ECE537: Lab 4 Report

In the first part of the lab, we will be simulating low pass Guassian, and then we will use the spectrual analysis to have a better understanding of it.

Throughout this lab, the Distributions.jl package in Julia has been utilized to be able to use the probability constructs in code.

```
using Distributions, StatsBase, StatsPlots, LinearAlgebra, LaTeXStrings, PlutoUI, DSP
, Plots
```

```
PlotlyBackend()
```

1. Generating Low-Pass Random Processes

Given an infinite sequence of i.i.d. Gaussian random variables, $\{X_k\}$, we can construct a bandlimited White Gaussian Noise (WGN) process by the Shannon-Nyquist sampling theorem.

$$X(t) = \sum_{-\infty}^{\infty} X_k \text{sinc}\left(\frac{t-kT}{T}\right),$$

where $X_k = X(kT)$. and for this lab, we choose $X_k \sim N(0,1)$ and the summation above matches the convolution $\sum_k X(kT)\delta(t-kT) * 2Bsinc(2Bt)$ for $B = 1/2T$, implying that $X(t)$ is a bandlimited (low-pass filtered) white noise signal with bandwidth $|f| < B$.

To numerically approximate,

$$X(t) \approx \sum_{k_t-m}^{k_t+m} X_k \text{sinc}\left(\frac{t-kT}{T}\right),$$

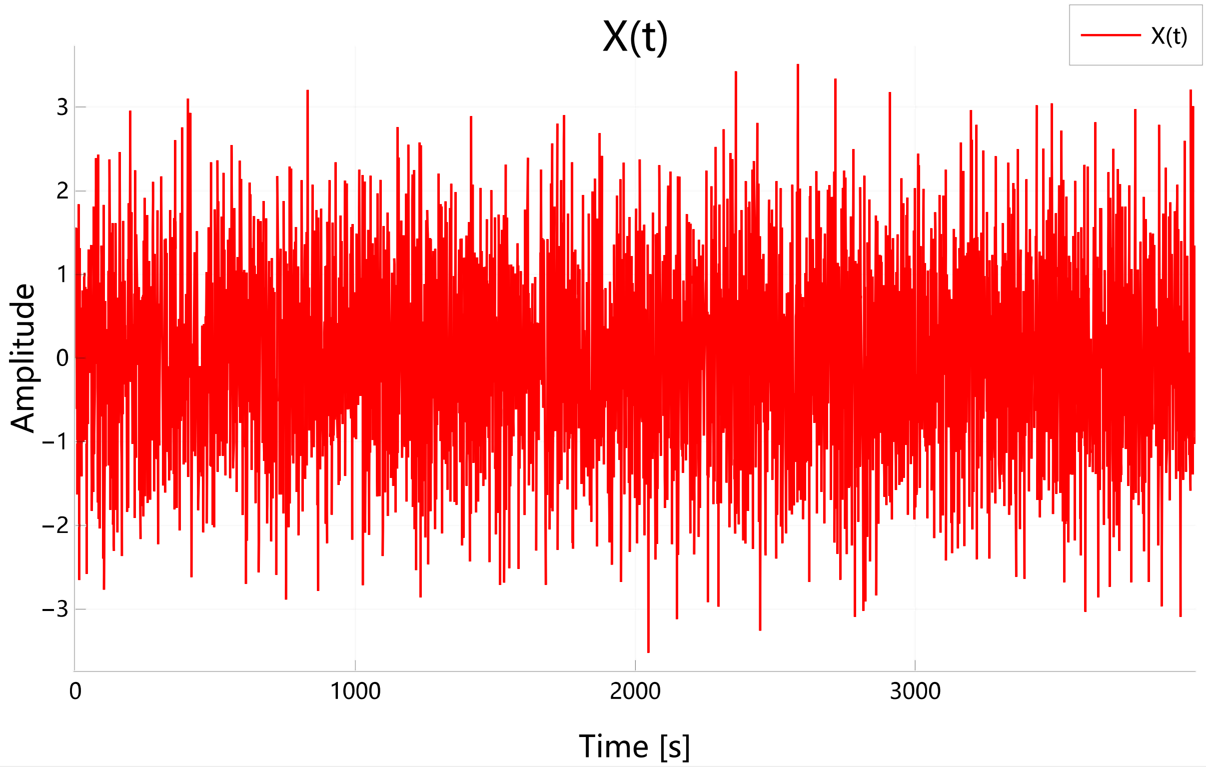
where $k_t = \lfloor t/T \rfloor$. For the purpose of this lab, we choose the approximation limit $m = 5$ and sampling time $T = 1$. and we assume that $X(t) = 0$ for $t < 0$

```
X_t (generic function with 1 method)
```

```julia
begin
    # basic parameters
    mu = 0;
    sigma = 1;
    m = 5;
    T = 1;
    deltaT = T / m;

    # some useful functions to calculate the X(t)
    # calculate the X_k
    X_k(n) = [Normal(mu, sigma) for k = 1:1:n];

    # calculate the k_t - m to k_t + m
    k_t(t) = floor(Int, t / T);
    k_t(t, m) = max(k_t(t) + m, 1);

    # calculate the sum of target values
    X_t(X_k, t, T, m) = sum([X_k[k] * sinc(t / T - k) for k = k_t(t, -m):k_t(t, m)]);
end
```

```
[0.0, 0.188287, 0.447049, 0.746496, 1.04299, 1.28742, 1.49414, 1.55781, 1.4545, 1.19716, 0
```

```julia
begin
    # generate the X_k array and define the true function
    Xk   = rand.(X_k(20_000));
    Xt(t) = X_t(Xk, t, T, m);

    # generate the t series and calculate the corresponding function value
    ts   = 0:deltaT:(4000 * T);
    Xts = Xt.(ts)
end
```




```
begin
    plot(ts, Xts, label="X(t)", xlabel="Time [s]", ylabel="Amplitude", title =
    "X(t)", color="red");
    plot!(; xlims=(ts[1],ts[end]));
end
```

