach. Poniżej znajduje się link prowadzący do napisanego programu.

- [Link do pliku Google Colab](#)



Rys. 19. Podgląd pliku, w którym zapisany jest program projektu.

## 10. Bibliografia

Wykonanie projektu nie byłoby możliwe bez skorzystania z dodatkowych źródeł. Poniżej znajduje się lista z pomocnymi wykładami i literaturą, które towarzyszyły przy realizacji projektu.

- Wykłady “Podstawy Radiokomunikacji”  
*dr inż. Krzysztof Cichoń*
- Ćwiczenia z przedmiotu “Podstawy Radiokomunikacji”  
*dr inż. Krzysztof Cichoń*
- “Projektowanie i obliczenia w radiokomunikacji”, Poznań 2005  
*prof. dr hab. inż. Hanna Bogucka*
- Rekomendacja ITU-R P.2040
- Złoty podział (*link*)
- Właściwości i rodzaje anten (*link*)