ULB

BRUFACE
BRUSSELS FACULTY
OF ENGINEERING

VUB

ELEC-H-415 - Communication Channels

Channel modeling for 5G small cells

MA1 - Electronics and Information Technology

Professor : Philippe De Doncker

Oscar Van Slijpe
Théo Lepoutte

Academic Year 2020-2021

Contents

Introduction 1
1 Program description 2
2 Global parameters 3
2.1 Power received and SNR 3
2.2 Delay spread and Rice factor 7
3 Local parameters 10
3.1 Theoretical analyses 10
3.2 Case application 11
4 Propagation model 17
4.1 Path loss model 17
4.2 Fading variability 17
4.3 Cell range 18
4.4 Penetration depth of coverage 19
Conclusion 20
A Appendix : Codes used for physical computations 21
B Appendix : Simulation validation 26

Introduction

Modeling the coverage area of a telecommunication antenna is an important issue of our era. The deployment of new telecommunication technologies such as 5G requires in-depth knowledge of the behaviour of the waves with the surrounding environment. Unlike the usual urban cells, 5G small cells base stations (BS) will be installed on urban furniture at an height equivalent to the height of the user equipment (UE) and will communicate at high frequencies with large bandwidth. Students are asked to model the channel of a small cell located in Rue de la Loi/Wetstraat using ray-tracing.

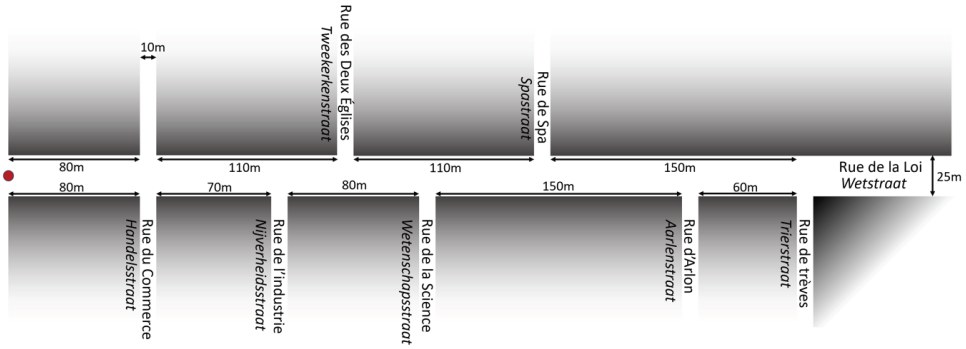


Figure 1: Case study: Rue de la Loi/Wetstraat

The physical constants and different parameters used for the project are shown in the Table 1.

c	Speed of light	$2.997\,924\,10^{8}$m/s
k	Boltzmann constant	$1.380\,659\,10^{-23}$J/K
T	Ambient temperature	293.15 K
f	Carrier frequency	27 GHz
B	Bandwidth	Up to 200 MHz
h	Antenna height	2 m
$EIRP$	Effective Isotropic Radiated power	2 W
F_{dB}	Noise figure at UE	10 dB
SNR_t	Target SNR at UE	2 dB
ε_r	Wall's relative permittivity	5
h	Antenna height	2 m

Table 1: Parameters

1 Program description

For this project, a ray-tracing simulation program has been made using Qt and C++. It is designed to work in general cases, the program recognises the main street zone based on the distance between the BS and the first wall/end of the map, taken horizontally and vertically[1]. This zone is used to decide whether or not we compute reflection on the building or diffraction, as asked for this project. The map can be manually drawn and saved and its size can be configured.

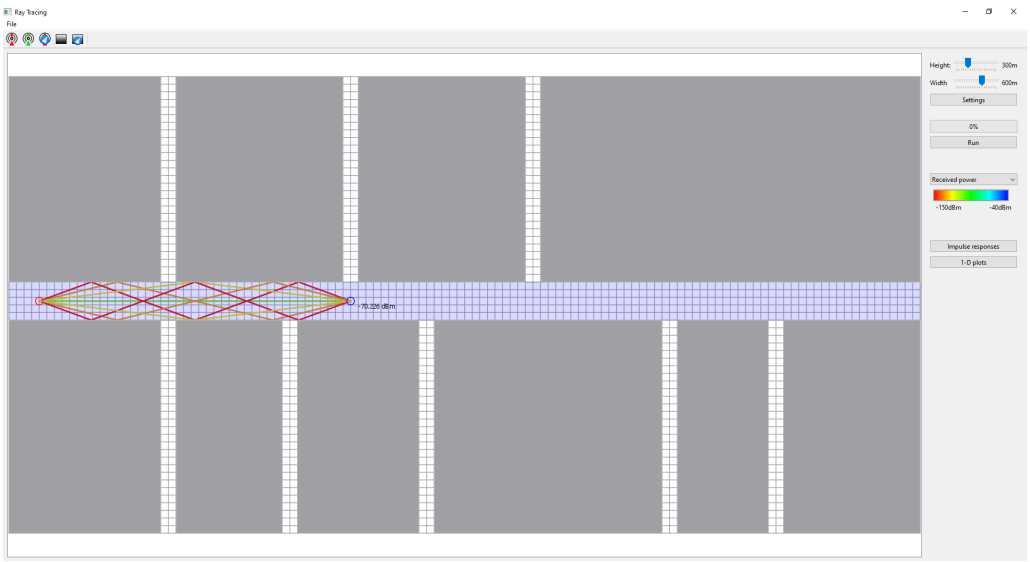


Figure 2: main windows

The ray paths and local parameters (Physical, TDL and uncorrelated scattering TDL impulse response) are computed when a BS and UE are placed, the rays are shown directly on the map and the impulses responses can be visualised in an external window. The global parameters (received power, SNR at UE, delay spread and rice factor) can be visualised globally on heat maps which are drawn on top of the map and also on 1D-plots along the main street. It is also possible to add multiple BS.
Some parameters as bandwidth, temperature, antenna height or noise figure can be changed directly on the program UI.

[1]This works well on the given geometry where the main street width is constant and the BS is not in the middle of an intersection but it should be improved in order to work properly on more complex geometry cases

2 Global parameters

The computation of received power, SNR, delay spread and rice factor has to be done everywhere on the map (at the centre of each 1m by 1m squares) and visualised with a heat map and a 1-D plot along the main street.

The validation of the simulation for those parameters is done on simple cases in Appendix B, by comparing the simulation results to the theoretical ones that are described here.

2.1 Power received and SNR

Theoretical analyses

The power received at the UE for all MPCs can be obtained using the urban canyon model as we are working with an outdoor small cell:

$$P_{RX}(d) = \frac{1}{8R_a}|\underline{V}_{oc}(\vec{r})|^2 \tag{1}$$

with R_a being the radiation resistance of the antenna $= \frac{720\pi}{32}$ and $\underline{V}_{oc}$ the sum of the voltages induced by each MPC considering incident plane waves. This voltage can be computed as follow:

$$\underline{V}_{oc}(\vec{r}) = \sum_{n=1}^{N} \vec{h}_e(\theta_n, \phi_n)\, \underline{\vec{E}}_n(\vec{r}) \tag{2}$$

where $\vec{h}_e$ is the effective height of an antenna:

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

For vertical half-wave dipole antennas on the z-axis we know that

$$\begin{cases} \vec{r} &= z\vec{1}_r \\ \underline{I}(z) &= \underline{I}_a cos(\beta z) \\ \underline{\vec{J}}(\vec{r})dV &= \underline{I}(z)dz\vec{1}_z \end{cases}$$

so that the effective height becomes

$$\vec{h}_e(\theta, \phi) = -\frac{1}{\underline{I}_a} \int_{-\lambda/4}^{\lambda/4} \underline{I}(z) e^{j\beta z(\vec{1}_z\,\vec{1}_r)} dz\vec{1}_z \vec{h}_e(\theta) = -\frac{\lambda}{\pi} \frac{cos(\frac{\pi}{2}cos(\theta))}{sin^2(\theta)} \vec{1}_z \tag{3}$$

In particular, for a horizontal plane wave, $\theta = 90°$ which give us the maximal effective height:

$$\vec{h}_{e,max} = -\frac{\lambda}{\pi}\vec{1}_z \tag{4}$$

For half-wave dipole antennas, the electric field is given by:

$$\underline{\vec{E}} = j\frac{Z_0 \underline{I}_a}{2\pi}\frac{cos(\frac{\pi}{2}cos(\theta))}{sin(\theta)}\vec{1}_\theta$$

From that, we can deduce the radiated intensity U, the total radiated power P_{ar} and the gain G for a lossless dipole under the approximation $\left(\frac{cos(\frac{\pi}{2}cos(\theta))}{win(\theta)}\right)^2 \approx sin^3(\theta)$.

$$\begin{cases} U(\theta) = r^2\frac{|\vec{E}|^2}{2Z_0} = Z_0\frac{\left|\underline{I}_a\right|^2}{8\pi^2}\left(\frac{\cos\left(\frac{\pi}{2}\cos\theta\right)}{\sin\theta}\right)^2 \\ P_{ar} = \int_0^{2\pi}\int_0^{\pi} U(\theta)\sin\theta d\theta d\phi = \frac{3}{32}Z_0\left|\underline{I}_a\right|^2 \\ G(\theta) = D(\theta) = \frac{U(\theta)}{P_{ar}/4\pi} = \frac{16}{3\pi}\sin^3\theta \end{cases} \tag{5}$$

The gain $G_{TX}(\theta)$ is also maximal for $\theta = 90°$:

$$G_{TX,max} = \frac{16}{3\pi} \tag{6}$$

We can write one electric field component arriving to the antenna with an incidence angle θ_n as:

$$\underline{\vec{E}}_n = \chi_n\sqrt{60G_{TX}(\theta_n)P_{TX}}\frac{e^{-j\beta d_n}}{d_n}\vec{1}_\theta \tag{7}$$

where χ_n is the product of all the coefficients due to reflection and diffraction (transmission is not considered in this project).
As the Effective Isotropic Radiated Power (EIRP) for this project is imposed at $2W$ we obtain a radiated power $P_{TX} = 1.178W$. This result is given following the relation:

$$P_{TX} = \frac{\mathrm{EIRP}\,L_{TX}}{G_{TX}} = \frac{3\pi\,\mathrm{EIRP}}{16} = \frac{3\pi}{8} \tag{8}$$

where $L_{TX} = 1$ as we consider lossless antennas.
The reflection can be either orthogonal (reflection over the ground for a vertically polarised antenna) or parallel (reflection on the building in the same horizontal plane for vertically polarised antenna). The coefficients for each case are given by:

$$\Gamma_\perp = \frac{\cos\theta_i - \sqrt{\varepsilon_r}\sqrt{1-\frac{1}{\varepsilon_r}\sin^2\theta_i}}{\cos\theta_i + \sqrt{\varepsilon_r}\sqrt{1-\frac{1}{\varepsilon_r}\sin^2\theta_i}}, \qquad \Gamma_\parallel = \frac{\cos\theta_i - \frac{1}{\sqrt{\varepsilon_r}}\sqrt{1-\frac{1}{\varepsilon_r}\sin^2\theta_i}}{\cos\theta_i + \frac{1}{\sqrt{\varepsilon_r}}\sqrt{1-\frac{1}{\varepsilon_r}\sin^2\theta_i}} \tag{9}$$

