Julia Admittance

Lingling Fan

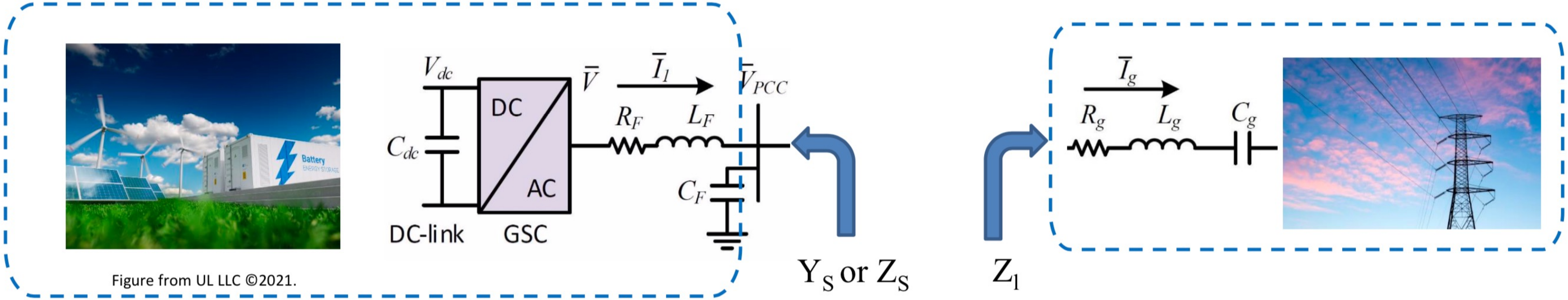
University of South Florida

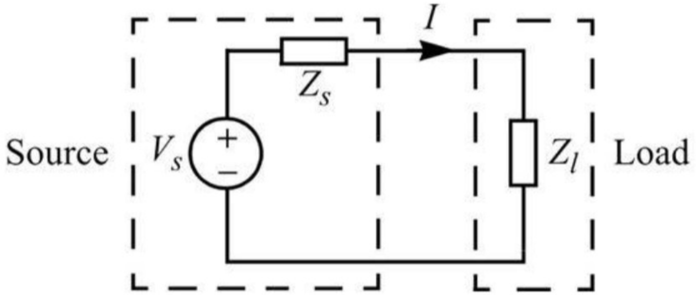
June 29, 2021

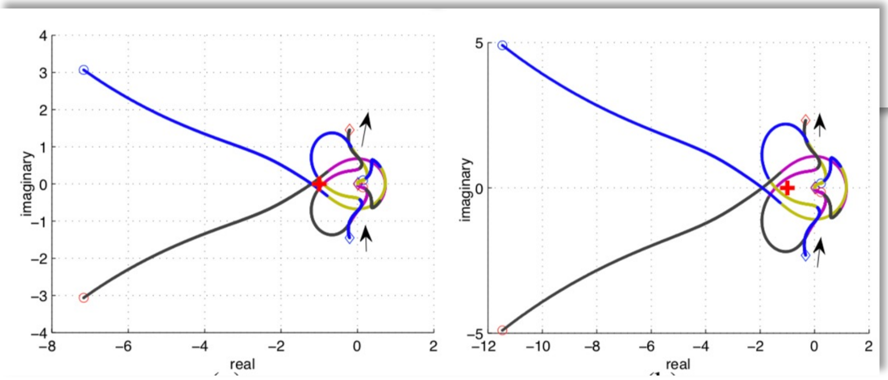
Http://power.eng.usf.edu

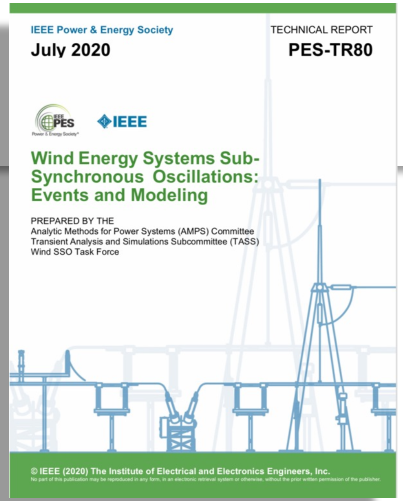
This video is for Juliacon 2021.

Motivation: stability analysis via admittance/impedance









Frequency-domain stability check by use of $Y_{souce}Z_{load}$ **or its eigenvalues**

RLC oscillations and a 2nd-order ODE

- A second-order ordinary differential equation has a pair of complex conjugate roots.

