LAB: Bearing Fault Monitoring and RUL Estimation

Name: AN GYEON HEAL

ID: 21900416

Date: 2024-10-04

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

First visualize the vibration signals in the time domain.

```
clear; close all; clc;
addpath('../../Library')
load("RULdata/bearing-vel-fulldata-rul.mat");
% rows: vel measurements // cols: measurement time
% idx_repair = 1431;

bearing=bearingFulldata(:,1:idx_repair);
```

```
bearingNew=bearingFulldata(:,idx_repair+1:end);

% Data Length for each measurement
% L=8192
L=length(bearing(:,1))
```

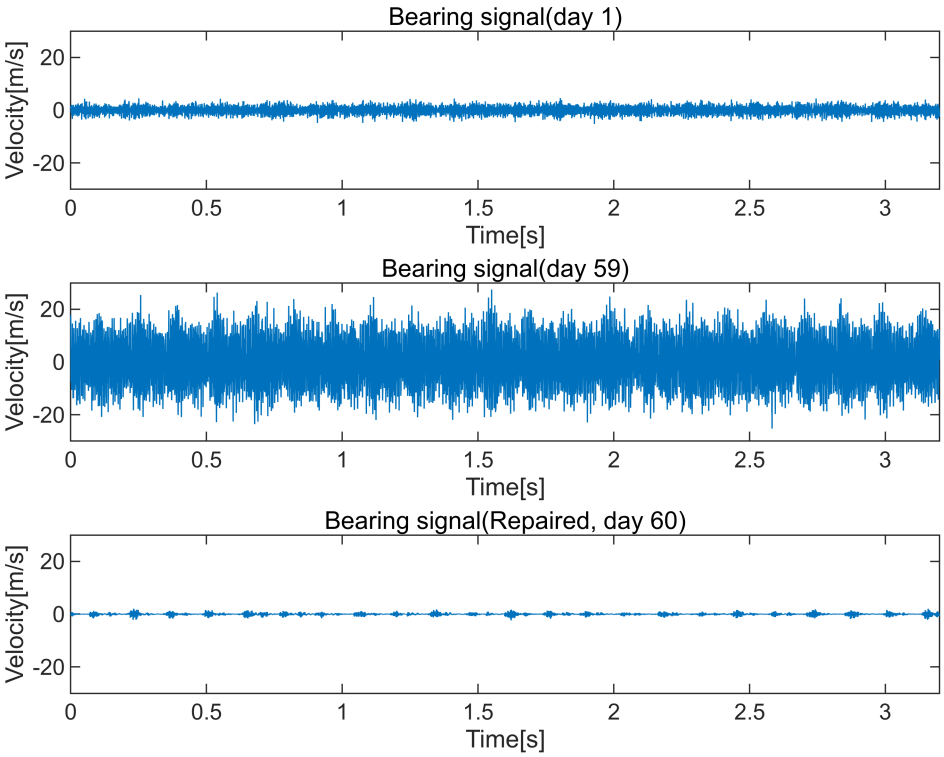
```
L = 8192
```

```
% Sampling Frequency
fs = 2560;
ts = 1/fs;
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 = idx_repair;    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 signal(day 1)');
subplot(3, 1, 2);       plot(t, bearing(:, idx2));
xlim([0 3.2]);  ylim([-30 30]); xlabel('Time[s]');  ylabel('Velocity[m/s]');
title('Bearing signal(day 59)');
subplot(3, 1, 3);       plot(t, bearingNew(:, idx3));
xlim([0 3.2]);  ylim([-30 30]); xlabel('Time[s]');  ylabel('Velocity[m/s]');
title('Bearing signal(Repaired, day 60)');
```