The diffraction coefficient for the Knife-edge model can be approximated with $d_1 >> h$ and $d_2 >> h$:

$$\begin{cases} |F(\nu)|^2[\text{ dB}] & \approx & -6,9 - 20\log\left(\sqrt{(\nu-0,1)^2+1}+\nu-0,1\right) \\ \text{Arg}\, F(\nu) & = & -\frac{\pi}{4}-\frac{\pi}{2}\nu^2 \\ \nu & = & h\sqrt{\frac{2}{\lambda}\left(\frac{1}{d_1}+\frac{1}{d_2}\right)} \approx \sqrt{\frac{2}{\pi}\beta\Delta r} \end{cases} \tag{10}$$

The SNR at the receiver is computed using the imposed receiver noise figure equal to 10 dB.

$$SNR\,[dB] = P_{RX}\,[\text{dBW}] - F_{dB} - 10\log_{10}(kTB) \tag{11}$$

$$= P_{RX}\,[\text{dBm}] - 30 - F_{dB} - 10\log_{10}(kTB) \tag{12}$$

Case application

The heat map of the received power (Figure 3) indicates that, as expected, the power decreases following the main street and that the secondary streets have low power in comparison due to the fact that diffraction is less "efficient" than direct ray and reflections added. On the SNR heat map of Figure 4, when the received SNR is lower than the minimum target SNR_t, it is displayed in a more transparent colour. We observe that the secondary streets are always below that target SNR while for the main street, the places that are far from the BS or where interferences between the direct ray and the ground reflected ray occur are also below the target SNR.

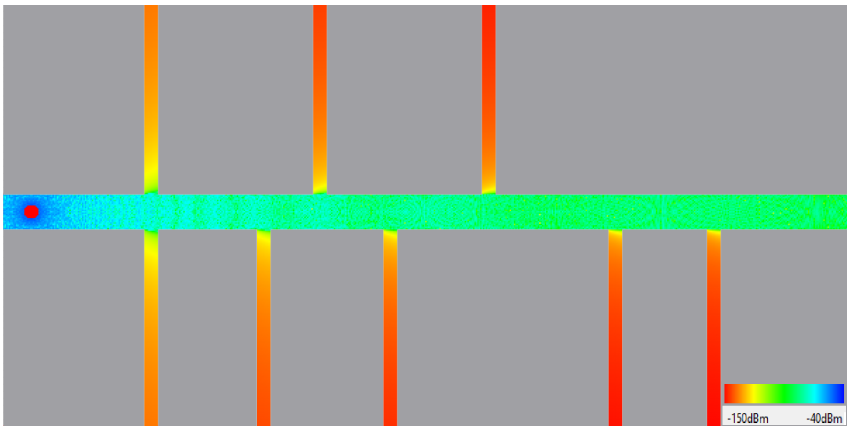


Figure 3: Heat map - Power

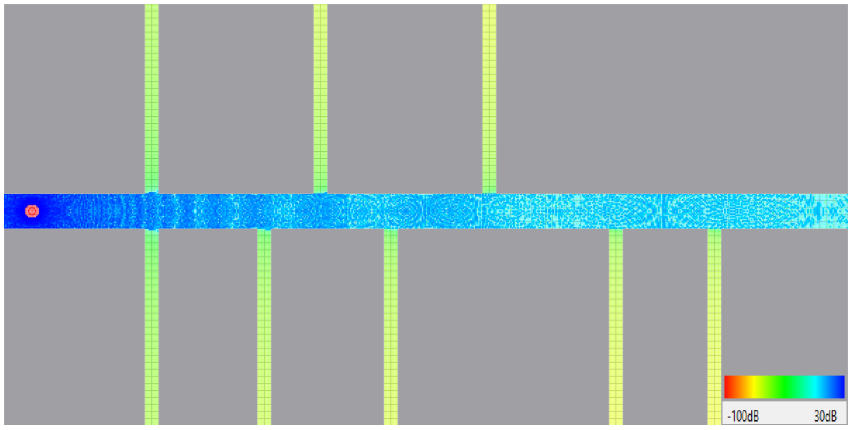


Figure 4: Heat map - SNR

The 1D-plots of the power and the SNR show a lot of interferences but overall, a linear decrease is observed when plotted with the log of the distance. This figure will be discussed more in Section 4.

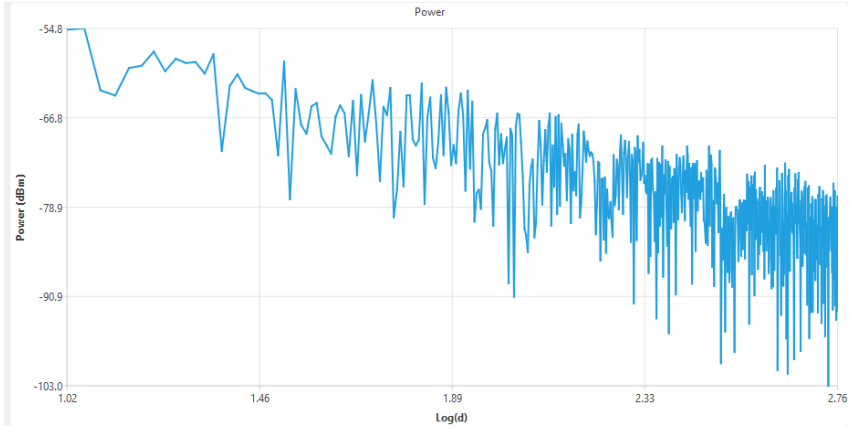


Figure 5: 1-D plot - Power

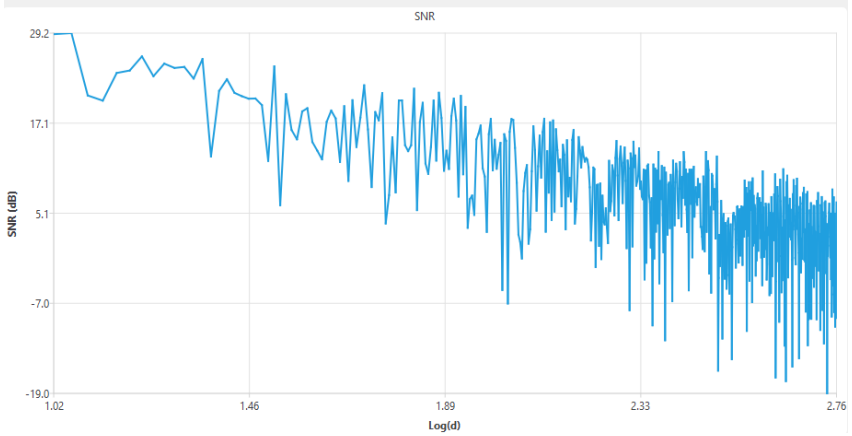


Figure 6: 1-D plot - SNR

2.2 Delay spread and Rice factor

Theoretical analyses

The **delay spread** is the maximal time difference between the first ray to arrive and the last one. The time of arrival can be obtained from the distance travelled by a ray, using the speed of light $\tau_i = \frac{d_i}{c}$.

$$\sigma_\tau = max|\tau_i - \tau_j| \quad \forall i, j \tag{13}$$

In particular, when there is only one ray (diffraction) $\sigma_\tau = 0$ and when there is only LOS and ground reflection with $d_{direct} >> 2h : d_{direct} \approx d_{ground}$ so that $\sigma_\tau \approx 0$. We expect to see the highest delay spread near the BS as the difference of distance travelled (and so delay) between the direct ray and the reflected ones will be maximal (in particular, for our application case, this will be only above or below the BS because there is no possible reflection at the beginning or at the end of the main street).

The **rice factor** is the ratio between the squared direct ray amplitude a_0 and the sum of all the other N squared rays amplitudes:

$$K = \frac{a_0^2}{\sum_{i=1}^{N} a_i^2} \tag{14}$$

Therefore, if there is no LOS, $K = 0$.

Case application

As expected, the delay spread is higher close to the BS because the difference of the path lengths- between the reflection and the direct ray is relatively higher than in the case where we are far away from the BS. In the secondary street we observe the expected delay spread = 0 due to only one ray (diffraction).

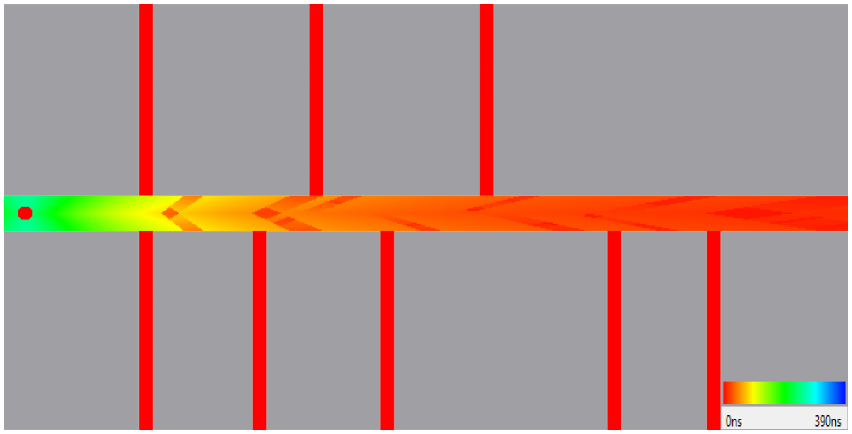


Figure 7: Heat map - Delay spread

Similarly, the rice factor is higher when close to the BS as the path length for the reflection is higher than for the direct ray. Far from the BS, the path length will be almost similar for reflected rays and direct ray and the absolute value of the wall reflection coefficients (Equation 9) increases for higher incident angles (for $d >> 2h$, $\Gamma_{\parallel} \approx -1$). The place where the rice factor is null (diffraction only) is represented by a completely transparent color.

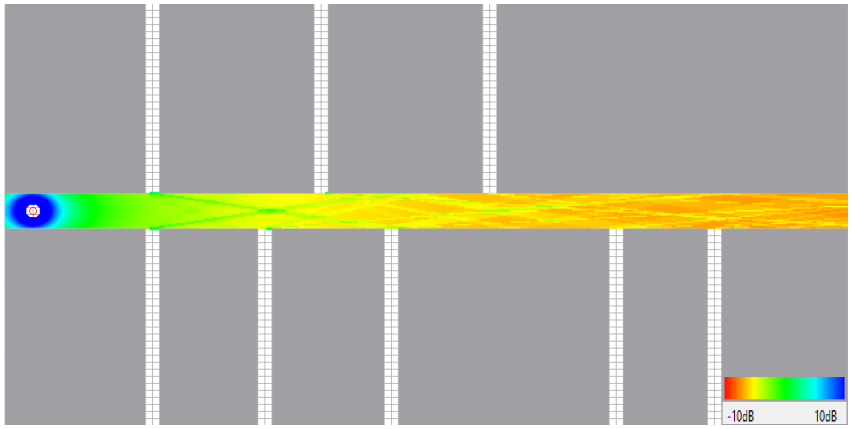


Figure 8: Heat map - Rice factor

These evolutions are easier to see on the 1-D plots (Figure 9 and 10). Here we can analyse the impact of the presence of secondary streets where the reflection is not possible as there is no wall. The delay spread decreases and the rice factor increases, which is due to the fact that there is less reflected rays arriving.