2. LTI Systems and Random Processes

Now, we consider a a linear time-invariant (LTI) filter with impulse response $h(t) = e^{-at}(u(t) - u(t-20))$, i.e., a truncated RC low-pass filter with parameter a that is cut to zero for $t > 20$ seconds. The input to the system will be the bandlimited WGN process, $X(t)$, produced above and the output random process, $Y(t)$, is a modified or an approximate Ornstein-Uhlenbeck process due to the time-windowed $h(t)$.

From analysis of deterministic signals and systems, we know that the output random process $Y(t) = X(t) * h(t)$ which is the continuous-time convolution of the input random process and the impulse response. For computational purposes, we can approximate it as a Riemann integral,

$$Y(t) \approx \sum_{k=-\infty}^{\infty} X(t-\tau)e^{-a\tau}(u(\tau) - u(\tau-20))d\tau,$$

where for the purpose numerical approximation of $X(t)$, following section 1, we take $\tau_t(k) = (t - \lfloor t/T \rfloor) + k\Delta t$ for which $d\tau_t(k) = \Delta t$. Also note that since h(t) is truncated, the maximum range of the samples will correspond to $20/\Delta t$ and will only be non-zero for $k > 0$. Thus, we get,

$$Y(t) \approx \sum_{k=0}^{20/\Delta t} X(\lfloor t/T \rfloor - k\Delta t)e^{-a((t-\lfloor t/T \rfloor)+k\Delta t)}\Delta t.$$

We implement the process $Y(t)$ in the code below and plot it for different values of the parameter a.

```
Y_t (generic function with 1 method)
```

```julia
begin
    # define some useful parameters
    alpha1 = 0;
    alpha2 = 0.2;

    # define some related functions
    t_t(t, k) = (t - k_t(t)) + k * deltaT;
    Y(t, X_k, a, T, m) = sum([X_t(t - t_t(t, k), X_k, T, m) * exp(-a * t_t(t, k))
    for k = 0:(20/deltaT)]) * deltaT;
    Y_t(t, a) = Y(t, Xk, a, T, m);
end
```

MethodError: no method matching floor(::Type{Int64}, ::Vector{Float64})

```
Closest candidates are:
floor(::Type{T}, !Matched::Missing) where T at missing.jl:154
floor(::Type{T}, !Matched::Rational) where T at rational.jl:458
floor(::Type{T}, !Matched::BigFloat) where T<:Union{Signed, Unsigned} at mpfr.jl:318
...
```

