Simplified Path Loss & Doppler Laboratory Session

WIRELESS COMMUNICATIONS 371-1-1903

Part 1 – Simplified Path Loss Simulation

Simplified Path Loss Model

The complexity of signal propagation makes it challenging to obtain a single model that accurately characterizes path loss across various environments. Accurate path loss models can be obtained from complex ray-tracing models or empirical measurements when tight system specifications must be met, or the best locations for base stations or access point layouts must be determined. However, for general tradeoff analysis of various system designs, it is sometimes best to use a simple model that captures the essence of signal propagation without resorting to complicated path loss models, which are only approximations of the real channel anyway. Thus, the following simplified model for path loss as a function of distance is commonly used for system design.

$$\frac{P_r}{P_t} = K\left[\frac{d_o}{d}\right]^{\gamma}$$

In this approximation $K[dB] = 20\log_{10}\frac{\lambda}{4\pi d_0}$, is a unitless constant which depends on the antenna characteristics and the average channel attenuation, d_0 is a reference distance for the antenna far-field and is the path loss exponent. Due to scattering phenomena in the antenna near-field, the model (shown above) is generally only valid at transmission distances $d > d_0$, where d_0 is typically assumed to be 1-10 m indoors and 10 -100 m outdoors.

The value of γ depends on the propagation environment: for propagation that approximately follows a free-space or two-ray model is set to 2 or 4, respectively. The value for more complex environments can be obtained via a minimum mean square error (MMSE) fit to empirical measurements.

Environment	γ range
Urban macrocells	3.7-6.5
Urban microcells	2.7-3.5
Office Building (same floor)	1.6-3.5
Office Building (multiple floors)	2-6
Store	1.8-2.2
Factory	1.6-3.3
Home	3

The table summarizes γ values for different indoor and outdoor environments and antenna heights at 900 MHz and 1.9 GHz.

Theoretical Questions

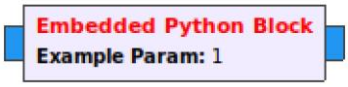
1. Consider the set of empirical measurements of P_r/P_t given in the table below for an indoor system at $2.4\ GHz$. Find the path loss exponent that minimizes the MSE between the simplified model and the empirical dB power measurements, assuming that $d_0 = 1\ \mathrm{m}$ and K is determined from the free space path loss formula at this d_0. Find the received power at $150\ m$ for the simplified path loss model with this path loss exponent and a transmit power of $1\ mW$ $(0\ dBm)$.

Distance from Transmitter	$M = P_r/P_t$
10m	$-65\ dB$
50m	$-75\ dB$
100m	$-95\ dB$
200m	$-105\ dB$
500m	$-135\ dB$

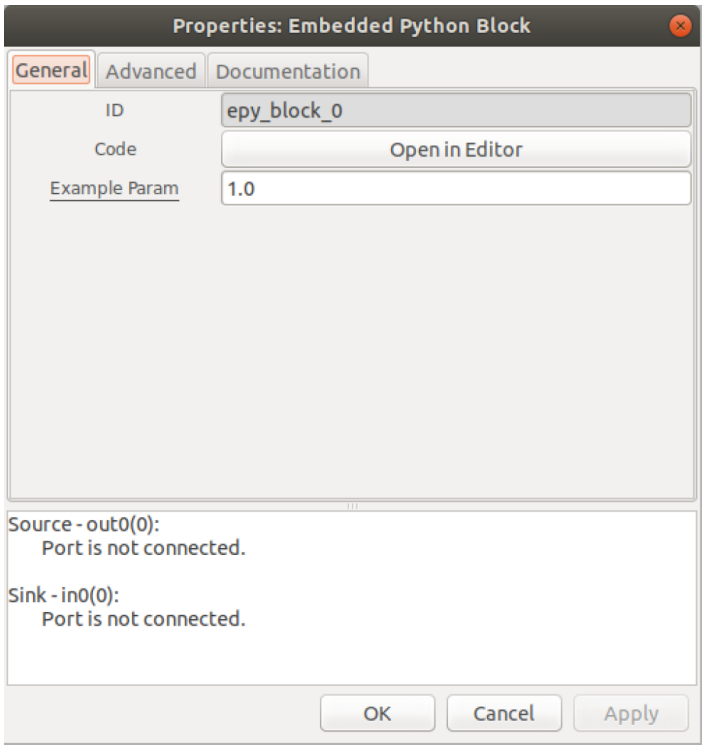
2. Under what conditions is the simplified path loss model the same as the free-space path loss model?