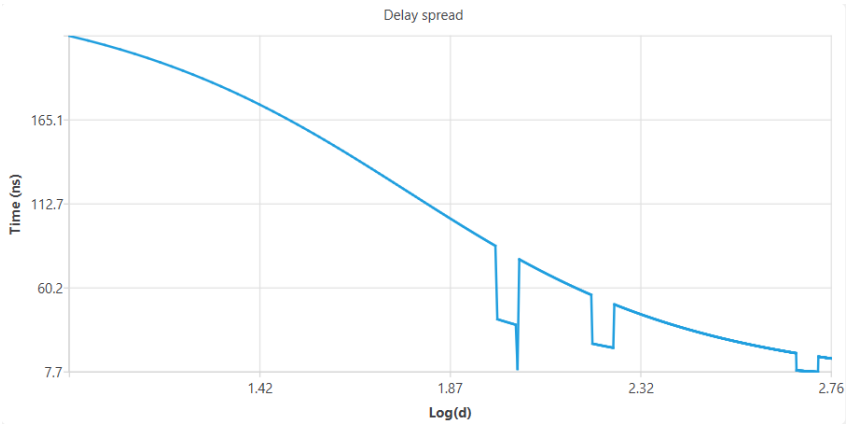


Figure 9: 1-D plot - Delay spread

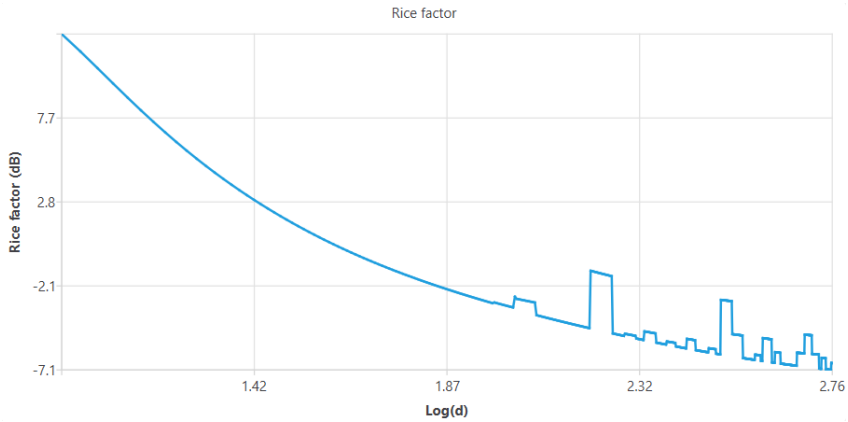


Figure 10: 1-D plot - Rice factor

3 Local parameters

The analyses of the time of arrival of each rays have to be done only for some points but for multiple bandwidths, from narrowband to $200MHz$. The narrowband limit is $B << \Delta f_c$ with the coherent bandwidth $\Delta f_c = \frac{1}{\sigma_\tau}$.

3.1 Theoretical analyses

Physical impulse responses

For the physical impulse responses, we have to consider the time of arrival of each ray, which is directly proportional to the travelled distance: $\tau = d/c$. We can then visualise the impulses due to each ray.

TDL impulse responses

The UE does not have infinite bandwidth, and for a bandwidth B it would need a sampling frequency $f_s = 2\,B$ which result in sample taps of duration $\Delta\tau = \frac{1}{2B}$. Knowing that the expression of a band-limited signal is $x(t-\tau) = \sum_{l=-\infty}^{\infty} x(t - l\Delta\tau) sinc(2B(\tau - l\Delta\tau))$, we can compute the Tapped Delay Line (TDL) impulse response:

$$\begin{cases} h_{\text{TDL}}(\tau, t) &= \sum_{l=0}^{L} h_l(t)\delta(\tau - l\Delta\tau) \\ h_l(t) &= \sum_{n=1}^{N} \alpha_n(t)\,\text{sinc}\left(2B\left(\tau_n - l\Delta\tau\right)\right) \end{cases} \tag{15}$$

Uncorrelated scattering TDL impulse responses

If we consider that the value of *sinc* is not negligible only when $\tau_n \; l\Delta\tau$, we can do a first approximation to obtain:

$$h_l(t) \simeq \sum_{\tau_n \in \text{ tap } l} \alpha_n(t) = \sum_{\tau_n \in \text{ tap } l} a_n(t) e^{j\phi_n(t)} e^{-j2\pi f_c \tau_n} \tag{16}$$

3.2 Case application

Intersection 1

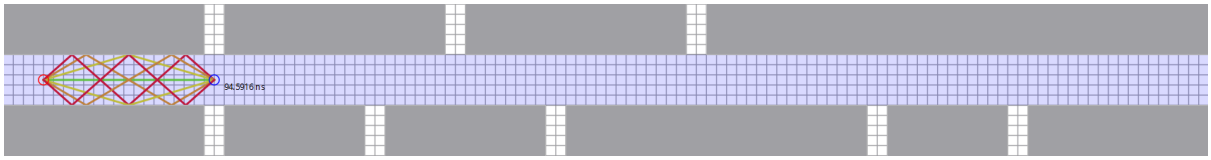
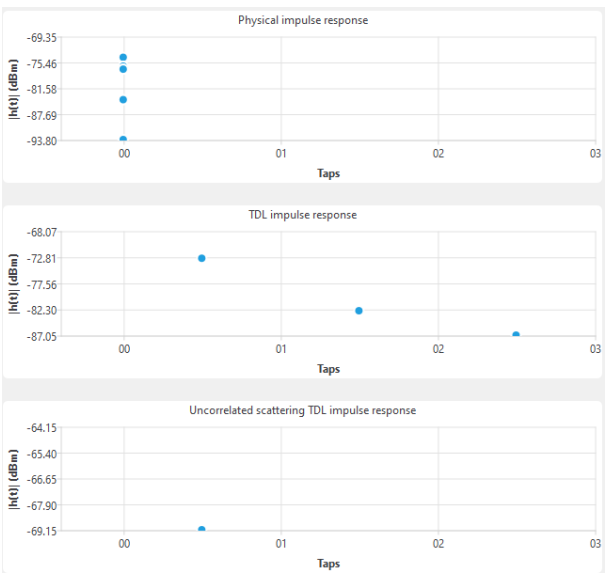


Figure 11: Intersection 1 - geometry

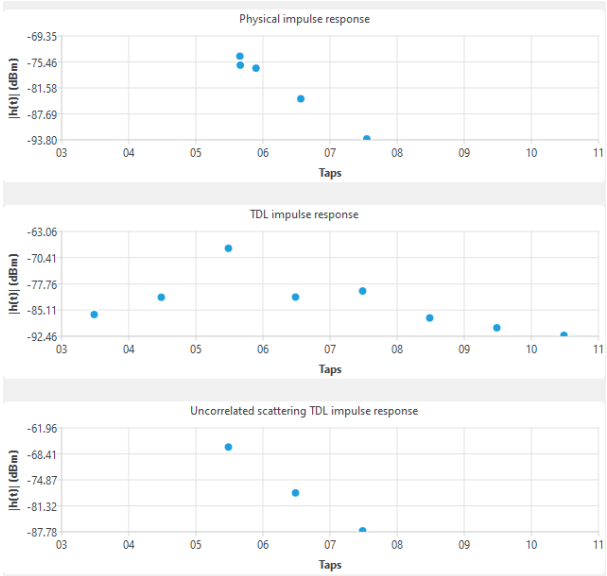
$\sigma_\tau = 94.6ns$	$SNR = 11.8dB$	$P_{rx} = -69.1dBm$	$K = -2.8dB$	$\Delta f_c = 10.6MHz$

Table 2: Intersection 1 - parameters

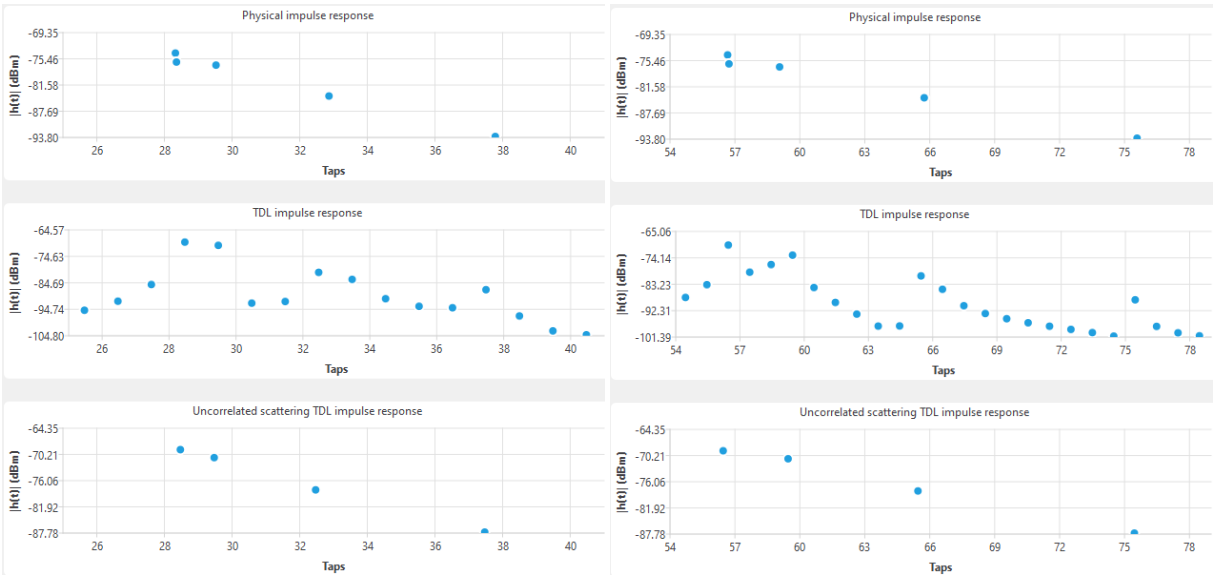
The first point is the closer one as shown in Figure 11 and global parameters for this point are on the Table 4. As we are close to the BS, the delay spread is high so that we can expect only to be in the narrowband scenario for low bandwidth. The impulse responses in Figure 14 confirms that the ray fall in the same taps only for $B = 1\,\text{MHz}$. We can also see that the uncorrelated scattering TDL impulse response has the same form for $B \geq 50\,\text{MHz}$.



