LAB: Bearing Fault Monitoring and RUL Estimation

Name: HanMinung

Due in 1 week

Instruction

In this LAB, you are required to create a simple code for Fault Monitoring and RUL estimation

You can refer to RUL estimation tutorial codes and related papers

Fault Monitoring

- Detect Fault Status
- Display WARNING MESSAGE

RUL Estimation

- Estimate RUL similar to Tutorial codes
- The end of Life can be assumed when the repair is done

Dataset

Raw Dataset of Bearing Velocity measured that shows a Bearing Life cycle before the repair is done. Also the velocity measurement after the repair is provided.

Download bearing dataset for this lab: [Download here]

Using the given dataset, develop simple program for

- Bearing velocity measured 24 times per day
- Each measurement is L=8192 with Fs=2560Hz
- Measurement for consecutive days until repair is done
- 24 times per day & 87 days : about 2104 columns

First visualize the vibration signals in the time domain.

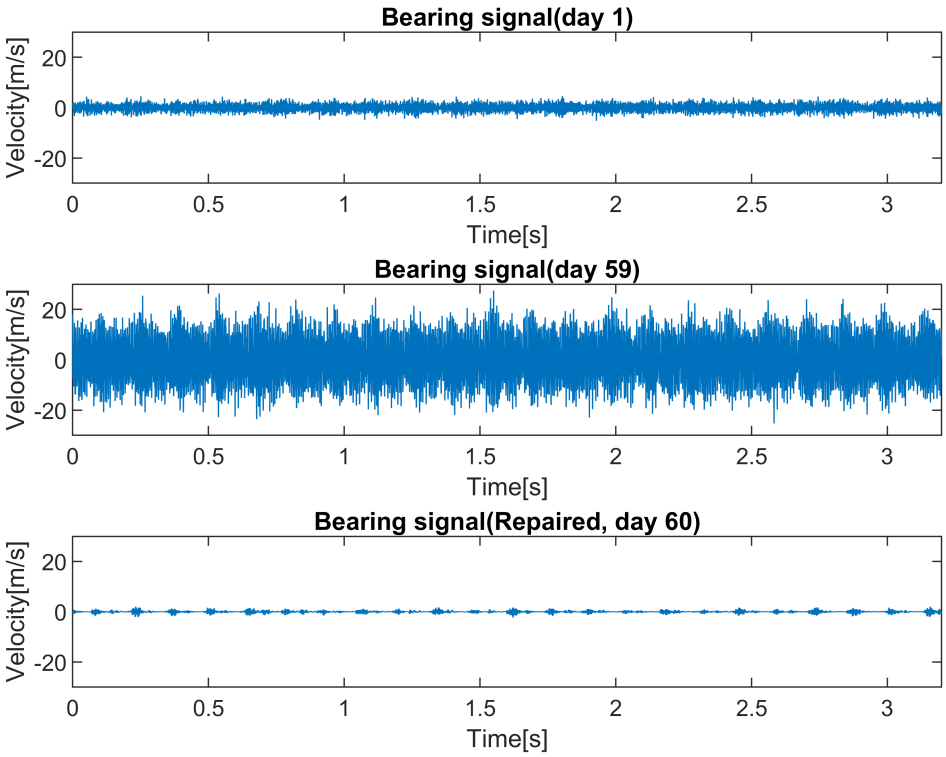
```
clear

addpath('../Assignment_3_Bearingdata');
addpath('Functions');

load('bearing-vel-fulldata-rul.mat');
```

```
% ----------------- Data strcuture -------------------------
% rows: vel measurements / cols: measurement time
% before repair : 59 days
% after repair : 28 days
% ------------------------------------------------------------

idxRepair = 1431;

bearing = bearingFulldata(:,1:idxRepair);               % before repair
bearingNew = bearingFulldata(:,idxRepair + 1 : end);    % after repair

% Data Length for each measurement
% L=8192
L = length(bearing(:,1));
fs = 2560;                          % sampling frequency [Hz]
ts = 1/fs;                          % sampling time
t = 0:ts:ts*(L-1);

% Day(time) unit upto repair
dataN = length(bearing(1, :));
d = (0:1:dataN)/24;
```

Plotting bearing velocity raw data

- Initial state: Day=0
- Just before repair: Day=59
- After repair: Day=60

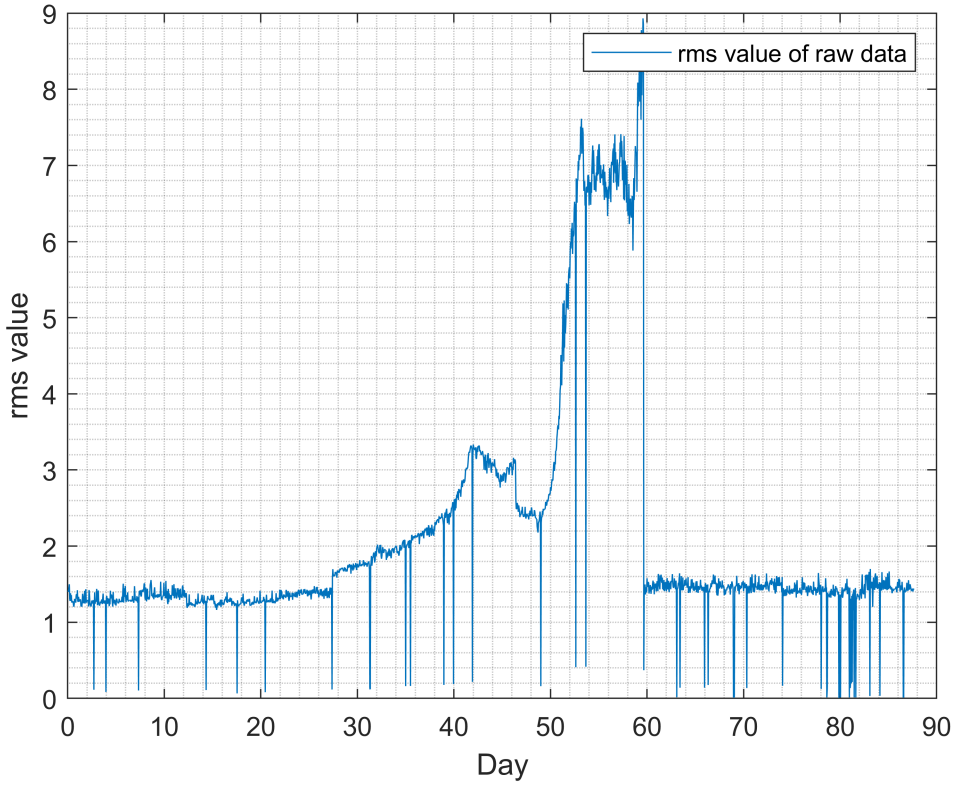
```
idx1 = 1;    idx2 = idxRepair;    idx3 = 1;

figure
subplot(3, 1, 1);       plot(t, bearing(:, idx1));
xlim([0 3.2]);  ylim([-30 30]); xlabel('Time[s]');  ylabel('Velocity[m/s]');    title('Bearing

subplot(3, 1, 2);       plot(t, bearing(:, idx2));
xlim([0 3.2]);  ylim([-30 30]); xlabel('Time[s]');  ylabel('Velocity[m/s]');    title('Bearing

subplot(3, 1, 3);       plot(t, bearingNew(:, idx3));
xlim([0 3.2]);  ylim([-30 30]); xlabel('Time[s]');  ylabel('Velocity[m/s]');    title('Bearing
```