$$Ri + L\frac{di}{dt} + v_c = v_s$$

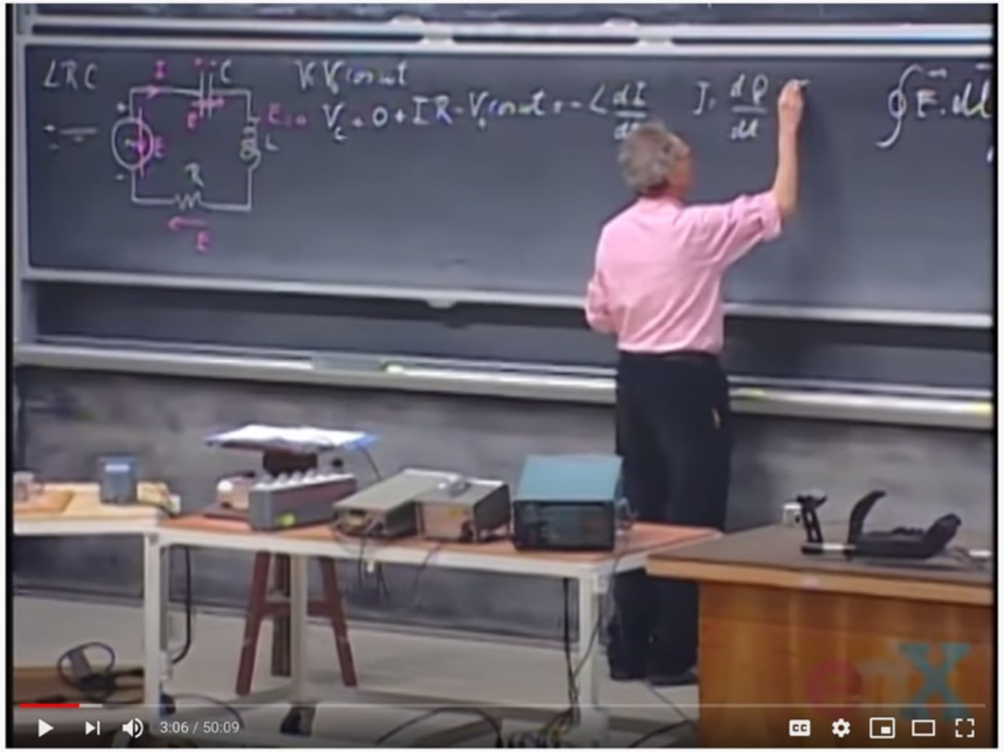
$$C\frac{dv_c}{dt} = i$$

Z(s)

$$\left(R + Ls + \frac{1}{Cs}\right) i = v_s$$

$$s^2 + \frac{R}{L}s + \frac{1}{LC} = 0$$

$$s = \sigma + j\omega \approx \frac{-R}{2L} \pm j\sqrt{\frac{1}{LC}}$$



8.02x - Lect 25 - Driven LRC Circuits. Metal Detectors

Prof. Walter Lewin's lecture: MIT 8.02.

Out[12]:

$$Ri + L\frac{di}{dt} + v_c = v_s$$

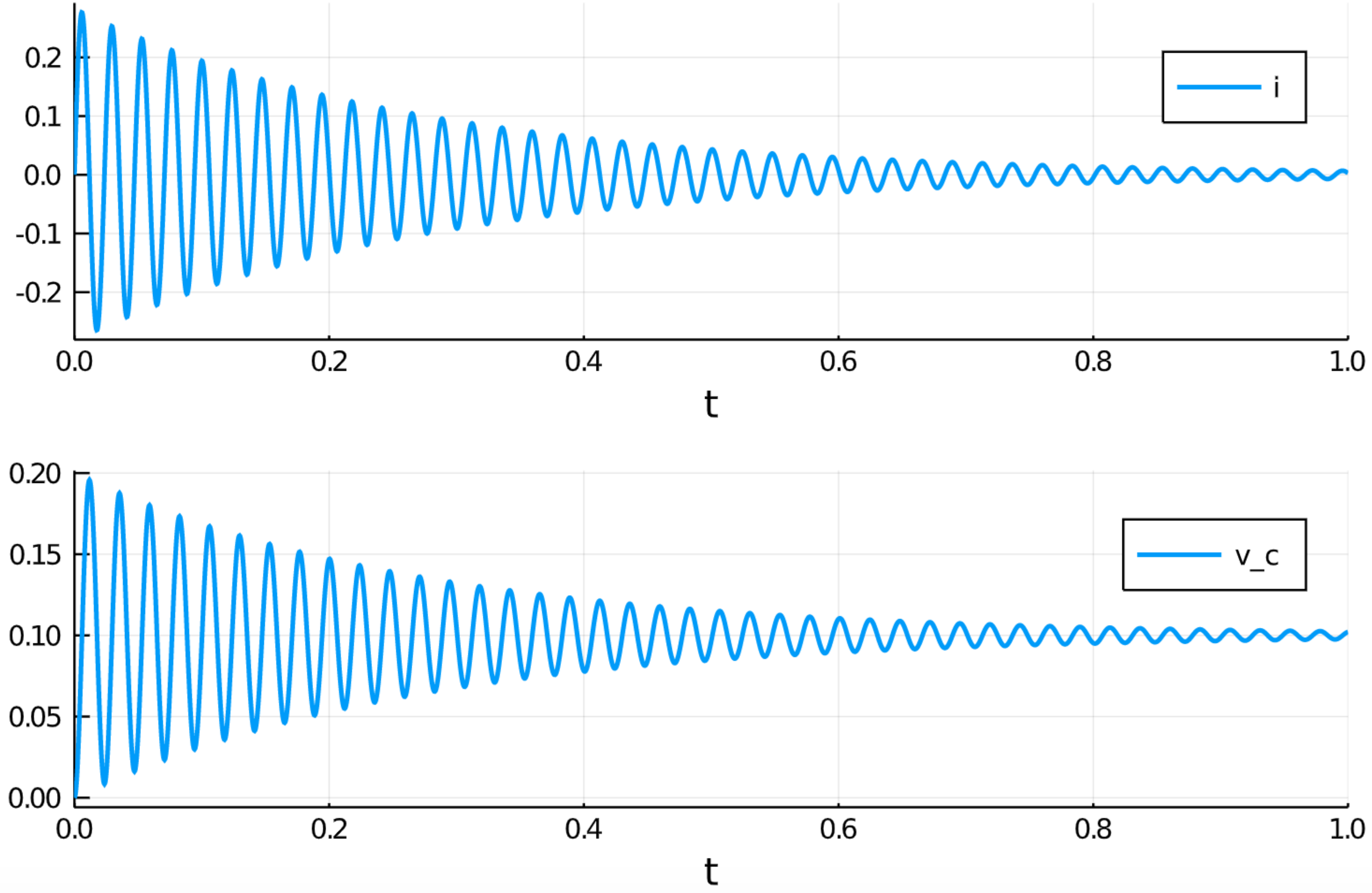
$$C\frac{dv_c}{dt} = i$$

In [14]:

```
using DifferentialEquations
using Plots

R = 0.01; L = 0.5/377.0; C = 1/(0.25*377.0);
v0 = 0.1;                                   # voltage drop across the RLC circuit

function dfun(du,u,p,t)
    i, vc = u
    du[1] = 1/L*(v0-R*i - vc)
    du[2] = 1/C*i
end
u₀ = [0, 0]                                 # initial state vector
tspan = (0.0,1.0)                           # time interval

prob = ODEProblem(dfun,u₀,tspan)
sol = solve(prob)

plot(sol,linewidth=2,xaxis="t",label=["i" "v_c"],layout=(2,1))
```

About 40 Hz oscillations

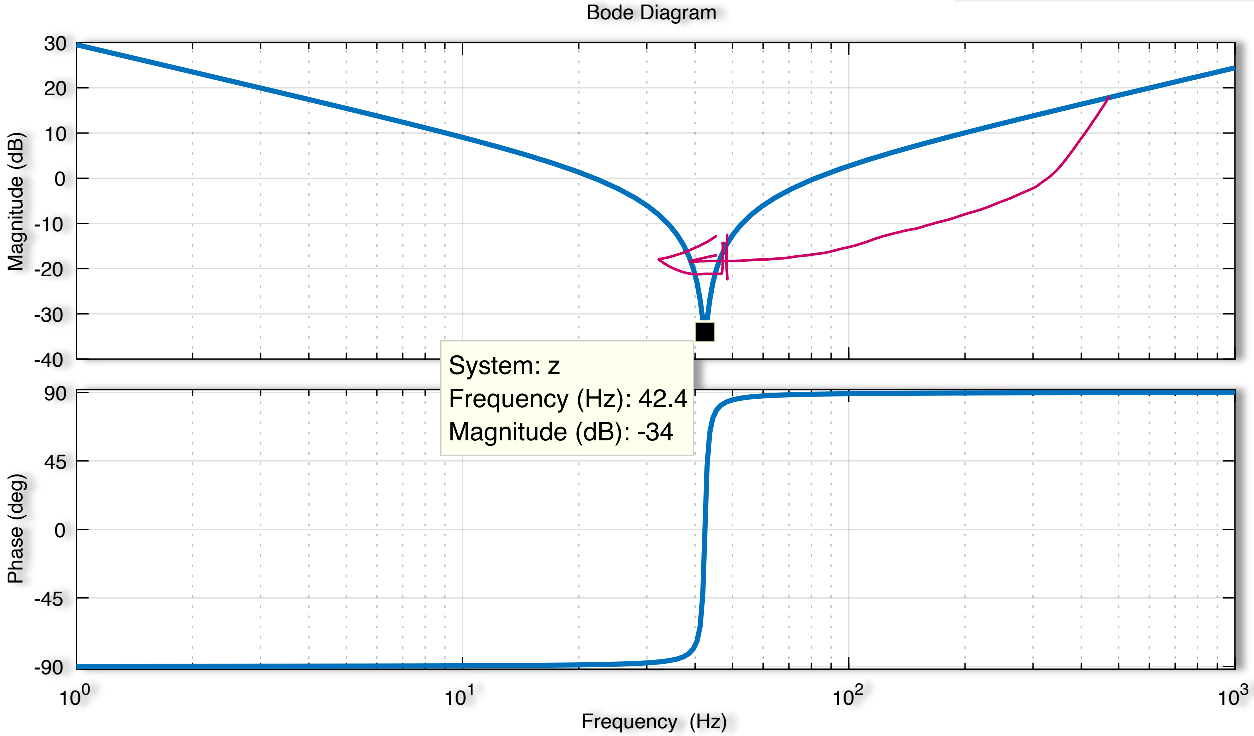
Out[14]:

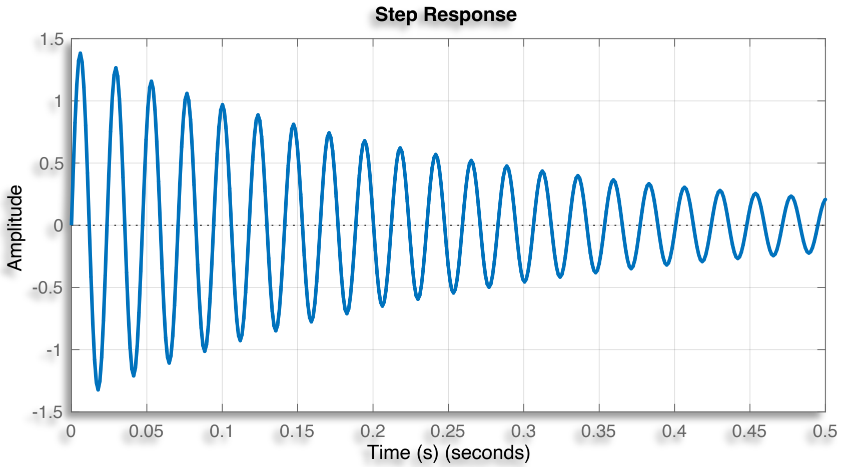


Bode diagrams

Impedance of the circuit: Z = R + Ls + 1/(Cs)

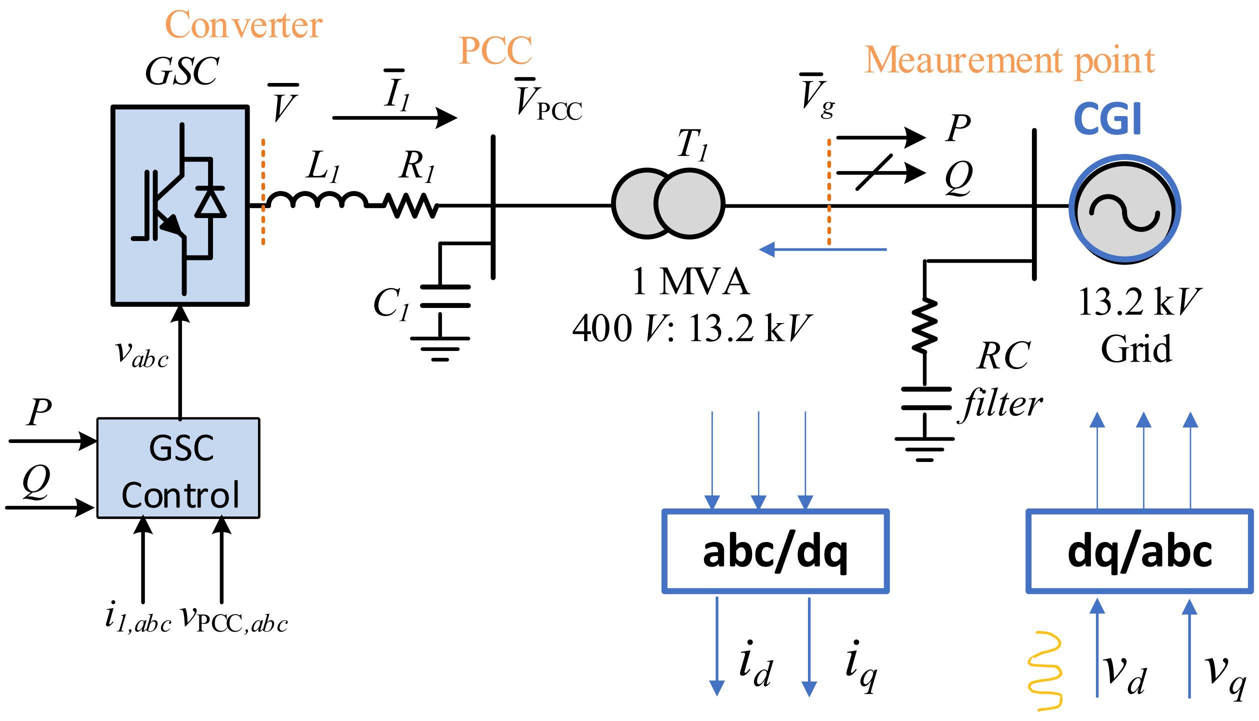
$$\omega_{\mathrm{LC}} = \sqrt{\frac{1}{LC}} = \omega_0\sqrt{\frac{1}{\omega_0 L \omega_0 C}} = \omega_0\sqrt{\frac{X_C}{X_L}} = \sqrt{\text{Compensation Level}} \times \omega_0$$





The above frequency-domain responses (a scalar impedance) can be measured.
How? → Frequency scan or harmonic injection method.
Next page shows an example of a more sophisticated matrix admittance.