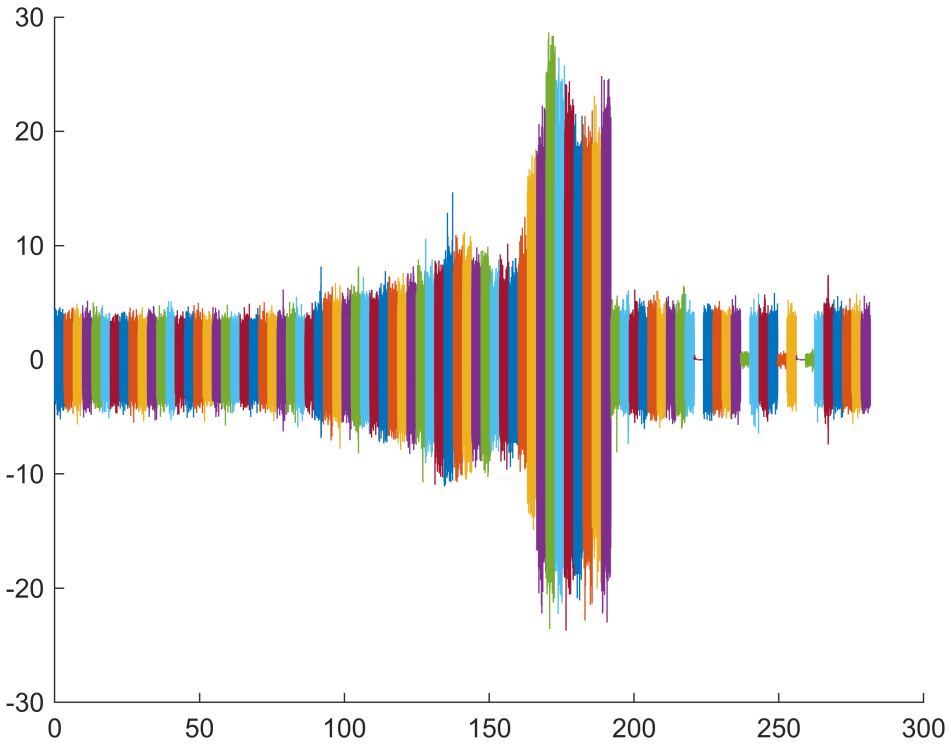


1. Data Exploration and Preprocessing

Time domain visualization

This code is for visualizing time-domain measurements of bearing data. The dataset is measured 24 times per day, and each measurement consists of L=8192L=8192L=8192 samples, sampled at Fs=2560HzFs = 2560HzFs=2560Hz. The code plots each day's measurements on the time axis, allowing comparison of data over multiple days.

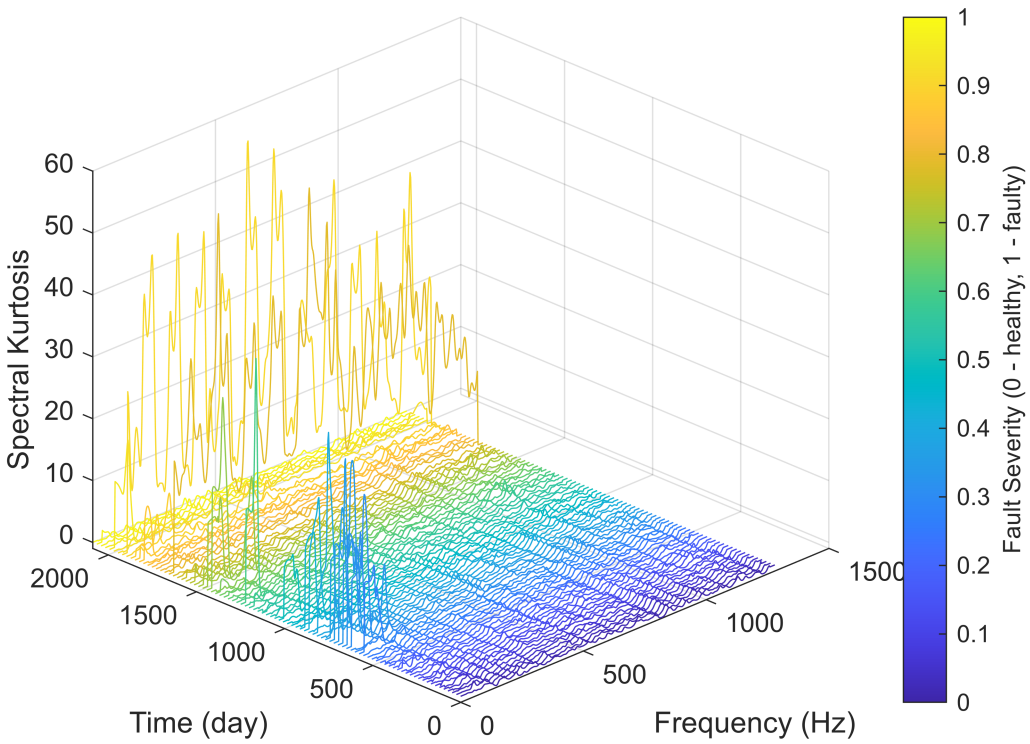
```
figure
hold on;
tstart = 0;
for day = 1:24:2104
    t = tstart + (1:L)/fs;
    plot(t, bearingFulldata(:,day))
    tstart = t(end);
end
hold off;
```



