Linkedin
Soundcloud
GitHub
CV

A glimpse to my mind

Ilari Angervuori

Last major update 6.10.2026

Contents

1 About me

I am a Finnish mathematician interested in communication engineering. Read about my life, or jump straight to my professional records in the subsections below. Check out my blog. You can also listen to my music on SoundCloud or check the GitHub.

I graduated in 2009 from Munkkiniemi high school. Mathematics was a subject I had naturally thrived in – so, after some bumps and turns, I found myself at the University of Helsinki studying mathematics. And yeah, indeed, I love mathematics—I love the apparent universality of it. This subject is without a doubt debatable, but, at least in some sense, I like to think that mathematical truths are universal in the truest sense of the word; they are eternal, and they are the same everywhere, regardless of the physical universe we live in. Aliens in another galaxy will end up with the same mathematical truths we do. Aliens in another universe will end up with the same mathematical truths we do. Mathematics has the power to explain what we see in our everyday life. Mathematics is not only natural science but a form of art and poetry. Mathematics is music—music is mathematics.

While studying mathematics, physics, and computer science, I took some courses on economics. That inspired me to write my bachelor's thesis on optimal control theory. I worked on the problem of how increasing public investments affects the GDP. I did not find any breakthrough, but it was an intriguing subject.

I proceeded with my graduate studies studying applied mathematics. I studied subjects like partial differential equations, functional analysis, dynamical systems, and—the University of Helsinki's pride—complex analysis. (My thesis advisor said that, in a moral sense, you cannot graduate from the University of Helsinki without taking some courses on Complex analysis, because a lot of the discipline has been developed at the university.) In addition, as a more "practical" subject, I studied some inverse problems. Summa summarum, I studied a wide range of fields in mathematics.

During my graduate studies, I spent half a year in Utrecht, Netherlands, studying more applied analysis of varying subjects (searching periodic orbits in the Lorentz attractor as an example of a course—that I failed). At Utrecht University, I got the inspiration for the subject for my future master's thesis; the finite element method (FEM). After I got back to Helsinki from the exchange, I had a chance to study more about the FEM in Aalto University's courses. (Aalto University is a consortium of the Helsinki University of Technology, the Helsinki School of Economics, and the University of Art and Design Helsinki.) While writing my thesis I also taught basic mathematics courses at the University of Helsinki and gained valuable experience in the pedagogical area.

In the binge of graduation, I started to look for future opportunities. I looked for coding jobs in Helsinki and Tallinn, jobs for mathematicians in the mapping industry, continuing at some universities to pursue a Ph.D., etc. I am glad I had the chance to use my creativity and continue in Aalto University's Department of Signal Processing and Acoustics to research low earth orbit satellite communications. The research methodology was from a stochastic geometry perspective, which was well aligned with my mathematical background.

My professional ambitions are in improving the lives of people globally. Communications play an essential in the picture. (But contain some challenging problems also, as we have seen with social media.) Through effective communication, we can share knowledge, control resources, discuss issues, etc.—however, globally, the communication infrastructure is still not nearly complete. My interests contain, but are not limited to, communications, particularly wireless networks and signal processing. Modulation and demodulation, bandpass and passband. My dream is to share my knowledge in the process toward a free and honest world. (Pardon me for the cliches.)

Figure 1: Me in my beloved home town Helsinki in the middle of the summer (well, technically I am in Espoo, Otaniemi—and it is a snowy and shiny day in February, frozen sea seen in the background)

I am keeping up a blog that you can find in the blog posts section. It handles stuff encountered in my daily professional life. Stuff that I have found interesting.

1.1 Education

- 2025, PhD, "Narrow-Beam Low Earth Orbit Communications and Stochastic Geometry: Non-Temporal and Temporal Interference Analysis," Department of Signal Processing, Aalto University

- 2018, Master of Philosophy, "Elementtimenetelmä (Finite Element Method)," Department of Applied Analysis, University of Helsinki, grade: 4/5

- 2016, Bachelor of Science, "Optimiohjausteoriasta ja sen sovelluksista (On Optimal Control Theory and its Applications)," University of Helsinki

1.2 Publications

- I. Angervuori and A. Afridi, "Spatial and Temporal Correlation of the Interference in a Narrow-Beam LEO Network with ALOHA Medium Access Control," 2025, in IEEE Letters on Communications, submitted.

- I. Angervuori and R. Wichman, "Order Statistics of the SIR and Interference Cancellation in a Narrow-Beam LEO Uplink," in IEEE Letters on Communications, 2025, submitted.
- I. Angervuori, M. Haenggi and R. Wichman, "Meta Distribution of the SIR in a Narrow-Beam LEO Uplink," in IEEE Transactions on Communications, 2025, doi: 10.1109/TCOMM.2025.3547734.
- I. Angervuori and R. Wichman, "A Closed-Form Approximation of the SIR Distribution in a LEO Uplink Channel," 2022 IEEE Globecom Workshops (GC Wkshps), Rio de Janeiro, Brazil, 2022, pp. 856-861, doi: 10.1109/GCWkshps56602.2022.10008485.
- N. Okati, T. Riihonen, D. Korpi, I. Angervuori and R. Wichman, "Downlink Coverage and Rate Analysis of Low Earth Orbit Satellite Constellations Using Stochastic Geometry," in IEEE Transactions on Communications, vol. 68, no. 8, pp. 5120-5134, Aug. 2020, doi: 10.1109/TCOMM.2020.2990993.
- A. Yastrebova et al., "Theoretical and Simulation-based Analysis of Terrestrial Interference to LEO Satellite Uplinks," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taipei, Taiwan, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9347980.
- Niloofar Okati, Islam Tanash, Taneli Riihonen, Ilari Angervuori, Wichman Risto, "Performance Evaluation of Low Earth Orbit Communication Satellites," Proceedings of XXXV Finnish URSI Convention on Radio Science, 2019
- Ilari Angervuori, Wichman Risto, Niloofar Okati, Taneli Riihonen, "On routing protocols in inter-satellite communications," Proceedings of XXXV Finnish URSI Convention on Radio Science, 2019

1.3 Personal grants

- 2022, Väisälä Fund, funding for a research visit to the University of Notre Dame, 4349€

1.4 Thesis's and other projects

- PhD dissertation: A sketch (not published), "Narrow-Beam LEO Communications and Stochastic Geometry: Non-Temporal and Temporal Analysis," preprint, Department of Electrical Engineering, Aalto University, 2025
- Master's thesis "Elementtimenetelmä (The Finite Element Method)," https://helda.helsinki.fi/handle/101 Department of Applied Analysis, University of Helsinki, 2018, grade: eximia cum laude (5/5)

- Bachelor's thesis "Optimiohjausteoriasta ja sen sovelluksista (On Optimal Control Theory and its Applications)," https://www.overleaf.com/project/563b9dec737da16f65bd24f2, University of Helsinki, 2016, grade: 4/5

- An university project for a poster session, "X-ray tomography with sparse data," https://www.dropbox.com/s/88pr2da24dil460/projektitosaulijamina.jpg?dl=0, 2015

1.5 Other certificates

- 2024, IEEE Transactions, Mobile Computing, 2024 Review Certificate

2 Blog posts 2026

2.1 January—Machine learning and stochastic geometry are a happy couple

The clear trend is that artificial intelligence (AI) and machine learning (ML) are current hot topics, particularly in wireless networks and signal processing. Another active research topic is stochastic geometry (SG). In this regard, some perceive SG and ML as competitors, suggesting ML has surpassed SG in practical impact. However, this kind of contraption is a poor perspective, and without saying, not the only way to look at it. The truth is that ongoing SG research remains valuable, not only as an academic exercise, but also beyond its academic context. In practice, SG can and will be utilized in modern ML by providing **hypothesis classes** for the ML algorithms. In this post, I will provide a concrete example of how.

Let me formulate the working definitions of the ML and SG. We focus on the signal processing of wireless networks, particularly interference modeling.

- SG encompasses **stochastic models** of wireless networks, from which many statistical properties, such as signal power distributions, can be inquired. The expressions are usually presented in mathematical forms; at best, tractable and closed-form formulas are available.

- ML encompasses learning algorithms that ultimately utilize empirical **raw data**, and they are powerful tools for signal interpolation and prediction. The results are usually numerical.

The landscape of the ML algorithms is vast; however, they are often based on Gaussian process regression (GPR). The GPR is a signal prediction method that relies on estimating the signal using a set of samples (or observations) and a correlation function. More specifically, an autocovariance function (also referred to as a kernel) is at the heart of the GPR. Such an autocovariance function can be arbitrary, and one should carefully choose an appropriate one to gain meaningful results. One way is to numerically estimate the autocovariance from training data. Whether the training data is empirical data from

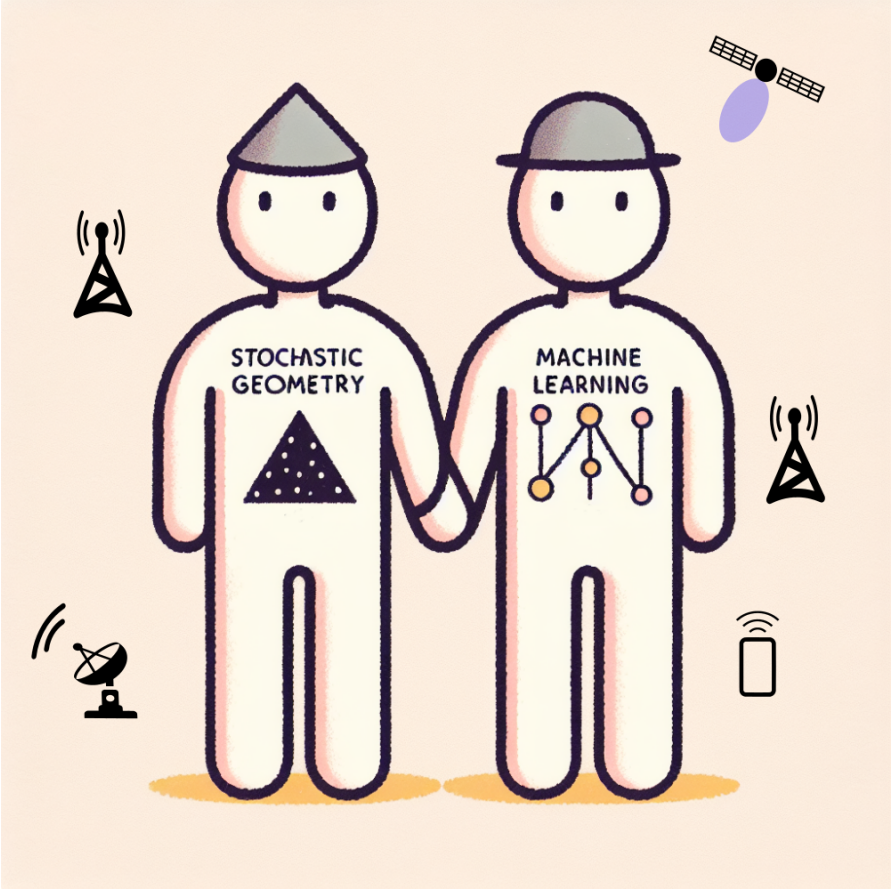


Figure 2: Image generated with the help of DALL-E, an AI by OpenAI.

an actual system or generated synthetically by using simulation tools, the numerical statistical estimates lack generality and physical insight, and have a concomitant dependence on local, empirical data and circumstances. Where to even start interpreting such data, if we don't have a rigid understanding of what qualities we are looking for? It is time for the SG to step into the arena:

> *A comprehensive, tractable, and solid SG system model will guide us in the right direction in inferring any empirical ML signal data set at hand.*

In the following, we demonstrate a simple example of estimating the power of a non-stationary Gaussian interference signal at a low Earth orbit (LEO) satellite receiver. The prior autocovariance function is derived from a theoretical estimate based on SG. Based on this autocovariance, the power estimation is inferred from empirical data using GPR. Furthermore, the estimation is compared to a moving average (MA) estimation.