3. Consider a receiver with noise power $-160\ dBm$ within the signal bandwidth of interest. Assume a simplified path loss model with $d_0 = 1\ m$, K obtained from the free space path loss formula, and $\gamma = 4$. For a transmit power of $P_t = 10\ mW$, find the maximum distance between the transmitter and receiver such that the received signal-to-noise power ratio is $20\ dB$.

The Embedded Python Block



The Embedded Python Block allows you to create a new (custom) block in Python without making and installing an Out of Tree Module (OOT). When you add the block to your flowgraph, the pre-populated code simply takes the input stream and multiplies it by a constant. Note that the structure of this Python block matches the structure of an Out of Tree Module (OOT) Python block. It's essentially a Python OOT block built into a grc flowgraph.



When entering the block's properties, you will notice the code (Open in Editor button) and the Example Param.

Let us dive into the code and understand what is going on behind the scenes.

Open the code in an editor of your choice (the default editor is fine). Inside, you will find the following code:

```python
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, example_param=1.0):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Embedded Python Block',   # will show up in GRC
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.example_param = example_param

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.example_param
        return len(output_items[0])
```

Let's explain some things we see there:

```
class blk(gr.sync_block)
```

is a class from `gnuradio`, which defines a synchronous 1:1 block (input to output) with history.

This allows us to process signals that go into this block and output a signal from this block.

```
def __init__(self, example_param=1.0):
```

is a function configuring the block upon initialization, meaning that this function is called only once at the beginning of the code.

You'll notice that besides `self` which refers to this block, we find the `example_param=1.0` we've seen in the block properties. We can change this argument (its name and default value) and add more arguments. We may need it for our new block.

```
name='Embedded Python Block',   # will show up in GRC
in_sig=[np.complex64],
out_sig=[np.complex64]
```

Inside the init function, you define the `name` and the inputs `in_sig` (and its type - `np.complex64`) and outputs `out_sig.`

The possible types can be found here https://numpy.org/devdocs/user/basics.types.html

*Recall the types accepted in GNU Radio.

```
self.example_param = example_param
```

Here we set the `example_param` with the default value of `1.0`

```
def work(self, input_items, output_items):
```

The work function is constantly called during runtime. It actually states the work to be done on each sample that enters the block and leaves it.

```
output_items[0][:] = input_items[0] * self.example_param
```

We can see here that we take the inputted streamed samples (just the first one `[0]`), multiply it by the `example_param,` and deliver it to the output `output_items[0][:].`

For further information regarding GNU Radio functions and more, refer to the GNU Radio documentation: https://www.gnuradio.org/doc/doxygen

In this experiment, we want to create a block that simulates the simplified path loss model for given parameters (K, d_0, d and γ).

1. Set the sample rate to $32Msps$.

2. Add the "Import" component and enter "import numpy" in the Import field.

3. Add another "Import" component and enter "$\text{import scipy.constants}$" in the Import field.

   (If not installed, write in terminal: sudo apt-get install python-scipy)

4. Add "Variable" wavelength and enter the value $\text{scipy.constants.c/frequency}$

5. Add another "QT GUI Range" for the distance – from 10m to 15Km in 10m steps and a default value of 10m.

6. Add "Variable" d0 and enter the value $20m$

7. Add "Variable" K and enter the formula $K[dB] = 20\log_{10}\frac{\lambda}{4\pi d_0}$, if needed convert to non dB (remember that GNU radio accepts python expressions)

8. Create a Python block that takes a signal at the input (P_t) and applies the effect of the channel, which is modeled by the Simplified Path Loss Model.

$$\frac{P_r}{P_t} = K\left[\frac{d_o}{d}\right]^{\gamma}$$

   The result should be outputted as the output signal (P_r).