2. Spectral Kurtosis

Visualizing Spectral Kurtosis (SK) across different time measurements for bearing fault data. The goal is to analyze how the kurtosis changes over time and frequency, which helps in identifying faulty conditions in the system

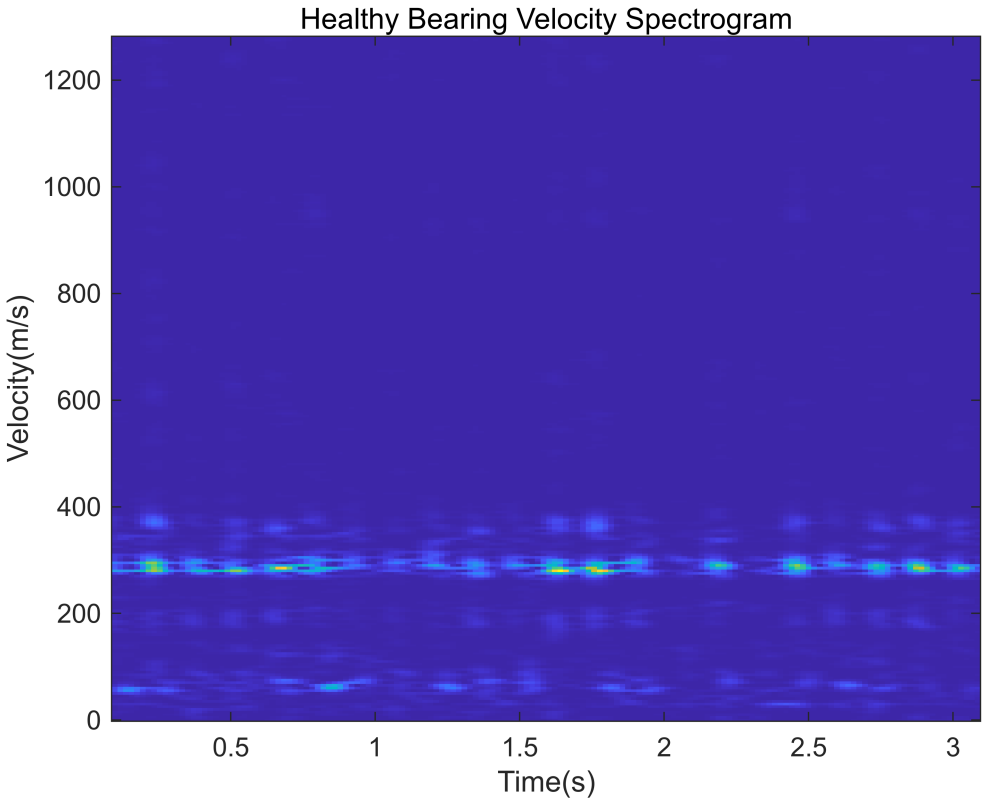
```
colors = parula(2104);
figure
hold on
wc = 128;
for times = 1:24:2104
    [SK, Fout] = pkurtosis(bearingFulldata(:,times), fs, wc);
    a = times*ones(size(Fout));
    plot3(Fout, times*ones(size(Fout)), SK, 'Color', colors(times, :))
end
hold off
xlabel('Frequency (Hz)')
ylabel('Time (day)')
zlabel('Spectral Kurtosis')
grid on
view(-45, 30)
cbar = colorbar;
ylabel(cbar, 'Fault Severity (0 - healthy, 1 - faulty)')
```