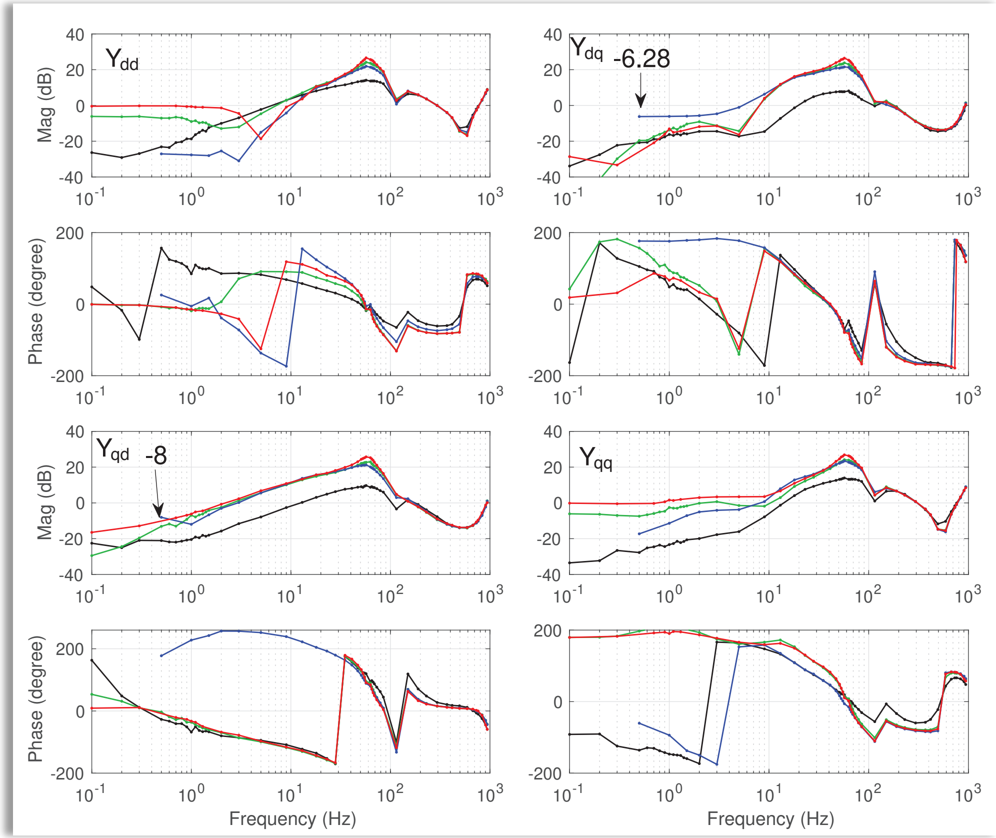
Example: A 2.3-MVA battery inverter admittance measuring test bed



CGI: Controllable Grid Interface

DQ-domain: grid voltage at 1 pu 0.

$$\begin{bmatrix}\bar{I}_d\\ \bar{I}_q\end{bmatrix} = \underbrace{\begin{bmatrix}Y_{dd}(j\omega) & Y_{dq}(j\omega)\\ Y_{qd}(j\omega) & Y_{qq}(j\omega)\end{bmatrix}}_{Y^{m}_{dq}(j\omega)} \begin{bmatrix}\bar{V}_d\\ \bar{V}_q\end{bmatrix}$$