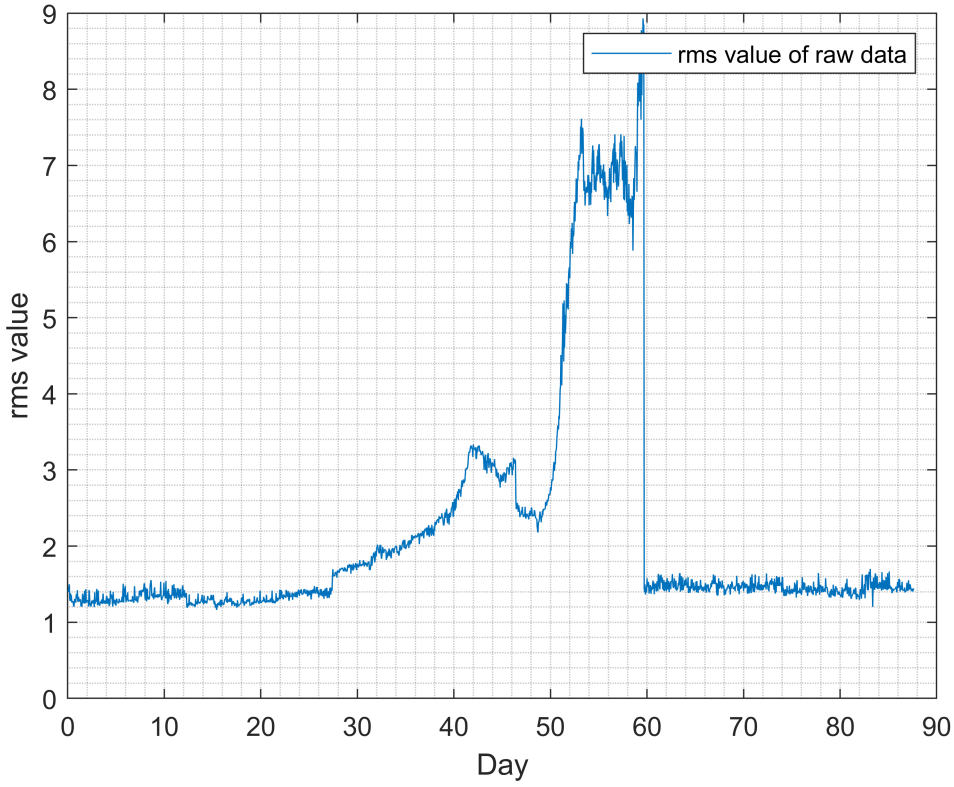
Data Exporation and Preprocessing

- Since rotation of shaft goes to zero temporarily in the middle of the whole process, it needs to be preprocessed.
- To see the variation due to the effect of preprocessing, RMS value is appropriate to use.
- rmsRaw : rms value of raw data each day
- rmsPro : rms value of preprocessed data

```
% Variables to see the result for 87days
Size = width(bearingFulldata);
perDay = 24;
idxDay = (1:Size)/perDay;

rmsRaw = rms(bearingFulldata(:,1:Size));

figure();
plot(idxDay, rmsRaw);
xlabel("Day");      ylabel("rms value");        grid minor;     legend("rms value of raw data")
```




```
rmsUnder = find(rmsRaw < 0.5);

for Idx = 1 : length(rmsUnder)

  bearingFulldata(:,rmsUnder(Idx)) = bearingFulldata(:,rmsUnder(Idx)-1);
end

rmsProcess = rms(bearingFulldata(:,1:Size));

figure();
plot(idxDay, rmsProcess);
xlabel("Day");      ylabel("rms value");        grid minor;     legend("rms value of raw data")
ylim([0 9]);
```

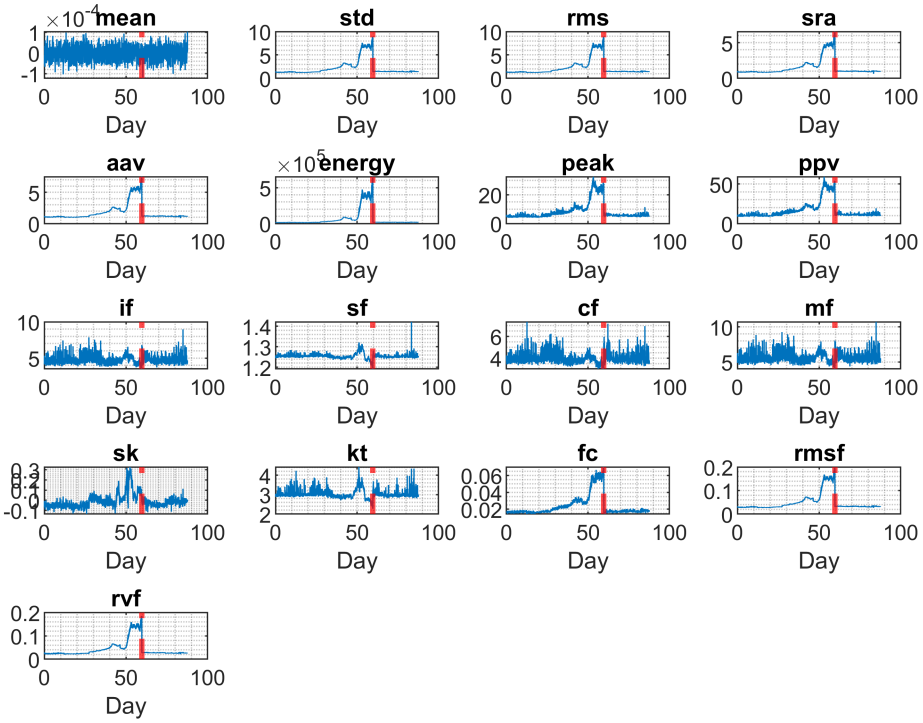


Feature Extraction and Analysis

- All other time-domain features also need to be tested in the same way that the trend was identified with the rms value above.
- In order to reliably analyze the life tendency of bearings, it is important to select features that show a clear difference in tendency before and after repair, such as the rms value above.
- As we can see in the figure below, some features in time domain and frequency domain (std, rms. sra, peak ... & fc, rmsf, rvf) shows good performance to distinguish.

```
featureName = ["mean", "std", "rms", "sra", "aav", "energy", "peak", "ppv", "if", "sf", "cf", "

timeFeatures = getTimefeatures(bearingFulldata);
freqFeatures = getFreqfeatures(bearingFulldata, fs);
staticFeatures = [timeFeatures freqFeatures];
staticFeatures = table2array(staticFeatures);

timeFeatures = timeFeatures(:, {'std', 'rms', 'sra', 'ppv', 'if', 'sf', 'cf', 'mf', 'sk', 'kt'}

figure()

for Idx = 1 : width(staticFeatures)

   subplot(5,4,Idx)
   plot(idxDay', staticFeatures(:,Idx));
```

```
    xlabel("Day");    title(featureName(Idx));   grid minor;
    xline(idxRepair/perDay, 'r--', Linewidth = 2);
end
```

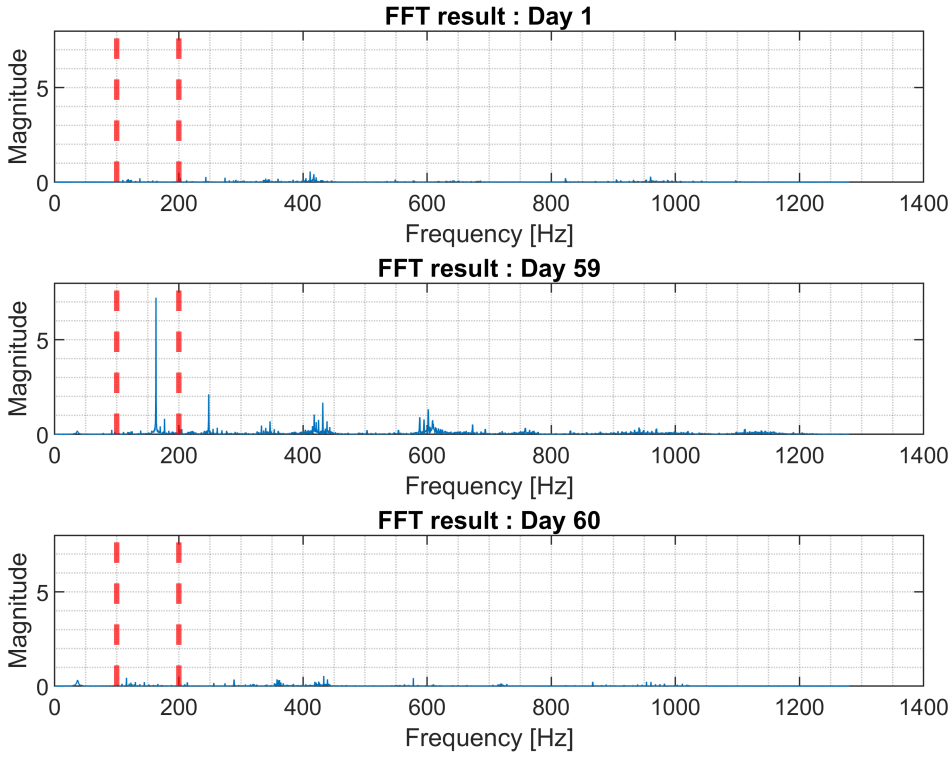


FFT of each cases

- Normal time : 1
- Largest fault : day 59 * 24
- Repair : day 60 * 24
- As we can see in the FFT result of each case (normal, largest cracked, repaired) in same scale (y limit), Day 59 (largest cracked) has the large magnitude of multiple frequency components.
- To see the variation of frequency components according to the time, it is efficient to get the STFT result in the next step.
- Usually, frequency components those are from faults are usually in the range of [100 200] Hz (BPFO, BPFI).

```
normalIdx = 1;
faultIdx  = 59 * 24;
repairIDX    = 60 * 24;

for Idx = 1 : width(bearingFulldata)

    [freq, mag] = getFFT(bearingFulldata(:,Idx),L, fs);
```

```
    if(Idx == normalIdx)

        subplot(3,1,1);
        plot(freq, mag);
        xline(100, 'r--', Linewidth = 2);      xline(200, 'r--', Linewidth = 2);
        xlabel("Frequency [Hz]");   ylabel("Magnitude");    grid minor;     title("FFT result :
    end

    if(Idx == faultIdx)

        subplot(3,1,2);
        plot(freq, mag);
        xline(100, 'r--', Linewidth = 2);      xline(200, 'r--', Linewidth = 2);
        xlabel("Frequency [Hz]");   ylabel("Magnitude");    grid minor;     title("FFT result :
    end

    if(Idx == repairIDX)

        subplot(3,1,3);
        plot(freq, mag);
        xline(100, 'r--', Linewidth = 2);      xline(200, 'r--', Linewidth = 2);
        xlabel("Frequency [Hz]");   ylabel("Magnitude");    grid minor;     title("FFT result :
    end

end
```