The theoretical settings are:

1. The interferers (natural sources or transmitters) of bandwidth 1 kHz are distributed according to the Poisson point process (PPP) on the Earth surface, causing interference at the LEO satellite.

2. The LEO satellite has a narrow beam of -3 dB beamwidth $\varphi_{\mathrm{RX}} = 1.6^\circ = 0.027925$ rad steered towards the Earth center.

3. The LEO satellite is at $h = 200$ km moving at its orbital speed $v_{\mathrm{sat}} = 7.4$ km/s.

4. For simplicity, no fast-fading, shadowing, or other attenuation is considered. For the narrow beam, the Doppler shifts are very close to each other for all transmitters.

As the satellite moves, the magnitude of the aggregate interference varies due to the randomly distributed interferers. The key insight acquired from the SG analysis is as follows. (The details can be found in my thesis, Sections 3.3 and 3.4.)

5. Under fairly general settings (like assuming a Gaussian waveform for each interferer), the interference waveform at the LEO satellite is approximately

$$I = I(t) = \sqrt{P(t)}X(t),$$

   where $X = X(t)$ is a zero-mean Gaussian waveform with $\mathbb{E}(|X|^2) = 1$, which is modulated by the power $P = P(t)$.

6. The normalized interference power (*i.e.*, variance of I) is approximately gamma distributed with the mean $\mathbb{E}(P) = 1$ and the variance $\mathrm{var}(P) = 1/2$ (see thesis, Theorems 3.3.6 and 3.3.7).

7. The autocovariance of $P(t)$ at lag τ has the Gaussian form

$$K_P(\tau) = k \exp\left\{D\tau^2 \log(2)\right\} = \frac{p_t^2}{2\log(2)} \pi\lambda \left(\frac{h\varphi_{\mathrm{RX}}}{\sin^2(\epsilon)}\right)^2 \exp\left\{v_{\mathrm{sat}}^2 \tau^2/(h^2\varphi_{\mathrm{RX}}^2)\log(2)\right\},$$

where $k, D > 0$ are constant parameters that depend on the -3 dB width of the antenna gain φ_{RX}, interference source power p_t, the elevation angle ϵ, the altitude of the LEO satellite h, the orbital speed v_{sat}, and the density of the Earth signal sources λ.

In our settings, since the antenna is directed to the Earth center, $\epsilon = 90^\circ = \pi/2$ rad. Further, for a normalized $\mathbb{E}(P)$ (p_t can be scaled accordingly) and for our satellite settings 2-3, $k = 0.5$ and $D \approx 0.1352$.

Note that, given an antenna pattern width, the constant $D = v_{\mathrm{sat}}^2\tau^2/(h^2\varphi_{\mathrm{RX}}^2)$ in (2) is directly defined by the altitude h of the satellite: the orbital speed v_{sat} follows single-handedly from the orbital altitude. It can be empirically verified that within a fairly general density region, the assertion of D is more crucial for the GPR estimation than k (especially in the non-causal prediction): too large and too small D yield overfitting and underfitting the data, respectively. In a refined version of the GPR, k and D could also be treated as hyperparameters, which are learned from the data, but we treat them as fixed.

The prior model we will utilize in the GPR is the properties 5-7. The described signal power $P(t)$ with the Gaussian autocovariance is the hypothesis class of the set of functions within which we search our estimation from, and the waveform $I(t)$ is treated as a non-stationary Gaussian process. In practice, we will use the samples from $|I(t)|^2$ to estimate the interference power $P(t)$. Each such sample z_i is interpreted as noisy measurements of $P(t)$: $z_i = P(t_i) + \mathcal{N}(0, \sigma^2)$, where $\sigma^2 = 3$ is set to correspond to the expected variance of $|I(t)|^2$.

For a random realization of the power $P(t)$ (generated according to the gamma distribution and its second-order statistics), which is "not known" prior to the empirical samples, in Figures 1 (a) and 1 (b), we estimate $P(t)$ from a sampled (0.1 kHz) realization of the interference waveform $I(t)$ by using GPR and non-causal MA. The window size of the MA is empirically optimized, but it could also be learned from training data. Apart from the slight overfitting, the MA captures $P(t)$ reasonably well (as long as the window size is optimized). However, the GPR prediction is clearly better in capturing the smoothness of $P(t)$. Furthermore, the GPR prediction captures the $P(t)$ from $t = 0$ onward, which is impossible for the non-causal, centralized MA, which starts at $t \approx 1.5$s. The forecast region in the figure corresponds to a causal estimation of the future signal based on the past values. Not so surprisingly, the GPR is superior to MA in this regard. Both predictions can be improved by increasing the sampling frequency.

Figure 1 (a) speaks its own language for the benefit of the prior SG-based hypothesis class. On the contrary, without any hypothesis classes at hand, signal estimation would be much more difficult and arbitrary: even if the Gaussian

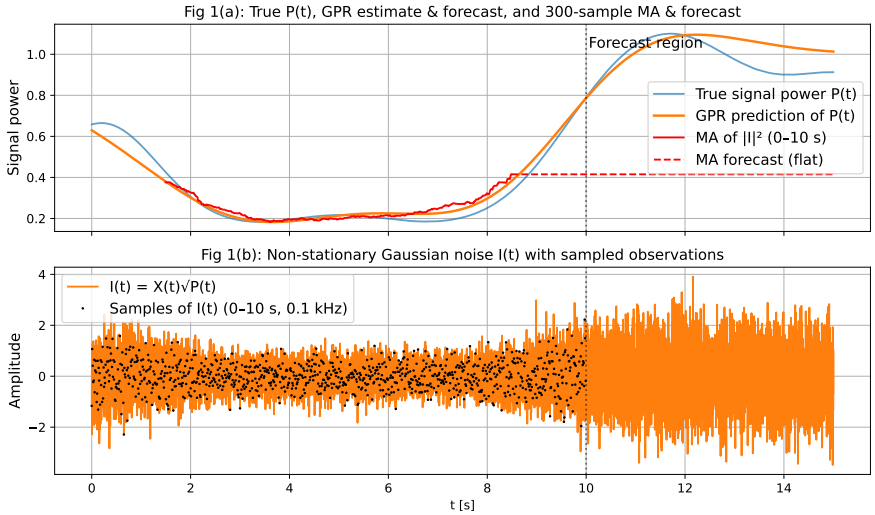


Figure 3: Signal power estimated from scarce signal samples.

correlation function is a commonly used initial guess for the autocovariance, and could be guessed without the SG analysis, the insight into the hyperparameters and their dependency on the orbital and network properties is invaluable. This was a simple, however solidly grounded, example of how SG analysis of wireless networks can assist in producing prior knowledge for the ML algorithms. A similar inquiry extends to signals affected by fading and other signal attenuation in LEO networks. Of course, terrestrial network settings are also feasible. The potential applications include interference cancellation, error correction, and medium access control.

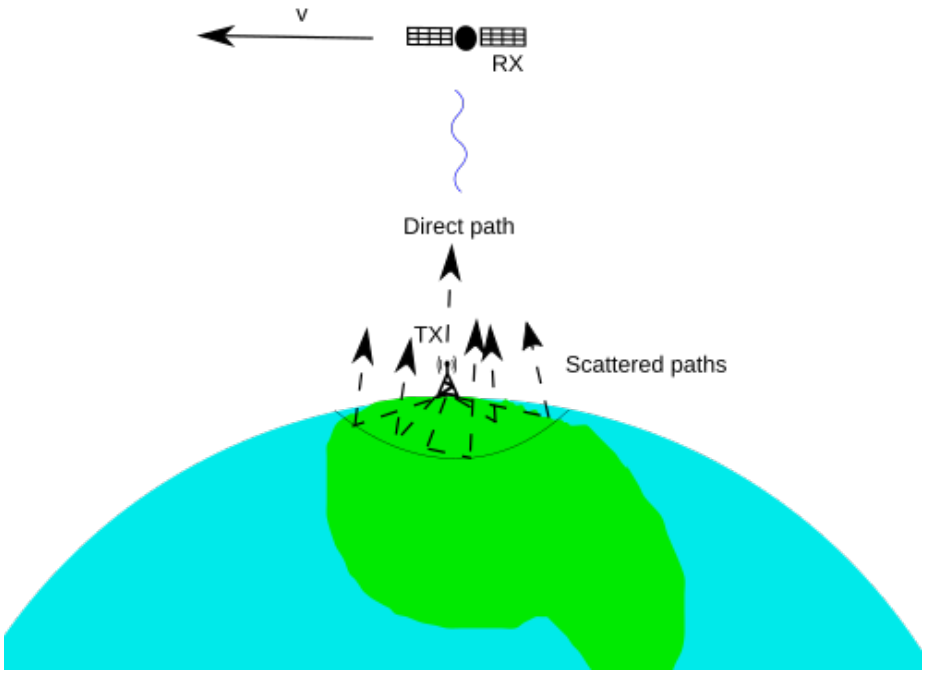
The following Python code generates the plots. (Written with the help of ChatGPT.)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, gamma
from scipy.linalg import cholesky
from numpy.linalg import solve

# === Parameters ===
T_total = 15.0       # simulate P(t), I(t) on [0, 15] s
T_obs = 10.0         # we only "observe" up to 10 s
dt = 0.001           # fine time step for continuous path (1000 Hz)
t = np.arange(0, T_total, dt)
N = len(t)

fs_sample = 100.0     # sampling of |I|^2 (0.1 kHz)
```

```
sample_interval = 1.0 / fs_sample  # 0.1 s
# indices of sampled points up to T_obs
indices = np.arange(0, int(T_obs / dt), int(sample_interval / dt))
t_sampled = t[indices]

# Gamma marginal for P(t): mean=1, var=1/2 -> Gamma(k=2, theta=0.5)
k_shape = 2.0
theta_scale = 0.5
mean_P = 1.0

# === 1. Construct latent Gaussian process for P(t) with covariance C() = 0.5 * exp(-3 ^2) =
tau_full = t[:, None] - t[None, :]
k_emp = 0.5
D_emp = 0.1352
C_full = k_emp * np.exp(-D_emp * tau_full**2)

# numerical jitter for Cholesky
jitter = 1e-10
C_full += jitter * np.eye(N)

L = cholesky(C_full, lower=True)

# latent Gaussian process
z_latent = L @ np.random.randn(N)

# Gaussian copula to get Gamma process P(t)
u = norm.cdf(z_latent)
P = gamma.ppf(u, a=k_shape, scale=theta_scale)

# === 2. Generate I(t) = X(t)*sqrt(P(t)) ===
X = np.random.randn(N)
I = X * np.sqrt(P)

# === 3. Sample I and form |I|^2 on [0, T_obs] at 0.1 kHz ===
I_sampled = I[indices]
power_sampled = np.abs(I_sampled)**2   # z_i = |I(t_i)|^2

# === 4. 300-sample moving average on sampled powers (0..T_obs) ===
window_size = 300
kernel = np.ones(window_size) / window_size
power_ma = np.convolve(power_sampled, kernel, mode='valid')

# time axis for centered MA
if window_size > 1:
    t_ma = t_sampled[(window_size-1)//2 : -(window_size//2)]
else:
```

```
    t_ma = t_sampled

# Simple MA "forecast": extend last MA value flat from last t_ma to T_total
t_ma_forecast = np.arange(t_ma[-1], T_total + 1e-9, sample_interval)  # 0.1 kHz grid to end
power_ma_forecast = np.full_like(t_ma_forecast, power_ma[-1])

# Approximate noise variance for |I|^2
# Var(|I|^2 | P=p) = 2 p^2, E[P^2]=1.5 => average Var  3
noise_var = 3.0

# Variance of 300-sample MA under iid noise with var=noise_var
ma_var = noise_var / window_size
ma_std = np.sqrt(ma_var)

# === 5. GPR prediction of P(t) on [0, 15] from noisy power samples on [0, 10] ===
# GP prior: P ~ GP(mean=mean_P, cov(t,s) = 0.5*exp(-3(t-s)^2))
# Observations: z_i = |I(t_i)|^2  P(t_i) + N(0, noise_var)

