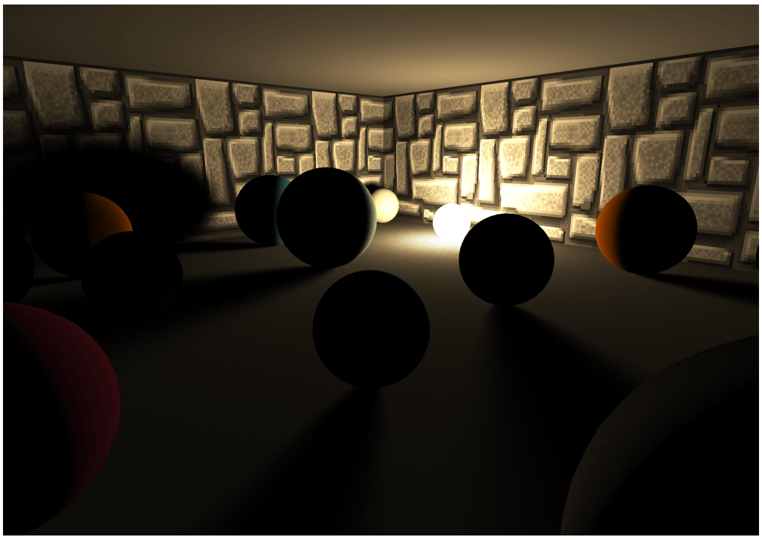

Development of a Ray-Tracing Program

Author
BOVING Alexandre

Date : 5 April 2021.

Contents

List of Figures

1 Introduction

This project involves the design and implementation of a ray-tracing-based electromagnetic wave propagation calculation code. The algorithm "follows" the waves emitted by the transmitter during their propagation.

The scenario is as follows: The Metropolitan Museum (MET) would like to cover the entire museum with a 27GHz 5G network and has approached us to accomplish this task. According to the museum's requirements, we need to deploy a minimal number of base stations that will allow the museum to achieve the highest possible data rate, enabling visitors to access additional information about the artworks on their tablets.

Specifically, the main objective of this project is to "track" the waves emitted by one or more transmitters and analytically calculate the reflection and transmission coefficients when these waves encounter obstacles. The algorithm then computes the reflected and transmitted waves using these coefficients and measures the received power of the waves passing through a local $1m^2$ area at a specific location. In practice, this involves utilizing the method of images with recursion to account for the paths of the reflected waves and tallying the number of obstacles along the direct path to the receiver in order to iteratively adjust the transmission coefficient.

All of this is performed using the Matlab software provided to us for the numerical wave analysis. During this project, we will employ code acceleration tools for the image method, as it can be time-consuming. Thin-walled obstacles with simple shapes, taking into account their thickness for the coefficients, are used as modeled walls. It is then interesting to analyze the coverage area of a base station as well as the received data rate based on the position of one or more receivers in order to propose a map displaying the distribution of 5G data rate within the museum.

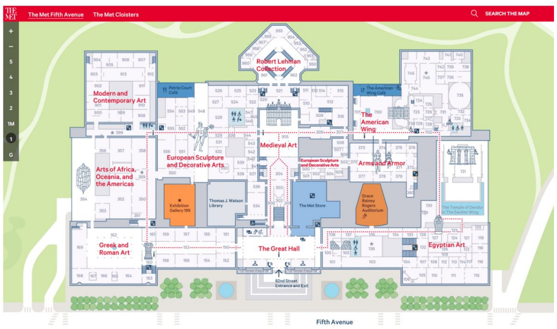

Figure 1: The MET

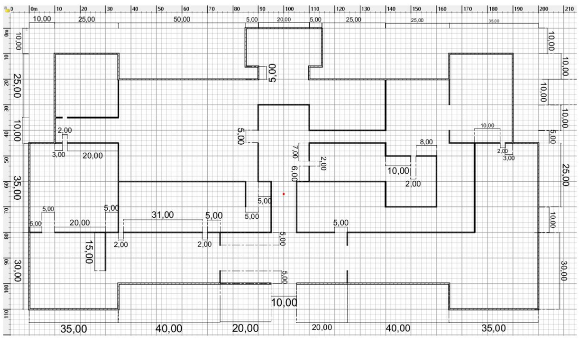

Figure 2: The MET map

2 Theory

The development of a ray-tracing-based electromagnetic wave propagation calculation code operates under the assumption of far-field conditions. This means that for sources varying sinusoidally at a certain frequency (such as our 27GHz antennas), we find that the dimensions of the system are comparable to the wavelength. Consequently, we transition from the quasi-static regime to the realm of electrodynamics, and by positioning ourselves far from the source, we can approximate the spherical wave as a plane wave.

2.1 Approximation of the far-field

The project instructions state that the waves are locally assumed to be plane waves. For this approximation to be valid, three conditions need to be satisfied.

1. Source size: The dimensions of the source (antenna) should be small compared to the wavelength of the emitted wave. This allows us to ignore the spatial variations of the field in the transverse dimensions of the source.
   - $r > \frac{\lambda}{2\pi}$
2. Propagation distance: The distance between the transmitting source and the observation point $|\vec{r} - \vec{r}'| \simeq r$ must be significantly larger than the characteristic size of the source (antenna). This condition ensures that the spatial variation of the field is negligible over the considered propagation distance.
   - $r >> D$ where D represents the dimension of the transmitter.
3. Aperture size: The size of the aperture through which the wave passes must be sufficiently large compared to the wavelength of the emitted wave. This condition ensures that diffraction effects are negligible and that the wave propagates in a straight line. It is necessary that $\vec{r}$ and $\vec{r} - \vec{r'}$ are almost parallel.
   - $r > \frac{2D^2}{\lambda}$

By satisfying these three conditions, we can locally consider the wave as a plane wave, which facilitates the analysis and calculations of the electromagnetic field. However, it is important to note that this approximation is only valid in the vicinity of the observation point and does not account for complex diffraction or reflection effects that may occur when the wave interacts with obstacles or surrounding structures.

2.1.1 Boundary between the near field and the far field

Arbitrarily, we define the boundary between the near field and the far field as the boundary located at a distance $r_{\text{ff}} = \text{Max}\left\{1, 6\lambda; 5D; \frac{2D^2}{\lambda}\right\}$ from the transmitter. These values are derived from the conditions mentioned earlier. Let's see if we can satisfy the far-field assumptions. The transmitting antenna is a half-wavelength dipole antenna emitting waves at a frequency of 27GHz. Therefore, the antenna's length is $\frac{c}{2f} = \frac{\lambda}{2} = \frac{1}{180}$ m. Comparing the conditions for r_{ff} we obtain $r_{\text{ff}} = 5D = 0.0277$ m, which is equivalent to 2.77 cm. The received power is calculated at a minimum distance of 1m from the base station(s). In conclusion, we can satisfy the far-field assumptions.

2.2 Description of the antennas $\frac{\lambda}{2}$

Within this project, we utilize antennas that are vertically oriented half-wavelength dipole antennas, considered lossless, operating at a frequency of 27GHz. Furthermore, our study situation is purely two-dimensional, and our field is polarized along the vertical axis within the xy plane. Consequently, we establish that our field is oriented along the z-axis and perpendicular to the xy plane. The transmitting and receiving antennas are placed at the same height.

2.2.1 Transmitting antenna

Characterizing a transmitting antenna involves describing its properties in terms of the emission direction. By considering the Poynting vector $\overrightarrow{S}$, we can deduce the radiated intensity and, from there, obtain the directivity and gain of an antenna. The Poynting vector represents the locally transported power density by the fields. Its magnitude represents the power density in $\mathrm{W/m^2}$, and its orientation indicates the direction of energy propagation. We can define the following formulas:

$$\mathcal{S} = \frac{1}{2Z_0}|\underline{\vec{E}}|^2 \quad \text{The magnitude of the Poynting vector} \tag{1}$$

$$U(\theta,\phi) = r^2\mathcal{S}(r,\theta,\phi) \quad \text{The radiated intensity (which depends only on the orientation)} \tag{2}$$

$$D(\theta,\phi) = \frac{U(\theta,\phi)}{P_{ar}/4\pi} \quad \text{The antenna's directivity} \tag{3}$$

$$G(\theta,\phi) = \eta D(\theta,\phi) \quad \text{The antenna's gain} \tag{4}$$

In this context, P_{ar} represents the power converted into radiated power (the only power consumed as there are no losses). η denotes the efficiency of the antenna, which is equal to 1 since the antenna is considered lossless. $Z_0 = \frac{\mu_0}{\epsilon_0}$ represents the characteristic impedance of free space, where μ_0 is the permeability of free space and ϵ_0 is the permittivity of free space. Lastly, $\vec{E}$ represents the electrical field. According to the assumptions mentioned earlier, it can be shown that the elevation angle is maximized at $\theta = 90°$.Therefore, we refer to it as an omnidirectional antenna within a plane. In this configuration, the directivity, gain, and radiated intensity reach their maximum values. (See figure 3). In the case of an omnidirectional half-wavelength dipole antenna within a plane, the radiated intensity can be derived from the Poynting vector. It is defined as follows:

$$U(\theta) = Z_0\frac{|I_a|^2}{8\pi^2}\left(\frac{\cos\left(\frac{\pi}{2}\cos\theta\right)}{\sin\theta}\right)^2 \Rightarrow U(\theta = 90°) = Z_0\frac{|I_a|^2}{8\pi^2} \tag{5}$$

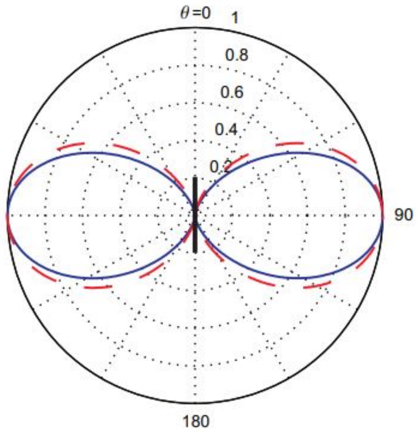

Figure 3: Normalized elevation radiation patterns for a half-wavelength dipole (solid line) and a current element (dashed line) are shown in the following diagrams

From the radiated intensity, we can deduce the total radiated power P_{ar} and the radiation resistance R_{ar} Additionally, we can determine the maximum directivity and gain (D_{max}, G_{max}). In this project, the equivalent resistance is equal to $R_a = R_{ar} + R_{al} = R_{ar}$ since the antennas are considered lossless. The term R_{al} represents the ohmic resistance, which is not considered in this context.

$$P_{ar} = \int_0^{2\pi} \int_0^{\pi} U(\theta) \sin\theta d\theta d\phi \approx \frac{3}{32} Z_0 \left|\underline{I}_a\right|^2 = \frac{1}{2} R_{ar} \left|\underline{I}_a\right|^2 \tag{6}$$

$$R_{ar} = \frac{720\pi}{32} \simeq 71\Omega \quad \text{By isolating it} \tag{7}$$

$$G_{max} = D_{\max} = D\left(\theta = 90^\circ\right) = \frac{16}{3\pi} \simeq 1,7 \quad \text{By using the formula from (4)} \tag{8}$$

The formula approximation (6) has been done by using the function $sin^3\theta$.

2.2.2 Receiving antenna

In the case of communication between two antennas in free space, the incident power density on the receiving antenna can be determined using the following formula:

$$\mathcal{S} = G_{TX}\left(\theta_{TX}, \phi_{TX}\right) \frac{P_{TX}}{4\pi d^2} = \frac{1}{2Z_0} |\underline{\vec{E}}|^2 \quad \text{where} \tag{9}$$

d is the distance travelled by the wave.
G_{TX} is the gain of the transmitting antenna.
P_{TX} is the power of the receiving antenna.