Narrowband feature

- 150 - 190 [Hz] : frequency related to bearing fault frequency
- Process of extracting narrowband feature
- 1) Apply FFT for everyday signal
- 2) Extract indidces those are contained in the bandwidth (150 - 190 [Hz])
- 3) Various features such as rms, rmsf, and rvf for the magnitude components of the frequency corresponding to the bandwidth were calculated.
- 4) As figure below in this block, three narrow band feartures show good performance and the following monotocity would be good.

```
% nDays = width(bearingFulldata);
% nbFeatures = zeros(nDays,3);
%
% for day = 1 : nDays
%
%     [freq, mag] = getFFT(bearingFulldata(:,day),L, fs);
%     N = length(mag);
%
%     narrowIdx = find(freq >= 150 & freq <= 190);
%
%     narrowMag = mag(narrowIdx);
%
%     nbCenterFreq = sum(narrowMag)/N;
%     nbRmsf = sqrt(sum(narrowMag.^2)/N);
%     nbRvf = sqrt(sum((narrowMag - nbCenterFreq).^2)/N);
%
%     nbFeatures(day, 1) = nbCenterFreq;
%     nbFeatures(day, 2) = nbRmsf;
%     nbFeatures(day, 3) = nbRvf;
% end
%
% nbVariableName = ["nbCF", "nbRMSF", "nbRVF"];
% nbFeaturesDay = nbFeatures(1:perDay:length(idxDay),: );
% nbFeaturesDay = array2table(nbFeaturesDay, 'VariableNames', nbVariableName)
% figure
% subplot(3,1,1)
% plot(idxDay, nbFeatures(:,1));
% xline(60,'r--','LineWidth',2);
% xlabel('day');      ylabel('cf');   grid minor;
% title('narrow band : center frequency');
%
% subplot(3,1,2)
% plot(idxDay,nbFeatures(:,2));
% xline(60,'r--','LineWidth',2);
% xlabel('day');      ylabel('rmsf');    grid minor;
% title('narrow band std value');
% title('narrow band : rmsf');
```

```
%
% subplot(3,1,3)
% plot(idxDay,nbFeatures(:,3));
% xline(60,'r--','LineWidth',2);
% xlabel('day');      ylabel('rvf');    grid minor;
% title('narrow band std value');
% title('narrow band : rvf');
```

Spectral kurtosis

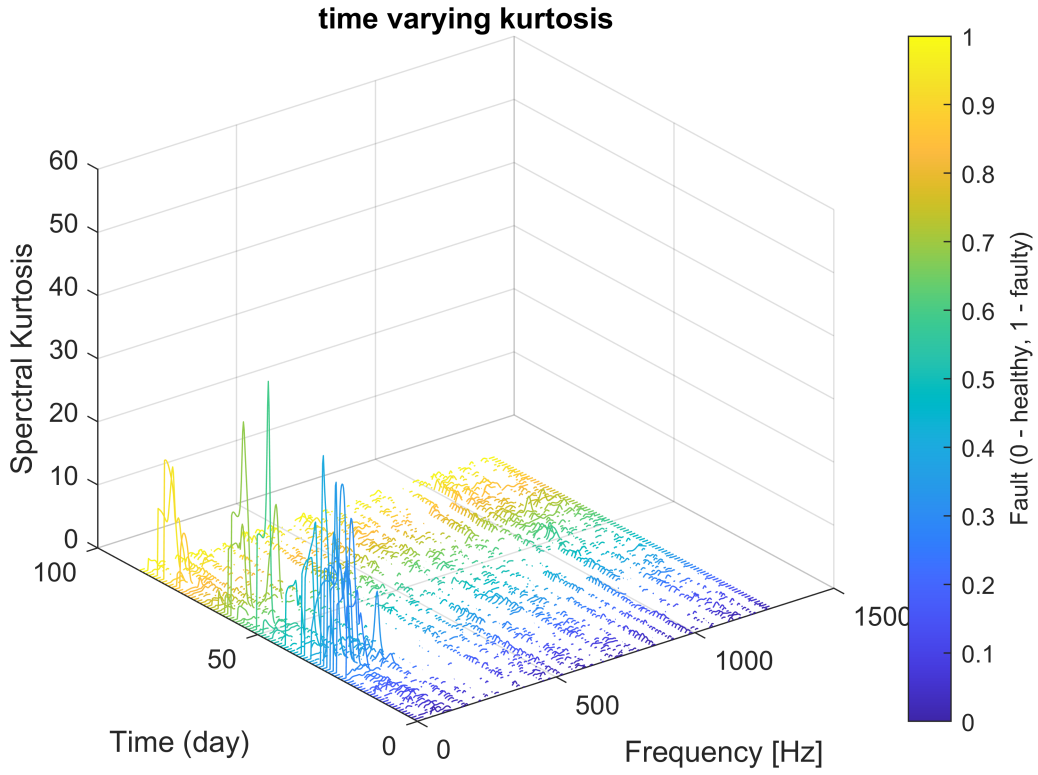
- As figure above shows, spectral kurtosis has a poor monotocity.
- Spectral kurtosis over time can be analyzed.

```
wc = 128;
Index = 1;

rawData = [bearing bearingNew];

% ----- Which data to see ? -----
% 1 : raw data
% 2 : processed data

selectData = 2
```

```
selectData = 2
```

```
switch selectData

    case 1
        dataToanalyze = rawData;            % rawdata
    case 2
        dataToanalyze = bearingFulldata;    % preprocessed data
end

colors = parula(width(dataToanalyze));

figure()

for Idx = 1 : perDay : width(dataToanalyze)

    kurtosisDat = table();

    [kurtosisSK, kurtosisF] = pkurtosis(dataToanalyze(:,Idx), fs, wc);

    kurtosis_sk(:,Index) = kurtosisSK;

    kurtosisDat.SpectralKurtosis = {table(kurtosisF, kurtosisSK)};
    plot3(kurtosisF, (Idx * ones(size(kurtosisF)))/perDay, kurtosisSK, 'Color', colors(Idx, :))
    zlim([0 60]);   grid on;    hold on;
```

```
    Index = Index + 1;
end

xlabel("Frequency [Hz]");   ylabel("Time (day)");    zlabel("Sperctral Kurtosis");
cbar = colorbar;    ylabel(cbar, 'Fault (0 - healthy, 1 - faulty)');     title('time varying ku
```




```
kurtosisResult = struct('mean',     mean(kurtosis_sk),     ...
                        'std',      std(kurtosis_sk),      ...
                        'skewness', skewness(kurtosis_sk), ...
                        'kurtosis', kurtosis(kurtosis_sk));

skFeature = [kurtosisResult.mean' kurtosisResult.std' kurtosisResult.skewness' kurtosisResult.k

skName = ["skMean", "skStd", "skSkewness", "skKurtosis"];
skFeature = array2table(skFeature, "VariableNames",skName);
```

Selection of Features

- In order to prepare the training data, data until normal operation by repair is required , and processing is performed accordingly.
- Variable 'allFeatureTbl' shows the data explained above.

```
timeFeaturesDay = timeFeatures(1 : perDay : length(idxDay),: );
freqFeaturesDay = freqFeatures(1 : perDay : length(idxDay),: );

% Shows all features of the period
```

```
% allFeatureTbl = [table2array(timeFeaturesDay(2:60,: ))      table2array(freqFeaturesDay(2:60,
allFeatureTbl = [table2array(timeFeaturesDay(1:59,: ))      table2array(freqFeaturesDay(1:59,:

tbltimeFeatureName = ["std", "rms", "sra", "ppv", "if", "sf", "cf", "mf", "sk", "kt", "center f

% tableLabel = [tbltimeFeatureName  skName  nbVariableName];
tableLabel = [tbltimeFeatureName  skName ];

dateReal = (datetime(2021,4,13) : days(1) : datetime(2021,7,11));
dateTime = (dateReal(2:60));


allFeatureTbl = array2table(allFeatureTbl, "VariableNames", tableLabel);
allFeatureTbl = table2timetable(allFeatureTbl, 'RowTimes', dateTime)
```

allFeatureTbl = 59×17 timetable

...