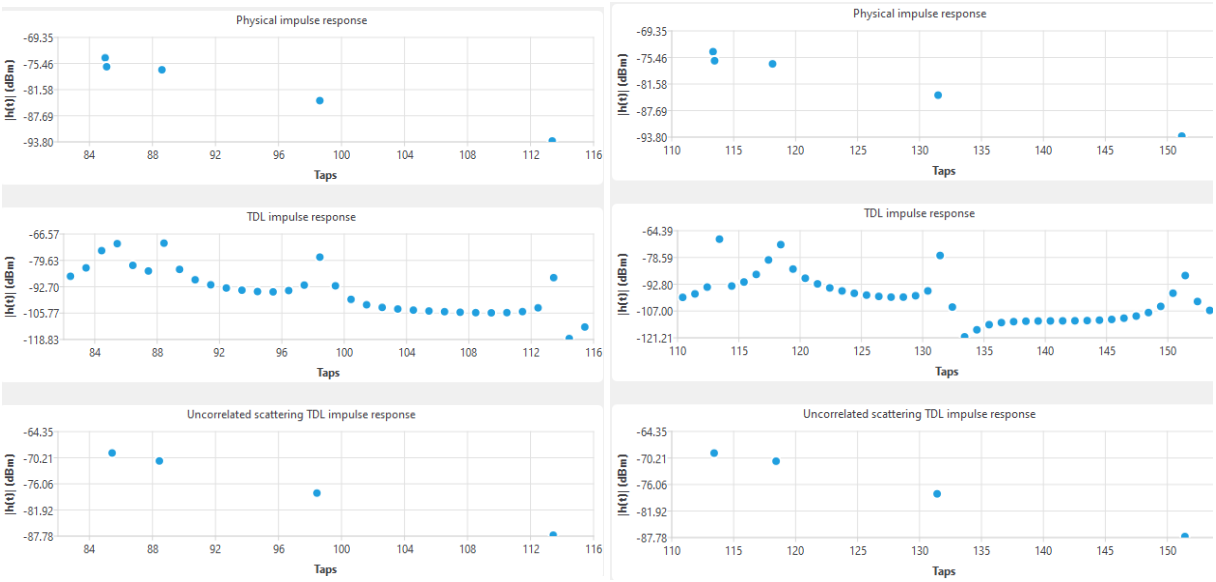
(a) Bandwidth = $1MHz$



(b) Bandwidth = $10MHz$



(a) Bandwidth = $50MHz$ (b) Bandwidth = $100MHz$



(a) Bandwidth = $150MHz$ (b) Bandwidth = $200MHz$

Figure 14: Intersection 1 - Impulse responses

Intersection 2

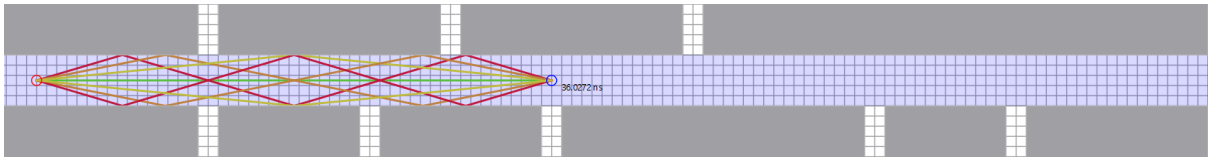
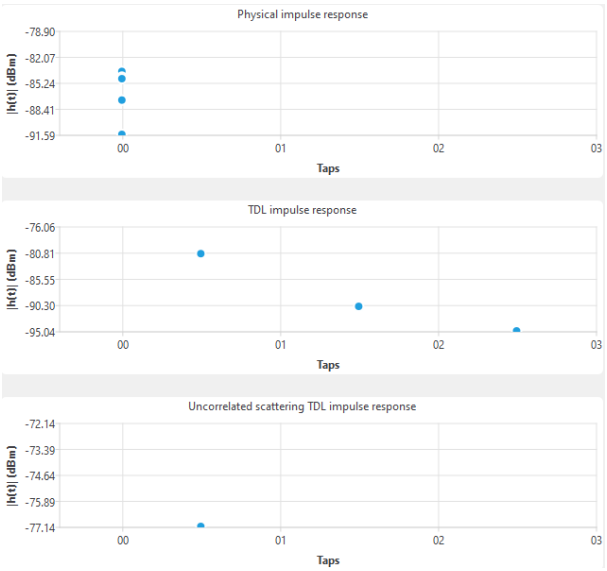


Figure 15: Intersection 2 - geometry

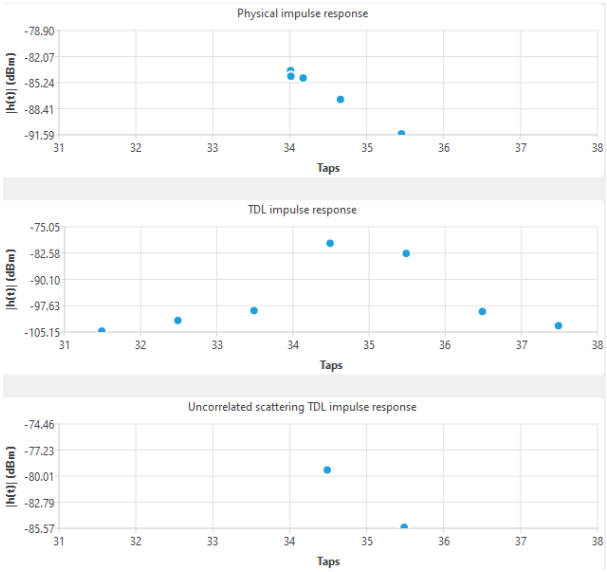
$\sigma_\tau = 36.0ns$	$SNR = 3.8dB$	$P_{rx} = -77.1dBm$	$K = -5.7dB$	$\Delta f_c = 27.8MHz$

Table 3: Intersection 2 - parameters

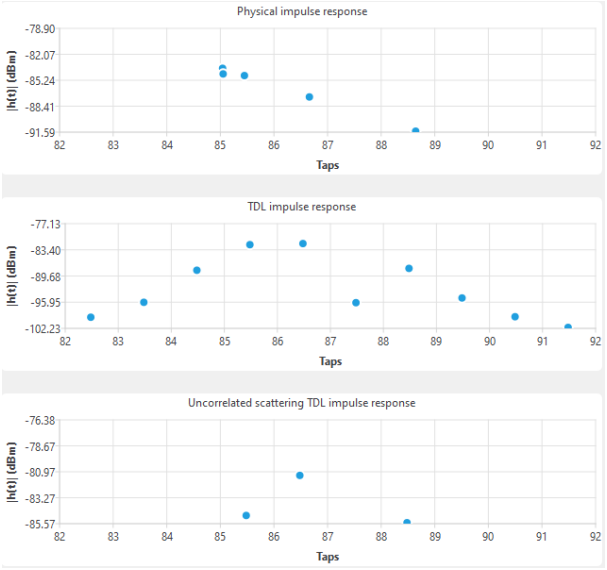
Similarly, the intersection point is shown in Figure 15 and the parameters are given in Table 3. In Figure 18, we observe that for 20 MHz, the last ray arrives just after the first tap. This means that narrowband limit is a bit lower than that, as expected from the coherent bandwidth.



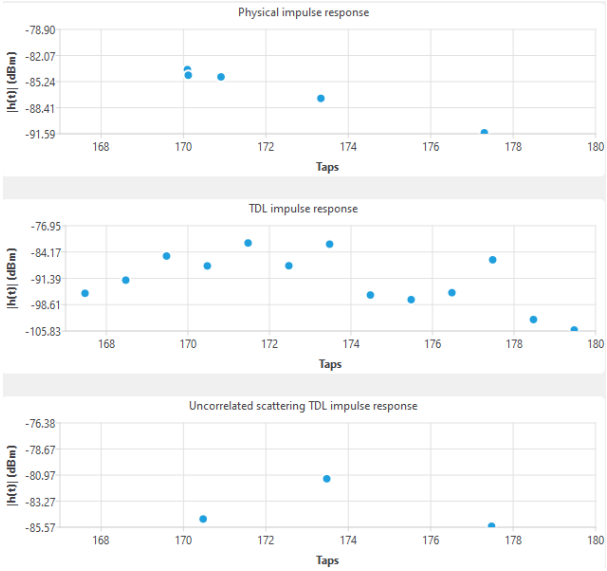
(a) Bandwidth = $1MHz$



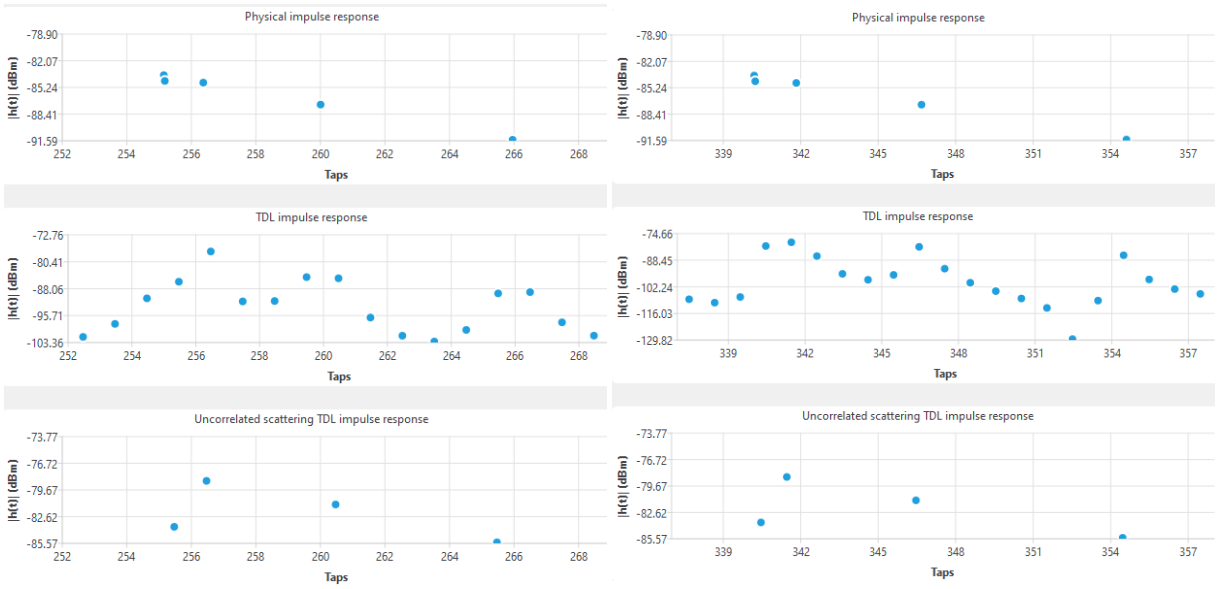
(b) Bandwidth = $20MHz$



(a) Bandwidth = $50MHz$



(b) Bandwidth = $100MHz$



(a) Bandwidth = $150MHz$ (b) Bandwidth = $200MHz$

Figure 18: Intersection 2 - Impulse responses

Intersection 3

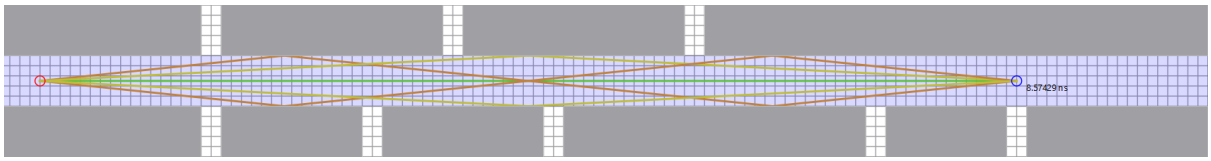
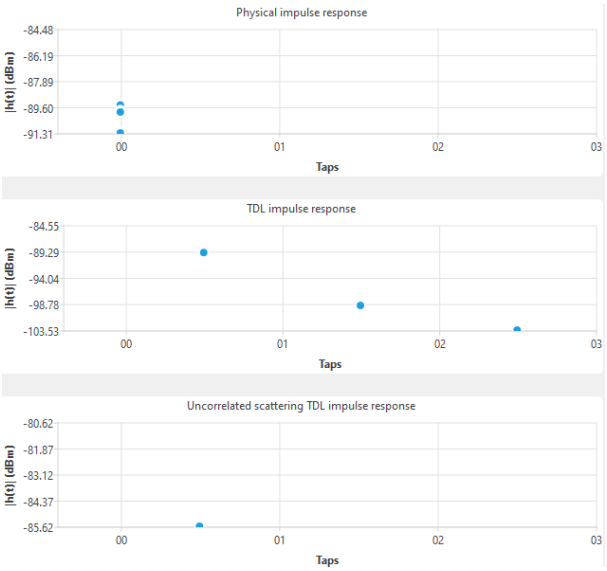