   - Implement K, d_0, d and γ as parameters (Just like the `example_param=1.0`, you can use it by writing `self.example_param`)

   - The input and outputs:
     `in_sig=[np.complex64],`
     `out_sig=[np.complex64]`
     It does not have to be `np.complex64`, you can change it as you wish (`np.float32,` for example). You can even add inputs or outputs like so: `in_sig=[np.complex64, np.float32].`

9. Add a "Signal source" as you wish, at your chosen frequency, and connect it to your model's input.

10. Add a float "QT GUI Time Sink" and connect it to your model's output (You can add a Control Panel = Yes for better control)

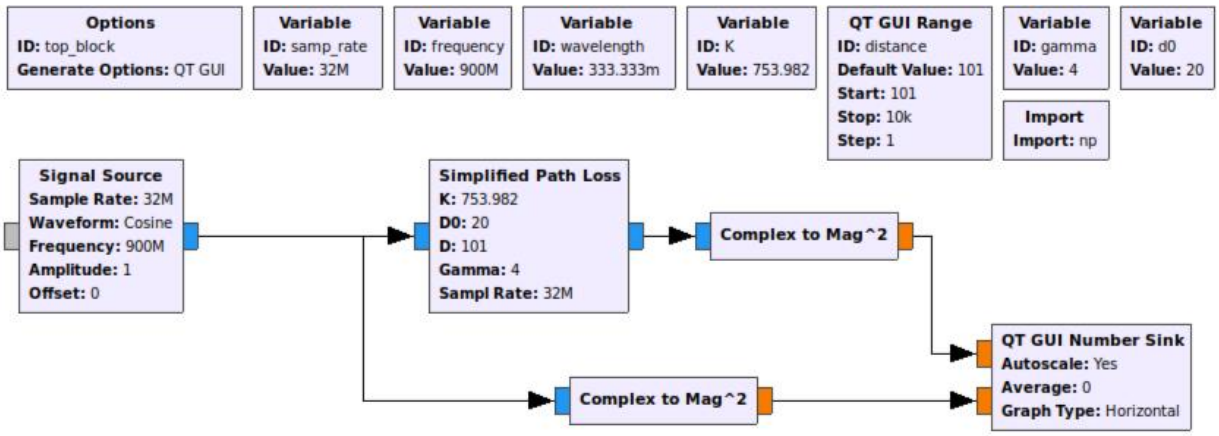
11. An alternative: add a complex to Mag^2 and input it to a “QT GUI Number sink.”

12. Compare the original signal to the one passed through your model.

13. Save and run your code.

Save your code and add it to your submission!

If you performed all the stages correctly, your system’s code should look like this:



*Does not have to be complex

You can change all parameters as you wish to obtain a system that is within working ranges.

Practical Questions

4. Explain the Mag^2 block and why we use it.

5. Play around with d_0 and explain what it is and how it affects the model.

6. Play around with γ and explain what it represents and how it affects the model.

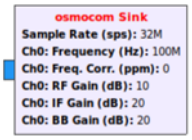
7. Choose 5 distances and measure the signal’s power after the model. Calculate analytically (same as question 1 above) and show that the results match your implementation.

8. Repeat 7 for a different γ and explain the changes in the ability to receive data correctly (better or worse).

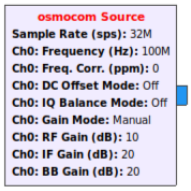
Bonus (20%) – Make real measurements (Empirical model)

In this part, you may leave the lab to investigate a channel of your choice somewhere in the university. You need 2 laptops with GNURadio installed (VM will do), a USB 3 port, and enough charge on your battery. Find a place that is not entirely obstacle free and perform the following steps:

- The transmitter is a CW (continuous wave) signal source block – sine would do fine, connected to the transmitter block.



- The receiver is just a receiver block with some display that would allow the signal to be measured.



- Both sides should be set to the same frequency. Listen to the channel using the receiver side to ensure the channel is clear of interference.

- Set the transmitter to maximal gain and receiver to minimal gain, place them close to each other, and measure the signal (similar to what you did in lab 1). Ensure you are not stronger than the maximal reception power at the receiver (signal must not be clipped). Record the values; they will be your reference power and distance.

- Take the receiver and its laptop on a ride; try to stay in a straight line as you distance away from the transmitter) **[distance traveled in cm is not good enough for this experiment]**. Try to measure the traveled distance by counting your steps and measuring the distance/size of your typical step. (for example, a foot is roughly 30cm; if you take 10 steps heel to tow, you will travel roughly 3 meters).