	Time	std	rms	sra	ppv	if	sf	cf
1	2021-04-14	1.3984	1.3983	0.9435	9.8100	4.1151	1.2536	3.7331
2	2021-04-15	1.2797	1.2796	0.8639	9.2900	4.4784	1.2539	3.6887
3	2021-04-16	1.2901	1.2900	0.8694	10.6000	4.7907	1.2561	4.4030
4	2021-04-17	1.3382	1.3381	0.9082	10.2200	4.8253	1.2513	3.8563
5	2021-04-18	1.3463	1.3462	0.9141	9.5600	4.6676	1.2492	3.7365
6	2021-04-19	1.3094	1.3094	0.8765	9.9100	4.5852	1.2586	3.9256
7	2021-04-20	1.2644	1.2643	0.8553	8.7400	4.0058	1.2505	3.7095
8	2021-04-21	1.2972	1.2972	0.8806	9.0600	4.0749	1.2496	3.7235
9	2021-04-22	1.2941	1.2940	0.8776	10.0800	4.4187	1.2512	4.2581
10	2021-04-23	1.3529	1.3528	0.9156	9.9900	4.2471	1.2518	3.9916
11	2021-04-24	1.3444	1.3443	0.9033	10.0400	4.4123	1.2567	3.9574
12	2021-04-25	1.3148	1.3147	0.8878	9.7700	4.5433	1.2522	3.8031
13	2021-04-26	1.3986	1.3985	0.9499	10.4500	4.5787	1.2482	3.8040
14	2021-04-27	1.2371	1.2370	0.8394	9.1400	4.3729	1.2492	3.8885
15	2021-04-28	1.2996	1.2995	0.8774	9.7600	4.5138	1.2534	3.9092
16	2021-04-29	1.3447	1.3446	0.9097	9.6700	4.6332	1.2510	3.7036
17	2021-04-30	1.2604	1.2603	0.8517	9.7000	4.2412	1.2518	4.3085
18	2021-05-01	1.2581	1.2580	0.8525	9.6300	4.5825	1.2505	3.9904
19	2021-05-02	1.2842	1.2841	0.8641	10.5900	4.7209	1.2551	4.4857
20	2021-05-03	1.2449	1.2448	0.8469	8.9500	4.2560	1.2495	3.7836
21	2021-05-04	1.2530	1.2529	0.8412	9.3300	4.9270	1.2573	3.9188
22	2021-05-05	1.2619	1.2618	0.8548	9.5400	4.5012	1.2511	3.9625

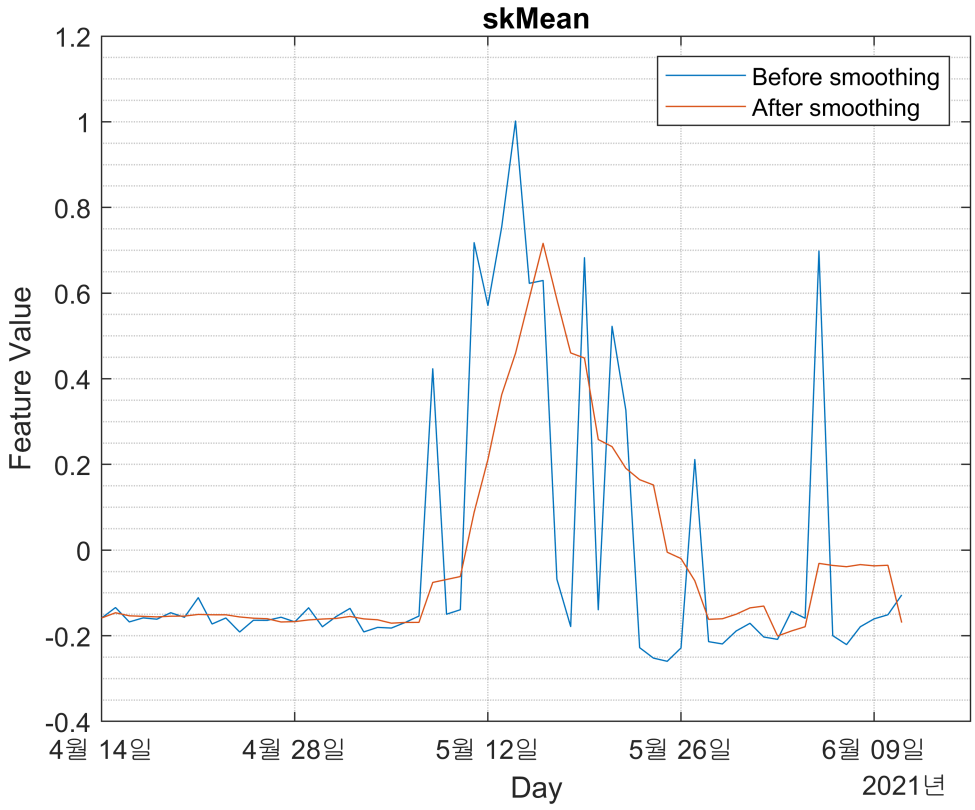
	Time	std	rms	sra	ppv	if	sf	cf
23	2021-05-06	1.3238	1.3237	0.8957	9.7200	4.7699	1.2527	3.8076
24	2021-05-07	1.3472	1.3471	0.9060	10.1800	4.4672	1.2563	4.0013
25	2021-05-08	1.4181	1.4180	0.9418	12.4400	5.5038	1.2669	4.4289
26	2021-05-09	1.3610	1.3609	0.9214	10.6000	4.6630	1.2516	4.0636
27	2021-05-10	1.3660	1.3659	0.9206	11.4100	5.0678	1.2540	4.3121
28	2021-05-11	1.3915	1.3914	0.9380	10.0800	4.4965	1.2563	3.6654
29	2021-05-12	1.6530	1.6529	1.1079	15.0300	6.2349	1.2614	4.9429
30	2021-05-13	1.7505	1.7504	1.1764	13.4800	4.6087	1.2585	4.0392
31	2021-05-14	1.7703	1.7702	1.1797	14.5000	4.8001	1.2644	4.3951
32	2021-05-15	1.7554	1.7553	1.1705	13.3500	4.5397	1.2629	4.0107
33	2021-05-16	1.9872	1.9871	1.3268	16.3800	5.2018	1.2636	4.1266
34	2021-05-17	1.8588	1.8586	1.2642	13.7700	4.8748	1.2480	3.9061
35	2021-05-18	1.9971	1.9970	1.3659	13.2200	3.8165	1.2433	3.5504
36	2021-05-19	1.9698	1.9697	1.3431	15.0790	4.9271	1.2489	3.9453
37	2021-05-20	2.1412	2.1410	1.4579	14.2800	4.2476	1.2475	3.4049
38	2021-05-21	2.2290	2.2288	1.5220	15.5000	4.2554	1.2447	3.5355
39	2021-05-22	2.3170	2.3169	1.5734	16.4200	4.4110	1.2494	3.5565
40	2021-05-23	2.4033	2.4032	1.6516	19.5100	4.5352	1.2413	4.4650
41	2021-05-24	2.6163	2.6161	1.7763	19.6600	5.0621	1.2494	4.0518
42	2021-05-25	2.8271	2.8269	1.9342	19.6500	3.8314	1.2449	3.8735
43	2021-05-26	3.3076	3.3074	2.2492	25.7500	5.5285	1.2473	4.4325
44	2021-05-27	3.1813	3.1811	2.1733	21.6500	4.2742	1.2440	3.4359
45	2021-05-28	3.0577	3.0575	2.0616	21.7300	4.6010	1.2549	3.6664
46	2021-05-29	2.8806	2.8804	1.9717	19.2800	4.2468	1.2431	3.4162
47	2021-05-30	2.9298	2.9296	1.9863	20.1600	4.2263	1.2519	3.5055
48	2021-05-31	2.4412	2.4411	1.6772	16.8700	4.4487	1.2411	3.5845
49	2021-06-01	2.5093	2.5091	1.6927	19.7900	5.0863	1.2549	4.0532
50	2021-06-02	2.3118	2.3117	1.5348	18.2930	4.8863	1.2687	4.0620
51	2021-06-03	2.7239	2.7238	1.8033	22.3100	5.8339	1.2702	4.5929
52	2021-06-04	4.1034	4.1032	2.5877	33.3400	5.9025	1.3134	4.4941
53	2021-06-05	5.8691	5.8688	3.7647	44.4800	4.9145	1.2957	3.7930
54	2021-06-06	7.1069	7.1065	4.4617	52.2400	5.2888	1.3114	4.0330
55	2021-06-07	6.7848	6.7844	4.4083	48.7400	5.0122	1.2842	3.9031

	Time	std	rms	sra	ppv	if	sf	cf
56	2021-06-08	7.2808	7.2803	4.9572	48.1100	4.1690	1.2439	3.3515
57	2021-06-09	6.5609	6.5605	4.5646	42.6000	4.0444	1.2324	3.2817
58	2021-06-10	6.7226	6.7222	4.5086	44.6500	4.0766	1.2559	3.3962
59	2021-06-11	6.3404	6.3401	4.3716	43.1800	4.5176	1.2383	3.6482