Figure 19: Intersection 3 - geometry

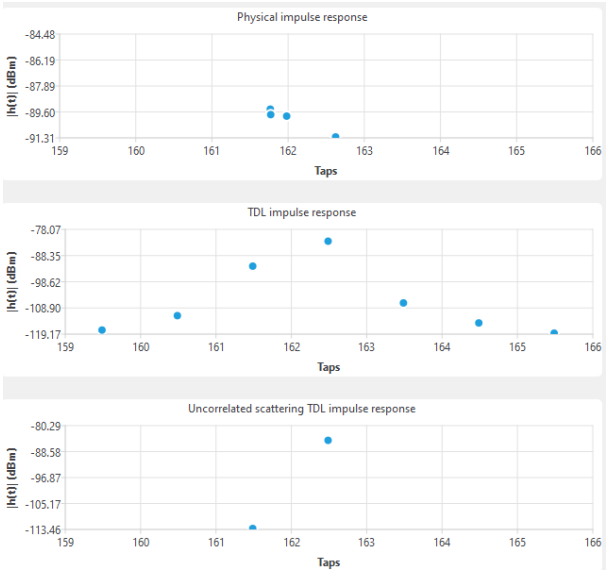
$\sigma_\tau = 8.6ns$	$SNR = -4.7dB$	$P_{rx} = -58.6dBm$	$K = -6.1dB$	$\Delta f_c = 116.3MHz$

Table 4: Intersection 1 - parameters

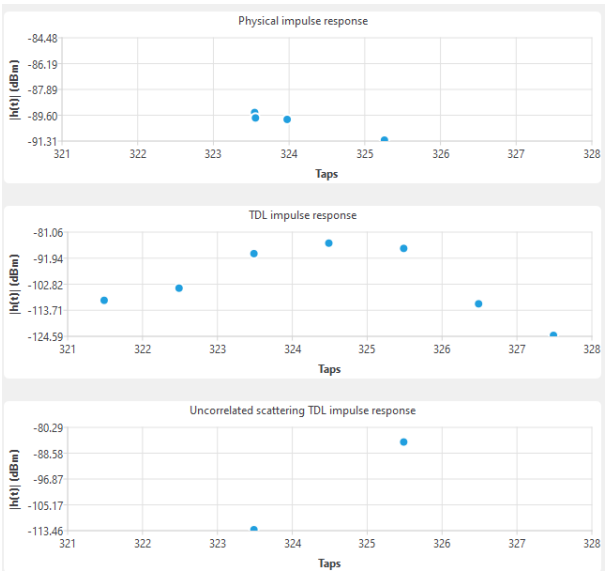
This intersection being the further away, the delay spread is small (due to distance being way higher than the width of the street and to the fact than we only consider 2 reflections), so the narrowband should be extended to higher bandwidth. We can indeed see in Figure 22 that for $B = 50MHz$, the rays arrive almost on the same tap. Even for $B = 200MHz$, there is still only one ray arriving on another tap, the USTDL is similar from $B = 50MHZ$ to $B = 200MHz$.



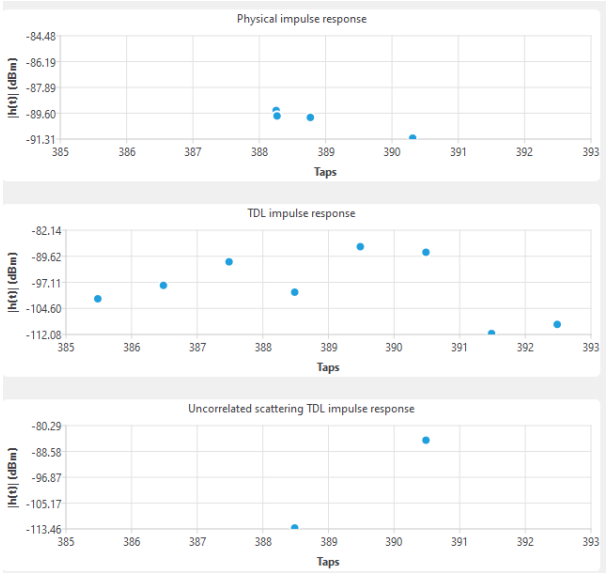
(a) Bandwidth = $1MHz$



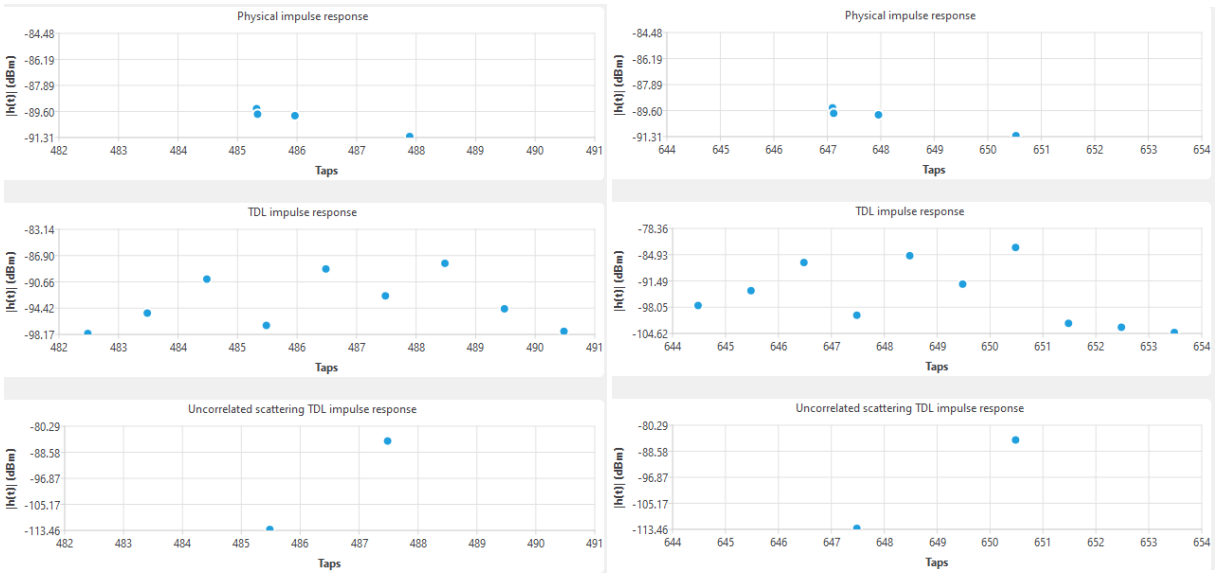
(b) Bandwidth = $50MHz$



(a) Bandwidth = $100MHz$



(b) Bandwidth = $120MHz$



(a) Bandwidth = $150MHz$ (b) Bandwidth = $200MHz$

Figure 22: Intersection 3 - Impulse responses

4 Propagation model

We can analyse the data from Figure 5 to deduce a model for the propagation of the power.

4.1 Path loss model

The path loss model can be expressed as (in decibel scale):

$$L_0(d) = L_0(d_0) - 10n \log \frac{d}{d_0}$$

with n the path loss exponent to determine. We can use the data from Figure 5 for this, because the slope corresponds to the $10n$ term of the expression above. The "simple linear regression" method has been used to compute this exponent (the code is in the Appendix A). The linear approximation is displayed on red in Figure 23 for the reference $L_o(d_o) = 0$ at $d_0 = 10$. The value of n is about 1.49.

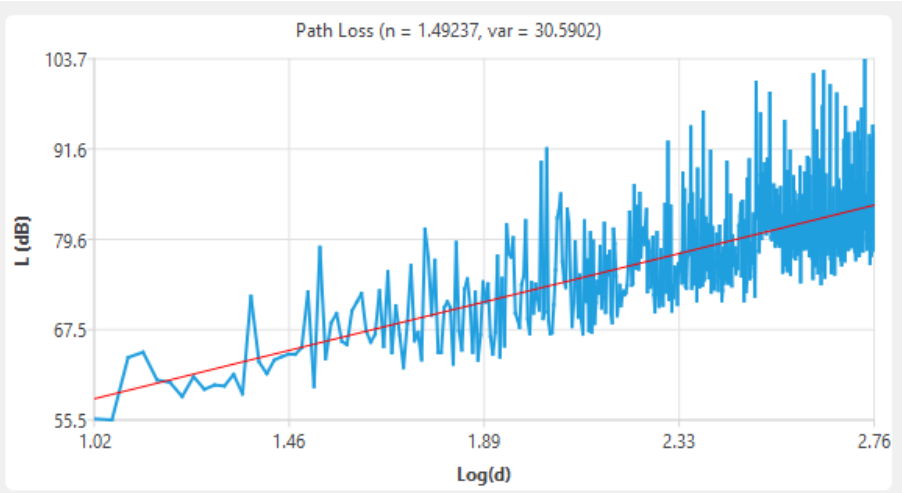


Figure 23: 1D plot - Path loss

$$L_0(d) = 58 + 14.9 \log \frac{d}{10}$$

4.2 Fading variability

The fading variability can also be computed as the variance of the power P_i around the mean $\overline{P_i}$ (the linear approximation):

$$\sigma_L^2 = \frac{1}{n}\sum^{n}(P_i - \overline{P_i})^2$$

That gives us $\sigma^2 = 14.86\,\text{dB}$. The propagation model for power is therefore given by the mean (from the linear approximation) and a random normal distribution L_{σ_L} with zero mean and variance σ_L:

$$L(d) =< L(d) > +l_{\sigma_L}$$

with $< L(d) >$ being the path loss L_0 determined above.