# training times (sampled up to T_obs)
t_train = t_sampled
z_train = power_sampled

# Covariance among training points
tau_ss = t_train[:, None] - t_train[None, :]
k_prior = 0.5
D_prior = 0.1352
C_ss = k_prior * np.exp(-D_prior * tau_ss**2 )
# Covariance between all times (0..T_total) and training points
tau_fs = t[:, None] - t_train[None, :]
C_fs = k_prior * np.exp(-D_prior * tau_fs**2 )


# Add observation noise to training covariance
C_ss_noisy = C_ss + noise_var * np.eye(len(t_train))

# Posterior mean: m_post(t) = mean_P + C_fs * K^{-1} * (z - mean_P)
z_centered = z_train - mean_P
alpha = solve(C_ss_noisy, z_centered)
P_gp_mean = mean_P + C_fs @ alpha

# Posterior variance: diag(C_ff - C_fs K^{-1} C_sf)
C_ff_diag = np.full(N, 0.5)  # since C_ff(t,t) = 0.5 for all t
V = solve(C_ss_noisy, C_fs.T)  # shape: (n_train, N)
C_fs_V = np.sum(C_fs * V.T, axis=1)
P_gp_var = C_ff_diag - C_fs_V
P_gp_std = np.sqrt(np.maximum(P_gp_var, 0.0))
```

```
# === 6. Plot ===
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)

# ---------- Subplot (a): P(t), GPR forecast, MA forecast ----------
# True P(t) over [0, 15]
ax1.plot(t, P, label='True signal power P(t)', color='C0', alpha=0.7)

# GPR mean and ±2 over [0, 15]
ax1.plot(t, P_gp_mean, label='GPR prediction of P(t)', color='C1', linewidth=2)
# ax1.fill_between(t,
#                  P_gp_mean - 2*P_gp_std,
#                  P_gp_mean + 2*P_gp_std,
#                  color='C1', alpha=0.2, label='GPR ±2')

# 300-sample MA on [0, T_obs]
ax1.plot(t_ma, power_ma, 'r-', label='MA of |I|² (0{10 s)')
# ax1.fill_between(t_ma,
#                  power_ma - 2*ma_std,
#                  power_ma + 2*ma_std,
#                  color='r', alpha=0.15, label='MA ±2 (approx, 0{10 s)')

# Flat MA forecast beyond last MA point
ax1.plot(t_ma_forecast, power_ma_forecast, 'r--', label='MA forecast (flat)')

# Raw power samples (0{10 s)
# ax1.plot(t_sampled, power_sampled, 'kx', label='Samples |I(t)|² (0.1 kHz)', markersize=6)

# Mark observation horizon
ax1.axvline(T_obs, color='k', linestyle=':', alpha=0.5)
ax1.text(T_obs+0.05, ax1.get_ylim()[1]*0.9, 'Forecast region', fontsize=12)

ax1.set_ylabel('Signal power', fontsize=12)
ax1.set_title('Fig 1(a): True P(t), GPR estimate & forecast, and 300-sample MA & forecast',
ax1.grid(True)
ax1.legend(fontsize=12)

# ---------- Subplot (b): I(t) and samples ----------
ax2.plot(t, I, label='I(t) = X(t)P(t)', color='tab:orange')
ax2.plot(t_sampled, I_sampled, 'ko', label='Samples of I(t) (0{10 s, 0.1 kHz)', markersize=1
ax2.axvline(T_obs, color='k', linestyle=':', alpha=0.5)
ax2.set_xlabel('t [s]')
ax2.set_ylabel('Amplitude', fontsize=12)
ax2.set_title('Fig 1(b): Non-stationary Gaussian noise I(t) with sampled observations', font
ax2.grid(True)
ax2.legend(fontsize=12)
```




```
plt.tight_layout()
plt.show()
```

References:

- Angervuori, Ilari., Narrow-Beam Low Earth Orbit Communications and Stochastic Geometry: Non-Temporal and Temporal Interference Analysis, Thesis Draft; Aalto University
- Oksanen, Maiju., Stokastinen geometria langattoman tietoliikenteen analyysissä, opinnäytetyö; aalto-yliopisto.
- Y. Hmamouche, M. Benjillali, S. Saoudi, H. Yanikomeroglu and M. D. Renzo, "New Trends in Stochastic Geometry for Wireless Networks: A Tutorial and Survey," in Proceedings of the IEEE, vol. 109, no. 7, pp. 1200-1252, July 2021, doi: 10.1109/JPROC.2021.3061778

3 Blog posts 2023

3.1 May—Signal fading in a satellite receiver and the coherence time

Let us study a faded signal in an omnidirectional satellite receiver sent by an earth transmitter. We consider that the transmitter (TX) is sending a constant QAM symbol, *i.e.*, the TX baseband signal is given by

(1)

for some $\theta \in [0, 1]$ and $A \in \mathbb{R}_+$. The passband signal experiences fading due to terrestrial obstacles. We use a model where 100 scatterer objects are Poisson distributed inside a circular area with a diameter of 180 m. The phases of the scattered signals are independently uniformly random. Initially, the satellite, which is moving at its orbital speed, is in the zenith w.r.t. the earth transmitter. The receiver's (RX) baseband signal is a linear combination of the down-conversions of the scattered, Doppler shifted and attenuated passband signal components, and it is given by

$$S_{\mathrm{RX}}(t) = \frac{A}{l(d_0)}\sqrt{\frac{K}{K+1}}e^{-2i\pi\tau_0(t)f_c}e^{-2i\pi\theta} + \sum_{j=1}^{100} + \frac{A}{l(d_j)\sqrt{100}\sqrt{K+1}}e^{-2i\pi\tau_j(t)f_c}e^{-2i\pi\theta}, \tag{2}$$

where $l(d_i)$ is the path-loss function at distance d_j from the scatterer j, d_0 is the direct-path distance to the transmitter, K is the parameter that determines the energy ratio of the direct-path (LOS) component and the scattered paths, and $\tau_j(t) = d_j(t)/c$ is the propagation delays restricted by the speed of light c, which is dependent on time t as the satellite moves, and f_c is the carrier frequency.

Given this model, we are interested in the time scale during which $S_{\mathrm{RX}}(\cdot)$ varies, *i.e.*, the coherence time. The coherence time depends on the Doppler shift, and the satellite's orbital speed is given by

$$v = \sqrt{\frac{GM}{h + R_\oplus}}, \tag{3}$$

where $GM \approx 3.9 \cdot 10^{14}\mathrm{m}^3/\mathrm{s}^2$, and the denominator is the sum of the satellite's altitude and the earth's radius, respectively. This is fast: around 7 km/s at $h = 600$ km. However, Doppler shift is only caused by the velocity component that is directed towards the transmitter. The Doppler shift is zero when the satellite is in the zenith and grows at lower elevation angles. In our model, the scatterers were distributed in an area of diameter 180 m below the satellite, and the maximum Doppler shift initially is 50 Hz for $f_c = 12 \cdot 10^9$; hence, the Doppler spread is $D_s = 100$ Hz. A small estimate for the coherence time is

Figure 4: The baseband signal with partial LOS $K = 10$.

Figure 5: The baseband signal under Rayleigh fading $K = 0$.

given by

$$T_c = \frac{1}{8D_s} \approx 10^{-3}\text{s}. \tag{4}$$

The satellite moves about 8 meters during this period.

The following figures illustrate the RX baseband signal changing in time while the satellite moves. As expected, the variance in the signal strength is larger in the Rayleigh faded case.

Here is the Octave code:

```
function signals = satellite_baseband_simulation()
  pkg load statistics
  close all;
  clear all;
  clear imread;
  clear imwrite;

  h = 600*1000; #Altitude of satellites.
  K = 0; #Rician parameter.
  t = 0; #Initial time.
  [refs bbsignals] = scatteredsignals(K);  #Random baseband signals ''bbsignals' and the sca
  signal = RXbaseband(refs, bbsignals);
  N = 200;
  history =[];
  filename = 'basebandgif.gif';
  if(exist(filename))
    delete(filename);
  end


  for iii = 1 : N
    ##Observe the progress.
    if(mod(iii,10) == 0)
      iii
    end
    refs = rotateearth(refs); #Rotate Earth.
    signal = RXbaseband(refs, bbsignals); #New received baseband signal
    history = [history [real(signal); imag(signal)]]; #History of the signals.
    ##Write GIF.
    quiver(0,0,real(signal), imag(signal));
```

```
    hold on;
    plot(history(1,:), history(2,:))
    axis([[-0.1, 0.1], [-0.1, 0.1]]);
    t = t + 1/(4*8*100); #Time hop. t = 1/(8*300) is the initial coherence time of the examp
     text(0.03, 0.03, mat2str(t, 3), 'fontsize',25);
    text(0.075, 0.03, 's', 'fontsize',25);
    frame = getframe();
    imwrite(frame.cdata, filename,'gif','writemode','append','DelayTime',0.05, 'Compression'
    hold off;
  end
end


##Returns a table of 101 (1 LOS signal and 100 signals from the scattered paths) randomly ph
function [refs, signals] = scatteredsignals(K)
  ##First, generate the Poisson distributed random obstacle locations.
  refs = [0; 0]; #LOS component.
  yMin = 1-0.0000000001; yMax = 1;
  xMin = -pi; xMax = pi;
  xDelta = xMax - xMin; yDelta = yMax - yMin; #Rectangle dimensions
  numbPoints =100;    #Number of points.
  x = xDelta*(rand(numbPoints,1)) + xMin;    #Pick points from uniform distribution
  y = yDelta*(rand(numbPoints,1)) + yMin;
  refs = [refs [pi/2-asin(y)' ; x'] ]; #Map referencepoints to spherical coordinates
  A = 30000; #Amplitude.
  signals = [A*sqrt(K/(K+1)).*exp(-rand(1,1).*i*2*pi)];
  signals = [signals A/(sqrt(100)*sqrt(1+K)).*exp(-rand(1,length(refs(1,:))-1).*i*2*pi)];
end


##Derives the received baseband signal.
function bb = RXbaseband(refs, signals)
  R = 6378*1000; #Radius of Earth in m.
  h = 600*1000; #Altitude of the satellite.
  d = @(gamma) sqrt((cos(gamma).*(R+h)-R).^2+(sin(gamma).*(R+h)).^2); ##Distance to the sate
  c = 299792458; ##Speed of light.
  if(!isempty(refs))
    a = 1./d(refs(1,:));
    tau  = d(refs(1,:))/c;
  else
    a = 0;
    tau = 0;
  end
  fc = 12*10^9; ##Modulation frequency.
  ab = a.*exp(-i*2*pi*tau.*fc); ##Received individual signals.
  bb = sum(ab.*signals); ##Aggregate received signal.
```