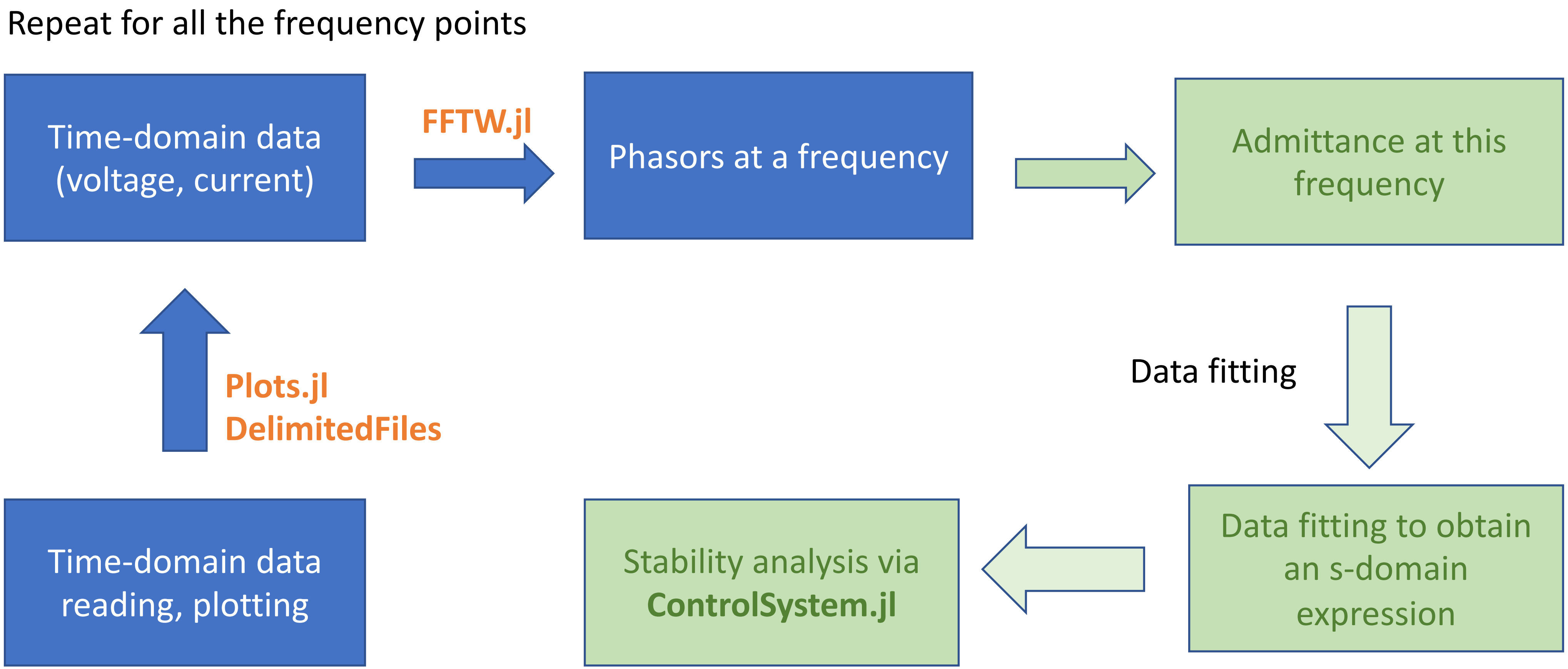


Case 1: P=0 kW, Q=0kVAr Case 2: P =500 kW, Q=0 kVAr

Case 3: P =0 kW, Q=500 kVAr Case 4: P =1000 kW, Q=0 kVAr

Lingling Fan, Zhixin Miao, Przemyslaw Koralewicz, Shahil Shah, and Vahan Gevorgian, **"Identifying DQ-Domain Admittance Models of a 2.3-MVA Commercial Grid-Following Inverter Via Frequency-Domain and Time-Domain Data,"** *IEEE TEC 2020*. pdf

Frequency-domain admittance measuring steps



Data: 3 buses/ 39 frequency points

Total size of data: 1.4 GB

- Bus4_id_freq_0.3_Hz
- Bus4_id_freq_0.7_Hz
- Bus4_id_freq_1_Hz
- Bus4_id_freq_1.5_Hz
- Bus4_id_freq_2_Hz
- Bus4_id_freq_3_Hz
- Bus4_id_freq_5_Hz
- Bus4_id_freq_9_Hz
- Bus4_id_freq_13_Hz
- Bus4_id_freq_18_Hz
- Bus4_id_freq_23_Hz
- Bus4_id_freq_28_Hz
- Bus4_id_freq_35_Hz
- Bus4_id_freq_41_Hz
- Bus4_id_freq_47_Hz
- Bus4_id_freq_51_Hz
- Bus4_id_freq_55_Hz
- Bus4_id_freq_56_Hz
- Bus4_id_freq_58_Hz
- Bus4_id_freq_59_Hz
- Bus4_id_freq_63_Hz
- Bus4_id_freq_67_Hz
- Bus4_id_freq_79_Hz
- Bus4_id_freq_85_Hz
- Bus4_id_freq_115_Hz
- Bus4_id_freq_150_Hz
- Bus4_id_freq_190_Hz
- Bus4_id_freq_230_Hz
- Bus4_id_freq_295_Hz
- Bus4_id_freq_370_Hz
- Bus4_id_freq_417_Hz
- Bus4_id_freq_494_Hz