4.3 Cell range

As the path loss is know considered as a random variable due to fading, we have to define a fade margin such that, if we respect this margin, we know that the probability of the communication is high enough. The probability of having an increase of the average path loss bigger than γ is given following the complementary error function *erfc*:

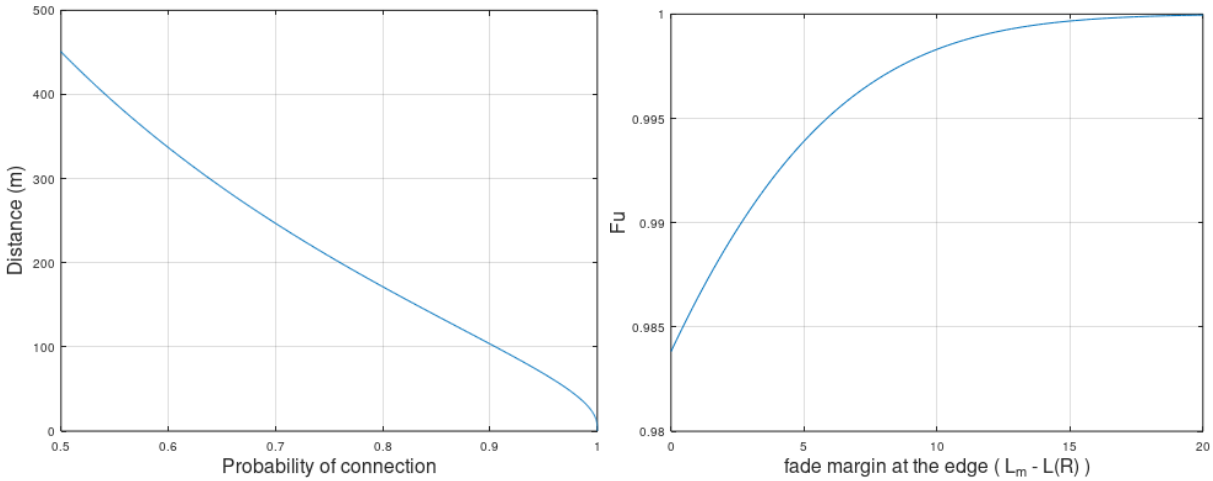
$$Pr\left[L_{\sigma_L} > \gamma\right] = \frac{1}{2} erfc\left(\frac{\gamma}{\sigma_L\sqrt{2}}\right)$$

Therefore we can deduce the probability of connection for a given distance d using SNR and SNR_t, which can be visualised in Figure 24 (a).

$$\begin{aligned}
Pr\left[SNR(d) \geq SNR_t\right] &= Pr\left[P_{rx} - 30 - F_{dB} - 10log_{10}(kTB) \geq SNR_t\right] \\
&= Pr\left[-L_{\sigma_L} \geq SNR_t + 30 + F_{dB} + 10log_{10}(kTB) - P_{tx} + < L(d) >\right] \\
&= 1 - Pr\left[L_{\sigma_L} > P_{tx} - SNR_t - 30 - F_{dB} - 10log_{10}(kTB) - L_0(d_0) - 10nlog_{10}(\frac{d}{d_0})\right] \\
&= 1 - \frac{1}{2} erfc\left(\frac{24.64 - 14.9\log\frac{d}{10}}{10.5}\right)
\end{aligned}$$

From that we can determine the probability of coverage as a function of the fade margins $L_m - L(R)$ at the edge of the cell as shown in Figure 24 (b).

$$\begin{cases}
F_u &= 1 - \frac{1}{2}\,\text{erfc}(a) + \frac{1}{2}e^{2a/b+1/b^2}\,\text{erfc}\left(a + \frac{1}{b}\right) \\
a &= \frac{L_m - L(R)}{\sqrt{2}\sigma_L} \\
b &= \frac{1}{\sqrt{2}\sigma_L} 10n\log e
\end{cases}$$



(a) at the edge of the cell

(b) for the whole cell area

Figure 24: Probability of connection

4.4 Penetration depth of coverage

We can see on the heat map of the SNR (Figure 4) that, as soon as we are in a secondary street and that there is no line of sight, the SNR will be way too low to be able to have a connection. The SNR will therefore be close to the main street one if there is line of sight (the very beginning of the secondary street) and too low to have any connection if there is no line of sight (diffraction is not enough).

Conclusion

The model implemented for this project gives us a good approximation of the behaviour of the communication channel of a 5G small cell in a realistic configuration such as Rue de la Loi. The need of using multiple base stations for crossing streets due to the low power contribution of diffracted waves has been illustrated. Depending on the required probability of connection, the distance between the cells in order to cover an entire street has also been computed. Even if the possibility to add multiple base stations has been implemented into the simulator, the challenge could not be taken up due to an excessive computation time. A better optimisation, and therefore more time should have been necessary to determine the best configuration possible in order to find the highest probability of connection for the smallest number of cells.

A Appendix : Codes used for physical computations

Direct ray and ground reflection

```
//Find and compute the direct and the ground reflected ray
QLineF directLine = QLineF(*(transmitter),*(receiver));
if(!lineIsBlocked(&directLine)){
  std::complex<qreal> E_direct = std::polar(sqrt(60*Ptx*GtxMax)/(directLine.length()/ *(this->px_per_m)),-beta*(directLine.length()/ *(this->px_per_m)));
  std::complex<qreal> T_direct = E_direct*heMax;
  qreal d_ground = sqrt(pow(directLine.length()/ *(this->px_per_m),2)+pow(2*h,2));
  qreal theta_ground = pi/2 - atan((2*h)/directLine.length() * *(this->px_per_m));
  qreal coef = coefReflGround(theta_ground);
  std::complex<qreal> E_ground = std::polar(coef*sqrt(60*Ptx*Gtx(pi - theta_ground))/(d_ground),-beta*(d_ground));
  std::complex<qreal> T_ground = E_ground * he(pi - theta_ground) * sin(pi - theta_ground);
  this->tension += T_direct + T_ground;
  this->los_tension_mod += norm(T_direct);
  this->nlos_tension_mod += norm(T_ground);
  this->delayCheck(1e9 * directLine.length()/ *(this->px_per_m));
  this->delayCheck(1e9 * d_ground);
  this->rayData.append(QPair<qreal,std::complex<qreal>>(1e9 *directLine.length()/ *(this->px_per_m)/c,T_direct));
  this->rayData.append(QPair<qreal,std::complex<qreal>>(1e9 *d_ground/c,T_ground));
  Ray* directRay = new Ray(directLine);
  directRay->coef*=1;
  directRay->setPen(rayPen);
  this->raysList.push_back(directRay);
}
```

Reflection on buildings

```
//Recursive function to find and compute the rays reflected on building. Recursion depth = number of reflections
for(Building* building:*this->building_list){
  for(QLineF wall:*(building->getWalls())){
    if((walls.isEmpty() || wall!=*(walls.last()))){
      if(wallIsValid(wall)){
        //Compute the new mirror pt based on the last one (tx pos for the 1st reflection)
        QList<QLineF*> tempWalls = walls;
        tempWalls.push_back(&wall);
        QList<QPointF> tempMirrorPoints = mirrorPoints;
        QPointF mirrorPoint;
        if(tempMirrorPoints.isEmpty()){
          mirrorPoint = mirrorPointMaker(&wall, transmitter);
        }
        else{
          mirrorPoint = mirrorPointMaker(&wall, &(tempMirrorPoints.last()));
        }
        tempMirrorPoints.push_back(mirrorPoint);
        //Compute the n_reflection for the current list of mirrorPoints and walls
        QList<Ray*> rays = QList<Ray*>();
```

```
        qreal coef = 1;
        qreal total_length = 0;
        QPointF lastIntersectionPoint = *(receiver);
        int n_mirror = tempMirrorPoints.size();
        for(qint16 i = 0; i<=n_reflection; i++){
          QLineF lineLIPtoIP;
          if(i != n_reflection){//Ray reflecting on a wall
            QLineF* currentWall = tempWalls.value(tempWalls.size()-(i+1));
            QLineF lineLIPtoMP(lastIntersectionPoint,tempMirrorPoints.value(n_mirror-(i
                +1)));
            QPointF intersectionPoint;
            if(currentWall->intersects(lineLIPtoMP,&intersectionPoint)==QLineF::
                BoundedIntersection && intersectionPoint != lastIntersectionPoint){//Is
                there an intersection on the wall ?
                lineLIPtoIP = QLineF(lastIntersectionPoint,intersectionPoint);
              if(!lineIsBlocked(&lineLIPtoIP)){//Is there no other wall blocking the ray
                  ?
                Ray* ray = new Ray(lineLIPtoIP);
                qreal angleD = lineLIPtoIP.angleTo(*currentWall);
                qreal incidenceAngle = (pi/180)*angleD;
                coef *= coefReflWall(incidenceAngle);
                total_length += lineLIPtoIP.length();
                rays.push_back(ray);
                lastIntersectionPoint = intersectionPoint;
              }else{break;}
            }else{break;}
          }
          else{//Last ray, going from last mirror point to TX
            lineLIPtoIP = QLineF(lastIntersectionPoint,*(transmitter));
            if(!lineIsBlocked(&lineLIPtoIP)){//Is there no other wall blocking the ray ?
              Ray* ray = new Ray(lineLIPtoIP);
              total_length += lineLIPtoIP.length();
              rays.push_back(ray);
            }else{break;}
        }}
        if(rays.size() == n_reflection +1){//Valide ray path (no rays intersects the
            walls)
          std::complex<qreal> E_refl = std::polar(coef*sqrt(60* GtxMax * Ptx )/ (
              total_length/ *(this->px_per_m)),(-this->beta*(total_length/(*(this->
              px_per_m)))));
          std::complex<qreal> T_refl = E_refl * heMax;
          this->tension += T_refl;
          this->nlos_tension_mod += norm(T_refl);
          this->delayCheck(1e9 * total_length/(*(this->px_per_m)));
          this->rayData.append(QPair<qreal,std::complex<qreal>>(1e9 *total_length/(*(
              this->px_per_m))/c,T_refl));
          for(Ray* ray:rays){
            ray->setPen(rayPen);
            this->raysList.push_back(ray);
        }}
        //More reflections iterations
        if(n_reflection < maxReflection){
          makeWallReflection(tempMirrorPoints,tempWalls,n_reflection+1);//Call itself
              for more reflections
}}}}}
```