```
end

##Rotates the positions in 'refs' as the satellite "moves". In this model, the satellite sta
function refs = rotateearth(refs)
  t = 1/(4*8*100); #Time hop in seconds.
  h= 600*1000; #Altitude of the satellite.
  R = 6378*1000; #Radius of Earth.
  GM = 3.986*10^14; #Gravitational constant.
  orbitalspeed = sqrt(GM/(h + R)); #Satellite speed in m/s.
  angularspeed = orbitalspeed/(h + R); #Angular speed of the satellite.
  rotation = angularspeed*t; #Rotation of Earth.
  eucpos  = pol2euc(refs); #Transform the polar coordinates to euclidean coordinates.
  newpos = [[1 0 0]; [0 cos(rotation) -sin(rotation)]; [0 sin(rotation) cos(rotation)]]*eucp
  refs = euc2pol(newpos); #Back to the polaroordinates.
end


function p  = euc2pol(e)
  R = 6378*1000; #Radius of Earth.
  p = [acos(e(3,:)./R); (e(2,:) >= 0).*atan2(e(2,:),e(1,:)) + ...
   (e(2,:) < 0).*(atan2(e(2,:),e(1,:)) + 2*pi)]; #Polar coordinates from the given Euclidean
end

function e = pol2euc(p)
  R= 6378*1000; #Radius of Earth.
  e = [R*cos(p(2,:)).*sin(p(1,:));...
       R*sin(p(2,:)).*sin(p(1,:));...
       R*cos(p(1,:))]; #Euclidean coordinates from the given polar coordinates in 'p'.
end
```

References:

- Tse, David., and Pramod. Viswanath. Fundamentals of Wireless Communication. Cambridge, U.K.; Cambridge University Press, 2005.

4 Blog posts 2022

4.1 June—Gaussian vs. impulsive noise

Noise is often modeled to have a Gaussian waveform. However, it might be unrealistic in some circumstances if the outlier events are more likely than in the Gaussian distribution. In this case, we might have to model the noise with fat-tailed distributions with significant mass in the tail distribution.

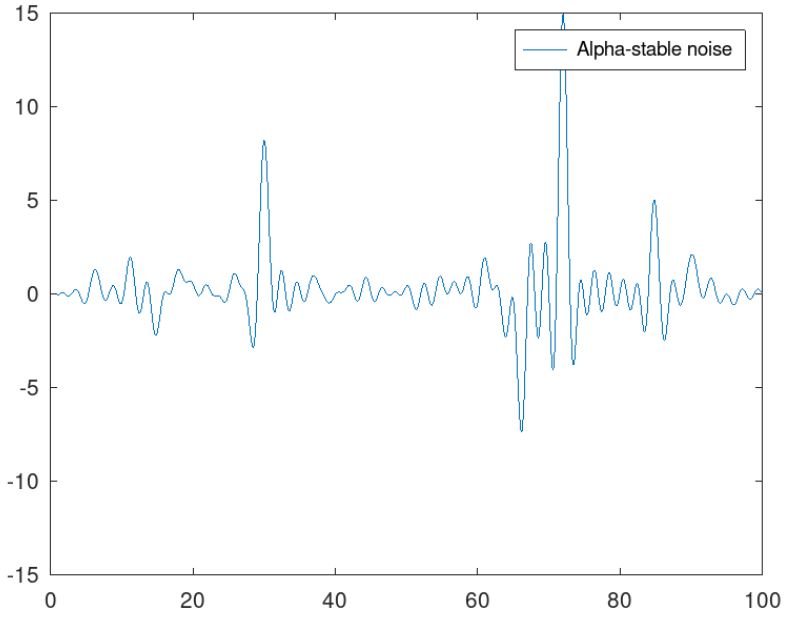
The α-stable distributions are used as noise models in various applications, including economics and electromagnetic communications. Also, the normal

distribution belongs to the α-stable family of α-stable distributions ($\alpha = 2$), the only distribution in the family with a finite variance. With values $\alpha \leq 1$, the mean is undefined. The qualitative difference to the normal distribution is clear: imagine investing money in a share that has a fixed expected profit value—say, 10 euros in a year—versus investing in a share whose value fluctuates so rapidly that the expected profit (mean profit over time) can be arbitrary.

Other impulsive noise distribution models are the *Middleton distributions* representing the envelope of non-Gaussian electromagnetic noise.

In the following, we compare the Gaussian and α-stable noises. The in-phase and quadrature components are distributed as Gaussian distribution or α-stable distribution with $\alpha = 1$, accordingly. The following GNU Octave code produces plots of bandwidth-limited analog Gaussian and α-stable noises with $alpha = 1$.

```
pkg load statistics

clear all;
close all;

%%Function returns the values at times T of the analog signal of the corresponding digital s
function xb = digitaltoanalog(T, digitalsignal, Fs)
  xb = [];
  for t = T
    xb =  [xb sum(digitalsignal.*sinc(Fs*t - (0 : length(digitalsignal) - 1)))];
  end
end


%%Inverse of the alpha stable CDF with alpha = 1.
invalphaCDF = @(x) tan(0.5*(-1 + 2*x)*pi);
%%Generate N Alpha-stable samples.
N = 100;
alphasamples = invalphaCDF(unifrnd(0,1,1,N));

%%Generate normal samples.
normalsamples = normrnd(0,1,1,N);
%%Convert to analog signal.
T = 1 : 0.1 : N;
alphasamples = digitaltoanalog(T, alphasamples, 1);
normalsamples = digitaltoanalog(T, normalsamples, 1);

alphasamples = alphasamples/(mean(abs(alphasamples))); %Normalize by mean instantaneous ampl
normalsamples = normalsamples/(mean(abs(normalsamples)));
figure(1)
plot(T, alphasamples)
legend('Alpha-stable noise')
axis([0 100 -15 15])
```



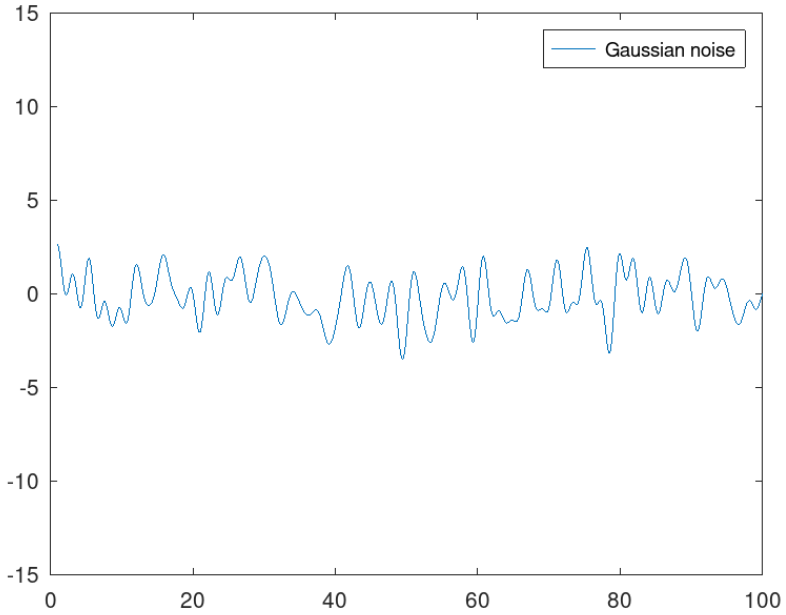

```
figure(2)
plot(T, normalsamples)
legend('Gaussian noise')
axis([0 100 -15 15])
```

Comparing the figures, one can easily understand where the term "impulsive noise" comes from.

It is also interesting to compare the audiolized noise signals: my Soundcloud.

References:

- Samorodnitsky, Gennady., and Murad S. Taqqu. Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance. New York: Chapman Hall, 1994. Print.
- Middleton, David. Statistical-Physical Models of Man-made and Natural Radio Noise parts I - , 1976.
- Tse, David., and Pramod. Viswanath. Fundamentals of Wireless Communication. Cambridge, U.K.; Cambridge University Press, 2005. Print.



5 Blog posts 2021

5.1 January—Poisson process on a sphere

The Poisson point process can be generalized to general manifolds. In Particular, the Poisson process on a three-dimensional sphere surface is useful. Nicely enough, the Poisson process on a unit sphere is equivalent to the process in a two-dimensional area $A = [-\pi, \pi] \times [-1, 1]$ through the area-preserving mapping from A to geographical coordinates

$$(x, y) \mapsto (1, x, \sin^{-1}(y)). \tag{1}$$

The resulting process interpreted in geographical coordinates (r, θ, φ) is a Poisson point process on a sphere of radius r. The following code returns a scatter plot of Poisson points on the unit sphere.

GNU Octave or Matlab:

```
%Plot random points on a unit sphere. Returns the points in a vector ref in cartesian coordi
function refc = poissononsphere(density)
  yMin = -1; yMax = 1;
  xMin = -pi; xMax = pi;

  xDelta = xMax - xMin; yDelta = yMax - yMin; %Rectangle dimensions
  numbPoints = poissrnd(density);    %Number of points in the area is a Poisson variable of
```

```
  x = xDelta*(rand(numbPoints,1)) + xMin;    %Pick points from uniform distribution
  y = yDelta*(rand(numbPoints,1)) + yMin;    %Map referencepoints to geographical coordinate

  refs = [x'; asin(y)'];%Map geographical coordinates to Cartesian coordinates on a unit cir
  r = 1;
  refc = [r*sin(refs(2,:)+pi/2).*cos(refs(1,:)+pi);...
          r*sin(refs(2,:)+pi/2).*sin(refs(1,:)+pi);...
          r*cos(refs(2,:)+pi/2)];

  figure(1)    %Plot
  [X, Y, Z] = sphere;
  surf(X,Y,Z,'EdgeColor','none','FaceColor','black');
  hold on
  scatter3(refc(1,:),refc(2,:),refc(3,:),10,...
           'MarkerFaceColor','yellow',...
           'MarkerEdgeColor','red');
  axis equal
end
```

Python:

```
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

#Rectangle dimension
xMin = -np.pi; xMax = np.pi;
yMin = -1; yMax = 1;
xDelta = xMax - xMin; yDelta = yMax - yMin; #rectangle dimensions

#Density parameter of the Poisson point process. Mean number of points on the sphere
lambda0=1000;

#Simulate Poisson point process

#Number of point in the area is a Poisson variable of intensity lambda0
numbPoints = scipy.stats.poisson( lambda0 ).rvs()
x = xDelta*scipy.stats.uniform.rvs(0,1,((numbPoints,1)))+xMin
y = yDelta*scipy.stats.uniform.rvs(0,1,((numbPoints,1)))+yMin

#Transform to geographical coordinates
x = x
y = np.arcsin(y)
#Plotting
fig = plt.figure()
ax = plt.axes(projection="3d")
```

```
ax.scatter(np.sin(y+np.pi/2)*np.cos(x+np.pi),np.sin(y+np.pi/2)*np.sin(x+np.pi),np.cos(y+np.p
plt.show()
```

Wolfram Language:

```
(*lambda is the mean number of points on the unit sphere*)
  poissononsphere[lambda_] :=
  Module[{nrofpoints, phi, theta, radius, refc, polarp},
   nrofpoints = RandomVariate[PoissonDistribution[lambda]];
   polarp =
    Table[{RandomVariate[UniformDistribution[{-Pi, Pi}]],
      ArcSin[RandomVariate[UniformDistribution[{-1, 1}]]]},
     nrofpoints];
   radius = 1;
   refc =
    Table[{radius*Sin[polarp[[i]][[2]] + Pi/2]*
       Cos[polarp[[i]][[1]] + Pi],
      radius*Sin[polarp[[i]][[2]] + Pi/2]*Sin[polarp[[i]][[1]] + Pi],
      radius*Cos[polarp[[i]][[2]] + Pi/2]}, {i, nrofpoints}];
   refc
   ];
   ListPointPlot3D[poissononsphere[500], BoxRatios -> {1, 1, 1}]
```

References:

- D. J. Daley and D. Vere-Jones, The General Poisson Process in "An introduction to the theory of point processes". New York: Springer, 2003, pp. 39.
- Stoyan, Dietrich. et al. "Stochastic Geometry and its Applications". 3rd ed. Chichester: Wiley, 2013. Print.
- H. Paul Keeler's Blog

5.2 March—Signal propagation in a city

Mobile telephone signal propagation in a city is characterized by obstacles, such as buildings and cars, that attenuate, reflect, refract and diffract the signal. Should there be a pure line-of-sight from the transmitter to the receiver, the receiver will always receive a constant power from the transmitter conditioned that the transmitter stays at a constant distance from the receiver—this can be the case, for example, if you are transmitting to a near-by high base-station antenna. Without the line-of-sight component, the signal strength will vary according to Rayleigh fading as the transmitter or the obstacles move. If only a limited line-of-sight element is present, the signal strength will follow Rician fading statistics.

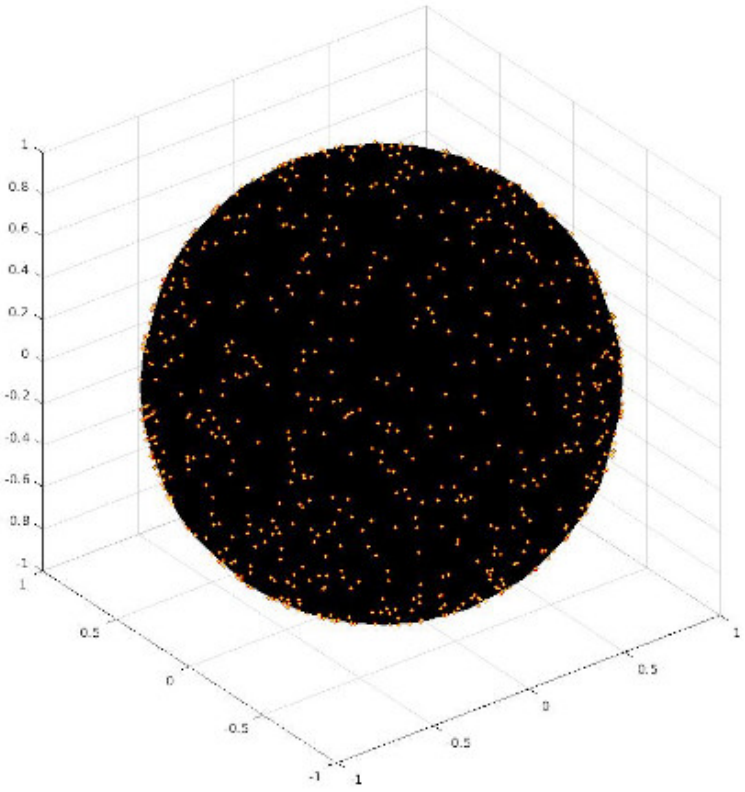


Figure 6: Are the stars Poisson distributed in the sky?

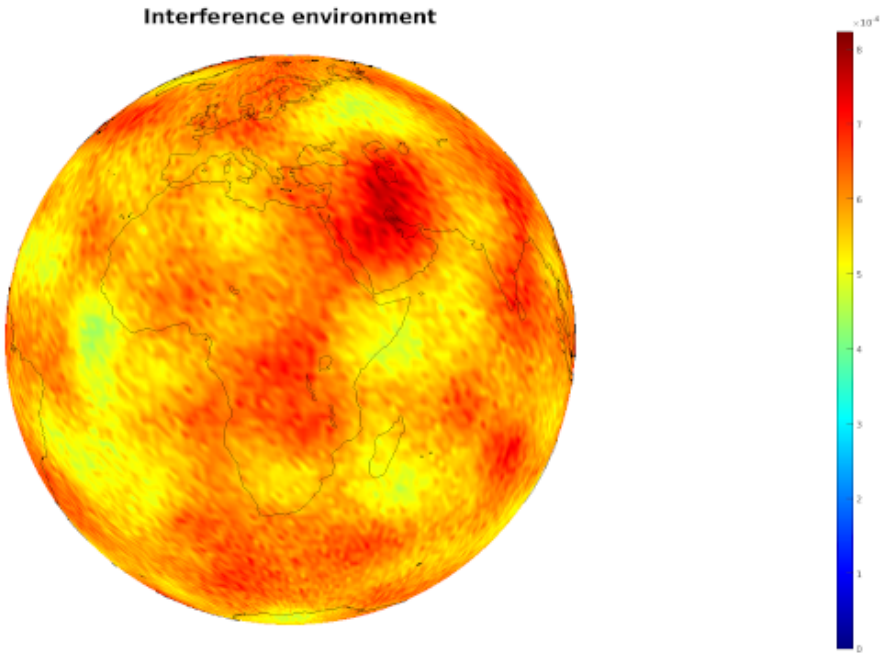


Figure 7: Aggregate interference in a satellite. A color represents aggregate interference power at the given location. The interfering sources are considered Poisson distributed on Earth.

Figure 8: The white boxes represent buildings, and the little white circle is the transmitter. One can see how reflections will cause the aggregate signal strength to fluctuate by location depending on the phases of the incoming reflected waves.

Figure 9: The interference power field develops in time as the interfering transmitters are moving in random directions at each time step. Red color represents the highest mean power of interference, and deep blue represents the absence of any interference. We assume Rician fading. The interference in the red occurrences will reduce the performance remarkably.

I animated a couple of GIFs to help us perceive what is happening. The first figure demonstrates how a simple sine signal propagates between buildings. You can observe how the multi-path components sum up to a signal amplifying in some locations and degenerating in others. In the first figure, you can look for Rician conditions (lower right) and Rayleigh conditions (lower left) and see how the signal behaves. The figure was obtained by solving the Helmholtz equation by finite element method.

The second figure shows how the aggregate signal power from many mobile transmitters develops in time under Rician fading conditions. The aggregate power can be considered as interference in a receiver. You can see how the high peaks of interference power emerge at random locations. These kinds of peaks can cause decreased data rates. The figure was obtained by simulating a random walk of points in a realization of the Poisson point process on a plane.

References:

- François Baccelli, Bartlomiej Blaszczyszyn Stochastic Geometry and Wireless Networks, Volume I -Theory
- Nicolae Cindea (2021). Movie to GIF Converter, MATLAB Central File Exchange. Retrieved June 14, 2021.

5.3 May—Rayleigh fading audiolized

When a signal propagates through multiple paths, each signal component in each path will be in a different phase and of varying strength when received. Should there be no line-of-sight component present, the additive signal will fade according to the Rayleigh fading.

For example, a simple sine wave

wave (links to soundcloud.com—VOLUME ALERT)

can after some multi-path propagation sound like

this:

Assuming that the receiver is moving, the variation of Doppler shifts in different signal paths will cause the signal strength to vary randomly in time.

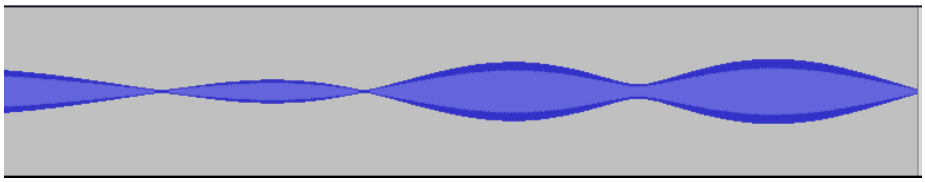


Figure 10: The Envelope of a multi-path faded sine signal.

If we add some Gaussian white noise, we notice that the original signal is somehow recognizable from the

noised signal (listen carefully, and you will hear the sine signal in the background).

but the

faded signal

will sometimes get buried under the noise—these events are referred to as *deep fades*.

Here is a GNU Octave or Matlab code for the used Rayleigh simulator, which outputs the signal as audio:

```
%Rayleigh simulator. Jakes model. In Octave remember to load the statistics package.

close all
clear all

tic

N = 20; %Number multipaths.
T = linspace(0,10000,100000); %Time.
v = 0.001; %Speed of the receiver.
randan = pi*rand(1,N); %Random angles w.r.t. receiver.
rI = 1000*rand(1,N); %Random distances of the sources.

%Geometrical stuff. Check for the Jakes model in the reference.
An = @(t) (atan(sin(randan).*rI./(cos(randan).*rI-v*t))).*(cos(randan).*rI>v*t)+...
(pi-atan(sin(randan).*rI./(v*t-cos(randan).*rI))).*(cos(randan).*rI<=v*t);
phis = 2*pi.*(rand(1,N));
wc =pi/2; %Frequency of the signal.
Beta = 2*pi/(1/(wc/(2*pi)));
theta = @(t) cos(An(t)).*Beta*v.*t+phis;

powers = rand(1,N); %Random powers of the signals in the multipaths.
powers = powers./sum(powers); %Normalize the powers.
Ez = @(t) sum(powers.*cos(wc*t+theta(t)));
EZ = []; %Faded signal.
REF = []; %Original signal.
```

```
NOISE = []; %Additional Gaussian noise.
for t =T
EZ = [EZ Ez(t)];
REF = [REF cos(wc*t)];
NOISE = [NOISE 0.9*stdnormal_rnd(1)];
end

%Write the audio files at sampling rate 8000.
audiowrite('EZ.wav',EZ,8000)
audiowrite('EZNOISE.wav',1/2*(EZ+NOISE),8000)
audiowrite('REFNOISE.wav', 1/2*(REF+NOISE), 8000)


toc
plot(T,EZ)
```

And the same in Python:

```
  import numpy as np
  import math
  import matplotlib.pyplot as plt
  import sounddevice as sd
  import time

  #Jakes Rayleigh simulator. Please check for the reference in this site for further details

  N = 20 #Number of multipaths.
  T = np.linspace(0, 10000, 100000) #Time vector.
  v = 0.001 #Speed of the receiver.



  randan = np.random.rand(1, N) * math.pi #Random angles w.r.t. receiver.
  rI = np.random.rand(1, N) * 1000 #Random distances of the sources.
  phis = np.random.rand(1, N) * 2 * math.pi

  def An(t):
  return (np.arctan(np.sin(randan) * rI / (np.cos(randan) * rI - v * t))) * (
  np.cos(randan) * rI > v * t
  ) + (np.pi - np.arctan(np.sin(randan) * rI / (v * t - np.cos(randan) * rI))) * (
  np.cos(randan) * rI <= v * t
  )

  wc = np.pi/2 #Frequency of the signal.
  Beta = 2 * math.pi / (1 / (wc / (2 * math.pi)))

def theta(t):
```

```
    return np.cos(An(t)) * Beta * v * t +phis

powers = np.random.rand(1,N)*2 #Random powers of the signals in the multipaths.
powers = powers/np.sum(powers) #Normalize powers..

def Ez(t):
    return np.sum(powers * np.cos(wc * t + theta(t)))

EZ = np.vectorize(Ez)(T)
REF = np.vectorize(lambda time : 2*np.cos(wc * time))(T)