3. Spectrogram Analysis

Performing a spectrogram analysis on two sets of bearing data: one representing a healthy bearing and the other representing a faulty bearing. The goal is to visualize the time-frequency representation of the velocity data and compare the peak frequencies of both healthy and faulty bearings.

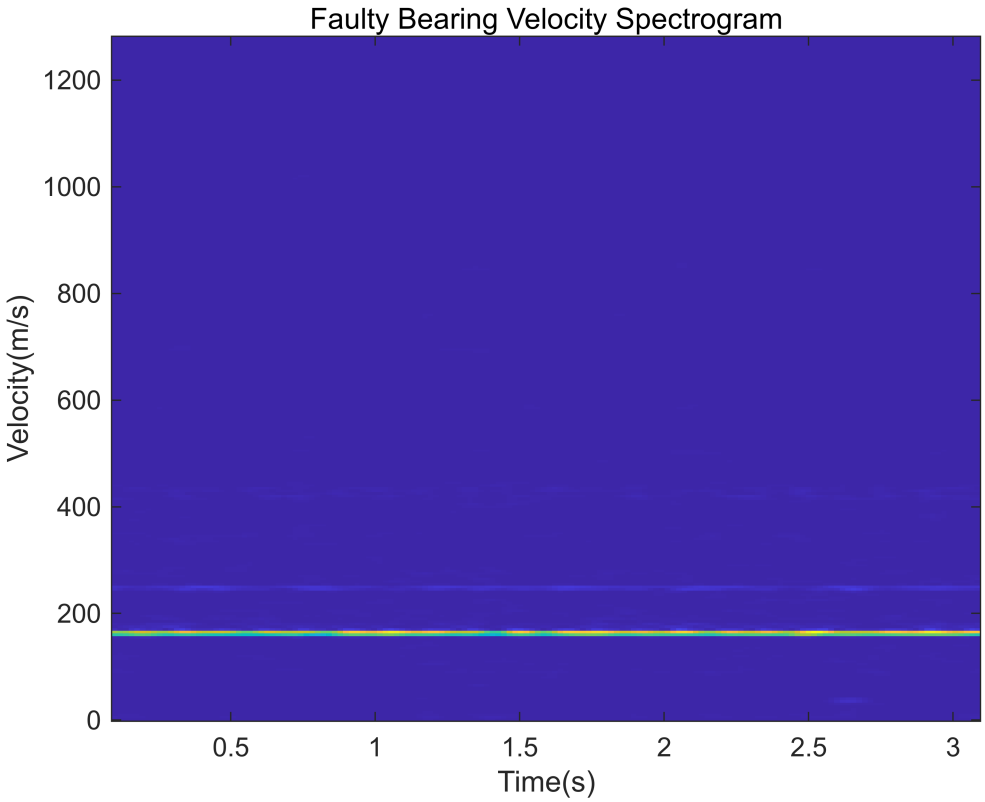
```
filtered_bearing = medfilt1(bearingNew(:, 1),5);

[~,fvec,tvec,P0] = spectrogram(filtered_bearing(:,1),500,450,512,fs);
clf;
imagesc(tvec,fvec,P0)
xlabel('Time(s)');
ylabel('Velocity(m/s)');
title('Healthy Bearing Velocity Spectrogram');
axis xy
```




```
[~,I0] = max(P0);                % Find out where the peak frequencies are located.
meanPeakFreq0 = mean(fvec(I0))   % Calculate mean peak frequency.
```

```
meanPeakFreq0 = 264.7078
```

```
filtered_bearing = medfilt1(bearing(:, 1420),5);

[~,fvec,tvec,P0] = spectrogram(filtered_bearing(:,1),500,450,512,fs);
clf;
imagesc(tvec,fvec,P0)
xlabel('Time(s)');
ylabel('Velocity(m/s)');
title('Faulty Bearing Velocity Spectrogram');
axis xy
```




```
[~,I0] = max(P0);                % Find out where the peak frequencies are located.
meanPeakFreq0 = mean(fvec(I0))   % Calculate mean peak frequency.
```

```
meanPeakFreq0 = 165
```