Diffraction

```
//Find and compute diffracted rays
for(Building* building:*building_list){
  for(QPointF corner:*(building->getCorners())){
    if(cornerIsValid(corner)){
      QLineF lineTXtoEP(*(transmitter),corner);
      QLineF lineEPtoRX(corner,*(receiver));
      QPointF point1 = makeNormalPoint(lineTXtoEP);
      QPointF point2 = makeNormalPoint(lineEPtoRX);
      QPointF checkPoint1 = corner+2*point1;
      QPointF checkPoint2 = corner+2*point2;
      qreal angle = lineTXtoEP.angleTo(lineEPtoRX);
      bool diff = false;
      if(checkPoint1 == checkPoint2 || (checkPoint1.y() == checkPoint2.y() && (
          checkPoint1.x() == corner.x() || checkPoint2.x() == corner.x())) || (
          checkPoint1.x() == checkPoint2.x() && (checkPoint1.y() == corner.y() ||
          checkPoint2.y() == corner.y()))){
        if(building->rect().toRect().contains(checkPoint1.toPoint())){if(angle<=90){diff
            = true;}}
        else {if(angle>=270 || angle == 0){diff = true;}}}
      if(diff && !lineIsBlocked(&lineTXtoEP) && !lineIsBlocked(&lineEPtoRX)){
        Ray* rayTXtoDP = new Ray(lineTXtoEP);
        Ray* rayDPtoRX = new Ray(lineEPtoRX);
        QLineF direct = QLineF(*(this->transmitter),*(this->receiver));
        qreal startAngle = direct.angleTo(lineTXtoEP)*(pi/180);
        qreal h = lineTXtoEP.length()/ *(this->px_per_m) * abs(sin(startAngle));
        qreal Dr = (pow(h,2)/2) * ( 1/(lineTXtoEP.length()/ *(this->px_per_m)) + 1/(
            lineEPtoRX.length()/ *(this->px_per_m)) );
        std::complex<qreal> F = coefDiff(Dr);
        qreal total_length = (lineTXtoEP.length() + lineEPtoRX.length())/ *(this->
            px_per_m);
        std::complex<qreal> E_diff = std::polar(abs(F)*sqrt(60* GtxMax * Ptx )/
            total_length, (-this->beta*total_length) + arg(F));
        std::complex<qreal> T_diff = E_diff * heMax;
        this->tension += T_diff;
        this->nlos_tension_mod += norm(T_diff);
        this->delayCheck(1e9 * total_length);
        this->rayData.append(QPair<qreal,std::complex<qreal>>(1e9 *total_length/c,T_diff
            ));
        rayTXtoDP->setPen(rayPen);
        rayDPtoRX->setPen(rayPen);
        this->raysList.push_back(rayTXtoDP);
        this->raysList.push_back(rayDPtoRX);
}}}}
```

Power and SNR

```
//Compute the power from the total induiced tension
this->received_power = (1/(8*Ra)) * pow(abs(this->tension),2);
this->received_power_dbm = 10*log10(this->received_power/0.01);
qreal power_dbw = this->received_power_dbm - 30;
qreal SNR = power_dbw - noise_figure - 10*log10(boltzman*BW*temp);
```

Delay spread and rice factor

```
 //Compute the rice factor (if LOS)
if(this->los_tension_mod != 0){
  this->rice_factor = 10*log10(this->los_tension_mod / this->nlos_tension_mod);
}
//Compute the delay spread (if at least 1 ray)
if(this->delay_min != INFINITE && this->delay_max > 0){
  this->delay_spread = this->delay_max - this->delay_min;}
```

Physical impulse response

```
//rayData list contains induce tension and arrival time for each rays
for(QPair<qreal,std::complex<qreal>> data :this->rayData){
  qreal taps = data.first/this->deltaT;
  qreal pwr = 10*log10((1/(8*Ra)) * pow(abs(data.second),2)/0.01);
  seriePhy->append(QPointF(taps,pwr));}
```

TDL impulse response

```
for(int i= round(minDelay/this->deltaT)-3;i<round(maxDelay/this->deltaT)+2;i++){
  qreal taps = i+0.5;
  std::complex<qreal> hl = 0;
  for(QPair<qreal,std::complex<qreal>> data :this->rayData){
    hl += (data.second*this->sinc(2*this->BW*(data.first - (i+0.5)*this->deltaT)));
  }
  qreal pwr = 10*log10((1/(8*Ra)) * pow(abs(hl),2)/0.01);
  serieTDL->append(QPointF(taps,pwr));}
```

USTDL impulse response

```
for(int i= round(minDelay/this->deltaT)-3;i<round(maxDelay/this->deltaT)+2;i++){
  std::complex<qreal> hl = 0;
  qreal taps = i+0.5;
  qreal pwr = 0;
  for(QPair<qreal,std::complex<qreal>> data :this->rayData){//TODO copy list and remove
      elem
    if(data.first >= (taps-0.5)*this->deltaT && data.first < (taps+0.5)*this->deltaT){
      hl += data.second;
    }
  }
  if(abs(hl) != 0){
    pwr = 10*log10((1/(8*Ra)) * pow(abs(hl),2)/0.01);
    serieUSTDL->append(QPointF(taps,pwr));}}
```

Path loss exponent

```
//Make the linear regression
const auto n    = x.size();
const auto s_x  = std::accumulate(x.begin(), x.end(), 0.0);
const auto s_y  = std::accumulate(y.begin(), y.end(), 0.0);
const auto s_xx = std::inner_product(x.begin(), x.end(), x.begin(), 0.0);
```

```
const auto s_xy = std::inner_product(x.begin(), x.end(), y.begin(), 0.0);
const auto a    = (n * s_xy - s_x * s_y) / (n * s_xx - s_x * s_x);
const auto b    = (s_y - a * s_x) / n;
return QPair<qreal,qreal>(a,b);
```

Fading variability

```
//a is the slope of the linear regression and b the initial value at log(d) = 1
qreal var = 0;
qreal mean = 0;
qreal diff = 0;
for(int i=0;i < x.size();i++){
  mean = b+x.value(i)*a;
  diff = mean - y.value(i);
  var += pow( diff ,2);
}
var /= x.size();
return var;
```

Connection probability

This codes is from matlab and not from Qt/C++.

```
%% Parameters
SNRt = 2;
FdB = 10;
k = 1.38*10^(-23);
T = 293.15;
B = 100*10^(6);
L_0 = 58;
n = 1.49;
d_0 = 10;
Ptx = 10*log10(3*pi/8);
sigma = 10*log10(sqrt(30.59));

%% Probability at edge
f = @(x) (1 - (1/2)*erfc((Ptx - L_0 - SNRt - 30 - FdB - 10*log10(k*T*B) - n.*10.*log10(x
    ./10))./(sigma*sqrt(2))));

x = [0.001:500];
figure(1);
plot(f(x),x);grid on;
xlabel("Probability of connection","fontsize", 16);xlim([0.5,1]);
ylabel("Distance (m)","fontsize", 16);ylim([0,500]);

%% Probability for whole cell
y = [0:0.2:20]
a = y./(sqrt(2)*sigma);
b = 10*n*log10(e)/(sqrt(2)/sigma);
Fu = @(a) (1 - (1/2).*erfc(a) + (1/2).*exp(2.*a./b + 1/(b^2)) .* erfc(a + 1/b));

figure(2);
plot(y,Fu(a));grid on;
xlabel("fade margin at the edge ( L_m - L(R) )","fontsize", 16);
ylabel("Fu","fontsize", 16);
```

B Appendix : Simulation validation

Direct ray and reflection on the ground:

To verify our simulation, we can take a basic case where the distance between BS and UE is 50m (and their height is 2m), which gives $d_d = 50$, $d_g = \sqrt{50^2 + 4^2} \approx 50.16$ and $\theta_{i,g} = 85.42$.

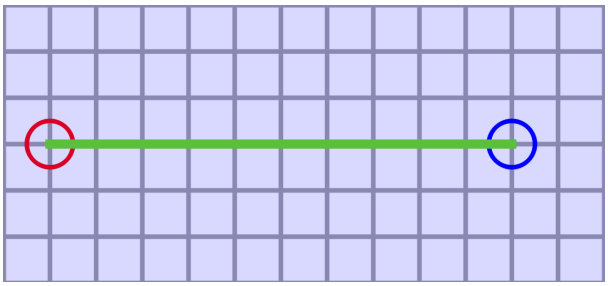


Figure 25: Simple case 1 - LOS and reflection on the ground

The electric field arriving at the receiver is directly given by the electric field of a plane wave (7), with the maximal gain (6). The induced voltage is obtained using also the max effective height (4).

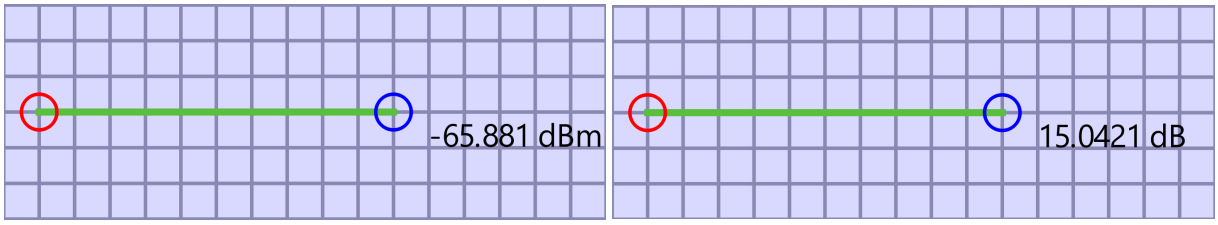
$$\begin{aligned}\underline{V}_{oc,d} &= h_{e,max}\,\underline{E}_d \\ &= -\frac{\lambda}{\pi}\sqrt{60\,G_{tx,max}P_{tx}}\frac{e^{-j\beta d_d}}{d_d} \\ &= -7.7433\,10^{-4}\,e^{-j0.2306\,\pi}\end{aligned}$$

For the reflection on the ground, the reflection is parallel. The angle of incidence of the wave (at the base station and the user endpoint) is not 90° anymore, that means that the effective height ad the gain have to be computed by (3) and (5) respectively. The total distance is determined with $\sqrt{d_{direct}^2 + (2\,h)^2}$ where h is the antenna height (considered as the same for both base station and user endpoint). The angle of incidence for the reflection is $\theta_{i,g} = \frac{\pi}{2} - atan(\frac{2h}{d_d})$ and the corresponding one at the antenna $\theta_{i,rx} = \theta_{i,tx} = \pi - \theta_{i,g}$. From that we can find the reflection coefficient $\Gamma_{\parallel}(\theta_{i,g}) = -0.6678$. The angle of incidence at the UE also means that the product of the effective height with the gain has to be done with the cosine of the angle $\frac{\pi}{2} - \theta_{i,rx}$. The induced voltage is therefore:

$$
\begin{aligned}
\underline{V}_{oc,g} &= \vec{h}_e(\theta_{i,rx})\,\vec{\underline{E}}_g \\
&= h_e(\theta_{i,rx})\,\Gamma_{\parallel}(\theta_{i,g})\,\sqrt{60\,G_{tx}(\theta_{i,rx})P_{tx}}\,cos(\frac{\pi}{2}-\theta_{i,rx})\,\frac{e^{-j\beta d_g}}{d_g} \\
&= 5.1057\,10^{-4}\,e^{-j1.0045\,\pi}
\end{aligned}
$$

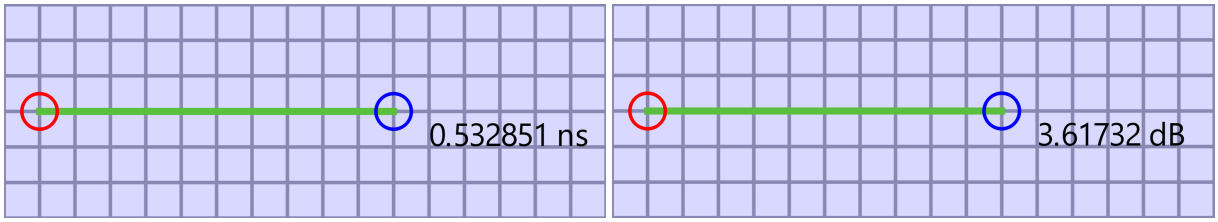
By adding these two voltages we can compute the maximal power received and compare it to the simulation results.[2]

$$
\begin{cases}
\underline{V}_{oc,tot} &= 12.08\,10^{-4}\,e^{-j0.1417\,\pi} \\
\underline{P}_{RX} &= 2.57\,10^{-9} = -65.9
\end{cases}
$$



(a) Power (b) SNR

Figure 26: Simple case 1 - power and SNR



(a) Delay spread (b) Rice factor

Figure 27: Simple case 1 - delay spread and rice factor

Reflection on buildings:

We can take the last case and add some building to have reflections.