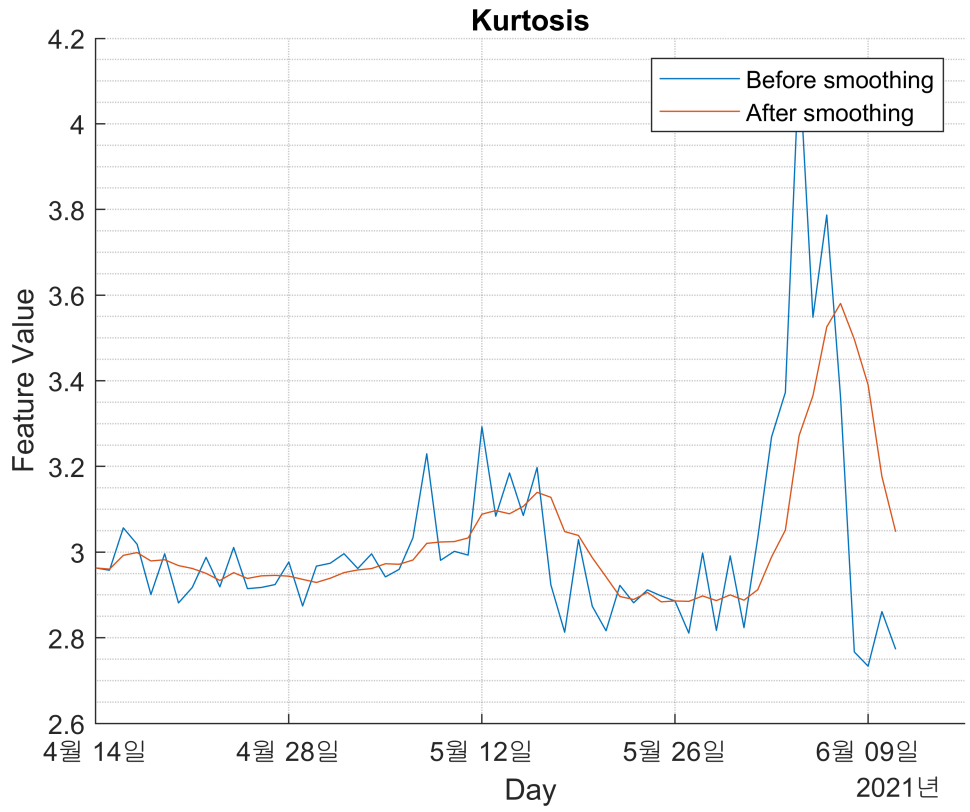
Feature postprocessing

- Since data without filtering contains noise, it needs to be filtered using median filter for smoothing.
- Result of filtering will be used for training.

```
variableNames = allFeatureTbl.Properties.VariableNames;
featureTableSmooth = varfun(@(x) movmean(x, [5 0]), allFeatureTbl);
featureTableSmooth.Properties.VariableNames = variableNames;

featureValues = zeros(1, 59);

figure

plot(allFeatureTbl.Time, allFeatureTbl.skMean);
hold on;
plot(allFeatureTbl.Time, featureTableSmooth.skMean);
hold off;
xlabel("Day");  ylabel("Feature Value");    legend('Before smoothing', 'After smoothing');  gri
title("skMean");
```




```
figure
hold on
plot(allFeatureTbl.Time, allFeatureTbl.kt);
plot(allFeatureTbl.Time, featureTableSmooth.kt)
hold off
xlabel("Day");  ylabel("Feature Value");    legend('Before smoothing', 'After smoothing');  gri
title("Kurtosis");
```



Training

- breakpoing : index 42 was selected for preparing data sets
- This breakpoint allows about 50 percent of date index (42/89) to be trained.

```
% About 50% selection for trianing
breaktimeFirst = datetime(2021, 4,29);
breaktime = datetime(2021, 5, 26);

breakpoint = find(featureTableSmooth.Time < breaktime , 1, 'last');

trainData  = featureTableSmooth(1:breakpoint, :)
```

trainData = 42×17 timetable

...

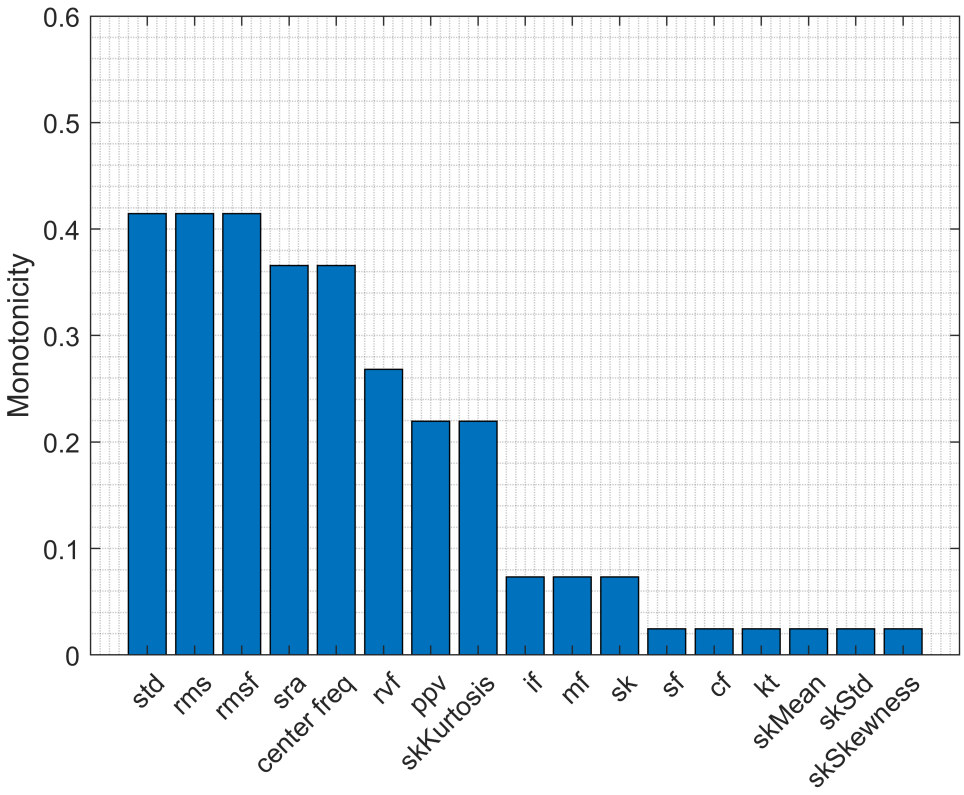
	Time	std	rms	sra	ppv	if	sf	cf
1	2021-04-14	1.3984	1.3983	0.9435	9.8100	4.1151	1.2536	3.7331
2	2021-04-15	1.3390	1.3389	0.9037	9.5500	4.2967	1.2538	3.7109
3	2021-04-16	1.3227	1.3226	0.8923	9.9000	4.4614	1.2546	3.9416
4	2021-04-17	1.3266	1.3265	0.8963	9.9800	4.5524	1.2537	3.9203
5	2021-04-18	1.3305	1.3304	0.8998	9.8960	4.5754	1.2528	3.8835
6	2021-04-19	1.3270	1.3269	0.8959	9.8983	4.5770	1.2538	3.8905

	Time	std	rms	sra	ppv	if	sf	cf
7	2021-04-20	1.3047	1.3046	0.8812	9.7200	4.5588	1.2533	3.8866
8	2021-04-21	1.3076	1.3075	0.8840	9.6817	4.4916	1.2526	3.8924
9	2021-04-22	1.3083	1.3082	0.8854	9.5950	4.4296	1.2517	3.8682
10	2021-04-23	1.3107	1.3106	0.8866	9.5567	4.3332	1.2518	3.8908
11	2021-04-24	1.3104	1.3103	0.8848	9.6367	4.2907	1.2531	3.9276
12	2021-04-25	1.3113	1.3112	0.8867	9.6133	4.2837	1.2520	3.9072
13	2021-04-26	1.3337	1.3336	0.9025	9.8983	4.3792	1.2516	3.9229
14	2021-04-27	1.3236	1.3236	0.8956	9.9117	4.4288	1.2516	3.9504
15	2021-04-28	1.3246	1.3245	0.8956	9.8583	4.4447	1.2519	3.8923
16	2021-04-29	1.3232	1.3231	0.8946	9.8050	4.5090	1.2518	3.8443
17	2021-04-30	1.3092	1.3091	0.8860	9.7483	4.4805	1.2510	3.9028
18	2021-05-01	1.2997	1.2997	0.8801	9.7250	4.4871	1.2507	3.9340
19	2021-05-02	1.2807	1.2806	0.8658	9.7483	4.5108	1.2518	4.0477
20	2021-05-03	1.2820	1.2819	0.8670	9.7167	4.4913	1.2519	4.0302
21	2021-05-04	1.2742	1.2741	0.8610	9.6450	4.5601	1.2525	4.0318
22	2021-05-05	1.2604	1.2603	0.8519	9.6233	4.5381	1.2525	4.0749
23	2021-05-06	1.2710	1.2709	0.8592	9.6267	4.6263	1.2527	3.9914
24	2021-05-07	1.2858	1.2857	0.8681	9.7183	4.6070	1.2537	3.9932
25	2021-05-08	1.3081	1.3081	0.8811	10.0267	4.7375	1.2556	3.9838
26	2021-05-09	1.3275	1.3274	0.8935	10.3017	4.8053	1.2560	4.0304
27	2021-05-10	1.3463	1.3462	0.9067	10.6483	4.8288	1.2554	4.0960
28	2021-05-11	1.3679	1.3678	0.9206	10.7383	4.8280	1.2563	4.0465
29	2021-05-12	1.4228	1.4227	0.9560	11.6233	5.0722	1.2578	4.2357
30	2021-05-13	1.4900	1.4899	1.0010	12.1733	5.0958	1.2581	4.2420
31	2021-05-14	1.5487	1.5486	1.0407	12.5167	4.9785	1.2577	4.2364
32	2021-05-15	1.6144	1.6143	1.0822	12.9750	4.9580	1.2596	4.2275
33	2021-05-16	1.7180	1.7179	1.1499	13.8033	4.9803	1.2612	4.1966
34	2021-05-17	1.7959	1.7957	1.2043	14.4183	5.0434	1.2598	4.2367
35	2021-05-18	1.8532	1.8531	1.2473	14.1167	4.6403	1.2568	4.0047
36	2021-05-19	1.8898	1.8897	1.2750	14.3832	4.6933	1.2552	3.9890
37	2021-05-20	1.9516	1.9515	1.3214	14.3465	4.6013	1.2524	3.8240
38	2021-05-21	2.0305	2.0304	1.3800	14.7048	4.5539	1.2493	3.7448
39	2021-05-22	2.0855	2.0853	1.4211	14.7115	4.4221	1.2470	3.6498