By isolating the magnitude of the electric field and adding its phase, we obtain:

$$\underline{E} = \sqrt{60G_{TX}\left(\theta_{TX},\phi_{TX}\right)P_{TX}}\frac{e^{-j\beta d}}{d} \quad \text{where} \tag{10}$$

The exponential term represents the phase difference between the transmitter and the receiver. Furthermore, we define the equivalent height as follows:

$$\vec{h}_e(\theta,\phi) = -\frac{1}{\underline{I}_a}\int_{\mathcal{D}} \underline{\vec{J}}(\vec{r})e^{j\beta\left(\vec{r}\cdot\overrightarrow{1}_r\right)}dV \tag{11}$$

Its value depends on the direction (θ,ϕ). However, we know that $\theta = 90°$, which means that $\vec{h}_e(\theta = 90°,\phi) = \frac{-\lambda}{\pi}\overrightarrow{1_z}$ and is parallel to the electric field $\overrightarrow{E}$. Generally, we define the induced voltage at the terminals of a receiving antenna located at a point $\overrightarrow{r}$ as follows:

$$\underline{V}_{oc}(\vec{r}) = \vec{h}_e(\theta,\phi)\cdot\vec{E}(\vec{r}) \tag{12}$$

we find that the maximum power collected at the load is as follows:

$$P_{Lmax} = \frac{1}{8}\frac{|\underline{V}_{oc}|^2}{R_a} \tag{13}$$

Note that this power formula is only maximum when the impedance Z_A* and Z_L are matched ($Z_A* = Z_L$), where Z_A* represents the antenna impedance, including the antenna resistance R_A (wire resistance and radiation resistance) as well as its reactance X_A and Z_L is simply the load connected to the antenna.

Explaining the induced voltage term in formula (13) gives :

$$P_{RX} = \frac{1}{8R_a}\left|\vec{h}_e\left(\theta = 90°,\phi\right)\cdot\underline{\vec{E}}(\vec{r})\right|^2 \tag{14}$$

This represents the power received at the receiver. For the multi-path components, it is necessary to correct equation (10) using the reflection and transmission coefficients. Additionally, we adopt the index n for the n-th multi-path component. Let's denote it as:

$$\underline{E}_n = \underbrace{\Gamma_1\Gamma_2\ldots}_{\text{Reffexions}}\ \underbrace{T_1T_2\ldots}_{\text{Transmissions}}\ \sqrt{60G_{TX}\left(\theta_{TXn},\phi_{TXn}\right)P_{TX}}\frac{e^{-j\beta d_n}}{d_n} \tag{15}$$

The induced voltage at the receiver and the received power are slightly modified by adding the sum of the contributions from all the multi-path components.

$$\underline{V}_{ac}(\vec{r}) = \sum_{n=1}^{N}\vec{h}_e\left(\theta_n,\phi_n\right)\cdot\underline{\underline{E}}_n(\vec{r}) \tag{16}$$

$$P_{RX} = \frac{1}{8R_a}\left|\sum_{n=1}^{N}\vec{h}_e\left(\theta_n,\phi_n\right)\cdot\underline{\vec{E}}_n(\vec{r})\right|^2 \tag{17}$$

2.3 The reflection and the transmission

Reflection is defined as the change in direction and sense of the incident wave within the same medium, while transmission refers to the phenomenon in which the incident wave is deviated within an obstacle before resuming a direction of the same inclination with respect to the incident wave. However, as an assumption for our Ray-Tracing code, we exclude the thickness of the different walls. Nonetheless, from a numerical perspective, the transmission coefficient takes into account this deviation through its parameter s, which represents the propagation of the wave within the wall (see Figure 4).

2.3.1 Snell' laws

The laws of Snell inform us that the wave is partially reflected at an angle θ_r and partially transmitted at an angle θ_t. The laws of Snell state the following:

$$\begin{aligned} \theta_i &= \theta_r \\ \sqrt{\varepsilon_1}\sin\theta_i &= \sqrt{\varepsilon_2}\sin\theta_t \end{aligned} \tag{18}$$

The second law of Snell is valid regardless of the polarization of the incident field. In our case, the polarization of the electric field in the plane of incidence is perpendicular to the plane of incidence. Using the laws of Snell, we can write the continuity condition for the tangential component of $\overrightarrow{E}$ and $\overrightarrow{B}$ as follows:

$$E_i + \Gamma_\perp E_i = T_\perp E_i \tag{19}$$

$$\sqrt{\epsilon_1}\cos\theta_i E_i - \sqrt{\epsilon_1}\cos\theta_i \Gamma_\perp E_i = \sqrt{\epsilon_2}\cos\theta_t T_\perp E_i \tag{20}$$

where $\Gamma_\perp$ and $T_\perp$ are respectfully the reflective coefficient and the transmission coefficient for the perpendicular polarization. By isolating the coefficients, we obtain:

$$\Gamma_\perp = \frac{Z_2\cos\theta_i - Z_1\cos\theta_t}{Z_2\cos\theta_i + Z_1\cos\theta_t} \tag{21}$$

$$T_\perp = \frac{2Z_2\cos\theta_i}{Z_2\cos\theta_i + Z_1\cos\theta_t} \tag{22}$$

where $Z_k = \sqrt{\frac{\mu_0}{\tilde{\varepsilon}_k}} = \sqrt{\frac{\mu_0}{\varepsilon_k - j\frac{\sigma_k}{\omega}}}$ represents the impedance of the medium.

It can be observed from the equations above that the reflection and transmission coefficients depend on geometric parameters such as the angle of incidence θ_i and the thickness of the wall ω, as well as the material's characteristics, particularly its permittivity and conductivity.

2.3.2 Transmission and reflection on a wall

Let's assume an incident wave with polarization perpendicular to the plane of incidence, incident on a lossless wall (we will consider losses in the formula for the reflection coefficient later) with a permittivity ϵ_m and thickness l (see Figure 4).

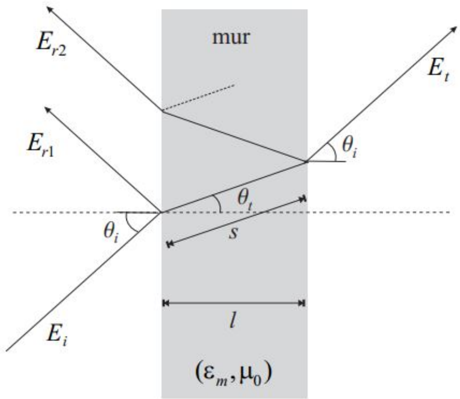

Figure 4: Transmission and reflection by a wall of thickness l

In the case of reflection, the incident wave is followed through its multiple reflections inside the wall. As can be observed in Figure 4, there are multiple contributions to the reflected wave. The first contribution is given by:

$$E_{r1} = \Gamma_{\perp}(\theta_i)\, E_i \tag{23}$$

Taking into account the transmission through the wall, the propagation of the wave from one end to the other over a distance s, and the wall $\Rightarrow$ air reflections, if we add up all the successive contributions, we obtain:

$$E_r = \Gamma_m E_i \tag{24}$$

where Γ_m is the reflection coefficient on a wall, which is given by the following formula:

$$\Gamma_m(\theta_i) = \Gamma_{\perp}(\theta_i) - \left(1 - \Gamma_{\perp}^2(\theta_i)\right) \frac{\Gamma_{\perp}(\theta_i)\, e^{-2\gamma_m s} e^{j\beta 2s \sin\theta_t \sin\theta_i}}{1 - \Gamma_{\perp}^2(\theta_i)\, e^{-2\gamma_m s} e^{j\beta 2s \sin\theta_t \sin\theta_i}} \quad \text{where} \tag{25}$$

$\Gamma_{\perp}(\theta_i)$ is the reflection coefficient for perpendicular polarization.
γ_m propagation constant in the wall.
$\beta = \frac{2\pi}{\lambda}$ wave number.
s is the wave propagation in the wall.
θ_i is the angle of incidence at the wall.
θ_t is the transmission angle within the wall.

Note that initially, the assumption was that the walls are considered lossless. We have added γ_m, the propagation constant within the wall, instead of $j\beta_m$ because the walls have low losses. We characterize $\gamma_m = j\omega\sqrt{\mu_0 \hat{\varepsilon}}$, where the real part affects the level of energy dissipation within the wall. Finally, according to the laws of Snell, we obtain the transmission coefficient, which is given by the following formula:

$$T_m\left(\theta_i\right)=\frac{\left(1-\Gamma_{\perp}^{2}\left(\theta_i\right)\right)e^{-j\beta_m s}}{1-\Gamma_{\perp}^{2}\left(\theta_i\right)e^{-2j\beta_m s}e^{j\beta 2s\sin\theta_t\sin\theta_i}} \tag{26}$$

3 The *Ray-Tracing*

To date, there are three types of numerical methods used for field calculations. The method of integral equations determines the induced current on or in an obstacle, and the finite difference method is based on a discrete solution of Maxwell's equations using finite differences. The last method, known as Geometrical Theory of Diffraction (GTD) or commonly referred to as Ray-Tracing, is the numerical method used in this project. The Ray-Tracing method is particularly suitable for analyzing wave propagation in complex environments with obstacles. It involves tracing the paths of rays and considering their reflections, transmissions, and diffractions to calculate the field distribution. This method provides a computationally efficient approach to model and analyze wave propagation, making it well-suited for the purposes of this project.

3.1 *Ray-Tracing* principle

The Ray-Tracing method involves "tracing" the wave during its propagation, taking into account its interactions or lack thereof with various obstacles, before calculating its power at a point representing a receiver. Analytically calculating the reflection, transmission, and diffraction coefficients (the latter coefficient is omitted from the project) is possible with the angle of incidence between the emitted ray and the obstacle. These coefficients are used to determine the reflected, transmitted, and diffracted waves. This method is particularly useful when the obstacles (i.e., walls) are much larger in size compared to the wavelength, although the accuracy is typically within a factor of 2 to 10. To obtain the incident angle at the point of impact between the emitted wave and a wall, as well as the total distance traveled by the wave, the method of images is employed.

3.2 The image method

Based on the principle of recurrence, the method of images constructs an "image antenna" relative to our transmitting antenna for each wall. Recursively, depending on the type of reflections (single, double, or triple), we repeat the same process of constructing an "image antenna" relative to the previously found image antenna. The construction of the method of images can be observed in Figures 5 and 6, which we will detail.

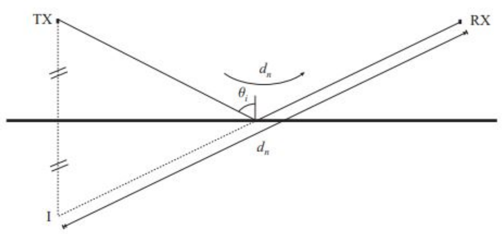

Figure 5: 1-reflection image method

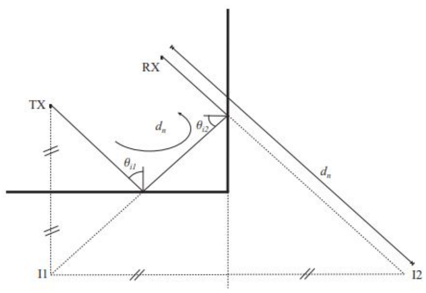

Figure 6: 2-reflection image method

Firstly, we determine the position of the image antenna for the base station by taking its symmetrical position with respect to the wall. Then, we deduce the theoretical reflection point, which is the intersection point between the image antenna and the receiving antenna. At this point, we can determine the angle of incidence formed by the segment [T'_x,R_x]. Finally, we determine the segment of length d_n representing the total distance traveled by the multi-path component n.

3.3 Coverage area representation

One of the key features of this calculation code is that it provides the distribution of the average power from the transmitting antenna captured at any point on a building floor. The obtained power values at each point in space are displayed using a "Heat Map" model using Matlab. The power values are adjusted based on a gauge where high data values are represented by a reddish color and low data values are represented by a bluish color. In telecommunications, these power values are often expressed in Watts or dBm (decibels relative to milliwatts).

$$P[\text{dBm}] = 10 \log \frac{P[\text{ W}]}{1 \text{ mW}} \tag{27}$$

For this project, the sensitivity range of our average received power is set between -73dBm and -82dBm. It is interesting to display these results using a gauge that measures the received data rate in order to observe the extent of the coverage area. Conversely, we can identify the areas not covered by the base station to propose a better deployment location for one or more base stations. To accomplish this, there exists a linear relationship between sensitivity and data rate. Please refer to the table below.