#Play and plot.
fs = 8000
sd.play(EZ,fs,blocking = True)
#sd.play(REF,fs,blocking = True)
plt.plot(T, EZ)
plt.show()
```

References:

- William C. Jakes, "Microwave Mobile Communications", IEEE PRESS, 1974.

5.4 June—Why does MIMO work?

Multiple-input and multiple-output MIMO antenna technology is used to exploit the multi-path propagation to improve the capacity of a communication link. In principle, the link's capacity can be increased merely by increasing the power of the transmitting antenna. However—apart from being energy-consuming—this also increases interference to any other receivers should they operate in the same frequency band. In MIMO, the energy is divided among multiple antennas, and the link capacity is improved without using any extra energy and without increasing interference towards other transmitters.

Why does it work? Here is how I came up with a simple argument based on stochastic geometry. First, let us make some assumptions. We assume a flat and infinite Earth (this is a "tin foil assumption," but it is often reasonable). In addition, our communication channel environment consists of many obstacles so that our transmitting and receiving antennae can not see each other. We are in a city full of houses, cars, trees, etc. Then, our data signal propagates to the receiver through multiple paths, for example, through distinct streets around different houses. The aggregate signal in the receiver will be Rayleigh faded. The city is stormed with mobile phones exchanging data with their base stations, which further hand the data to the receiving mobile phones' base station and, on the other hand, cause interference to the other base stations. We assume that the interfering base stations are independently located and are distributed according to the Poisson point process.

Let's consider that person A is sending a message to person B. We are interested in the probability that the base station serving person A successfully transmits the message to a base station serving person B. Under the assumptions above and some other simplified assumptions, as derived in here, we can express the probability of a successful transmission as:

$$\mathbf{P}[\text{Successful single-antenna transmission}] = e^{-\frac{\pi^2}{2\sqrt{P}}}, \tag{1}$$

where P denotes the mean transmitting power of the base stations.

In MIMO, we use multiple distinct antennas to transmit the same message. In each antenna, the encoding of the message should be different so that it can't mix with the information that the other antennas are transmitting—this is possible by orthogonal modulation. Assuming that each message from each antenna independently propagates to the receiver, we can calculate from equation (1) by the complementary probability that **at least one message transmission gets through:**

$$\mathbf{P}[\text{Successful MIMO transmission with N antennas}] = 1 - \left(1 - e^{-\frac{\pi^2}{2\sqrt{P/N}}}\right)^N, \tag{2}$$

where N is the number of transmitting antennas. Notice that we divided the transmitting power P by the number of antennas, so we didn't increase the aggregate power. On the other hand, should we have no MIMO technology at hand, we could improve the link quality by increasing the power P of a single antenna. In the following figure, we compare these two cases.

It is evident that MIMO is an excellent solution for increasing the throughput of a wireless communication link. Using multiple antennas, we can save power and achieve better data rates than by merely increasing the power of a single transmitter. In MIMO, using multiple antennas, statistically speaking, a significant fraction of the channels are in a good state (as well as a bad state), and the performance is consistent.

References:

- François Baccelli, Bartlomiej Blaszczyszyn Stochastic Geometry and Wireless Networks, Volume 2 - Applications

5.5 August—Radio waves in a tunnel

Almost everyone has some experience of how radio turns static when driving into a tunnel. The radio stations only work if a special radiating cable is installed inside the tunnel. Once upon a night, I could not sleep, pondering why the FM radios are unsuable in the tunnels. From my experience, mobile phone signals that operate at higher frequencies work better. Someone might have told me that FM radio wavelength is so large that the wave "does not fit the tunnel".

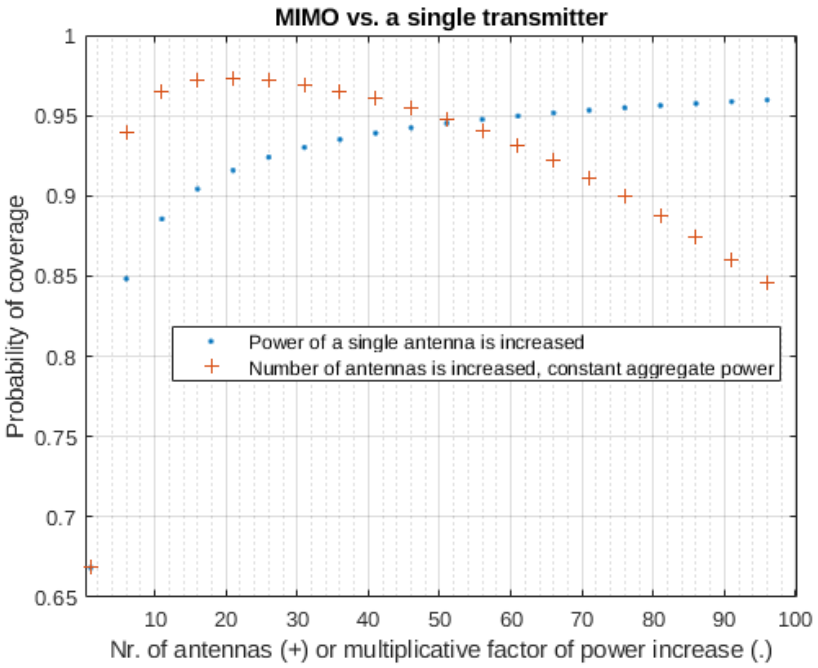


Figure 11: MIMO saves energy and improves performance.

Figure 12: High frequencies fit in the tunnel.

Figure 13: Wavelengths bigger than the diameter of the tunnel will not get through.

But as intuitively acceptable as this explanation could be (I don't think it is even that), this answer did not satisfy me.

After some research, I found that the explanation lies in Maxwell equation's behavior (well, what a surprise). Everything backs down to the boundary conditions of the electromagnetic field at the tunnel wall: the tangential component of the electric field component of the electromagnetic wave has to be near zero in the tunnel interface—this leads to a situation where the wave has, in a sense, no room to oscillate inside the tunnel.

Intuitive or not, this is what the equations tell us. To demonstrate this, I solved using FEM, the Helmholtz equation for a plane wave in a two-dimensional setting mimicking a tunnel and its entrance. The electromagnetic wave's polarization is so that the electrical component is pointing at the right angle w.r.t. the plane (either towards or against the reader of this page). Boundary conditions inside the 2D tunnel is zero. The coloring represents the electrical component's magnitude.

We compare the behavior of two different wavelengths:

.

References:

- Scattering Problem: Matlab PDE Modeler App
- Interface conditions for electromagnetic fields (Wikipedia article)

6 Miscellaneous notes

6.1 Controlling your passwords with pass

Pass is a nice Unix style free and open source wallet for keeping your passwords safe. Here is a brief look how to set it up in Linux.

- Install the application in the terminal.

```
sudo apt install pass
```

- Check for the existing GPG keys.

```
gpg --list-keys
```

- If no keys were found generate a key pair.

  ```
  gpg --generate-key
  ```

- Copy the name of the key and initialize pass.

  ```
  pass init ABCDEFGHIJKLMNOPQRSTUV1234
  ```

  where ABCDEFGHIJKLMNOPQRSTUV1234 is the name of the key.

- Generate a password with

  ```
  pass generate keyfolder/newkey
  ```

  List the passwords.

  ```
  pass
  ```

  Copy a password to the clipboard.

  ```
  pass keyfolder/newkey -c
  ```

  For more commands:

  ```
  man pass
  ```

Connect Pass to Git and it is easy to keep track of passwords with multiple machines.

- Export your public and private key to a file with

  ```
  gpg --export --output public.key ABCDEFGHIJKLMNOPQRSTUV1234
  gpg --export-secret-key --output private.key ABCDEFGHIJKLMNOPQRSTUV1234
  ```

  where ABCDEFGHIJKLMNOPQRSTUV1234 is your key name.

- Now, we can initialize the Git repository with these keys. Move the public key and private key through a safe channel to a computer you wish to use Pass in. Import the keys to the machine:

```
gpg --import public.key
gpg --import private.key
```

- After importing the keys to a new machine you can list the keys:

  ```
  gpg --list-keys
  ```

- and initialize Pass with

  ```
  pass init ABCDEFGHIJKLMNOPQRSTUV1234
  ```

- First, remember to add your public SSH key to GitHub. We will initialize the GitHub repository. If you do this for the first time, create a new repository named "pass-store" before the following commands. If the first command asks you to identify yourself, follow the instructions.

  ```
  pass git init
  pass git remote add origin git@repo.com:myname/pass-store
  ```

- Get password data from the server (from a non-empty repository, otherwise skip)

  ```
  pass git pull origin master --allow-unrelated-histories
  pass git commit -am "firstcommit"
  ```

  If Git complains about "divergent branches" just choose the "merge" reconcileing and repeat the command.

- Do some changes and pass will automatically commit them. Push and set the "upstream".

  ```
  pass git push --set-upstream origin master
  ```

- From here on you can use the familiar git commands.

  ```
  pass git pull
  pass git push
  ```

  It can be the case that you have to raise the trust level of the public key. For that, check this article.

Stay safe :)

References:

- Password Store

6.2 October – Fast Fourier transform in GNU Octave

Fast fourier transform (FFT) is an algorithm that computes the discrete Fourier transform.

FFT is especially handy in real-time digital signal processing. Digital computers are working on discrete data, thus a input signal is always sampled with some sampling rate F_s (Hz). By the Nyquist-Shannon sampling theorem, sampling captures frequencies under $F_s/2$.

Octave calculates the FFT of a discrete signal. In the following code, we calculate the normalized FFT and plot the frequency spectrum w.r.t. the frequency. Signal vector and sampling frequency are given as input.

```
##Calculates the FFT and plots the frequency spectrum of a signal. Sampletimes have to start

function fftvector = plotfft(signal, Fs)
  N = length(signal); #Signal length.
  FFT =  fft(signal);
  if(mod(N,2) == 0) #Check if signal length is odd or even.
    FFT = 2*FFT(1 : N/2)/N;
    f = Fs*(0 : (N - 1)/2)/(N - 1);
  else
    FFT = 2*FFT(1 : (N - 1)/2)/N;
    f = Fs*(0 : (N - 2)/2)/(N - 1);
  end
  fftvector = [f; FFT];

  figure(1)
  plot(f, abs(FFT));
  title('Fast fourier transform')
  xlabel('Frequency (Hz)')
  print  plot.jpg
end
```

The following figure shows my track Tappimarssi in the frequency domain. The track was imported to Octave by

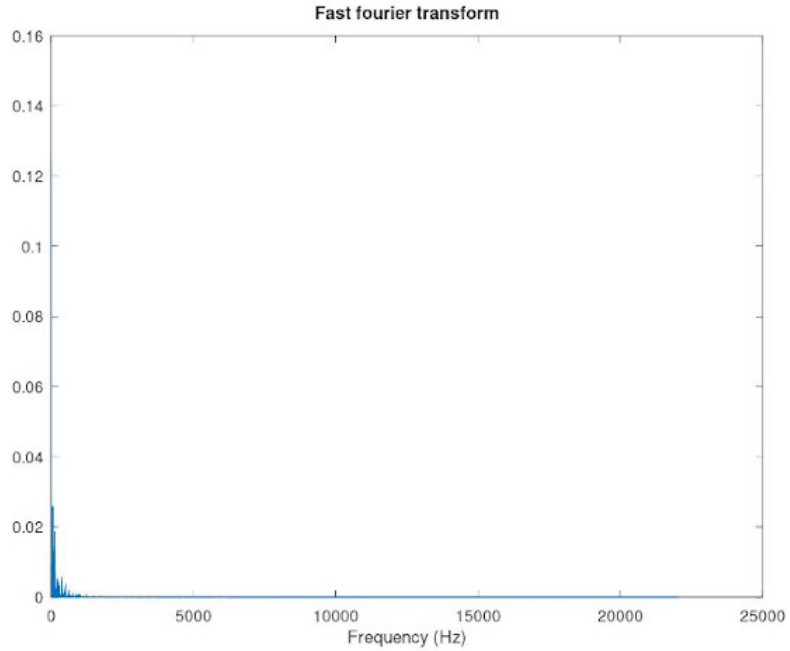
```
[signal, Fs] = audioread('Tappimarssi.wav');
```

- Octave Documentation – Signal Processing

6.3 November – Digital filtering

Let's construct a simple digital band-pass filter based in the sinc filter.

As everyone know, the sinc function is the impulse response of a low-pass filter. That is, the convolution of a signal $S(t)$ and the sinc function $t \mapsto$



$\frac{\sin(2\pi B_L t)}{\pi t}$ will produce a low-pass filtered signal $S(t)$ without frequencies higher than B_L. Similarly, the function $t \mapsto \delta(t) - \frac{\sin(2\pi B_H t)}{\pi t}$, where $\delta(t)$ represents the Dirac delta function, is the frequency response of the ideal high-pass filter of frequency B_H.

Heuristically, we can right away derive the discrete versions of the impulse responses:

$$\mathbb{Z} \ni i \mapsto \frac{\sin(2\pi \frac{B_L}{f_c} i)}{\pi i},$$

for the low-pass filter, and

$$i \mapsto \delta[i] - \frac{\sin(2\pi \frac{B_H}{f_c} i)}{\pi i},$$

for the high-pass filter. The variable f_c is the sampling frequency and $i \mapsto \delta[i] := \delta_{0i}$ is the Kronecker delta.

Now, having a discrete signal at hand, band-pass filtering is (in the simplest approach) just a matter of discrete convolution of the signal with the functions above. The following Octave code does the job.