	Time	std	rms	sra	ppv	if	sf	cf
40	2021-05-23	2.1762	2.1761	1.4856	15.6682	4.3655	1.2458	3.7429
41	2021-05-24	2.2794	2.2793	1.5540	16.7415	4.5731	1.2468	3.8265
42	2021-05-25	2.4223	2.4222	1.6526	17.5033	4.3904	1.2462	3.8145

Feature importance ranking

```
% Since moving window smoothing is already done, set 'WindowSize' to 0
% turn off the smoothing within the function
featureImportance = monotonicity(trainData, 'WindowSize', 0);
helperSortedBarPlot(featureImportance, 'Monotonicity');
ylim([0 0.6]);      grid minor;
```



- Making new dataset composed of selected features.

```
trainIdx = featureImportance{:,:} > 0.4;

trainDataSelected = trainData(:, trainIdx);
featureSelected = featureTableSmooth(:, trainIdx)
```

`featureSelected = 59×3 timetable`

	Time	std	rms	rmsf
1	2021-04-14	1.3984	1.3983	0.0309

	Time	std	rms	rmsf
2	2021-04-15	1.3390	1.3389	0.0296
3	2021-04-16	1.3227	1.3226	0.0292
4	2021-04-17	1.3266	1.3265	0.0293
5	2021-04-18	1.3305	1.3304	0.0294
6	2021-04-19	1.3270	1.3269	0.0293
7	2021-04-20	1.3047	1.3046	0.0288
8	2021-04-21	1.3076	1.3075	0.0289
9	2021-04-22	1.3083	1.3082	0.0289
10	2021-04-23	1.3107	1.3106	0.0290
11	2021-04-24	1.3104	1.3103	0.0290
12	2021-04-25	1.3113	1.3112	0.0290
13	2021-04-26	1.3337	1.3336	0.0295
14	2021-04-27	1.3236	1.3236	0.0292
15	2021-04-28	1.3246	1.3245	0.0293
16	2021-04-29	1.3232	1.3231	0.0292
17	2021-04-30	1.3092	1.3091	0.0289
18	2021-05-01	1.2997	1.2997	0.0287
19	2021-05-02	1.2807	1.2806	0.0283
20	2021-05-03	1.2820	1.2819	0.0283
21	2021-05-04	1.2742	1.2741	0.0282
22	2021-05-05	1.2604	1.2603	0.0278
23	2021-05-06	1.2710	1.2709	0.0281
24	2021-05-07	1.2858	1.2857	0.0284
25	2021-05-08	1.3081	1.3081	0.0289
26	2021-05-09	1.3275	1.3274	0.0293
27	2021-05-10	1.3463	1.3462	0.0297
28	2021-05-11	1.3679	1.3678	0.0302
29	2021-05-12	1.4228	1.4227	0.0314
30	2021-05-13	1.4900	1.4899	0.0329
31	2021-05-14	1.5487	1.5486	0.0342
32	2021-05-15	1.6144	1.6143	0.0357
33	2021-05-16	1.7180	1.7179	0.0380
34	2021-05-17	1.7959	1.7957	0.0397

	Time	std	rms	rmsf
35	2021-05-18	1.8532	1.8531	0.0409
36	2021-05-19	1.8898	1.8897	0.0418
37	2021-05-20	1.9516	1.9515	0.0431
38	2021-05-21	2.0305	2.0304	0.0449
39	2021-05-22	2.0855	2.0853	0.0461
40	2021-05-23	2.1762	2.1761	0.0481
41	2021-05-24	2.2794	2.2793	0.0504
42	2021-05-25	2.4223	2.4222	0.0535
43	2021-05-26	2.6167	2.6166	0.0578
44	2021-05-27	2.7754	2.7753	0.0613
45	2021-05-28	2.8989	2.8987	0.0640
46	2021-05-29	2.9784	2.9783	0.0658
47	2021-05-30	3.0307	3.0305	0.0670
48	2021-05-31	2.9664	2.9662	0.0655
49	2021-06-01	2.8333	2.8331	0.0626
50	2021-06-02	2.6884	2.6882	0.0594
51	2021-06-03	2.6328	2.6326	0.0582
52	2021-06-04	2.8366	2.8364	0.0627
53	2021-06-05	3.3265	3.3263	0.0735
54	2021-06-06	4.1041	4.1038	0.0907
55	2021-06-07	4.8167	4.8164	0.1064
56	2021-06-08	5.6448	5.6445	0.1247
57	2021-06-09	6.2843	6.2839	0.1388
58	2021-06-10	6.7209	6.7205	0.1485
59	2021-06-11	6.7994	6.7990	0.1502

Dimension reduction & feature fusion

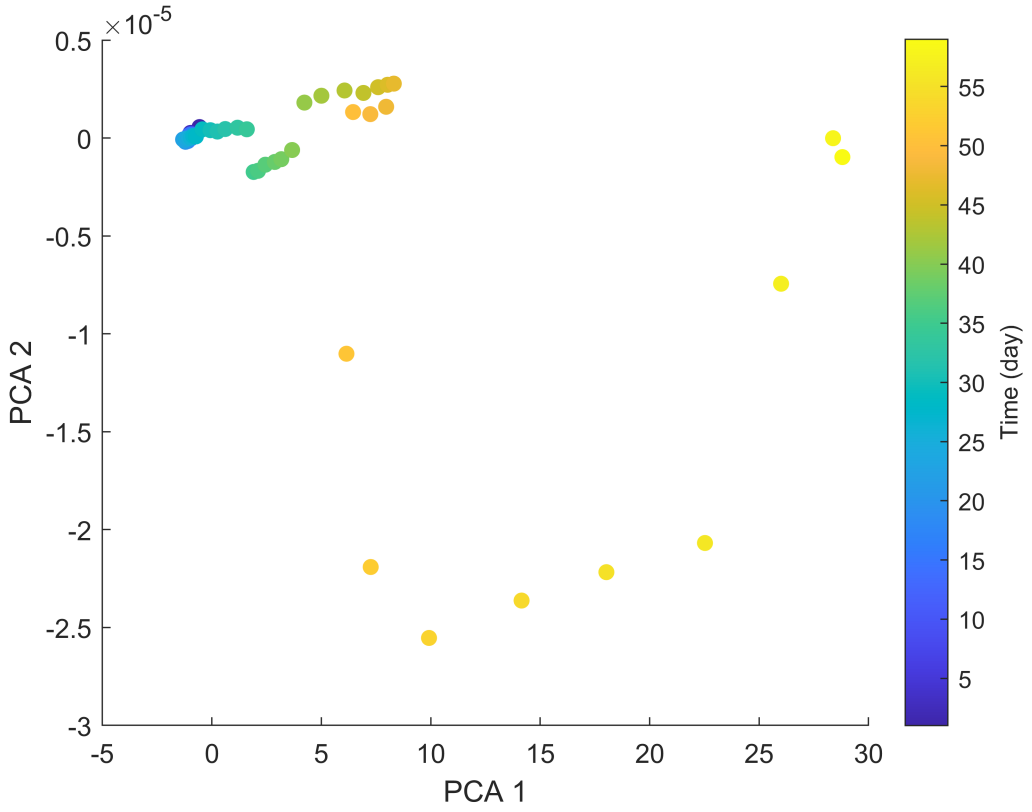
- Data normalization
- Apply principal component analysis with normalized datasets.(proprocessing for PCA)

```
meanTrain = mean(trainDataSelected{:,:});
sdTrain = std(trainDataSelected{:,:});
trainDataNormalized = (trainDataSelected{:,:} - meanTrain)./sdTrain;
```

```
% principal component vector
coef = pca(trainDataNormalized);

% PCA : 원본데이터가 PCA1, PCA2 주성분에 projection된 결과
% 이 값들은 원본 데이터의 차원을 축소한 결과로, 주요 패턴과 상관관계를 표현하기에 적합
PCA1 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 1);
PCA2 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 2);

timeUnit = 'day';
```