- As in part 1 questions, fill out a table of measured power and distance from the transmitter. And **obtain γ**. What is your inferred **environment?** Does it match the table in Part 1?

Notice, your work should be well documented (pictures and explanations) to be entitled to the full amount of bonus points. Explain if your results coincide with what you learned in class and exercises.

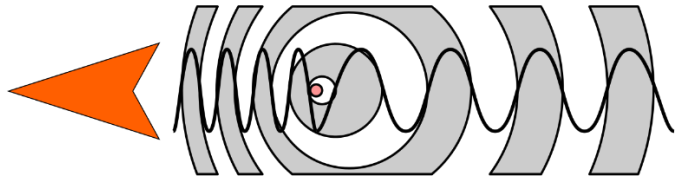
Feel free to add blocks or steps as you wish.

Try to think of your system; it has limitations (for example, with the simple antennas you have, would you be able to reach 1000m and still pick up the signal?)

Part 2 – Doppler Simulation

The Doppler effect or Doppler shift is the change in frequency of a wave in relation to an observer moving relative to the wave source.

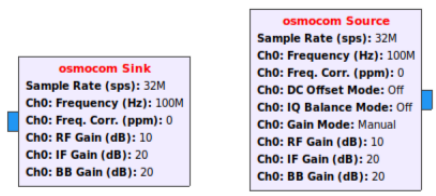
$$f_D = \frac{1}{2\pi}\frac{\Delta\phi}{\Delta t} = \frac{v cos\theta}{\lambda}$$



The Doppler effect is used in some types of radar to measure the velocity of detected objects. A radar beam is fired at a moving target — e.g., a motor car, as police use radar to detect speeding motorists — as it approaches or recedes from the radar source.

Practical experiment

Connect to an SDR and implement a system that can recognize if there is movement in the environment. As a reminder, you can use the osmocom block as a sink (the transmitter) and source (the receiver). Hint: Continuous-wave radar.



Use one of the possible QT GUIs to show movement when you engage your system, and explain why and how it works.

You can use any mathematical blocks and even write your own embedded Python blocks to implement the system.

Save your code and add it to your submission!

Practical Questions

9. Explain your work; include your code and a short clip showing it detects movement.

10. Assuming you implemented a system which can detect an object's movement, what would it take to figure out the speed and range of an object? (hint: radar)

11. How does the chosen frequency affect the system's ability to do a good job? (e.g., resolution)

12. How does transmitting power affect the system's ability to do a good job?

Theoretical Questions

13. ADS-B (Automatic dependent surveillance—broadcast) is a surveillance technology in which an aircraft, which determines its position via satellite navigation or other sensors, periodically broadcasts it, enabling it to be tracked.
    An aircraft that wishes to hide its identity is flying near an airport's radio base station, which is tuned to pick up all ADS-B transmissions up to $500Hz$ frequency deviation from the center frequency - $978MHz$. The airport's radio tower at a height of $60m$ and the aircraft was seemed to be cruising at a speed of $700Km/h$.
    a. The aircraft has been observed at a height of $3060m$. At which distance (facing the radio tower) can we expect the ADS-B to fail and the aircraft will remain anonymous?
    b. At which flight path, relative to the radio tower, will the frequency shift remain $0Hz$? (Aircrafts cannot stop mid-air, BTW)

14. A car traveling along a highway approaches a $50m$ cellular base station tower working at a frequency of $5GHz$. Knowing that the car is cruising at $140km/h$ directly towards the base station and is now $283.55m$ away from it. What is the Doppler frequency shift experienced by the driver's cellphone?

Resources

Wireless communications, Andrea Goldsmith, Stanford University, California, 2005, 9780511841224
https://en.wikipedia.org/wiki/Doppler_effect
https://en.wikipedia.org/wiki/Doppler_radar
https://en.wikipedia.org/wiki/Continuous-wave_radar
https://en.wikipedia.org/wiki/Automatic_Dependent_Surveillance%E2%80%93Broadcast
Rappaport Wireless Communications Principles And Practice 2Nd Edition, Prentice Hall, 2002
Theodore S. Rappaport, 978-0130422323

Good luck!