4. Feature Extraction and Analysis

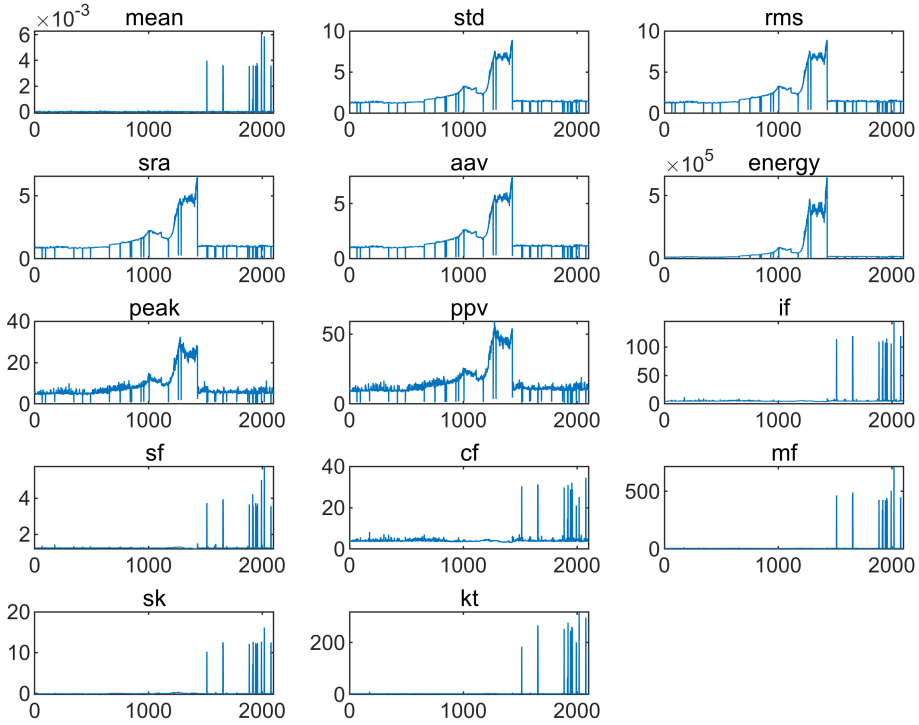
Time Domain Feature Extraction

This code performs feature extraction and visualization for bearing data in the time domain. It extracts time-domain features for 2104 time steps, plots them, and then applies a moving average filter to smooth the extracted features before plotting them again.

```
bearing_time_features = table;
for times = 1:2104
    bearing_time_features(times,:) = timeFeatures(bearingFulldata(:,times));
end
```

```
feature_names = {'mean', 'std', 'rms', 'sra', 'aav', 'energy', 'peak', 'ppv', 'if',
'sf', 'cf', 'mf', 'sk', 'kt'};
x = 1:2104;

figure;
hold on;
for i = 1:14
    subplot(5,3,i)
```

```
    plot(x, table2array(bearing_time_features(:, i)))
    title(feature_names{i});
end
hold off;
```