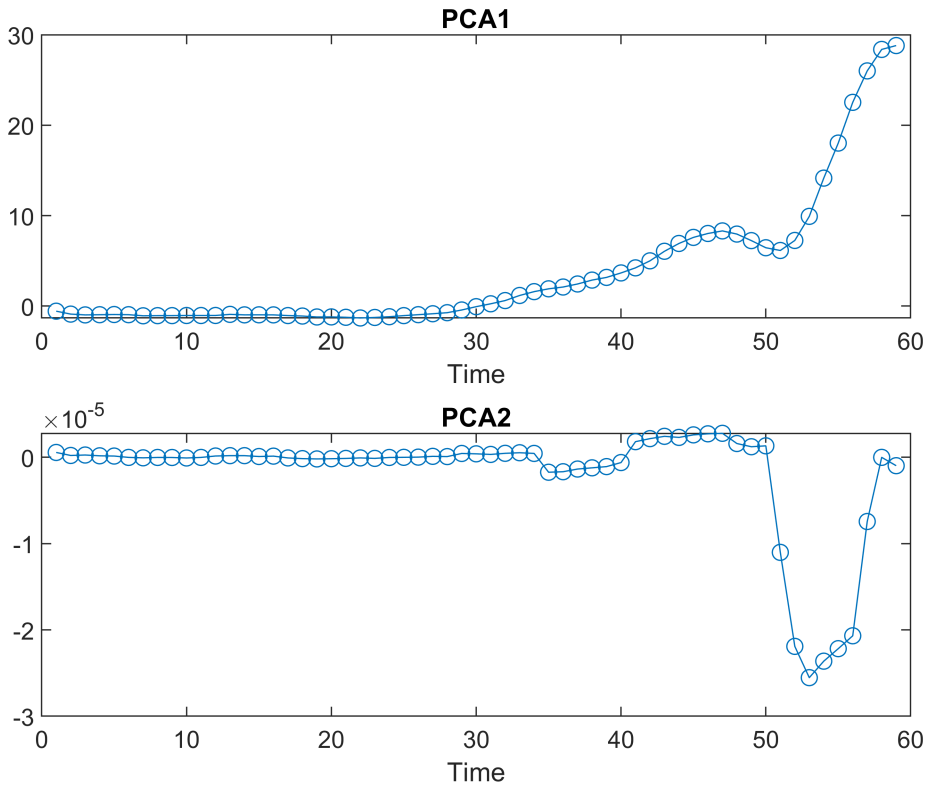
- visualizing

```
figure
numData = size(featureTableSmooth, 1);
scatter(PCA1, PCA2, [], 1:numData, 'filled');
xlabel('PCA 1');
ylabel('PCA 2');
cbar = colorbar;
ylabel(cbar, ['Time (' timeUnit ')']);
```

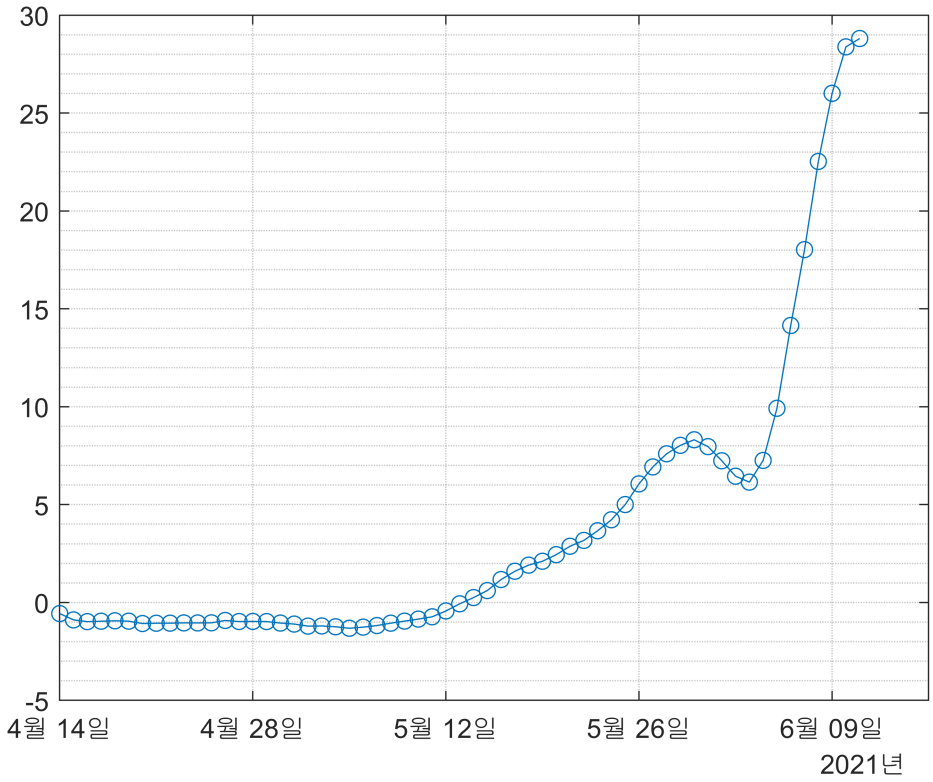



```
figure
subplot(2,1,1)
plot(PCA1, '-o')
xlabel('Time')
title('PCA1')
```

```
subplot(2,1,2)
plot(PCA2, '-o')
xlabel('Time')
title('PCA2')
```




```
figure
healthIndicator = PCA1;

plot(featureSelected.Time, healthIndicator, '-o');
grid minor;
```



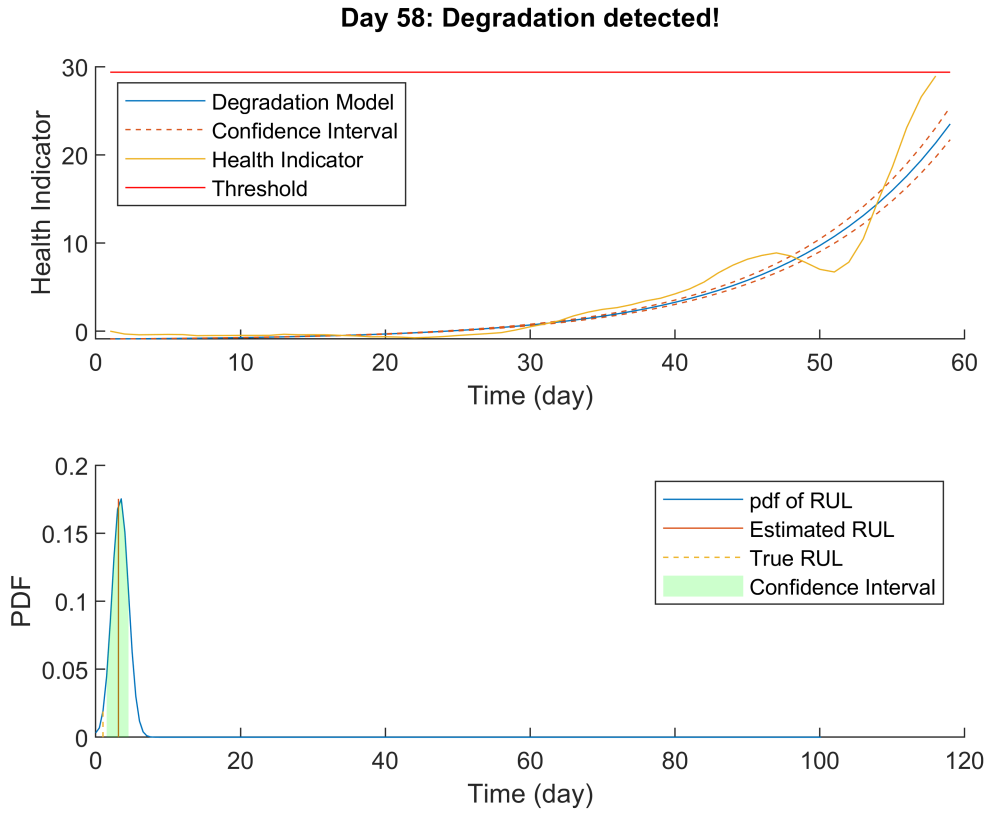
Fault Monitoring and Warning

RUL based on exponential degradation model

- pdf of RUL : probability density funtion of RUL, which represents the probability distribution of the predicted RUL values. Through this, uncertainty of RUL can be checked and range of possible RUL values can be estimated.
- `Estimated RUL`: The remaining useful life value predicted using the model. This value predicts how much life is left in the bearing, which helps in planning maintenance and replacements.
- `True RUL` : The actual Remaining Useful Life value, which is the real remaining life of the bearing. It is used to compare with the estimated RUL to evaluate the model's performance.
- `Confidence interval`: A range of values within which the true RUL is likely to fall with a certain level of confidence. It provides an indication of the uncertainty associated with the RUL estimation.

```
healthIndicator = healthIndicator - healthIndicator(1);
threshold = healthIndicator(end);


mdl = exponentialDegradationModel(...
    'Theta', 1, ...
    'ThetaVariance', 1e6, ...
    'Beta', 1, ...
    'BetaVariance', 1e6, ...
    'Phi', -1, ...
```

