a7_Q4_highlowencoding

May 3, 2021

1 Linear Systems and Noise Assigment 7 - Signals in noise

The following should translate fairly smoothly to Python, Matlab, etc. `else, question |> me!`

2 Random bitstream: high-low encoding

[18]:
```
using FFTW, Plots, LaTeXStrings, Plots.Measures
gr(grid=false,legend=false,size=(650,300));
```

[3]:
```
MHz = 1e6 # time unit for bits
T = 1/MHz # length of one bit in seconds.
Nt = 2^12 # fine grid for sampling
N = 32     # number of t grid points in one bit
dt = T/N  # time grid point spacing
t = (0:(Nt-1))*dt # total sampling time
Ns = 1000 # number of realizeations of the bitstream to average
```

[3]:
```
1000
```

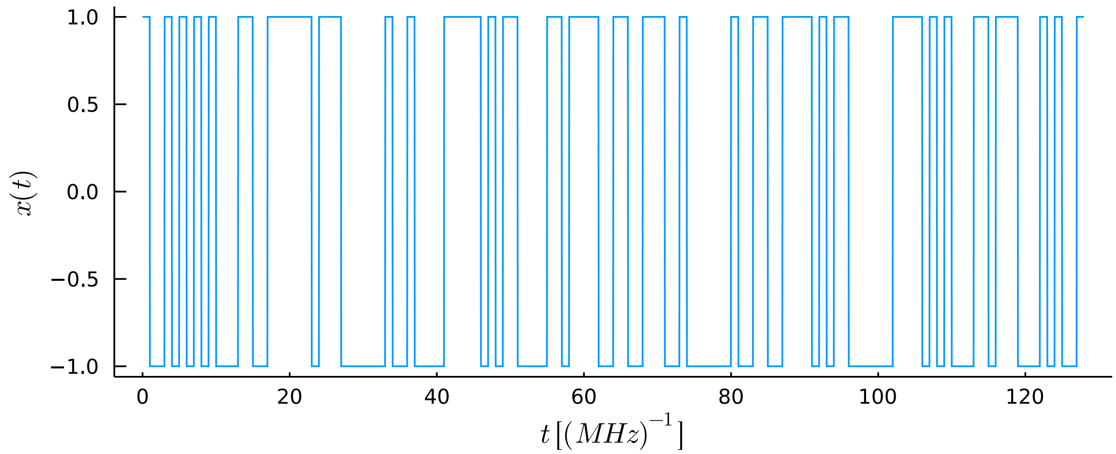
2.1 Plot one realization

Make a single random bitstream over $[0, N_t dt]$, broken into N_t/N bit intervals:

[4]:
```
x = zero.(t) # create empty vector to hold each realization
Nbits = Nt÷N # integer division
for k in 0:Nbits-1
    bit = N*k+1:N*k+N # bit location in x(t)
    x[bit] = rand() > 0.5 ? ones(N) : -ones(N)  #flip a coin (? : is shorthand␣
 ↪for if else)
end
Phi = abs2.(dt*fft(x))/(Nt*dt); # PSD for one realization.
# Two dt's here: first because we are approximating integral using DFT.
# Second to normalize correctly for average power
```

[20]:
```
plot(t*MHz,x);xlabel!(L"t \;[ (MHz)^{-1} ]",bottom_margin=1cm);ylabel!(L"x(t)")
```

[20]:



Average over N_s realizations of the bitstream:

[21]:
```
x = zero.(t) # zero vector to hold each realization
Phi = zero.(t) # zero vector to accumulate average power-spectral density
for i in 1:Ns
    # make a random bitstream over [0,Nt*dt], broken into Nt/N bit intervals
    Nbits = Nt÷N
    for k in 0:Nbits-1
        bit = N*k+1:N*k+N # bit location in x(t)
        x[bit] = rand() > 0.5 ? ones(N) : -ones(N)  #flip a coin
    end
    Phi .+= abs2.(dt*fft(x))/(Nt*dt); # Accumulate PSD for Ns realizations
end
Phi = fftshift(Phi/Ns);  # normalize and shift
```

[22]:
```
nu = (-Nt÷2:Nt÷2-1)/Nt/dt;
```

[23]:
```
Phi_a(nu)=T*(sinc(nu*T))^2 # Analytical PSD of bitstream, after averaging over␣
↪random bits
```

[23]:
```
Phi_a (generic function with 1 method)
```

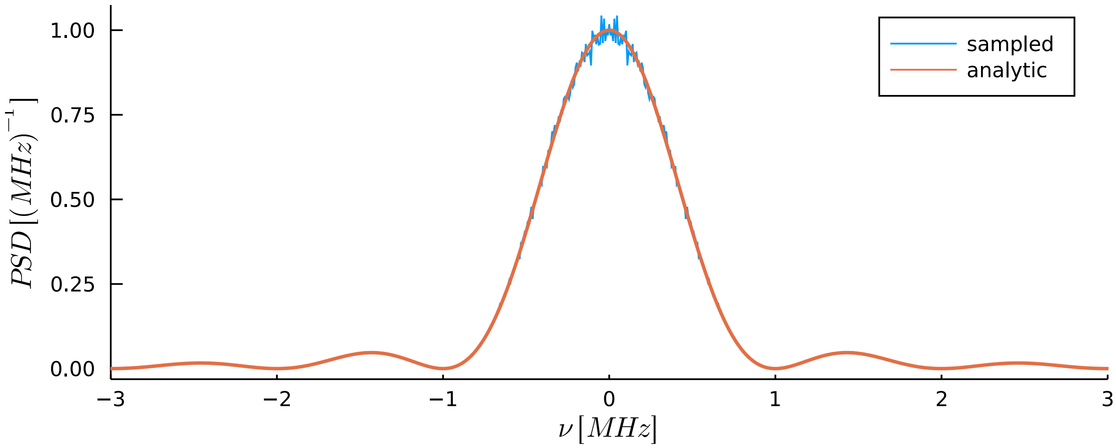
Let's work this out. From the hint in the assignment, if we consider a finite time interval (since the signal only has finite average power, and not finite energy)

[24]:
```
Phith = Phi_a.(nu);
```

[25]:
```
plot(nu/MHz,MHz*Phi,label="sampled",legend=:topright)
plot!(nu/MHz,MHz*Phith,label="analytic",lw=2)
```

```
xlims!(-3,3);ylabel!(L"PSD\; [(MHz)^{-1}]");xlabel!(L"\nu\; [
↪MHz]",bottom_margin=1cm)
```

[25]:



Can be made smoother with more samples, but the essential point is clear: the PSD has its peak power at zero frequency.