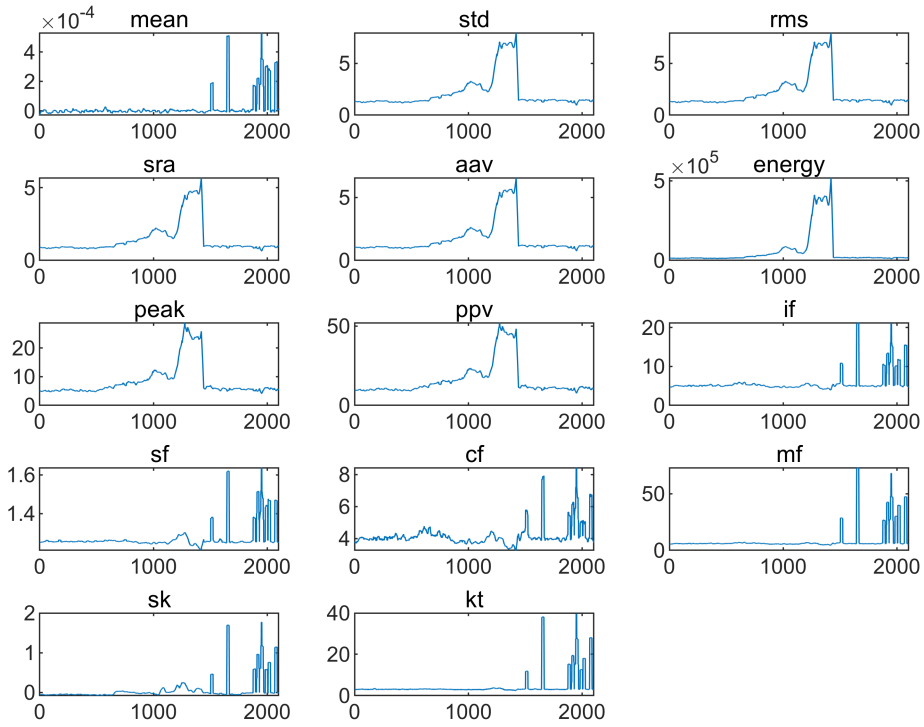


Smoothing - moving average filter applied

This code snippet is designed to smooth the time-domain features of the bearing data using a moving average filter, then visualize the smoothed features for `fc`, `rmsf`, and `rvf` over 2104 time steps.

```
windowSize = 21;

bearing_time_features_mv = bearing_time_features;
for i = 1:width(bearing_time_features)
    bearing_time_features_mv{:, i} = movmean(bearing_time_features{:, i},
windowSize);
end

feature_names = {'mean', 'std', 'rms', 'sra', 'aav', 'energy', 'peak', 'ppv', 'if',
'sf', 'cf', 'mf', 'sk', 'kt'};
x = 1:2104;
figure;
hold on;
for i = 1:14
    subplot(5,3,i)
    plot(x, table2array(bearing_time_features_mv(:, i)))
    title(feature_names{i});
```

```
end
hold off;
```