Sensitivity	bit rate
-82 dBm	40 Mb/s
-73 dBm	320 Mb/s

This table provides the corresponding data rates for different sensitivity levels. By mapping the received data rate according to the sensitivity, we can visualize the coverage area and identify areas where the data rate falls below a desired threshold. This information can help determine the optimal placement of base stations for better coverage and data rate distribution. The linear function is defined as the following:

$$y = \frac{1}{9} \times (280x + 23320) Mb/s \tag{28}$$

The input sensitivity (x) is injected to obtain the output bit rate (y).

4 Validation using elementary cases

To verify the correctness of our calculation code, we examine three elementary cases that we solve analytically by hand before comparing them to the results provided by the numerical code.

4.1 Defining constants

Before solving our calculations, we'll define a few constants needed to validate our code on simple cases.

1. The source frequency is $f = 27 * 10^9$ Hz.
2. The wave number is $\beta = \frac{\omega}{c} = \frac{2\pi}{f} = 565,878\ m^{-1}$.
3. The power of the source is 20 dBm, which is 0,1 W.
4. Wall thickness is w = 0,5 m.
5. The relative permittivity of concrete walls is $\epsilon_r = 5$.
6. The conductivity of concrete walls is $\sigma = 0.014$.
7. The complex permittivity of a material $\tilde{\varepsilon} = \varepsilon - j\frac{\sigma}{\omega}$.
8. Impedance of the concrete medium $Z_k = \sqrt{\frac{\mu_0}{\varepsilon_k - j\frac{\sigma_k}{\omega}}} = \sqrt{\frac{4\pi \cdot 10^{-7}}{\varepsilon_0 \cdot 5 - j \cdot \frac{0,014}{2\pi \cdot 27 \cdot 10^9}}} = 168,48 + 0,15703i$.
9. The propagation constant is $\gamma_m = j\omega\sqrt{\mu_0 \tilde{\varepsilon}} = 1,1794 + 1265,3i$.

4.2 Direct wave propagation

The propagation of a direct wave occurs without obstacles and, by definition, does not involve transmission and reflection coefficients. We place a source at (0,0) from which we would like to determine the data rate at an arbitrary point. We will consider T_{rx} as our receiving point located at (87,87). The situation is depicted in Figure 7.

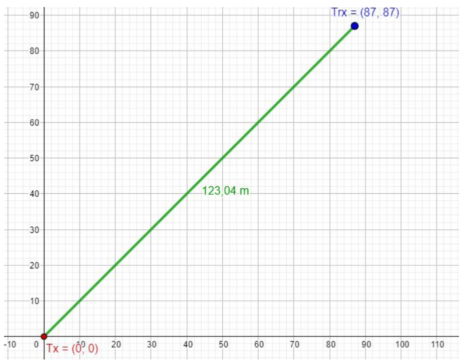

Figure 7: Diagram of the situation for a direct wave to the point (87;87)

As we saw earlier, the relationship giving us the received power is given by the formula (14) where $R_a = \frac{720\pi}{32}$, $h_e = \frac{-\lambda}{\pi}$, $G_{TX} = \frac{16}{3\pi}$ and $P_{TX} = 0.1$ and $E = \frac{\sqrt{60 G_{TX}(\theta_{TX},\phi_{TX})P_{TX}}}{d}$. It gives us:

$$P_{RX} = \frac{60 * 0.1 * 16 * 299792458^2}{8 * 73 * 87^2 * 3 * \pi^3 * (27 * 10^9)^2} = 14,392 \quad pW \tag{29}$$

Expressing this power in dBm, we obtain:

$$P[\text{dBm}] = 10 \log \frac{14.392 * 10^{-12}}{0.001} = -78,418 \quad dBm \tag{30}$$

Simply convert sensitivity to bit rate and conclude with:

$$D_b = \frac{1}{9} \times (-280 * 78,418 + 23320) = 151,44 \quad Mb/s \tag{31}$$

Here is the result calculated by the calculation code shown in Figure 8, where we obtain a value of 151.793 Mb/s.

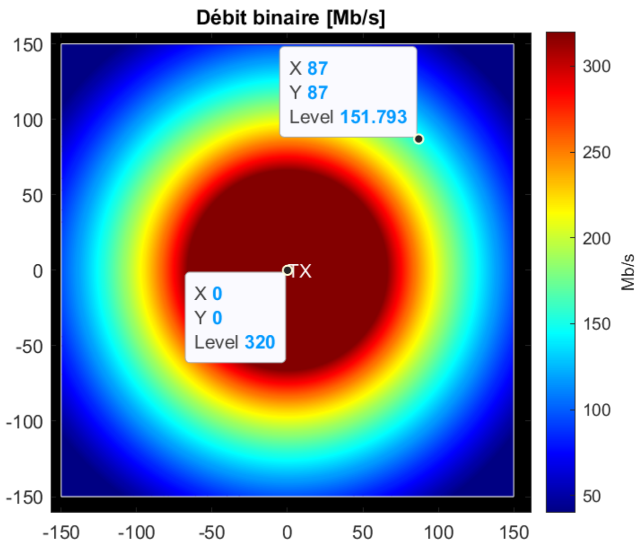

Figure 8: Bit rate distribution within a 300 m square. Source at origin (0,0)

4.3 Propagation of a wave with transmission

Taking a similar situation to the previous subsection, all we need to do is add a (concrete) wall between the source and the desired calculation point. We place a vertical wall point (30; 0) and a reception point at (40; 10). The situation is shown below in figure 9

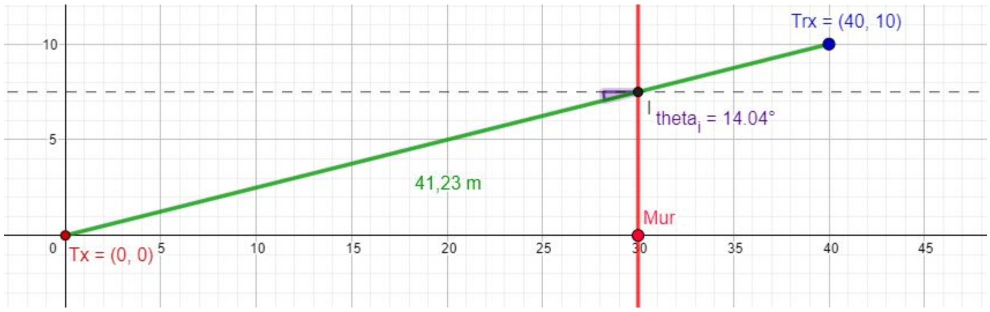

Figure 9: Diagram of the situation for a wave with transmission within a wall of conductivity $\sigma = 0.014$ S/m going towards the point (40;10)

4.3.1 Calculation of θ_t,s,$\Gamma_\perp$,T_m

Using Snell's laws, specifically the second equation of (18), we can find θ_t given that $\theta_i = 14,04°$. This yields:

$$\theta_t = \arcsin\left(\frac{1}{\sqrt{\varepsilon_r}} \cdot \sin(\theta_i)\right) = \arcsin\left(\frac{1}{\sqrt{5}} \cdot \sin(14.04°)\right) = 6,2285° \tag{32}$$

Then, using θ_t and the defined constants mentioned earlier, we can find the distance traveled by the wave within the wall s), the reflection coefficient $\Gamma_\perp$ for perpendicular polarization, and finally the transmission coefficient T_m. The results are provided below:.

$$s = \frac{w}{\cos(\theta_t)} = \frac{0.5}{\cos(18.4349°)} = 0.5030m \quad \text{distance of propagation inside the wall} \tag{33}$$

$$\Gamma_\perp = -0,3923 + 0,0004 \cdot j \quad \text{reflection coefficient (21) with a perpendicular polarization.} \tag{34}$$

$$T_m(14,04°) = -0,1305 - 0,4570 \cdot j \quad \text{transmission coefficient (26) on a wall.} \tag{35}$$

Note that the main calculation concerns power. What we're interested in is the modulus of the transmission coefficient, which will attenuate the power by a factor between 0 and 1.

4.3.2 Power and bit rate evaluation in T_{rx}

Using the values obtained in the previous section, we can determine the power received at the point T_{rx}, taking into account the transmission coefficient when a wall is placed between the source and the receiver.

$$P_{RX} = T_m^2 \frac{1}{8R_a}\left|\sum_{n=1}^{N} \vec{h}_e(\theta_n,\phi_n)\cdot \underline{\vec{E}}_n(\vec{r})\right|^2 = 0,4752^2 \frac{60 * 0.1 * 16 * 299792458^2}{8 * 73 * 41,23^2 * 3 * \pi^3 * (27 * 10^9)^2} = 28,9419 \quad pW \tag{36}$$

As with the direct wave, we deduce the power in dBm and then its bit rate. This gives us:

$$P[\text{dBm}] = 10 \log \frac{28,9419 * 10^{-12}}{0.001} = -75,3847 \quad dBm \tag{37}$$

$$D_b = \frac{1}{9} \times (-280 * 75,3847 + 23320) = 245,80 \quad Mb/s \tag{38}$$

Here is the result calculated by the calculation code shown in figure 9 where we obtain: 245,494 Mb/s.

4.4 Propagation of a wave with reflection

Before proceeding to the mapping of the data rate within the Metropolitan Museum (MET), it is necessary to conclude if the code is capable of validating the elementary case based on reflection. To simplify the calculation steps, we will consider the same scenario as in the case of direct wave propagation but with the addition of a wall behind the receiver. We place a wall passing through the point (125,0). The situation is depicted in figure 10.

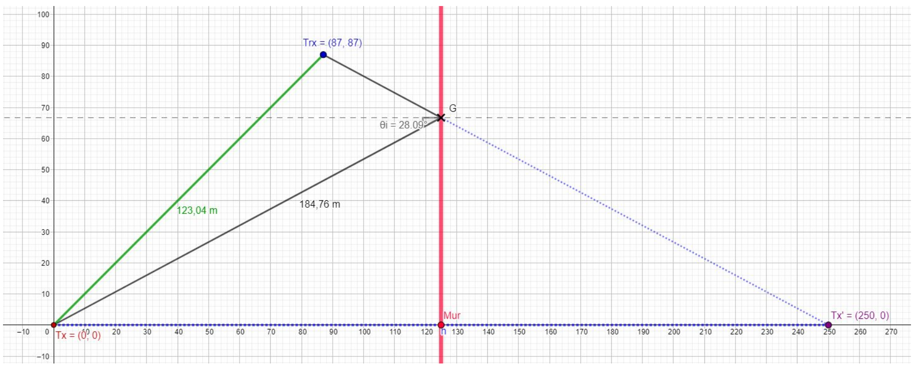

Figure 10: Diagram of the situation of a wave reflected on a wall with conductivity $\sigma = 0.014$ S/m and a direct wave to the point (87;87)

4.4.1 Calculation of θ_t,s,$\Gamma_\perp$,Γ_m

With an incident angle of $\theta_i = 28,09°$, we find θ_t, s, Γ and T_m. Therefore:

$$\theta_t = \arcsin\left(\frac{1}{\sqrt{\varepsilon_r}} \cdot \sin(\theta_i)\right) = \arcsin\left(\frac{1}{\sqrt{5}} \cdot \sin(28.09)\right) = 12.1560° \tag{39}$$

$$s = \frac{w}{\cos(\theta_t)} = \frac{0.5}{\cos(12.1560)} = 0.5115m \quad \text{distance travelled inside the wall} \tag{40}$$

$$\Gamma_\perp = -0.4249 + 0.0004 \cdot j \quad \text{reflection coefficient (21) with a perpendicular polarization.} \tag{41}$$

$$\Gamma_m\left(28,09^\circ\right) = -0,3580 + 0,0850 \cdot j \quad \text{reflection coefficient (26) on a wall.} \tag{42}$$

4.4.2 Power and bit rate evaluation in T_{rx}

In this case, the total power is the sum of the power of the reflected wave and the power of the direct wave. We can express the received power P_{rx} as $P_{rx} = P_{rx,rf} + P_{rx,direct}$, where $P_{rx,rf}$ represents the power of the reflected wave and $P_{rx,direct}$ represents the power of the direct wave. It is possible to add these powers since the phase of the electric wave $\overrightarrow{E}$ is not relevant, only its magnitude matters. Therefore, we obtain the following values:

$$P_{rx,rf} = \Gamma_m^2 \frac{60 * 0.1 * 16 * 299792458^2}{8 * 73 * 184,76^2 * 3 * \pi^3 * (27 * 10^9)^2} = 0,8655 \quad pW \tag{43}$$

$$P_{rx} = P_{rx,rf} + 14,392pW = 0,8655pW + 14,392pW = 15,2575pW \tag{44}$$

As with the direct wave, we deduce its power in dBm and then its bit rate. This gives us:

$$P[\text{dBm}] = 10 \log \frac{15,2575 * 10^{-12}}{0.001} = -78,165 \quad dBm \tag{45}$$

$$D_b = \frac{1}{9} \times (-280 * 78,165 + 23320) = 159,311 \quad Mb/s \tag{46}$$

On Figure 11 below, it can be observed that the value at points (87, 88) is 156.087 Mb/s and (87, 87) is 162.846 Mb/s. The slight difference between the algorithm's results and our calculations, where the value is 159.311 Mb/s, is due to the fact that the algorithm computes an average power over a $1m^2$ area. In our calculations, the power value is computed at a single point.