Bus4_id_freq_0.3_Hz

View | Group | Action | Share | Edit Tags | Search

Documents > 01_2021 > Bus4_id_freq_0.3_Hz

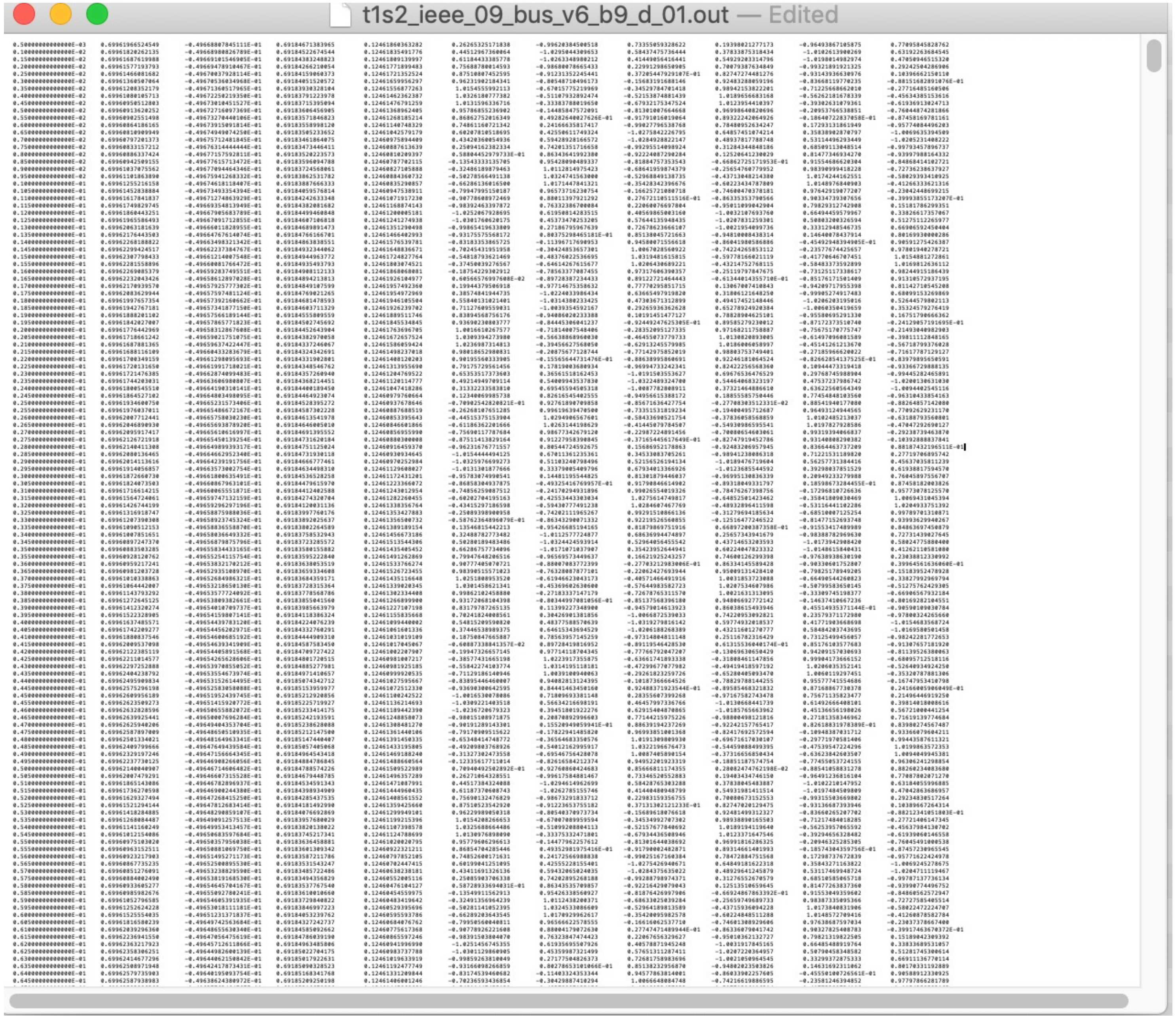
Name	Date Modified	Size	Kind
t1s2_ieee_09_bus_v6_gen_01.out	Dec 12, 2020 at 5:08 AM	2.7 MB	Document
t1s2_ieee_09_bus		2.7 MB	Document
t1s2_ieee_09_bus_v6_gen_03.out	Dec 12, 2020 at 5:08 AM	2.7 MB	Document
t1s2_ieee_09_bus_v6_gen_04.out	Dec 12, 2020 at 5:08 AM	2.7 MB	Document
t1s2_ieee_09_bus_v6_gen_05.out	Dec 12, 2020 at 5:08 AM	1.5 MB	Document
t1s2_ieee_09_bus_v6_gen.inf	Dec 12, 2020 at 5:06 AM	4 KB	Document
T4_5.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T4_6.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T5_7.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T6_9.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T7_8_2.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T7_8_3.out	Dec 12, 2020 at 5:06 AM	6 KB	Document
T8_9.out	Dec 12, 2020 at 5:06 AM	6 KB	Document

Show items as icons, in a list, in columns, or in a gallery

A data file

This package is used to read the files:

DelimitedFiles



Data structure

- Create a data structure to save all time-domain data for each event, each frequency measurement point, 6 different channels

```
begin
using DelimitedFiles
    n_totalFreq = length(Hz_string)
    Data_total_allfreq_allevents =[[],[],[],[],[],[]];
    for i_event in 1:2
    filenames =list_filenames[i_event]
        #Data_total_allfreq =[];
    #for i in 1:1
        for k in 1:n_totalFreq
            data_many =readdlm.(filenames[:,k],  header=false)
            Data_total = combine_datafiles(data_many,N_files);
            push!(Data_total_allfreq_allevents[i_event], Data_total)
            #push!(Data_total_allfreq_allevents, Data_total_allfreq)
        end #k for all freq
            #push!(Data_total_allfreq, combine_datafiles(readdlm.(filenames[:,k],
header=false),N_files))
        #end
    end # fro all events
        # for every frequency, conduct FFT for 12 signals.
        #for i in 1:n_totalFreq
end
```

```
#begin
    for i_event in 3:4
    filenames =list_filenames[i_event]
        for k in 1:n_totalFreq
            data_many =readdlm.(filenames[:,k],  header=false)
            Data_total = combine_datafiles(data_many,N_files);
            push!(Data_total_allfreq_allevents[i_event], Data_total)
        end #k for all freq
    end # fro all events
#end
```