1. **k_t**(::Vector{Float64}) @ *Other: 14*
2. **k_t**(::Vector{Float64}, ::Int64) @ *Other: 15*
3. **X_t**(::Float64, ::Vector{Float64}, ::Int64, ::Int64) @ *Other: 18*
4. (::Main.var"workspace#45".var"#1#2"{Float64, Vector{Float64}, Int64, Int64, Int64})(::Float64) @ *none:0*
5. **iterate** @ *generator.jl:47* [inlined]
6. **collect**(::Base.Generator{StepRangeLen{Float64, Base.TwicePrecision{Float64}, Base.TwicePrecision{Float64}, Int64}, Main.var"workspace#45".var"#1#2"{Float64, Vector{Float64}, Int64, Int64, Int64}}) @ *array.jl:787*
7. **Y**(::Float64, ::Vector{Float64}, ::Int64, ::Int64, ::Int64) @ *Other: 8*
8. **Y_t**(::Float64, ::Int64) @ *Other: 9*
9. **Yt₁**(::Float64) @ *Local: 2*
10. **_broadcast_getindex_evalf** @ *broadcast.jl:670* [inlined]
11. **_broadcast_getindex** @ *broadcast.jl:643* [inlined]
12. **getindex** @ *broadcast.jl:597* [inlined]
13. **copy** @ *broadcast.jl:899* [inlined]
14. **materialize**(::Base.Broadcast.Broadcasted{Base.Broadcast.DefaultArrayStyle{1}, Nothing, typeof(Main.var"workspace#45".Yt₁), Tuple{StepRangeLen{Float64, Base.TwicePrecision{Float64}, Base.TwicePrecision{Float64}, Int64}}}) @ *broadcast.jl:860*
15. **top-level scope** @ *Local: 3*

```julia
begin
    Yt₁(t) = Y_t(t, alpha1);
    plot(ts, Yt₁.(ts); legend=false)
    xlabel!("Time [s]");
    ylabel!("Amplitude");
    title!(L"Y(t; a=0)");
    plot!(; xlims=(ts[1],ts[end]))
end
```

3. Power Spectral Density of Random Processes

For LTI systems, we have the following statistical result,

$$S_Y(f) = S_X(f)|H(f)|^2,$$

where $S(f)$ is the power spectral density (PSD) of the respective signals and $H(f)$ is the Fourier transform of the impulse response. Recall that, in section 1, we have treated X_k as samples of $X(t)$ every T second intervals. For a bandlimited WGN process we have the result that the samples $X_k \sim \mathcal{N}(0, N_0 B)$ where $N_0 = 2$ is the net power of the signal $X(t)$ and $B = \frac{1}{2}$ is its bandwidth. Therefore, we have $S_X(f) = 1 = \frac{N_0}{2}$ for $|f| < \frac{1}{2}$. Furthermore, we have $H(f) = \frac{1-e^{-20(a+j2\pi f)}}{a+j2\pi f}$, and to get the squared magnitude of H, we multiply it with its conjugate H^*. Finally, we have,

$$|H(f)|^2 = \frac{1 + e^{-20a}(e^2 - 2\cos(40\pi f))}{a^2 + 4\pi^2 f^2}$$

We expect $S_Y(f) = \frac{N_0}{2}|H(f)|^2 = |H(f)|^2$, for $|f| < \frac{1}{2}$ and zero elsewhere. Thus, the theoretical PSDs of $Y(t)$ for filters with different values of parameter a are,

$$S_Y(f) = \begin{cases} \dfrac{1 + (e^2 - 2\cos(40\pi f))}{4\pi^2 f^2}, & a = 0 \\ \dfrac{1 + e^{-4}(e^2 - 2\cos(40\pi f))}{0.04 + 4\pi^2 f^2}, & a = 0.2 \end{cases}$$

Note that for for $a = 0$, S_Y is unbounded near $f = 0$, whereas for $a = 0.2$ it only reaches a maximum of ≈ 27.46 at 0 Hz. We have created a function to get the theoretical PSD for $Y(t)$ below.