[2] As the product βd for the phase computation gives high number of full rotation (around 4500 rotations for $d = 50m$), this mean that the phase will be really sensitive to any rounding approximation (the difference $d_g - d_d \approx 0.1597m$, but still result to ≈ 14 rotations).

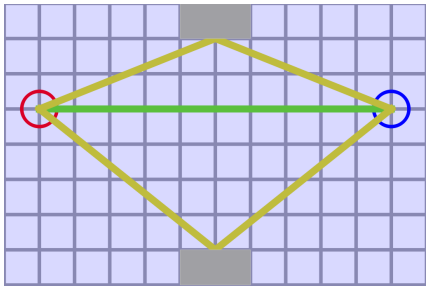


Figure 28: Simple case 2 - Reflection on building

Similarly to the reflection on the ground, the reflection on a building takes a coefficient, but the parallel one. However, this time the incidence angle at BS and UE is 90° so we can use $h_{e,max}$ and $G_{tx,max}$ and the cosine in product between the effective height and the gain is simply 1. The induced voltage for the reflection "below" is given by:

$$\begin{cases} \theta_{i,b1} &= 51.34 \\ d_{b1} &= 64.03m \\ \Gamma_{\parallel}(\theta_{i,b1}) &= -0.5407 \\ \underline{V}_{oc,b1} &= h_{e,max}\,\underline{E}_{b1} \\ &= -\frac{\lambda}{\pi}\,\Gamma_{\perp}(\theta_{i,b1})\,\sqrt{60\,G_{tx,max}P_{tx}}\,\frac{e^{-j\beta d_{b1}}}{d_{b1}} \\ &= 3.2693\,10^{-4}\,e^{j\,0.3914\pi} \end{cases}$$

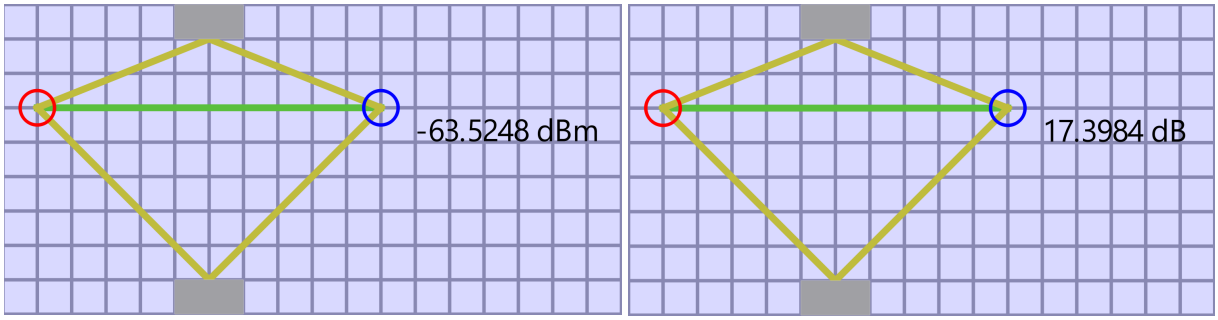
And for the reflection "above":

$$\begin{cases} \theta_{i,b2} &= 68.20 \\ d_{b2} &= 53.85m \\ \Gamma_{\parallel}(\theta_{i,b2}) &= -0.6912 \\ \underline{V}_{oc,b2} &= h_{e,max}\,\underline{E}_{b2} \\ &= -\frac{\lambda}{\pi}\,\Gamma_{\perp}(\theta_{i,b2})\,\sqrt{60\,G_{tx,max}P_{tx}}\,\frac{e^{-j\beta d_{b2}}}{d_{b2}} \\ &= 4.9695\,10^{-4}\,e^{j\,1.983\pi} \end{cases}$$

When added to the direct and ground reflected ray, the total power receiver becomes:

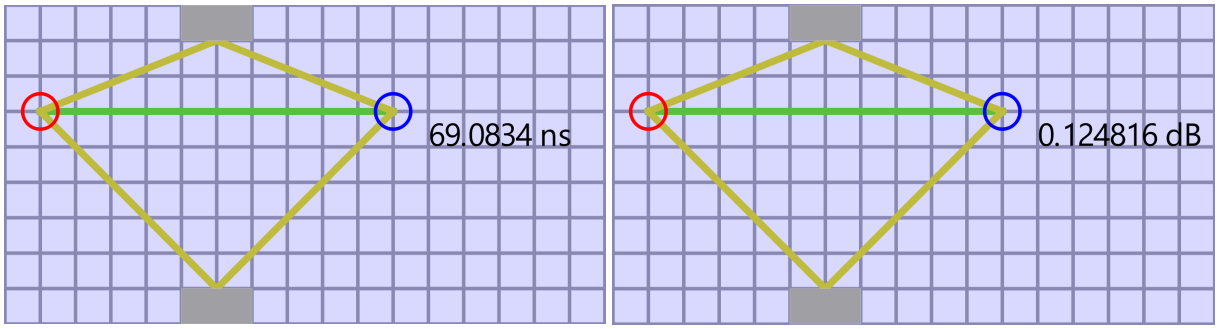
$$\begin{cases} \underline{V}_{oc,tot} &= 9.3688\,10^{-4}\,e^{-j\,0.327\pi} \\ \underline{P}_{RX} &= 1.55216\,10^{-9} = -68.1 \end{cases}$$

which corresponds to the result of the simulation as shown in Figure 29.



(a) Power (b) SNR

Figure 29: Simple case 2 - power and SNR



(a) Delay spread (b) Rice factor

Figure 30: Simple case 2 - delay spread and rice factor

Diffraction

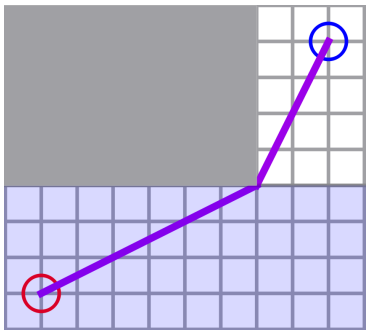
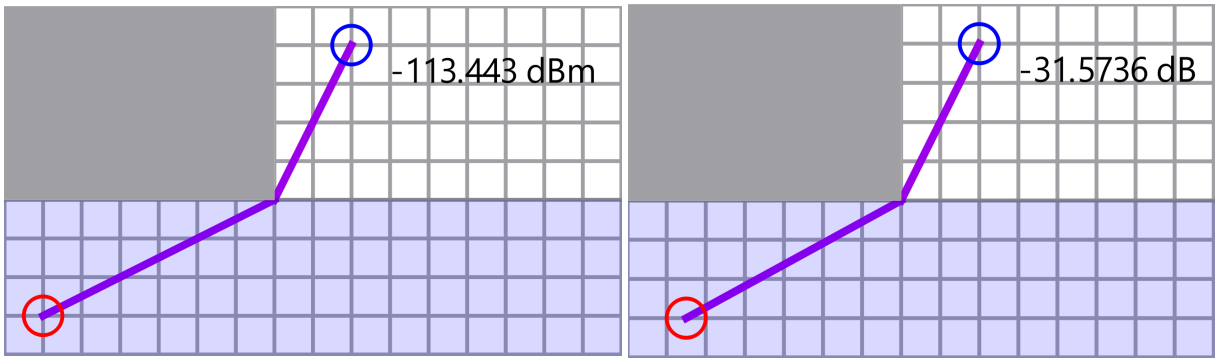


Figure 31: Simple case 3 - Diffraction

For the diffraction, the reflection is not considered and we have therefore only one ray. We have to find the Fresnel parameter ν which only depends on the geometry of this case:

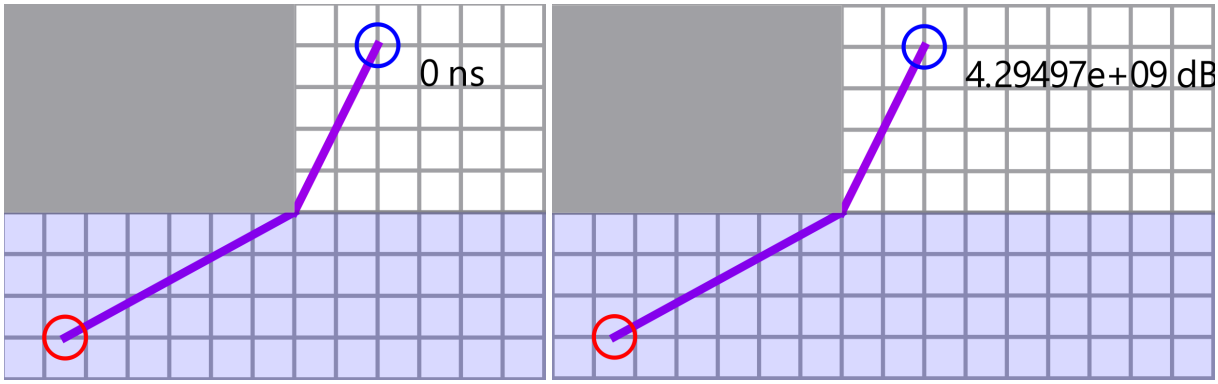
$$\begin{cases} h &= = 8.46m \\ \Delta r &= 2.67m \\ \nu &= 31.01 \\ |F(\nu)|^2[\text{ dB}] &= -42.72 \\ \text{Arg}\, F(\nu) &= -481.18\pi\ rad \\ \underline{V}_{oc,diff} &= -5.05\,10^{-6}\, e^{j\,0.2916\pi} \\ \underline{P}_{RX} &= 4.525\,10^{-14} \\ &= -113.45 \end{cases}$$

The simulation gives the same result.



(a) Power (b) SNR

Figure 32: Simple case 3 - power and SNR



(a) Delay spread (b) Rice factor

Figure 33: Simple case 3 - delay spread and rice factor