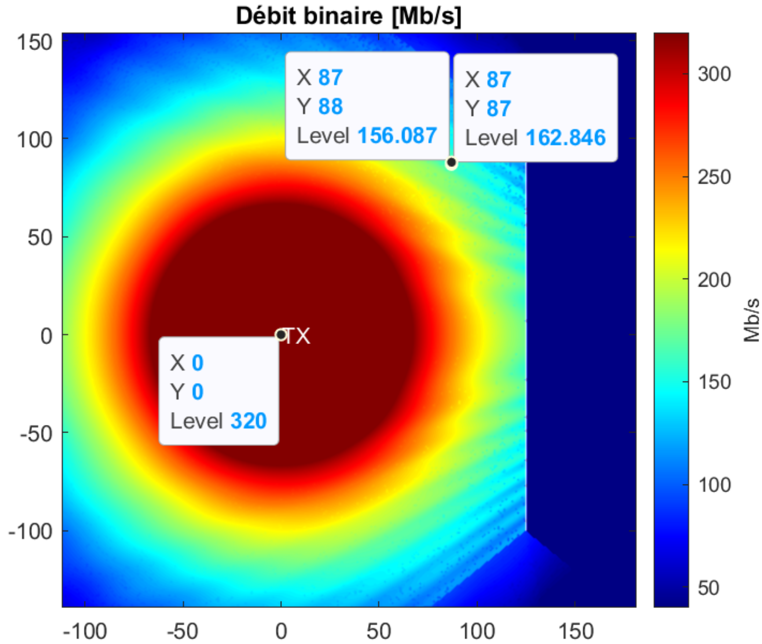

Figure 11: Binary flow distribution within a 500 m square with a vertical wall at x = 125 with conductivity = 0.014. The source is located at the origin (0,0).

5 Bit rate within MET

By placing a base station at (100,55) with a power of 20dBm and a frequency of 27GHz, the Ray-Tracing method allows us to obtain the spatial distribution of the data rate within the Metropolitan Museum (MET). Please refer to Figure 12 for the visualization of this distribution. It should be noted that we have inverted the y-axis so that the base station is located at (100,55) rather than (100,65) as indicated in the project statement. The power distribution in dBm can be found in the appendix.

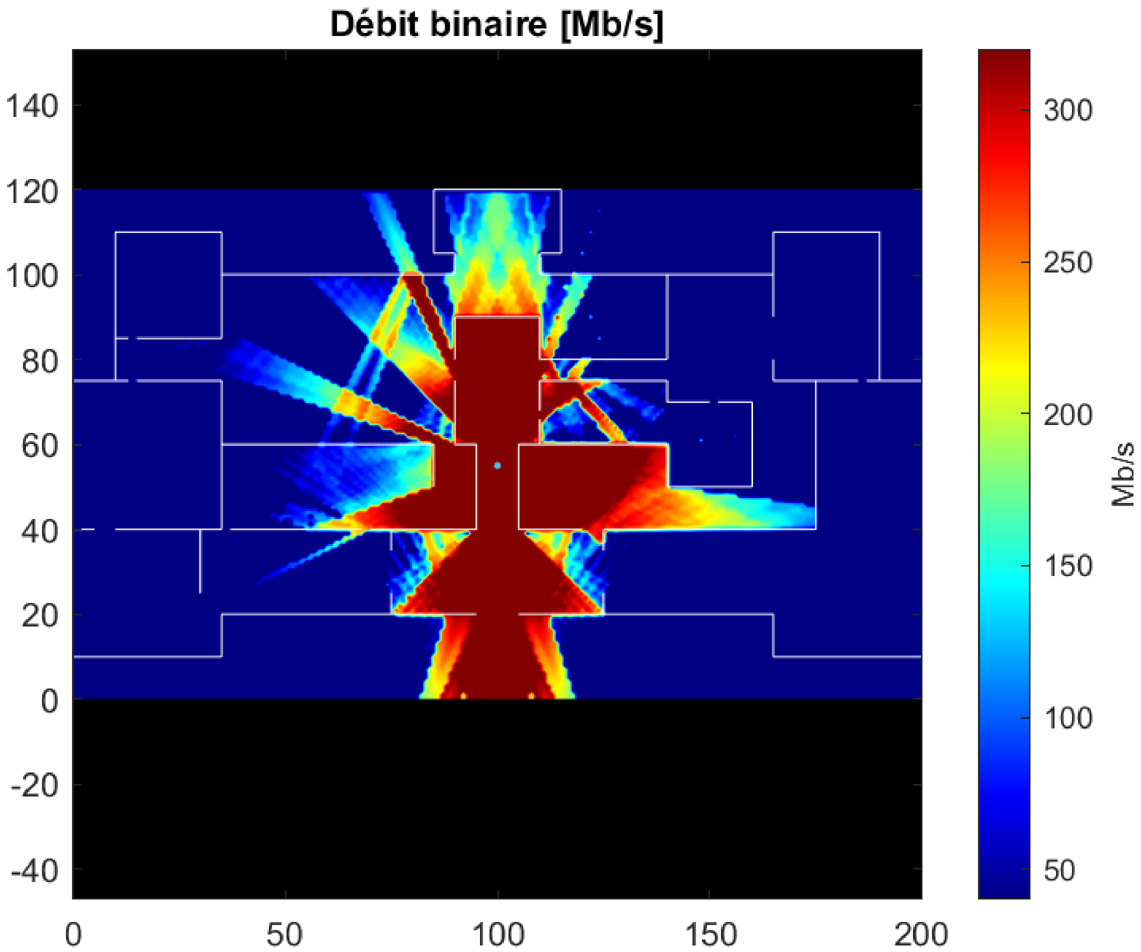

Figure 12: The bit rate distribution within the MET, expressed in [Mb/s]. The antenna is located at (100,55)

6 Code optimisation

As a reminder, the objective of this project is to deploy a minimal number of base stations to provide full coverage of the entire museum with 5G connectivity. The aim is to offer a sufficiently high data rate for the entire museum. Considering the computational time of approximately 45 minutes for a single transmitting antenna, considering three reflections, we have decided that for the deployment of multiple antennas, we will only consider the first reflection. By focusing on the first reflection, we can significantly reduce the computational complexity while still achieving a reasonable approximation of the coverage and data rate distribution within the museum. This approach allows us to strike a balance between computational efficiency and obtaining meaningful results for the deployment of multiple antennas.

6.1 The optimisation

The main idea behind maximizing the coverage area in terms of data rate is to populate the map with base stations and combine their power output. Essentially, the goal is to determine the minimum combination of transmitting antennas that can cover the majority of the map above a certain data rate threshold (the desired coverage rate is set arbitrarily). The code will iterate through each base station and calculate which one provides the highest coverage rate above the specified limit (coverage rate = $\frac{"covered\ area"}{"total\ area"}$). If such a base station exists, the code will stop and display its coverage area. If not, the code will proceed to evaluate combinations of two transmitting antennas by summing their power and repeat the same conditions mentioned above. This logical process can be repeated for up to five possible combinations, as the computational time on our computer imposes significant limitations.

6.2 In practice

We define our coverage rate as 85% with a threshold of 175 Mb/s. This means that 85% of the local areas of $1\mathrm{m}^{\mathbf{2}}$ throughout the entire map have a data rate of 175 Mb/s or higher. By selecting 12 transmitters (bearing in mind the already high computational time), they are placed as seen in figure 13. We request our numerical code to retrieve the best possible and minimal combination among these 12 base stations. Consequently, we obtain figure 14. According to the project instructions, the focus is solely on the distribution of the data rate within the museum. However, during the calculation of our distribution rate, we considered a square area of $200 * 120m^2$, which is not the exact surface area corresponding to the interior of the museum (the area delimited by the walls). It is important to note that this discrepancy is an error, but it should not be taken into account when optimizing the allocation of base stations in the code. Instead, the error lies in the creation of the walls within the numerical code itself. To address this issue, it would be necessary to accurately define the boundaries and dimensions of the museum's interior within the code, ensuring that it aligns with the actual layout and structure of the museum. By accurately modeling the walls, the code can provide a more realistic representation of the data rate distribution within the museum.

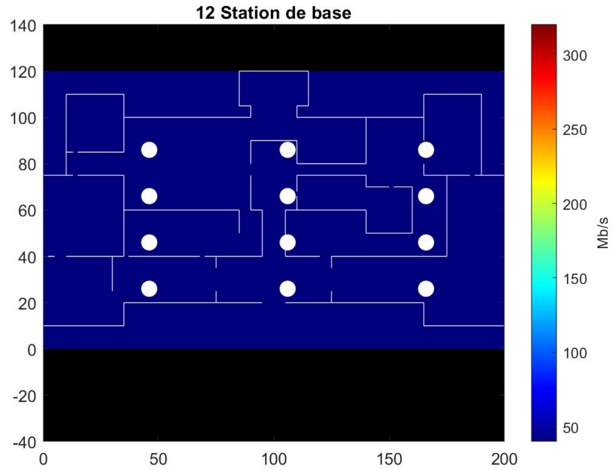

Figure 13: Arrangement of the 12 transmitters in the MET.

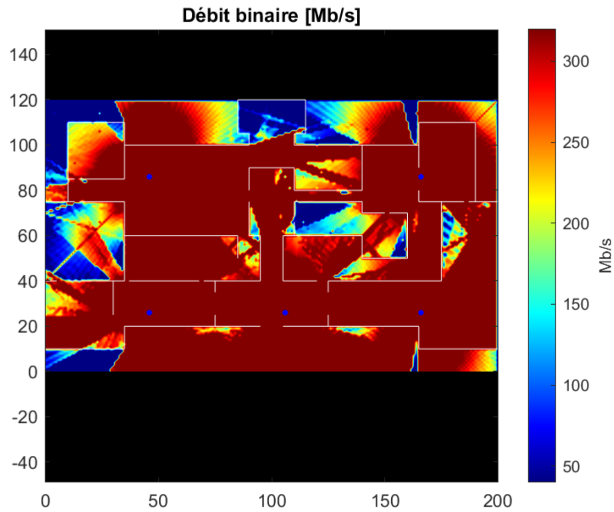

Figure 14: Minimal combination of 5 base stations covering 85.637 % of the MET.

6.3 Generalization

Our code can be generalized to the case where we place n transmitters on the map. By specifying the coverage rate above a certain threshold, it can provide us with the minimum combination of transmitting antenna(s) up to a maximum of 5 antennas (limited by the computer's processing time). With this functionality, we can determine the optimal configuration of the minimum number of transmitting antennas that achieves the desired coverage rate. By inputting the coverage threshold, the code will evaluate different combinations of antennas and identify the configuration that satisfies the coverage requirement while minimizing the number of antennas used. It is important to note that the computational time may be limited by the capabilities of the computer, especially as the number of antennas increases. Therefore, for practical purposes, the code is currently designed to handle up to 5 antennas.