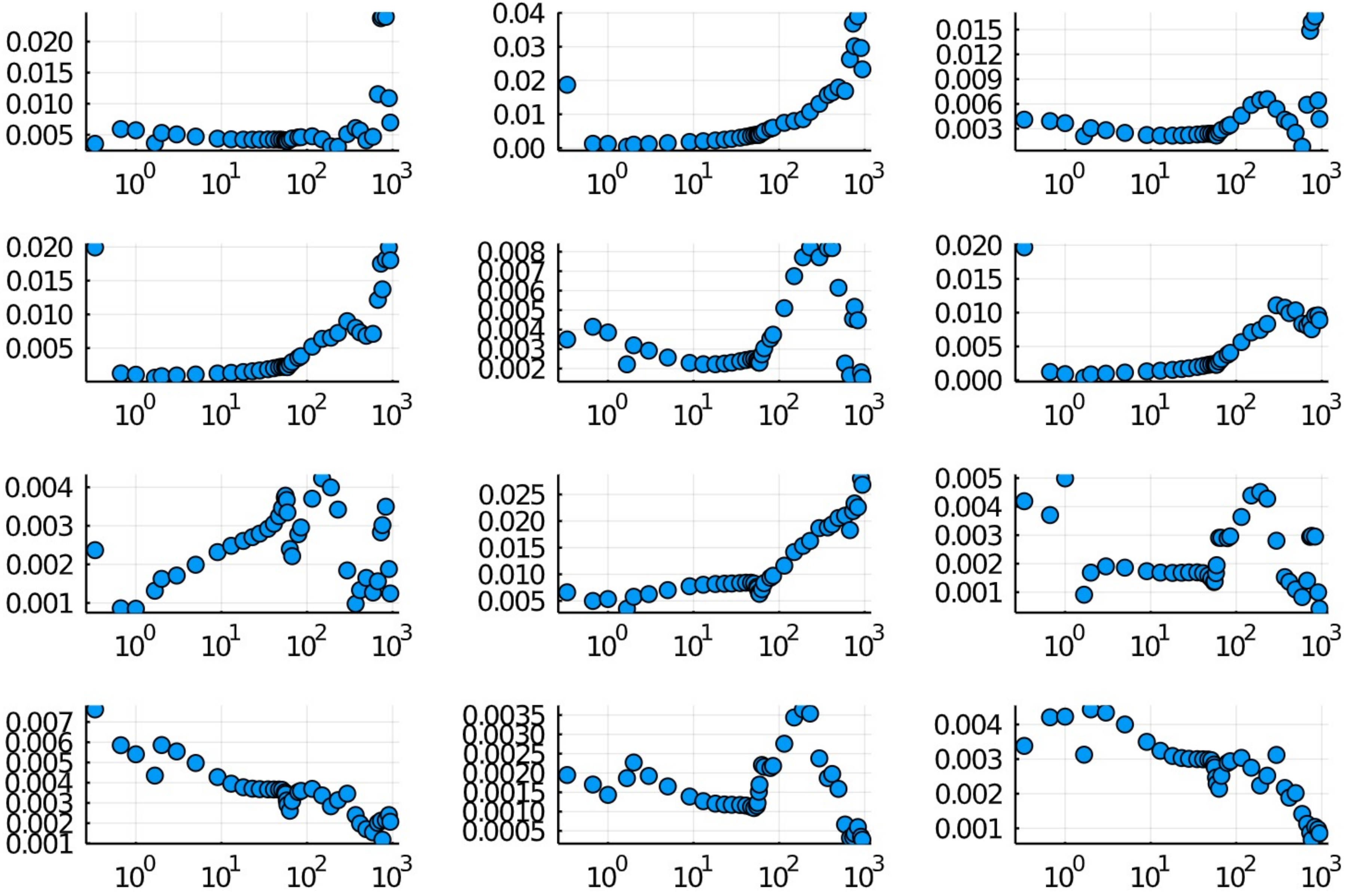
FFT function to read in time-series data of a single channel and find phasors

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

```julia
# from the input time and sinusoidal signals, take a time period, and output frequency and phasors.
function myFFT(tt, input_signal, t_start, t_end)
    dt = tt[2]-tt[1];
    N = Int(floor((t_end-t_start)/dt));

    N_start = Int(floor(t_start/dt))+1;
    N_end   = Int(floor(t_start/dt))+N;

    t = (t_start+dt):dt:(t_start+N*dt);

    nn_start = Int(floor(0.95/dt));
    # take of the initial values
    signal = input_signal[N_start:N_end]- input_signal[nn_start]*ones(length(t));
    # Fourier Transform of it
    F = fft(signal)/length(t)|>fftshift ;
    time_domain = plot(t, signal, title = "Signal");
    freqs = fftfreq(length(t), 1.0/dt) |>fftshift;
    freq_domain = plot(freqs, abs.(F), title = "Spectrum",xlim=(0, +100))
    #plot(time_domain, freq_domain, layout = (2,1))
    # find the index of max of F
    ind_pos =findall(x->x>0.0, freqs);
    F_plus = F[ind_pos];
    ind_max=findall(x->abs(x)==maximum(abs.(F_plus)), F_plus);
    return freqs[ind_pos][ind_max][1], F_plus[ind_max][1],time_domain, freq_domain
end
```

Call FFT to find phasors for each event, each frequency measurement

```
begin
    for i_event in 1:2
        for i in 1:39
            data = Data_total_allfreq_allevents[i_event][i]
            for j in 1:length(ind_channels)
               ch = ind_channels[j]+1;
                if(i<2)
               (FFT_freq, FFT_phasor) = myFFT(data[:,1], data[:,ch], 2.0, 5.0);
                else
                    a = max(0.35, parse(Float64,Hz_string[i])*0.9)
                (FFT_freq, FFT_phasor) = myFFT1(data[:,1], data[:,ch], 2.0, 5.0, a);
                end

               FFT_output1[i_event][j, i]= FFT_freq
               FFT_output2[i_event][j, i]= FFT_phasor
            end
        end
    end
end
```




```
scatter(FFT_output1[2][1,:], transpose(abs.(FFT_output2[2])),xaxis=:log,layout=
(4,3), leg= false)
```

Concluding remarks

- Julia is great to deal with data intensive applications.
- Several packages, DelimitedFiles, FFTW, are pretty handy.