```
##Sinc band-pass filter. f0 = B_L/fs, f1 = B_H/fs

function filtered = sincfilter(signal, f0, f1)
```

```
  if(mod(length(signal), 2) != 0) #Check if the signal length is even or odd.
    signal = signal(1 : length(signal) - 1);
  end

  M = 10000; #Increase this to increase accuracy.
  sincF = zeros(1, M);
  for m =  -M/2 + 1 : 1 : M/2
    if(!(m  == 0))
      sincF(m + M/2) = sin(2*pi*f1*m)./(pi*m);
    else
      sincF(m + M/2) = 2*f1;
    end
  end

  sincH = zeros(1,M);
  for m =  -M/2 + 1 : 1 : M/2
    if(!(m  == 0))
      sincH(m + M/2) = -sin(2*pi*f0*m)./(pi*m);
    else
      sincH(m + M/2) = -2*f0 + 1;
    end
  end

  ##Plot stuff.
  figure(1)
  plot(sincF)
  figure(2)
  plot(signal)

  tic
  filtered = conv(signal, sincF, "same");
  filtered = conv(filtered, sincH, "same");
  toc

  figure(3)
  plot(filtered)
end
```

For more accuracy and effectiveness, one should use windowed or recursive filters. One can check in the references for more sophisticated methods!

References:

- Steven W. Smith, "Digital Signal Processing – A Practical Guide for Engineers and Scientists", Elsevier Science, 2003.

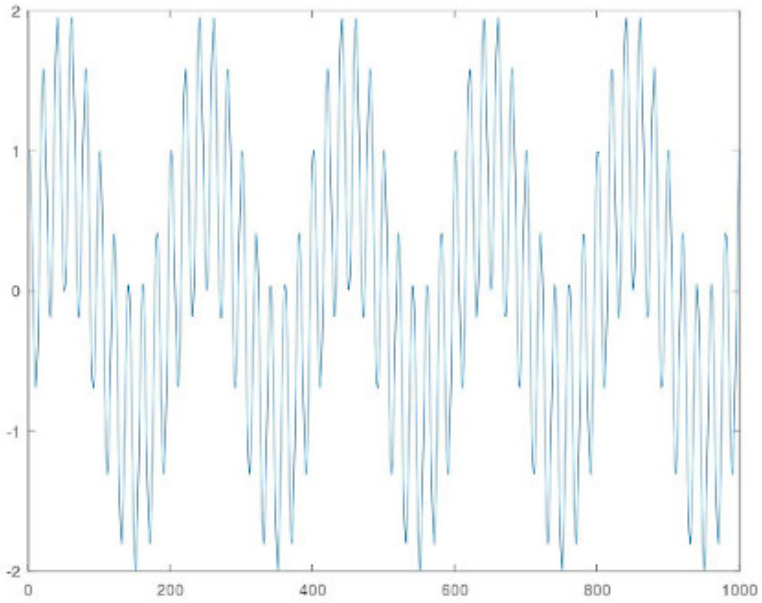


Figure 14: Signal of form $\cos(20\pi t) + \sin(2\pi t)$

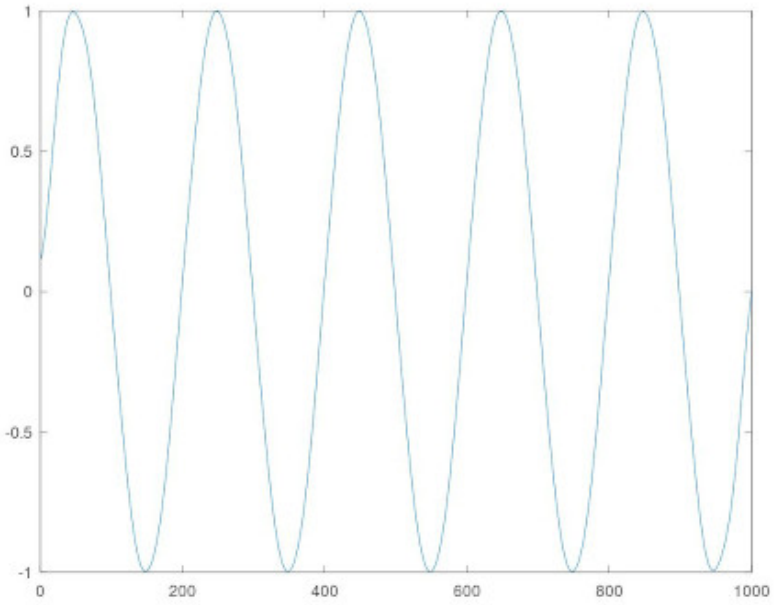


Figure 15: The same signal with the first term filtered away

6.4 Inverse of a Gaussian variable

There is a lot of literature on the inverse of a Gaussian distributed variable (this should not be fixed with inverse Gaussian distribution—it is a different matter). The inverse distribution is ill-behaved; the mean and variance do not generally exist.

I came up with a simple approximation that works well if the mean is large enough and the variance is small enough (I have yet to work out the details of the exact conditions for this approximation. However, the results can be verified, e.g., by Monte Carlo simulations).

First, approximate the Gaussian distributed variable $X \sim \mathcal{N}(\mu, \sigma^2)$ by a log-normally distributed variable $X \approx Y \sim \text{Lognormal}(\mu_{\text{LN}}, \sigma_{\text{LN}})$, with corresponding mean and variance, *i.e.*,

$$\mu = \exp\left(\mu_{\text{LN}} + \frac{\sigma_{\text{LN}}^2}{2}\right)$$

and

$$\sigma = (\exp(\sigma_{\text{LN}}^2) - 1)\exp(2\mu_{\text{LN}} + \sigma_{\text{LN}}^2).$$

We leave the solving of μ_{LN} and σ_{LN} as an easy exercise for the reader. (:

Using the theory of log-normal distribution, the inverse of X is now given by

$$1/X \approx 1/Y \sim \text{Lognormal}(-\mu_{\text{LN}}, \sigma_{\text{LN}}).$$

That's it!

References:

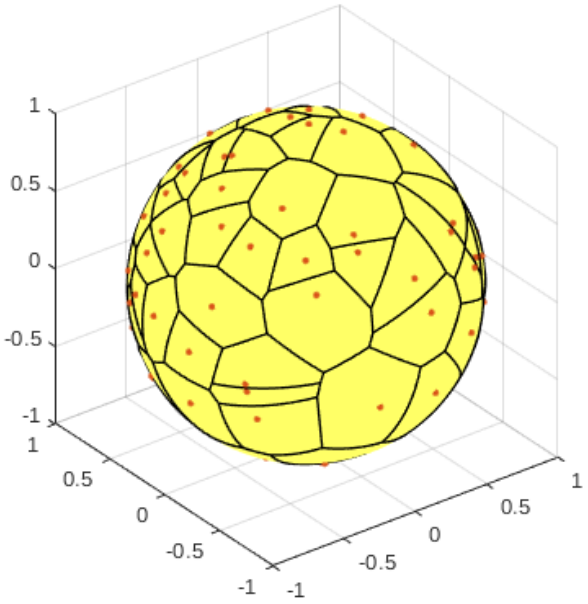
- Log-normal distribution
- Díaz-Francés, Eloísa; Rubio, Francisco J. (2012-01-24). "On the existence of a normal approximation to the distribution of the ratio of two independent normal random variables". Statistical Papers. Springer Science and Business Media LLC.

6.5 Voronoi tesselation on a sphere

Voronoi tessellation consists of neighborhood areas of a given set of points, or Voronoi cells: every cell is surrounded by the area that consists of locations that are closer to the cell than any other cell. The Voronoi Diagram, or Voronoi tessellation, has applications, *e.g.*, in wireless communications.

The next Matlab code produces a Voronoi tessellation on a sphere Poisson points as cells. It is based in Grady Wrights codes openly available in Github.

```
%%Create Voronoi tesselation around Poisson points on a Sphere.

clear all;
```




```
close all;

addpath(fullfile(cd,'rbfsphere'));
density = 100;
X = poissononsphere(density)'; %Poisson points in spherical coordinates.
voronoiSph(X); %This function is downloaded from gradywright's Github. It is placed in the f

function refc = poissononsphere(density)
  yMin = -1; yMax = 1;
  xMin = -pi; xMax = pi;

  xDelta = xMax - xMin; yDelta = yMax - yMin; %Rectangle dimensions
  numbPoints = poissrnd(density);    %Number of points in the area is a Poisson variable of
  x = xDelta*(rand(numbPoints,1)) + xMin;    %Pick points from uniform distribution
  y = yDelta*(rand(numbPoints,1)) + yMin;    %Map referencepoints to geographical coordinate
  ref = [x y]';

  refs = [x'; asin(y)'];%Map geographical coordinates to Cartesian coordinates on a unit cir
  r = 1;
  refc = [r*sin(refs(2,:)+pi/2).*cos(refs(1,:)+pi);...
          r*sin(refs(2,:)+pi/2).*sin(refs(1,:)+pi);...
          r*cos(refs(2,:)+pi/2)];
```

```
end
```

References:

- gradywright Github

6.6 May – Digital to analog modulation

Digital signal consists of values at discrete times, whereas analog signal is continuous in time t. As the orthogonal set of sinc functions

$$\{t \mapsto \text{sinc}(F_s t - n)\}_n,$$

spans the space of signals of bandwidth $F_s/2$, an analog signal can be (uniquely) produced from a digital signal of length N

$$\{x[n]\}_{n=0}^{N-1}$$

as a superposition of the sinc basis-functions

$$S(t) = \sum_{n=0}^{N-1} x[n]\text{sinc}(F_s t - n),$$

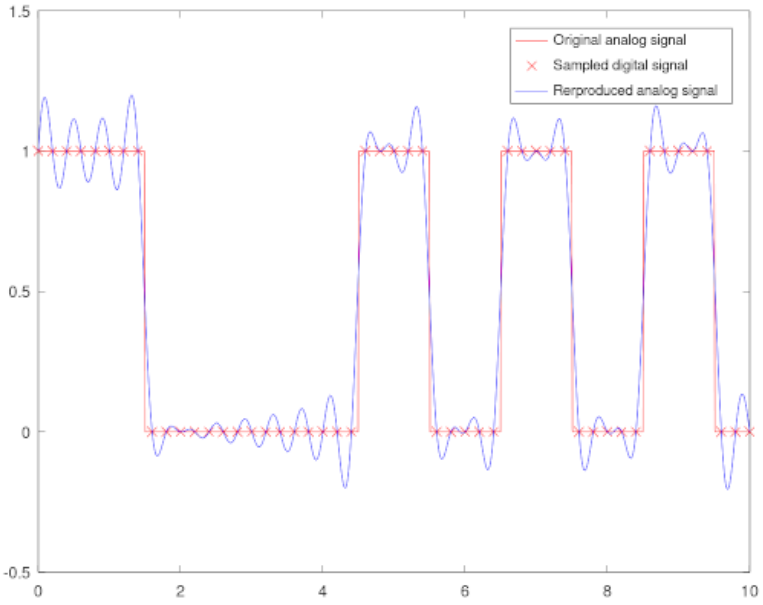
where F_s is the sampling rate of the digital signal.

The following GNU Octave function produces the analog signal from a given digital signal

```
%%Function returns the values at times T of the analog signal of the corresponding digital s
function xb = digitaltoanalog(T, digitalsignal, Fs)
  xb = [];
  for t = T
    xb =  [xb sum(digitalsignal.*sinc(Fs*t - (0 : length(digitalsignal) - 1)))];
  end
end
```

The following code simulates a rectangle pulse train signal given as input, samples the input to a digital signal x, and modulates the sampled digital signal back to an analog signal S. As the Nyquist rate restricts the bandwidth of the signal S to $F_s/2$, information is lost in the sampling stage, because the original rectangle train has an infinite bandwidth.

```
pkg load signal;
close all;
clear all;

input  = @(t) pulstran(t,[0,1,5,7,9],"rectpuls") %%The original analog signal at time t.
```



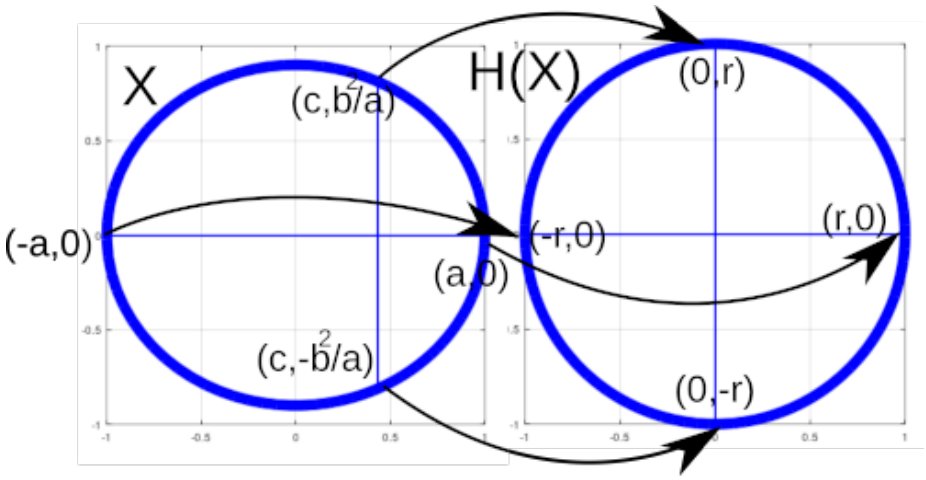

```
Fs = 5; %Sampling rate.
t = 10 %Time length of the signal.
Ts = 0 : 1/Fs : t; %Sampling time instances.
x =  input(Ts); %Sampled values, i.e. the digital signal.