7 Conclusion

For obstacles of very large dimensions and with a sufficiently powerful computer, ray-tracing remains a highly competent numerical method for retrieving power within a rectangular local area. Despite its approximate nature and computationally demanding processing time, this numerical code allows us to generate a spatial distribution of received power within a space with basic-shaped structures. By analytically calculating the reflection and transmission coefficients (in the course of this project), we were able to compute the reflected and/or transmitted waves and measure their power within a local zone. The diffraction coefficient could be included as a feature as the project progresses.

8 Appendix

8.1 Power distribution in the MET

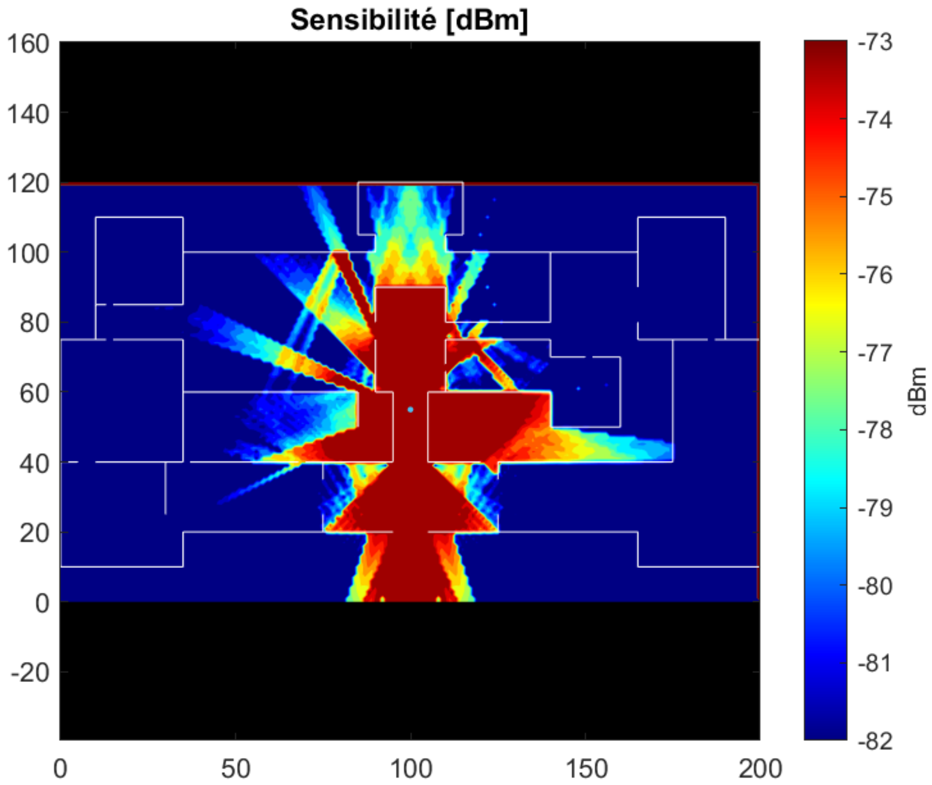

Figure 15: Power received in the museum in [dBm].

8.2 Insight of some functionalities

In our numerical code, the implementation of walls is done using a *.csv* file in Excel format. We can easily and quickly implement a large quantity of simple geometric shapes. The method of images allows for the analytical calculation of reflection and transmission coefficients and their application to the reflected and/or transmitted waves. This functionality is accomplished through the Method-Image function. The code returns the spatial distribution of power and data rate within a defined area based on the Excel file. The coverage zone figure is directly generated using the Main function. Finally, another noteworthy aspect of our code is the determination of the minimal combination of antenna(s) that offers the greatest data rate coverage within the map. In another "main" function, specifically called "main," we find the optimization of the deployment of transmitting antennas within the Metropolitan Museum (MET).

8.3 Numerical computer code

Main function in order to execute the code.

```matlab
%% Description du projet.

% This is the algorithm used for Ray-Tracing with the
% image method.

%% Initialisation.

tic

% Look up the technical specifications for more information about the
% values.

% Frequency initialization.

F = 27*10^9; % This is the frequency of our Wifi wave emitted by the antenna.
w = 2*pi*F; % The pulse.

% Wave initialization.

c = 3*10^8; % Wave velocity
Z_0 = 120*pi; % Void impedance
mu0 = 1.25663706e-6;
ep0 = 8.85418782e-12;
lambda = c/F; % Wavelength

% Antenna initialisation.

h = -lambda/pi; % Characteristic height. where theta = pi/2.
Ra = 73; % Radiation resistance.
Tx = Base([100,55]); % The location of our transmitting station. Tx is an object.

% Walls initialization.

perm_relative_brique = 4.6; % Bric.
perm_relative_beton = 5; % Concrete.

% Conductivity (sigma) of the materials.

conductivite_brique = 0.02; % Brick.
conductivite_beton = 0.014; % Concrete.

%% Treatment.

% Walls creation.

[wallxyz1,wallxyz2,wallX,wallY] = CSVexcel;

[M,N] = size(wallxyz1); % M : Number of walls. N : Number of columns.

% Initialize a cell matrix (for wall objects).
% It's important to initialize Mur correctly, otherwise it will create a new
% a new variable to store a wall object. This code is already
% code, you need to optimize it as much as possible.

% Initialization of object wall matrix.

a = zeros(1,length(wallxyz2));
Mur = num2cell(a);

for i = 1:M
   if i < 26 % The first 26 walls are made of concrete.
```

```
        Mur{i} = Wall(wallxyz1(i,1:N),wallxyz2(i,1:N),perm_relative_beton,conductivite_beton,Z_0,w,ep0);
    else % The rest is brick.
        Mur{i} = Wall(wallxyz1(i,1:N),wallxyz2(i,1:N),perm_relative_brique,conductivite_brique,Z_0,w,ep0);
    end
end

% Loaded with Rx receivers. (No collisions with walls)

S = zeros(max(max(wallX)),max(max(wallY)));
Matrice_recepteur = num2cell(S);
for i = 0:1:max(max(wallX))
    for j = 0:1:max(max(wallY))
        Matrice_recepteur{i+1,j+1} = Recepteur([i,j]);
        % scatter(i,j,300);
    end
end

% Application of the image method (vector method). + Calculation of
% received power.

Mb = zeros(max(max(wallX))+1,max(max(wallY))+1);
Sens = zeros(max(max(wallX)+1),max(max(wallY))+1);
Sens(max(max(wallX)+1,max(max(wallY)+1))) = -82;

for a = 1:max(max(wallX))
    disp(a)
    parfor g = 1:max(max(wallY))
        Nos_ondes = Methode_image(Tx,Matrice_recepteur{a,g},Mur);
        [Mb(a,g),Sens(a,g)]  = Prx(Tx,Nos_ondes,h,Ra);
    end
end

%% Display.

% In order to display our walls without the waves.

% x = 1:1:(max(max(wallX)));
% y = 1:1:(max(max(wallY)));
% [X,Y] = meshgrid(x,y);
%
% plot(wallX,wallY,'b')%visualisation des murs
% text(Tx.position(1),Tx.position(2),'TX','Color','b') % Mettre un point dans le futur.
% text(Rx.position(1),Rx.position(2),'RX','Color','b') % Mettre un point dans le futur.

x = 0:1:(max(max(wallX)));
y = 0:1:(max(max(wallY)));
[X,Y] = meshgrid(x,y);

%contour(X,Y,Mb)

% adding color map and showing it on the screen.

%ax1 = subplot(1,1,1);

figure

contourf(X,Y,Mb',150,'LineColor','none')
title('D bit binaire [Mb/s]')
```

```
set(gca, 'DataAspectRatio', [1 1 1])
colormap(jet);
caxis([40 320])
c = colorbar;
c.Label.String = 'Mb/s';
hold on
plot(wallX,wallY,'w')
%text(Tx.position(1),Tx.position(2),'TX','Color','w') % Mettre un point dans le futur.
%title('Puissance Recue Dans le MET, Station de base(100.65) : Ray Tracing avec 5G 27GHz')
% text(Rx.position(1),Rx.position(2),"RX",'Color','w') % Mettre un point dans le futur.
axis([0 200 0 200])
set(gca,'color','k')
set(gcf,'color','w')
scatter(100,55,5,'fill')

figure

contourf(X,Y,Sens',150,'LineColor','none')
title('Sensibilit  [dBm]')
set(gca, 'DataAspectRatio', [1 1 1])
colormap(hot);
caxis([-82 -73])
c = colorbar;
c.Label.String = 'dBm';
hold on
plot(wallX,wallY,'w')
%text(Tx.position(1),Tx.position(2),'TX','Color','w') % Mettre un point dans le futur.
%title('Puissance Recue Dans le MET, Station de base(100.65) : Ray Tracing avec 5G 27GHz')
% text(Rx.position(1),Rx.position(2),"RX",'Color','w') % Mettre un point dans le futur.
axis([0 200 0 200])
set(gca,'color','k')
set(gcf,'color','w')
scatter(100,55,5,'fill')

toc
```

This function calls up an Excel file containing all the information needed to build the walls.

```
function [wallxyz1,wallxyz2,wallX,wallY] = CSVexcel
[fileName,fileAddress] = uigetfile('*.csv'); % Return [file name, file address].
[~,~,pointData] = xlsread([fileAddress,fileName]); % returns [num, txt, raw],
    meaning only numbers, text, all the file.
% pointData
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pointData is a cell matrix. Every "cell" contains the information
% wether it is a number, text or a symbol.

%%
% strtok returns a string from left to right until delimitations.
% indicated here by ';'. The rest is inserted inside a 'remainer'.
% str2num converts a matrix of characters to a numerical matrix.
for i = 1:numel(pointData) % Numel returns the number of elements in the pointData
     matrix
    X(i,1) = str2num(strtok(pointData{i,1},';')); % Vertical matrix (abscissa x)
    [~,remain{i,1}] = strtok(pointData{i,1},';');
    Y(i,1) = str2num(strtok(remain{i},';')); % Vertical matrix (ordinate y)
end
%%
wallxyz1 = zeros(size(X,1)/2,2); % Generates a matrix [number of lines from X/2,3]
wallxyz2 = wallxyz1;
% Defining walls clockwise
for i = 1:numel(X)/2 % Returns the 4 corners of a wall. Every line
    wallxyz1(i,:) = [X((i*2)-1,1),Y((i*2)-1,1)]; % Takes the coordonate of the
    initial points.
    wallxyz2(i,:) = [X(i*2,1),Y(i*2,1)]; % Takes the coordonate of the initial
    points.
end
% wallxyz1
% wallxyz2

%%
wallX = [wallxyz1(:,1)';wallxyz2(:,1)']; % Line 1 = x ; Line 2 = y initial points.
     (Look Excel file)
wallY = [wallxyz1(:,2)';wallxyz2(:,2)'];
```

Class that constructs "wall" objects carrying information about the characteristics of their environment.

```
classdef Wall
    %MUR Summary of this class goes here
    %   Detailed explanation goes here

    properties
        debut
        fin
        perm_relative
        conductivite
        Z_2
    end

    methods
        function obj = Wall(debut,fin,perm_relative,conductivite,Z_0,w,ep0)
            %MUR Construct an instance of this class
            %   Detailed explanation goes here
            obj.debut = debut;
            obj.fin = fin;
            obj.perm_relative = perm_relative;
            obj.conductivite = conductivite;
            e_c_r = perm_relative - 1i*conductivite/(w*ep0); % epsilon complexe
            obj.Z_2 = Z_0/sqrt(e_c_r); % Imp dance du mur.
        end
    end
end
```