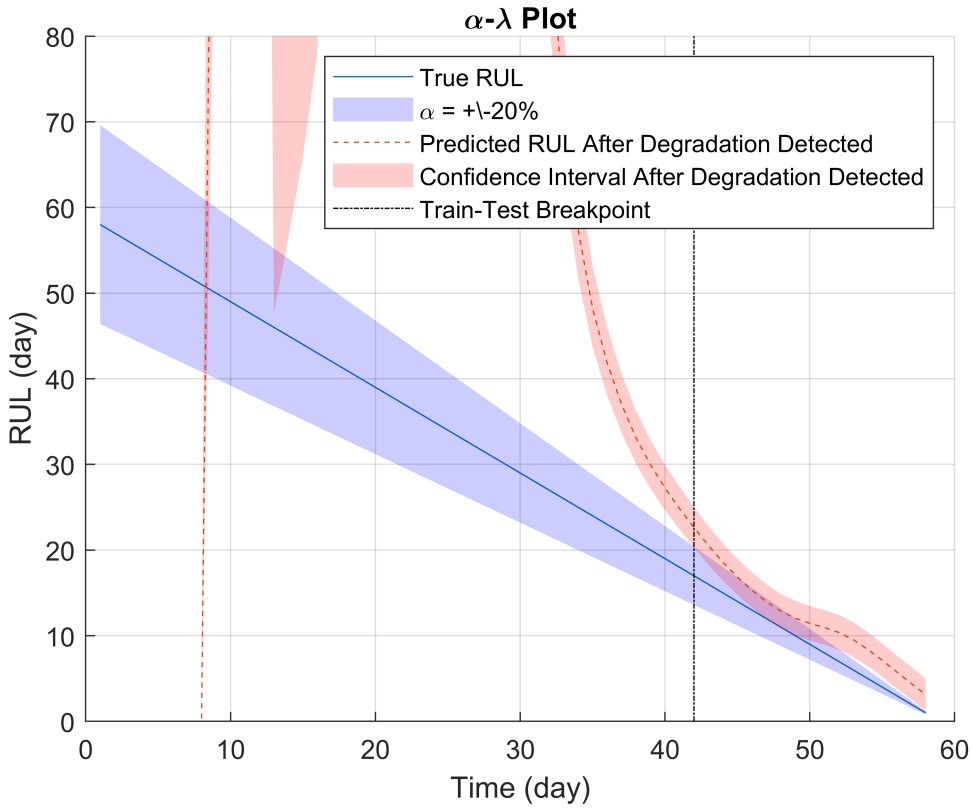
```
    'NoiseVariance', (0.1*threshold/(threshold + 1))^2, ...
    'SlopeDetectionLevel', 0.05);


% Keep records at each iteration
totalDay = length(healthIndicator) - 1;
estRULs = zeros(totalDay, 1);
trueRULs = zeros(totalDay, 1);
CIRULs = zeros(totalDay, 2);
pdfRULs = cell(totalDay, 1);

% Create figures and axes for plot updating
figure
ax1 = subplot(2, 1, 1);
ax2 = subplot(2, 1, 2);

for currentDay = 1:totalDay

    % Update model parameter posterior distribution
    update(mdl, [currentDay healthIndicator(currentDay)])

    % Predict Remaining Useful Life
    [estRUL, CIRUL, pdfRUL] = predictRUL(mdl, ...
                                         [currentDay healthIndicator(currentDay)], ...
                                         threshold);
    trueRUL = totalDay - currentDay + 1;

    % Updating RUL distribution plot
    helperPlotTrend(ax1, currentDay, healthIndicator, mdl, threshold, timeUnit);
    helperPlotRUL(ax2, trueRUL, estRUL, CIRUL, pdfRUL, timeUnit)

    % Keep prediction results
    estRULs(currentDay) = estRUL;
    trueRULs(currentDay) = trueRUL;
    CIRULs(currentDay, :) = CIRUL;
    pdfRULs{currentDay} = pdfRUL;

    % Pause 0.1 seconds to make the animation visible
    pause(0.1)
end
```



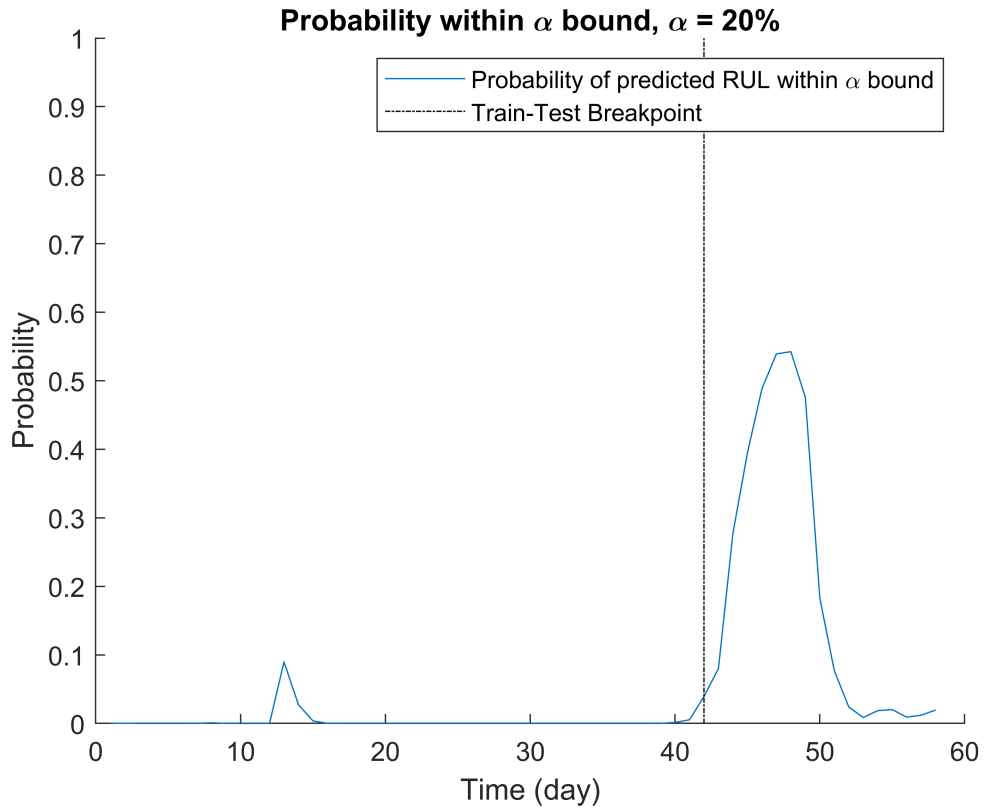
RUL Estimation

- Explained in the 'discussion'

```
alpha = 0.2;
detectTime = mdl.SlopeDetectionInstant;
prob = helperAlphaLambdaPlot(alpha, trueRULs, estRULs, CIRULs, ...
    pdfRULs, detectTime, breakpoint, timeUnit);
title('\alpha-\lambda Plot');
```




```
figure
t = 1:totalDay;
hold on
plot(t, prob)
plot([breakpoint breakpoint], [0 1], 'k-.')
hold off
xlabel(['Time (' timeUnit ')'])
ylabel('Probability')
legend('Probability of predicted RUL within \alpha bound', 'Train-Test Breakpoint')
title(['Probability within \alpha bound, \alpha = ' num2str(alpha*100) '%'])
```



Discussion

1) Data analysis

The experimental data was measured a total of 24 times per day for approximately 87 days. The measurements from day 1 to day 59 represent the operation before maintenance, and those from day 60 onwards represent the measurements after maintenance. As observed from the characteristics in the time domain and frequency domain, as well as the FFT results and spectral kurtosis, there is a clear difference between the measurements before and after maintenance. There are some parts where the bearing does not rotate intermittently, causing the values to appear to jump. To address this, I performed preprocessing to refer to the previous value for those sections. Additionally, for the future task of predicting RUL, it is essential to select features that clearly distinguish the measurements before and after maintenance. There are many features available, such as std, rms, sra, aav, fc, rmsf, rvf, among others.

2) Narrow band anlaysis

In this study, to extract features with higher monotonicity, we performed a Fourier Transform (FFT) analysis on the narrowband signals within the frequency range of 150-190Hz, where bearing faults are primarily observed. Features such as center frequency, rmsf, and rvf for the corresponding frequency band were computed. As a result, it was able to extract features with better monotonicity performance compared to the various time and frequency features previously obtained.

3) PCA & dimension reduction

The appearance of a graph showing a linearly increasing trend indicates a positive correlation between PCA1 and PCA2. This implies that the selected primary features exhibit a consistent relationship and change over time. These findings indicate that PCA effectively captures the main patterns, which can aid in predicting changes in bearing status or failure progression. Based on these results, it is possible to confirm that the primary features are changing in a consistent direction over time. This information can be used to develop a prediction model for estimating bearing Remaining Useful Life (RUL) or exploring other correlations within the system.

4) Fault monitoring & RUL estimation

The data before the Train-Test Breakpoint is used for model training. However, if at this point, the True RUL (Remaining Useful Life) and the prediction do not match completely, it may indicate that the model has not fully learned the patterns in the data. This is a common phenomenon that occurs during the training process, as the model gradually accumulates information about the data, resulting in improved prediction performance.

At this point, low prediction accuracy does not necessarily mean that the model is inadequate. On the contrary, it provides an opportunity to observe the model's learning process and evaluate its performance. Therefore, prediction errors at this stage can be considered as part of the model's learning and improvement process.

However, if the prediction accuracy continues to be low in the training data, it may indicate that the model is not fitting well or is underfitting. In this case, the model may need to be improved or a different model may need to be used. To address such issues, methods such as adjusting the model's hyperparameters, using different algorithms, or collecting additional data can be employed.