Ta = linspace(0, t, 1000); %Analog signal time instances for the plot.
S = digitaltoanalog(Ta, x, Fs); %Modulated analog signal.

%Plot.
plot(Ta, input(Ta), 'color', 'r');
hold on;
plot(Ts, x, 'x', 'color', 'r')
plot(Ta, S, 'color', 'b');
legend( 'Original analog signal','Sampled digital signal', 'Rerproduced analog signal')
```

References:

- Tse, David., and Pramod. Viswanath. Fundamentals of Wireless Communication. Cambridge, U.K.;: Cambridge University Press, 2005. Print.

6.7 Homography

Homography can be used to "change the perspective" of an image (set of vec-



tors). I have used homography for the satellite footprint in the computer simulations. For elevation angles smaller than 90, you can conveniently map transmitters inside the elliptical footprint to a circle for which the radially symmetric antenna pattern function can be used. The following homography matrix H transforms an ellipse of parameters a and b to a circle of radius r so that the right-hand side focus point maps to the origin.

$$H = \begin{pmatrix} -a & -0 & -1 & 0 & 0 & 0 & ar & 0 & r \\ 0 & 0 & 0 & -a & -0 & -1 & 0 & 0 & 0 \\ -c & -b^2/a & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -c & -b^2/a & -1 & cr & rb^2/a & r \\ a & 0 & -1 & 0 & 0 & 0 & ar & 0 & -r \\ 0 & 0 & 0 & a & 0 & -1 & 0 & 0 & 0 \\ -c & b^2/a & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -c & b^2/a & -1 & -rc & rb^2/ar & -r \end{pmatrix}$$

Here is a GNU Octave code:

```
%%Changes ellipses with parameters a > b  perspective to a sphere of radius r centered in or

function points = homography(refc, a,b,r)
  c = sqrt(a^2 - b^2);
  %%Construct the homography matrix.
  X1 = [[-a -0 -1 0 0 0 a*r 0*r r]; [0 0 0 -a -0 -1 a*0 0*0 0]];
  X2 = [[-c -b^2/a -1 0 0 0 c*0 b^2/a*0 0]; [0 0 0 -c -b^2/a -1 c*r b^2/a*r r]];
  X3 = [[a 0 -1 0 0 0 -a*-r 0*-r -r]; [0 0 0 a -0 -1 -a*0 0*0 0]];
  X4 = [[-c b^2/a -1 0 0 0 c*0 -b^2/a*0 0]; [0 0 0 -c b^2/a -1 c*-r b^2/a*r -r]];
  P = [X1; X2; X3; X4];
  [U,S,V] = svd(P); %Singular value composition.
```

```
  h = V(:,9);
  H = reshape(h, 3, 3)';
  points = [];
  for point = refc
    homopoint = [point; 1]; %Point presented in homogeneous coordinates.
    homopoint = H*homopoint;
    homopoint = [homopoint(1)/homopoint(3); homopoint(2)/homopoint(3)];
    points = [points homopoint];
  end
  figure(1)
  plot(points(1,:), points(2,:), 'b','linewidth',10);
end
```

In the following, we are rotating a "pyramid". It can be seen how the homography mapping can be interpreted as a change of perspective.

References:

- Stack exchange

6.8 Asymptotic decay rate of a probability distribution

Let us study the tail distributions of three related distributions. The asymptotic decay rate measures the thickness of a random variable's X tail distribution. It is defined by

$$\lim_{x\to\infty} -\frac{\log \mathbf{P}(X > x)}{x}, \tag{1}$$

where $\mathbf{P}(\cdot)$ denotes the probability of an event. Let us calculate the decay rates for the exponential distribution, gamma distribution, and normal distribution.

For **exponentially distributed** X with scale parameter θ:

$$\rho_{\text{Exponential}} = \lim_{x\to\infty} -\frac{\log \mathbf{P}(X > x)}{x} = \lim_{x\to\infty} -\frac{\log(e^{-x/\theta})}{x} \quad = \lim_{x\to\infty} -\frac{-x/\theta}{x} = 1/\theta. \tag{2}$$

For **gamma distributed** X with shape parameter k and scale parameter θ:

$$\rho_{\text{Gamma}} = \lim_{x\to\infty} -\frac{\log \mathbb{P}(X > x)}{x} = \lim_{x\to\infty} -\frac{\log(1-\gamma(k,x/\theta)/\Gamma(k))}{x} \quad = \lim_{x\to\infty} -\frac{\log(\Gamma(k,x/\theta)/\Gamma(k))}{x} \overset{(a)}{=} \lim_{x\to\infty} -\frac{\log}{} \tag{3}$$

In (a), we used the asymptotic behavior of the gamma distribution

$$\lim_{x\to\infty} \frac{\Gamma(s,x)}{x^{s-1}e^{-x}} = 1.$$

For **normal distributed** X with mean μ and variance σ^2:
x = $1\frac{}{\sigma^2 \lim_{x\to\infty} \frac{x^2-2x\mu+\mu^2}{x}=\infty,}$where in (b), we used the inequality

$$\operatorname{erfc}(x) \geq \sqrt{\frac{2e}{\pi}} \frac{\sqrt{\beta-1}}{\beta} e^{-\beta x^2},$$

that holds for $x \geq 0$ and $\beta > 1$ (we used $\beta = 2$).

The asymptotic decay rates for the exponential, gamma, and normal distribution are given in (2), (3), and (4), respectively. We see that the decay rate ρ of the normal distribution is always maximal, *i.e.*, $\rho_{\text{Normal}} = \infty$. In contrast, the decay rates of the exponential and gamma distributions depend on the scale parameter θ.

By normalizing the mean ($\mathbb{E}[X] = 1$) by setting $\theta = 1$ and $k = 1/\theta$, we can compare the decay rates of the exponential distribution and gamma distribution: then, the decay rate of the exponential distribution is $\rho_{\text{Exponential}} = 1$ and of the gamma distribution is $\rho_{\text{Gamma}} = k$. That is, keeping the mean normalized, the gamma distribution decays slower than the exponential distribution with shape parameters $k < 1$, and faster than the exponential distribution with $k > 1$. With $k = 1$, the distributions coincide. When $k \to \infty$, the gamma distribution approaches the normal distribution and $\rho_{\text{Gamma}} \to \rho_{\text{Normal}} = \infty$.

References:

- Error function
- Exponential distribution
- Gamma distribution
- Normal distribution

6.9 Summing the signal powers

Let us consider two sinusoidal signals of opposite phases during the time $t \in [0, 1]$: $S_1(t) = \cos(2\pi t)$ and $S_2(t) = \cos(2\pi t + \pi)$. Obviously, the signals cancel each other completely, and the mean power of the additive signal $S_1 + S_2$ is just 0:

$$\mathbb{E}[(S_1 + S_2)^2] = \int_0^1 (\cos(2\pi t) + \cos(2\pi t + \pi))^2 dt = \int_0^1 0 dt = 0, \qquad (1)$$

where $\mathbb{E}[\cdot]$ is the mean (slightly abusing the notation of the probabilistic expected value).

We could try to add the powers of the signals separately together:

$$\mathbb{E}[S_1^2] + \mathbb{E}[S_2^2] = \int_0^1 \cos^2(2\pi t) dt + \int_0^1 \cos^2(2\pi t + \pi) dt \quad = \int_0^1 2\cos^2(2\pi t) dt = \int_0^1 \cos(4\pi t) dt + 1 = 1. \qquad (2)$$

But (2) significantly differs from (1)! Can we sometimes sum the individual signal powers together, which would be handy in many applications? This question has everything to do with the correlation of the signals.

Let us assume that S_1 and S_2 are two signals. We have that

$$\mathbb{E}[(S_1 + S_2)^2] = \mathbb{E}[S_1^2 + S_2^2 + 2S_1S_2] = \mathbb{E}[S_1^2] + \mathbb{E}[S_2^2] + 2\mathbb{E}[S_1S_2]. \qquad (3)$$

Hencex, the identity $\mathbb{E}[(S_1+S_2)^2] = \mathbb{E}[S_1^2]+\mathbb{E}[S_2^2]$ holds *if and only if* the cross-correlation $\mathbb{E}[S_1S_2] = 0$, *i.e.*, if the signals S_1 and S_2 are not correlated—this is not the case with our initial signals, as then the cross-correlation is given by

$$\mathbb{E}[S_1S_2] = \int_0^1 \cos(2\pi t)\cos(2\pi t+\pi)dt = \int_0^1 -\cos^2(2\pi t)dt = -\frac{1}{2}\int_0^1 \cos(4\pi t)dt - \frac{1}{2} = -\frac{1}{2},$$

as it should be if we tie together the equations (1), (2) and (3).

We used deterministic signals, but the same remarks apply to random signals. For example, for uniformly random phases $\phi_1, \phi_2 \in [0, 2\pi]$, let $S_1(t) = (2\pi t + \phi_1)$ and $S_2(t) = \cos(2\pi t + \phi_2)$;

$$\frac{1}{2\pi}\int_0^{2\pi}\int_0^1 (\cos(2\pi t+\phi_1))^2 dtd\phi_1 + \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 (\cos(2\pi t+\phi_2))^2 dtd\phi_2 + \frac{1}{2\pi}\int_0^{2\pi}\int_0^{2\pi}\int_0^1 \cos(2\pi t+\phi_1)\cos(2\pi t+\phi_2)dtd\phi_1 d\phi_2 \quad = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1$$

The expected powers can be summed, as the expectation of the cross-correlation $\mathbb{E}[S_1S_2]$ is 0 for the two random signals S_1 and S_2, *i.e.*, $\mathbb{E}_{\phi_1,\phi_2}[\mathbb{E}[S_1S_2]] = 0$.

References:

- Maol-taulukot

6.10 A representation of the generalized hypergeometric function ${}_3F_2$

Here is a representation of the hypergeometric function ${}_3F_2(1, 1, b; 2, 2; \cdot)$ in terms of the polylogarithm: For $|x| < 1$ and $b \in \mathbb{N}$, the *hypergeometric series* representation is given by

$$;x) = \sum_{n=0}^{\infty} \frac{(1)_n(1)_n(1+b)_n}{(2)_n(2)_n}\frac{x^n}{n!} \quad = \sum_{n=0}^{\infty} \frac{(1+b)_n}{(n+1)^2 n!}x^n = \frac{1}{b!}\sum_{n=0}^{\infty}\frac{(n+1)_b}{(n+1)^2}x^n \overset{(a)}{=} \frac{1}{b!}\sum_{n=0}^{\infty}\frac{\sum_{k=1}^{b}\begin{bmatrix} b \\ k \end{bmatrix}(n+1)^k}{(n+1)^2}x^n \quad = \frac{1}{b!}\sum_{k=1}^{b}\begin{bmatrix} b \\ k \end{bmatrix}\sum_{n=0}^{\infty}\frac{x^n}{(n+1)^{2-k}} \overset{(b)}{=} \frac{1}{b!}$$

where $\begin{bmatrix} b \\ k \end{bmatrix}$ is the unsigned Stirling number of the first kind. In (a), we used the expansion of the rising Pochhammer factorial and in (b), we used the definition of the polylogarithm. Furthermore, the $x \in \mathbb{C}$ follows from the analytic continuation of the polylogarithm.