A class that constructs "wave" objects carrying all the necessary information about waves, such as their reflection and transmission coefficients, which can change along the way.

```
classdef Wave
    %WAVE Summary of this class goes here
    %   Detailed explanation goes here

    properties
        d % Distance travelled by the wave.
        Gamma % Wave reflection coefficient.
        T % Transmission coefficient.
        I_t % Transmission index. Takes into account the number of transmissions already made.
        I_r % Reflection index. Takes into account the number of reflections already performed.
    end

    methods
        function obj = Wave(d)
            %ONDE Construct an instance of this class
            %   Detailed explanation goes here
            obj.d = d;
            obj.Gamma = 1; % Initialize reflection and transmission coefficients
            obj.T = 1; % to 1 to indicate that the wave has not yet impacted a wall.
            obj.I_t = 0;
            obj.I_r = 0;
        end

        function obj = Indice_transmission(obj)
            %METHOD1 Summary of this method goes here
            % Returns the transmission index incremented by 1.
            obj.I_t = obj.I_t + 1;
        end

        function obj = Indice_reflexion(obj)
            %METHOD1 Summary of this method goes here
            % Returns the reflection index incremented by 1.
            obj.I_r = obj.I_r + 1;
        end

        function obj = Modif_Reflexion(obj,Ref)
            %METHOD1 Summary of this method goes here.
            obj.Gamma = obj.Gamma*Ref;
        end

        function obj = Modif_Transmission(obj,Trans)
            %METHOD1 Summary of this method goes here.
            obj.T = obj.T*Trans;
        end
    end
end
```

This function repeats the image method with the help of other functions such as Cramer's method or "$Impact_{t}ransmission$".

```matlab
function [Nos_ondes] = Methode_image(Tx,Rx,Mur)

% Vector method. (For image method).
% Returns the coordinates of P.
% Returns the incident angle and the distance between TXprime and RX.
% for i=1:length(wall.debut) Repeat for all subsequent walls.
% Calculates wall slope and y-intercept

%% Initialisation

% For our direct wave.

d_directe = norm(Rx.position - Tx.position); % Distance.
Onde_directe = Wave(d_directe);
for z = 1:length(Mur)
    obj = Mur{z};
    Onde_directe = Impact_transmission(Onde_directe,obj,Tx.position,Rx.position);
end
Nos_ondes{1} = Onde_directe;% Storing our waves. Object matrix for waves.

%% Image method.

% 1st reflection.

for i = 1:length(Mur)
    obj_mur = Mur{i};
    [Tx_image,Impact,P_1,theta_i,distance] = Methode_de_Cramer(Tx.position,Rx.position,obj_mur);
    %Tx_image;
    if (Impact == 1) % Impact of Cramer's method. Impact of 1st reflection.
        Onde = Wave(distance);
        for z = 1:length(Mur) % Look at all the collisions between the base station and the imapct point for reflection.
            obj = Mur{z};
            Onde = Impact_transmission(Onde,obj,Tx.position,P_1);
            Onde = Impact_transmission(Onde,obj,P_1,Rx.position);
        end
        Ref = Coeff_Reflex(obj_mur,theta_i);
        % Trans = Coeff_Trans(obj_mur,theta_i);
        Onde = Modif_Reflexion(Onde,Ref); % Modifie le coeff. de reflexion
        Onde = Indice_reflexion(Onde);
        % Onde = Modif_Transmission(Onde,Trans); % Modifie le coeff. de transmission
%           if (dessin==1)
%               hold on
%               l1 = line([Tx.position(1) P_1(1) Rx.position(1)],[Tx.position(2) P_1(2) Rx.position(2)]); % Tracer du rayon.
%               set(l1,'color','b')
%           end
        Nos_ondes{end+1} = Onde;
    end

% second reflection.

    Mur_2 = Mur;
    Mur_2(i) = []; % At the end, the element must be returned to its cell.
    for j = 1:length(Mur_2)
        obj_mur2 = Mur_2{j};
        [Tx_image2,Impact2,P_2_2,theta_i2,distance] = Methode_de_Cramer(Tx_image,
```

```
Rx.position,obj_mur2);
        if (Impact2 == 1)
            [Tx_im,Impact3,P_2_1,theta_i1,dist_Tx_P2] = Methode_de_Cramer(Tx.position,P_2_2,obj_mur);
            if (Impact3 == 1)
                Onde = Wave(distance);
                for z = 1:length(Mur) % Look at all the collisions between the base station and the imapct point for reflection.
                    obj = Mur{z};
                    Onde = Impact_transmission(Onde,obj,Tx.position,P_2_1);
                    Onde = Impact_transmission(Onde,obj,P_2_1,P_2_2);
                    Onde = Impact_transmission(Onde,obj,P_2_2,Rx.position);
                end
                Ref = Coeff_Reflex(obj_mur2,theta_i2)*Coeff_Reflex(obj_mur,theta_i1);
                % Trans = Coeff_Trans(obj_mur2,theta_i2)*Coeff_Trans(obj_mur,theta_i1);
                Onde = Modif_Reflexion(Onde,Ref); % Modifie le coeff. de reflexion
                Onde = Indice_reflexion(Onde);
                % Onde = Modif_Transmission(Onde,Trans); % Modifie le coeff. de transmission
%                 if (dessin==1)
%                     hold on
%                     l2 = line([Tx.position(1) P_2_1(1) P_2_2(1) Rx.position(1)],[Tx.position(2) P_2_1(2) P_2_2(2) Rx.position(2)]); % Tracer du rayon.
%                     set(l2,'color','r')
%                 end
                Nos_ondes{end+1} = Onde;
            end
        end

        % third reflection.

%         Mur_3 = Mur;
%         Mur_3(i) = [];
%         Mur_3(j) = [];
%         for k = 1:length(Mur_3)
%             obj_mur3 = Mur_3{k};
%             [Tx_image3,Impact3,P_3_3,theta_i3,distance] = Methode_de_Cramer(Tx_image2,Rx.position,obj_mur3);
%             if (Impact3 == 1)
%                 [Tx_image2,Impact2,P_3_2,theta_i2,dist_TxIm_P_3_3] = Methode_de_Cramer(Tx_image,P_3_3,obj_mur2);
%                 if (Impact2 == 1)
%                     [Tx_im,Impact1,P_3_1,theta_i1,dist_Tx_P_3_2] = Methode_de_Cramer(Tx.position,P_3_2,obj_mur);
%                     if (Impact1 == 1)
%                         Onde = Wave(distance);
%                         for z = 1:length(Mur) % Regarde toutes les collisions entre la station de base et le point d'imapct pour la r flexion.
%                             obj = Mur{z};
%                             Onde = Impact_transmission(Onde,obj,Tx.position,P_3_1);
%                             Onde = Impact_transmission(Onde,obj,P_3_1,P_3_2);
%                             Onde = Impact_transmission(Onde,obj,P_3_2,P_3_3);
%                             Onde = Impact_transmission(Onde,obj,P_3_3,Rx.position);
%                         end
%                         Ref = Coeff_Reflex(obj_mur3,theta_i3)*Coeff_Reflex(obj_mur2,theta_i2)*Coeff_Reflex(obj_mur,theta_i1);
%                         Trans = Coeff_Trans(obj_mur3,theta_i3)*Coeff_Trans(obj_mur2,theta_i2)*Coeff_Trans(obj_mur,theta_i1);
```

```
%                          Onde = Modif_Reflexion(Onde,Ref); % Modifie le coeff. de
     reflexion
%                          Onde = Indice_reflexion(Onde);
%                          Onde = Modif_Transmission(Onde,Trans); % Modifie le
    coeff. de transmission
% %                          if (dessin==1)
% %                              hold on
% %                              l3 = line([Tx.position(1) P_3_1(1) P_3_2(1) P_3_3
    (1) Rx.position(1)],[Tx.position(2) P_3_1(2) P_3_2(2) P_3_3(2) Rx.position(2)
    ]); % Tracer du rayon.
% %                              set(l3,'color','y')
% %                          end
%                          Nos_ondes{end+1} = Onde;
%                      end
%                  end
%              end
    end
end
end
```

Calculates the transmitter's image point.

```
function [Tx_image,Impact,P,theta_i,distance] = Methode_de_Cramer(Tx,Rx,obj)
% This function returns the coordinates of the image base station.
% Principle based on the vectorial method seen in TP with the assistant.
% Video Teams is the FAQ for the project.
% See if U and Proj_S should be removed.

if (obj.debut(1) == obj.fin(1)) % Case we have a vertical wall.
    if (Tx(1) > obj.debut(1)) && (Rx(1) > obj.debut(1))
        n = [1,0]; % vector normal to the wall.
        U = [obj.fin(1) - obj.debut(1),obj.fin(2) - obj.debut(2)]/(norm(obj.fin - obj.debut));
    elseif (Tx(1) <= obj.debut(1)) && (Rx(1) <= obj.debut(1))
        n = [-1,0];
        U = [obj.fin(1) - obj.debut(1),obj.fin(2) - obj.debut(2)]/(norm(obj.fin - obj.debut));
    else
        Tx_image = [2*obj.debut(1) - Tx(1),Tx(2)];
        U = NaN;
        Proj_S = NaN;
        Impact = 0;
        P = NaN;
        distance = NaN;
        theta_i = NaN;
        return
    end
elseif (obj.debut(2) == obj.fin(2)) % Case the wall is horizontal.
    if (Tx(2) > obj.debut(2)) && (Rx(2) > obj.debut(2))
        n = [0,1]; % vector normal to the wall.
        U = [obj.fin(1) - obj.debut(1),obj.fin(2) - obj.debut(2)]/(norm(obj.fin - obj.debut));

    elseif (Tx(2) < obj.debut(2)) && (Rx(2) < obj.debut(2))
        n = [0,-1];
        U = [obj.fin(1) - obj.debut(1),obj.fin(2) - obj.debut(2)]/(norm(obj.fin - obj.debut));
    else
        Tx_image = [Tx(1),2*obj.debut(2) - Tx(2)];
        U = NaN;
        Proj_S = NaN;
        Impact = 0;
        P = NaN;
        distance = NaN;
        theta_i = NaN;
        return
    end
end

% Wall direction vector
% Vector between initial position of wall and Tx
% Projection vector between Tx and wall.

S = Tx - obj.debut;
Proj_S = 2*dot(S,n)*n; %Projection between Tx and Wall.

% vector between TX and the start of a wall.

Tx_image = Tx - Proj_S;

% This vector-based function returns whether there is a
% a point of impact and the point of impact.
```

```
da = Rx - Tx_image; % Vector between Txprime and Rx receiver.
distance = norm(da); % Norm of the da vector.

% Find the point of intersection between the wall and the radius from
% Tx.

t = -(da(2)*(obj.debut(1)-Tx_image(1))-da(1)*((obj.debut(2)-Tx_image(2))))/(da(2)*
    U(1) - da(1)*U(2));
L = norm(obj.fin - obj.debut); % Wall's length.

if (t > 0) && (t < L)
    P = obj.debut + t*U; % Intersection position between wall and Tx radius.
    Impact = 1;
    y = (norm(Proj_S)/2);
    x = sqrt(norm(P - Tx)^2 - y^2); % Pythagor.
    theta_i =  atan(x/y); % theta_i angle.

else
    Impact = 0;
    theta_i = 0;
    P = NaN;
end
end
```

```
function [Collision,theta_i] = Intersect(obj,p1,p2)
% Returns 1 and the point of impact if there is an impact between the wall
% and the line segment from point1 to point2.
% otherwise 0. We use Cramer's method.

% The direction vector for the wave.

V = (p2 - p1)/norm(p2 - p1);
W = (obj.fin - obj.debut)/norm(obj.fin - obj.debut);

PD = V(2)*W(1)-V(1)*W(2);

theta_i = 0;

if (PD == 0)
    Collision = 0;
elseif isfinite(V)
        t = -(V(2)*(obj.debut(1)-p1(1))-V(1)*(obj.debut(2)-p1(2)))/PD;
        L = norm(obj.fin - obj.debut); % Wall's length.
        if (t > 0) && (t < L)
            P_impact = obj.debut + t*W;
            if (norm(P_impact - p1) < norm(p2-p1)) && (norm(P_impact - p2) < norm(p2-p1))
                P_impact;
                Collision = 1;
                W_3d = [W(1) W(2) 0];
                V_3d = [V(1) V(2) 0];
                %theta_i = pi/2 - acos((V(1)*W(1) + V(2)*W(2))/(norm(V)*norm(W)))
                theta_i = pi/2 - atan2(norm(cross(W_3d,V_3d)),dot(W_3d,V_3d));
                %theta_i = 0;
                %hold on
                %scatter(P_impact(1),P_impact(2),300);
            else
                P_impact = 0;
                Collision = 0;
            end
        else
            Collision = 0;
        end
else
    Collision = 0;
end
end
```

Calculate the point of intersection with the wall.

```
function Onde = Impact_transmission(Onde,obj,Txposition,Rxposition)
    [Collision,theta] = Intersect(obj,Txposition,Rxposition);
    if (Collision == 1)
        trans = Coeff_Trans(obj,theta); % Calculates the coefficient's multiplying parameter.
        Onde = Modif_Transmission(Onde,trans); % Modifies the transmission coefficient
    end
end
```

Calculate the reflection coefficient.

```
function [Ref] = Coeff_Reflex(obj,theta_i)
% COEFF_REFLEX Summary of this function goes here
% This function simply returns the reflection coefficient.

%% Initialisation

Z_1 = 120*pi; % void impedance.
Z_2 = obj.Z_2; % Return object impedance value.
F = 27*10^9; % Frequency.
w = 2*pi*F; % Omega.
c = 3*10^8; % Wave velocity.
b = w/c;

% Vacuum permeability and permittivity.

mu0 = 1.25663706e-6; % Ici  taient  nos erreurs.
ep0 = 8.85418782e-12; % Ici aussi.

% Permittivity of the material.

epsilon = obj.perm_relative*ep0;

% Calculation of alpha and beta.

alpha = w*sqrt(mu0*epsilon/2)*sqrt(sqrt(1+(obj.conductivite/(w*epsilon))^2) - 1);
beta = w*sqrt(mu0*epsilon/2)*sqrt(sqrt(1+(obj.conductivite/(w*epsilon))^2) + 1);

gamma = alpha + 1j*beta; % The propagation constant

theta_t = asin(sin(theta_i)/sqrt(obj.perm_relative));
s = 0.5/cos(theta_t); % 0.5 is the wall thickness.

%% Formulas and computation

% Reflection and transmission coefficients for perpendicular polarization.
% The expressions for reflection and transmission.

Tref_orthogonal = (Z_2*cos(theta_i) - Z_1*cos(theta_t))/(Z_2*cos(theta_i) + Z_1*cos(theta_t));
% Ttrans_orthogonal = 2*Z_2cos(theta_i)/(Z_2*cos(theta_i) + Z_1*cos(theta_t));

Ref = Tref_orthogonal - (1 - Tref_orthogonal^2)*Tref_orthogonal*exp(-2*gamma*s)*exp(2*1j*b*s*sin(theta_t)*sin(theta_i))/(1 - Tref_orthogonal^2*exp(-2*gamma*s)*exp(1j*b*2*s*sin(theta_t)*sin(theta_i)));

end
```

Calculate the transmission coefficient.

```
function [Trans] = Coeff_Trans(obj,theta_i)
%COEFF_TRANS Summary of this function goes here
% Cette fonction renvoie simplement le coefficient de transmission.

%% Initialisation

Z_1 = 120*pi; % void impedance.
Z_2 = obj.Z_2; % Return object impedance value.
F = 27*10^9; % Frequency.
w = 2*pi*F; % Omega.
c = 3*10^8; % Wave velocity.
b = w/c;

% Vacuum permeability and permittivity.

mu0 = 1.25663706e-6; % Ici  taient  nos erreurs.
ep0 = 8.85418782e-12; % Ici aussi.

% Permittivity of the material.

epsilon = obj.perm_relative*ep0;

% Calculation of alpha and beta.

alpha = w*sqrt(mu0*epsilon/2)*sqrt(sqrt(1+(obj.conductivite/(w*epsilon))^2) - 1);
beta = w*sqrt(mu0*epsilon/2)*sqrt(sqrt(1+(obj.conductivite/(w*epsilon))^2) + 1);

gamma = alpha + 1j*beta; % The propagation constant

theta_t = asin(sin(theta_i)/sqrt(obj.perm_relative));
s = 0.5/cos(theta_t); % 0.5 is the wall thickness.

%% Formulas and computation

% Reflection and transmission coefficients for perpendicular polarization.
% The expressions for reflection and transmission.

Tref_orthogonal = (Z_2*cos(theta_i) - Z_1*cos(theta_t))/(Z_2*cos(theta_i) + Z_1*
    cos(theta_t));
% Ttrans_orthogonal = 2*Z_2cos(theta_i)/(Z_2*cos(theta_i) + Z_1*cos(theta_t));

%expo = Tref_orthogonal*exp(-2*gamma*s)*exp(1j*2*b*s*sin(theta_t)*sin(theta_i))

Trans = (1 - Tref_orthogonal^2)*exp(-gamma*s)/(1 - Tref_orthogonal^2*exp(-2*gamma*
    s)*exp(1j*2*b*s*sin(theta_t)*sin(theta_i)));

end
```

Function that returns bit rate and sensitivity in a local area of 1m^2.

```
function [Mb,Sens] = Prx(Tx,Nos_Ondes,h,Ra)
%PRX Summary of this function goes here
% Calculation of the power received at the transmitter station.
% On output, the code returns the power, its sensitivity and the associated
% baud rate.
%% Initialisation

F = 27*10^9;
w = 2*pi*F;
c = 3*10^8;

% Calculate antenna gain and power.
% According to the course, the directivity of the antenna is 1.7 (theta = 90  ).
% This means that the gain is the same as the directivity.

G_tx = Tx.gain;
P_tx = Tx.puissance_emmission;

% In the void, beta :

b = w/c;

%% Calculation of multi-component waves.
E = zeros(1,length(Nos_Ondes)); % Probleme
for k = 1:length(Nos_Ondes)
    ray = Nos_Ondes{k};
    d = ray.d; % Distance travelled by the wave
    Gamma = ray.Gamma; % Reflection coefficient associated with the wave
    T = ray.T; % Wave transmission coefficient
    E(k) = Gamma*T*sqrt(60*G_tx*P_tx)*exp(-1i*b*d)/d; % Calculates the electric field for each ray
end

%% Power calculation.
% Sum of all fields received.
P_rx = (1/(8*Ra))*sum(abs(h*E).^2); % Average power received.

%% Transformation en DBm.
Sens = 10*log(P_rx)/log(10)+30; % Our sensitivity. between -73 and -82
if Sens < -82
    Sens = -82;
    Mb = 0;
elseif (-82 < Sens) && (Sens < -73)
    Sens= 10*log(P_rx)/log(10)+30;
    Mb = (Sens*280 + 23320)/9; % Our bit rate.
    % The linear relationship is deduced from the values given since
else
    Sens = -73;
    Mb = 320;
end

end
```

Object created to simulate receivers every m^2.

```
classdef Recepteur
    %RECEPTEUR Summary of this class goes here
    %   Detailed explanation goes here

    properties
        position
        gain
        puissance_emmission
    end

    methods
        function obj = Recepteur(position)
            %RECEPTEUR Construct an instance of this class
            %   Detailed explanation goes here
            obj.position = position;
            obj.gain = 1.7;
            obj.puissance_emmission = 10^(20/10-3);
        end
    end
end
```

Object created to simulate transmitting antennas.

```
classdef Base
    %BASE Summary of this class goes here
    %   Detailed explanation goes here

    properties
        position
        gain
        puissance_emmission
    end

    methods
        function obj = Base(position)
            %BASE Construct an instance of this class
            %   Detailed explanation goes here
            obj.position = position;
            obj.gain = 1.7; % See the technical specifications pdf.
            obj.puissance_emmission = 10^(20/10-3); % See the technical specifications pdf.
        end
    end
end
```

Function that takes the main main3.m from one base station, but this time applies it to several base stations.

```
clear; close all; clc

%% D finition des constantes
tic

F = 27*10^9; % Fr quence de notre onde Wifi (5G). Changer plus tard    27.
w = 2*pi*F; % Om ga.

% Perm abilit  et permittivit  du vide.

mu0 = 1.25663706e-6; % Ici   taient   nos erreurs.
ep0 = 8.85418782e-12; % Ici aussi.

c = 3*10^8; % C  l rit   de l'onde.
Z_0 = 120*pi; % Imp dance du vide.

lambda = c/F; % Notre longeur d'onde.

% Hauteur caract ristique. D'apr s le cours, c'est une constante car nous
% Tra tons avec un dip le transverse au plan xy.

h = -lambda/pi; % La hauteur caract ristique. (5.42) o   th ta = pi/2.

% On suppose que l'antenne   metrice   est sans perte (rendement de 1).
% Par cons quent, on n glige la r sistance ohmique et seul la r sistance de
% rayonnement est prise en compte.

Ra = 73; % R sistance de rayonnement.

% Coordonn e station de base + r cepteur.

%Tx = Base([100,55]); % Coordonn e de notre station de base. Tx est un objet.
%Rx = Recepteur([89,77]);

% Permittivit  relative epsilon_r.

perm_relative_brique = 4.6; % Celle de la brique.
perm_relative_beton = 5; % Celle du b ton.

% Conductivit  des mat riaux sigma.

conductivite_brique = 0.02; % Celle de la brique.
conductivite_beton = 0.014; % Celle du b ton.
dessin = 0;

% Pour l'optimisation du code

taux = 70;

%% Cr ation des murs

% Les coordonn es des murs proviennent d'un fichier Excel.
% La fonction 'CVSexcel' pouvant lire ces coordonn es provient d'un
% algorithme de la biblioth que d'aide de Mathworks.
% Lien :
% Date de derni re modification :
% Auteur :
% La fonction a   t   modifi e et arrang e pour qu'elle puisse correspondre
% aux attentes et besoin pour ce projet.
```

```
[wallxyz1,wallxyz2,wallX,wallY] = CSVexcel;

[M,N] = size(wallxyz1); % M : Nombre de murs.  N : Nombre de colonnes.

% Initialisation d'une matrice cellule (acceuille les objets mur).
% Important de bien initialiser Mur sinon il va cr er    chaque fois
% une nouvelle variable pour stocker un objet mur. Pour ce code d ja
% gourmand, il faut l'optimiser un maximum.

a = zeros(1,length(wallxyz2));
Mur = num2cell(a);

for i = 1:M
   if i < 26 % Les premiers 26 murs sont en b tons.
       Mur{i} = Wall(wallxyz1(i,1:N),wallxyz2(i,1:N),perm_relative_beton,
    conductivite_beton,Z_0,w,ep0);
   else % Le reste est en brique.
       Mur{i} = Wall(wallxyz1(i,1:N),wallxyz2(i,1:N),perm_relative_brique,
    conductivite_brique,Z_0,w,ep0);
   end
end

% x = 1:1:(max(max(wallX)));
% y = 1:1:(max(max(wallY)));
% [X,Y] = meshgrid(x,y);
%
% plot(wallX,wallY,'b')%visualisation des murs
% text(Tx.position(1),Tx.position(2),'TX','Color','b') % Mettre un point dans le
    futur.
% text(Rx.position(1),Rx.position(2),'RX','Color','b') % Mettre un point dans le
    futur.
% hold on
%
%
% Nos_ondes = Methode_image(Tx,Rx,Mur,dessin)
% [Mb,Sens]  = Prx(Tx,Nos_ondes,h,Ra)
%
% toc


%% Truff  la carte de r cepteur Rx. (Pas de collisions avec les murs)

S = zeros(max(max(wallX)),max(max(wallY)));
Matrice_recepteur = num2cell(S);
for i = 0:1:max(max(wallX)) % Modifier si la carte prend des valeurs n gatives.
    for j = 0:1:max(max(wallY)) % Modifier ici  galement  .
        Matrice_recepteur{i+1,j+1} = Recepteur([i,j]);
        % scatter(i,j,300);
    end
end

%% Cr ation d'une matrice de station de base.

M = zeros(3,4);
Matrice_base = num2cell(M);
for j = 1:3
    for pz = 1:4
        Matrice_base{j,pz} = Base([46 + (j-1)*60, 26 + (pz-1)*20]);
    end
end

```

```
%% Application de la m thode des images (M thode vectorielle). + Calcul de la puissance re ue.
Mb = zeros(max(max(wallX))+1,max(max(wallY))+1);
Sens = zeros(max(max(wallX)+1),max(max(wallY))+1);
Sens(max(max(wallX)+1,max(max(wallY)+1))) = -82;

for m = 1:size(Matrice_base,1)*size(Matrice_base,2) % Le nombre d'  lments   dans la matrice
    disp(m)
    for a = 1:max(max(wallX))
        disp(a)
        parfor g = 1:max(max(wallY))
            Nos_ondes = Methode_image(Matrice_base{m},Matrice_recepteur{a,g},Mur,dessin);
            [Mb(a,g),Sens(a,g)]  = Prx(Matrice_base{m},Nos_ondes,h,Ra);
        end
    end
    Matrice_base{m} = Modif_zone_de_couverture(Matrice_base{m},Mb);
end

%     Matrice_base{m} = Modif_zone_de_couverture(Matrice_base{m},Mb);
%     Mb = zeros(max(max(wallX))+1,max(max(wallY))+1);


%% Optimisation de l'algorithme

% Pour 1 combinaison d'antenne.

Mb = zeros(max(max(wallX))+1,max(max(wallY))+1);

Maxi = 0;
Seuil = 175;
taux = 85;

for a = 1:size(Matrice_base,1)*size(Matrice_base,2)
    k = 0;
    for i = 1:size(Matrice_base{a}.zone_de_couverture,1)
        for j = 1:size(Matrice_base{a}.zone_de_couverture,2)
            if Matrice_base{a}.zone_de_couverture(i,j) > Seuil
                k = k + 1;
            end
        end
    end
    if k > Maxi
        Maxi = k;
        Mb_1 = Matrice_base{a}.zone_de_couverture;
        L = [Matrice_base{a}.position(1),Matrice_base{a}.position(2)];
    end
end


compteur = 0;

for q = 1:size(Mb_1,1)*size(Mb_1,2)
     if Mb_1(q) > Seuil
         compteur = compteur + 1;
     end
end


Taux_de_couverture = compteur/(size(Mb,1)*size(Mb,2))*100; % Exprim  en %

```

```matlab
if Taux_de_couverture > taux
    if Mb == zeros(max(max(wallX))+1,max(max(wallY))+1)
        Mb = Mb_1;
        disp('1 combinaison')
    end
end

% Pour 2 combinaisons d'antennes.

Maxi = 0;

for a = 1:size(Matrice_base,1)*size(Matrice_base,2)
    for b = a+1:size(Matrice_base,1)*size(Matrice_base,2)
        k = 0;
        for i = 1:size(Matrice_base{a}.zone_de_couverture,1)*size(Matrice_base{a}.zone_de_couverture,2)
            if Matrice_base{a}.zone_de_couverture(i) + Matrice_base{b}.zone_de_couverture(i) > Seuil
                k = k + 1;
            end
        end
        if k > Maxi
            Maxi = k;
            Mb_2 = min(Matrice_base{a}.zone_de_couverture + Matrice_base{b}.zone_de_couverture, 320);% Si un  lment  plus que grand 320 alors il est satur .
            L_2_1 = [Matrice_base{a}.position(1),Matrice_base{a}.position(2)];
            L_2_2 = [Matrice_base{b}.position(1),Matrice_base{b}.position(2)];
        end
    end
end

compteur = 0;

for q = 1:size(Mb,1)*size(Mb,2)
     if Mb(q) > Seuil
         compteur = compteur + 1;
     end
end

Taux_de_couverture = compteur/(size(Mb,1)*size(Mb,2))*100; % Exprim  en %

if Taux_de_couverture > taux
    if Mb == zeros(max(max(wallX))+1,max(max(wallY))+1)
        Mb = Mb_2;
        L = L_2_1;
        L2 = L_2_2;
        disp('2 combinaison')
    end
end

% Pour 3 combinaisons d'antennes.

Maxi = 0;

for a = 1:size(Matrice_base,1)*size(Matrice_base,2)
    for b = a+1:size(Matrice_base,1)*size(Matrice_base,2)
        for c = b+1:size(Matrice_base,1)*size(Matrice_base,2)
           k = 0;
           for i = 1:size(Matrice_base{a}.zone_de_couverture,1)*size(Matrice_base{a}.zone_de_couverture,2)
              if Matrice_base{a}.zone_de_couverture(i) + Matrice_base{b}.
```

```
    zone_de_couverture(i) + Matrice_base{c}.zone_de_couverture(i)> Seuil
                        k = k + 1;
                    end
                end
                if k > Maxi
                    Maxi = k;
                    Mb_3 = min(Matrice_base{a}.zone_de_couverture + Matrice_base{b}.
    zone_de_couverture + Matrice_base{c}.zone_de_couverture, 320);
                    % Si un   lment   plus que grand 320 alors il est satur .
                    L_3_1 = [Matrice_base{a}.position(1),Matrice_base{a}.position(2)];
                    L_3_2 = [Matrice_base{b}.position(1),Matrice_base{b}.position(2)];
                    L_3_3 = [Matrice_base{c}.position(1),Matrice_base{c}.position(2)];
                end
            end
        end
end

compteur = 0;

for q = 1:size(Mb_3,1)*size(Mb_3,2)
     if Mb_3(q) > Seuil
         compteur = compteur + 1;
     end
end

Taux_de_couverture = compteur/(size(Mb,1)*size(Mb,2))*100; % Exprim  en %

if Taux_de_couverture > taux
    if Mb == zeros(max(max(wallX))+1,max(max(wallY))+1)
        Mb = Mb_3;
        L = L_3_1;
        L2 = L_3_2;
        L3 = L_3_3;
        disp('3 combinaison')
    end
end

% Pour 4 combinaisons d'antennes.

Maxi = 0;

for a = 1:size(Matrice_base,1)*size(Matrice_base,2)
    for b = a+1:size(Matrice_base,1)*size(Matrice_base,2)
        for c = b+1:size(Matrice_base,1)*size(Matrice_base,2)
            for d = c+1:size(Matrice_base,1)*size(Matrice_base,2)
                k = 0;
                for i = 1:size(Matrice_base{a}.zone_de_couverture,1)*size(
    Matrice_base{a}.zone_de_couverture,2)
                    if Matrice_base{a}.zone_de_couverture(i) + Matrice_base{b}.
    zone_de_couverture(i) + Matrice_base{c}.zone_de_couverture(i) + Matrice_base{d
    }.zone_de_couverture(i)> Seuil
                        k = k + 1;
                    end
                end
                if k > Maxi
                    Maxi = k;
                    Mb_4 = min(Matrice_base{a}.zone_de_couverture + Matrice_base{d
    }.zone_de_couverture + Matrice_base{b}.zone_de_couverture + Matrice_base{c}.
    zone_de_couverture, 320);
                    % Si un   lment   plus que grand 320 alors il est satur .
                    L_4_1 = [Matrice_base{a}.position(1),Matrice_base{a}.position
    (2)];
```

```
                    L_4_2 = [Matrice_base{b}.position(1),Matrice_base{b}.position(2)];
                    L_4_3 = [Matrice_base{c}.position(1),Matrice_base{c}.position(2)];
                    L_4_4 = [Matrice_base{d}.position(1),Matrice_base{d}.position(2)];
                    %disp(c)
                end
            end
        end
    end
end

compteur = 0;

for q = 1:size(Mb_4,1)*size(Mb_4,2)
     if Mb_4(q) > Seuil
         compteur = compteur + 1;
     end
end

Taux_de_couverture = compteur/(size(Mb_4,1)*size(Mb_4,2))*100; % Exprim  en %

if Taux_de_couverture > taux
    if Mb == zeros(max(max(wallX))+1,max(max(wallY))+1)
        Mb = Mb_4;
        L = L_4_1;
        L2 = L_4_2;
        L3 = L_4_3;
        L4 = L_4_4;
        disp('4 combinaison')
    end
end

% Pour 5 combinaisons d'antennes.

Maxi = 0;

for a = 1:size(Matrice_base,1)*size(Matrice_base,2)
    for b = a+1:size(Matrice_base,1)*size(Matrice_base,2)
        for c = b+1:size(Matrice_base,1)*size(Matrice_base,2)
            for d = c+1:size(Matrice_base,1)*size(Matrice_base,2)
                for e = d+1:size(Matrice_base,1)*size(Matrice_base,2)
                    k = 0;
                    for i = 1:size(Matrice_base{a}.zone_de_couverture,1)*size(Matrice_base{a}.zone_de_couverture,2)
                        if Matrice_base{a}.zone_de_couverture(i) + Matrice_base{e}.zone_de_couverture(i) + Matrice_base{b}.zone_de_couverture(i) + Matrice_base{c}.zone_de_couverture(i) + Matrice_base{d}.zone_de_couverture(i)> Seuil
                            k = k + 1;
                        end
                    end
                    if k > Maxi
                    Maxi = k;
                    Mb_5 = min(Matrice_base{a}.zone_de_couverture + Matrice_base{d}.zone_de_couverture + Matrice_base{e}.zone_de_couverture + Matrice_base{b}.zone_de_couverture + Matrice_base{c}.zone_de_couverture, 320);
                    % Si un   lment   plus que grand 320 alors il est satur .
                    L_5_1 = [Matrice_base{a}.position(1),Matrice_base{a}.position(2)];
                    L_5_2 = [Matrice_base{b}.position(1),Matrice_base{b}.position(2)];
```

```
                    L_5_3 = [Matrice_base{c}.position(1),Matrice_base{c}.position(2)];
                    L_5_4 = [Matrice_base{d}.position(1),Matrice_base{d}.position(2)];
                    L_5_5 = [Matrice_base{e}.position(1),Matrice_base{e}.position(2)];
                    end
                end
            end
        end
    end
end

compteur = 0;

for q = 1:size(Mb_5,1)*size(Mb_5,2)
     if Mb_5(q) > Seuil
          compteur = compteur + 1;
     end
end

Taux_de_couverture = compteur/(size(Mb_5,1)*size(Mb_5,2))*100; % Exprim  en %

if Taux_de_couverture > taux
    if Mb == zeros(max(max(wallX))+1,max(max(wallY))+1)
        Mb = Mb_5;
        L = L_5_1;
        L2 = L_5_2;
        L3 = L_5_3;
        L4 = L_5_4;
        L5 = L_5_5;
        disp('5 combinaison')
    end
end

%% Analyse des donn es re ues.
x = 0:1:(max(max(wallX)));
y = 0:1:(max(max(wallY)));
[X,Y] = meshgrid(x,y);

%contour(X,Y,Mb)

% adding color map and showing it on the screen.

%ax1 = subplot(1,1,1);

figure

contourf(X,Y,Mb',150,'LineColor','none')
title('D bit binaire [Mb/s]')
set(gca, 'DataAspectRatio', [1 1 1])
colormap(jet);
caxis([40 320])
c = colorbar;
c.Label.String = 'Mb/s';
hold on
plot(wallX,wallY,'w')%visualisation des murs
%text(Tx.position(1),Tx.position(2),'TX','Color','w') % Mettre un point dans le futur.
%title('Puissance Recue Dans le MET, Station de base(100.65) : Ray Tracing avec 5G 27GHz')
% text(Rx.position(1),Rx.position(2),"RX",'Color','w') % Mettre un point dans le
```

```
futur.
axis([0 200 0 200])
set(gca,'color','k')
set(gcf,'color','w')
scatter(L(1),L(2),10,'fill','blue')
scatter(L2(1),L2(2),10,'fill','blue')
scatter(L3(1),L3(2),10,'fill','blue')
scatter(L4(1),L4(2),10,'fill','blue')
scatter(L5(1),L5(2),10,'fill','blue')

% figure
%
% contourf(X,Y,Sens',150,'LineColor','none')
% title('Sensibilit  [dBm]')
% set(gca, 'DataAspectRatio', [1 1 1])
% colormap(hot);
% caxis([-82 -73])
% c = colorbar;
% c.Label.String = 'dBm';
% hold on
% plot(wallX,wallY,'w')%visualisation des murs
% %text(Tx.position(1),Tx.position(2),'TX','Color','w') % Mettre un point dans le futur.
% %title('Puissance Recue Dans le MET, Station de base(100.65) : Ray Tracing avec 5G 27GHz')
% % text(Rx.position(1),Rx.position(2),"RX",'Color','w') % Mettre un point dans le futur.
% axis([0 200 0 200])
% set(gca,'color','k')
% set(gcf,'color','w')
% scatter(100,55,5,'fill')

toc
```