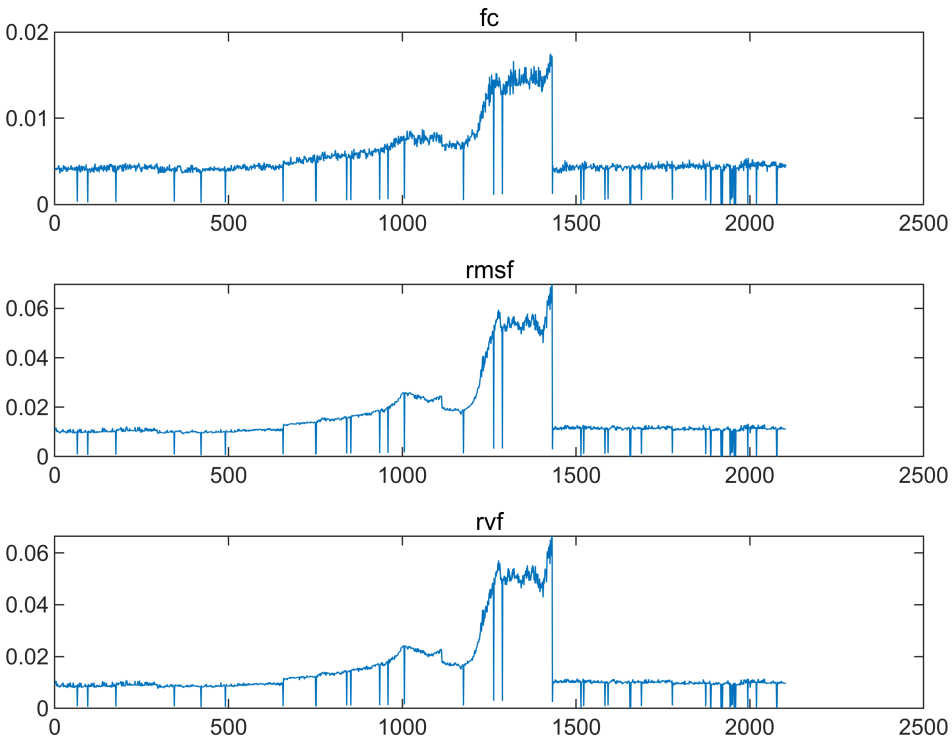


5. Frequency Domain Feature Extraction

This code is designed to perform feature extraction in the frequency domain on bearing data, extracting three specific frequency-domain features: frequency center (`fc`), RMS frequency (`rmsf`), and root variance frequency (`rvf`). It then plots these features for 2104 time steps to visualize the trends over time.

```
bearing_freq_features = table;
for times = 1:2104
    bearing_freq_features(times,:) = freqFeatures(bearingFulldata(:,times));
end
% row 1 = fc, row 2 = rmsf, row 3 = rvf
```

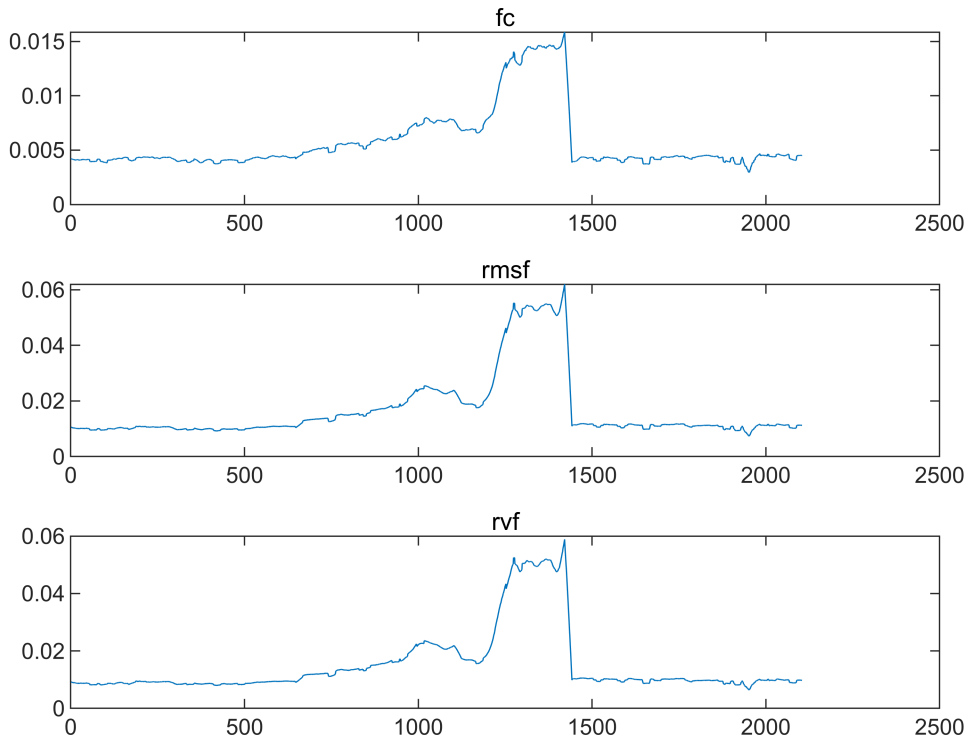
```
feature_names = {'fc', 'rmsf', 'rvf'};
x = 1:2104;
figure;
hold on;
for i = 1:3
    subplot(3,1,i)
    plot(x, table2array(bearing_freq_features(:, i)))
    title(feature_names{i});
end
hold off;
```



Smoothing Datasets - moving average filter applied

This code snippet is designed to smooth the frequency-domain features of the bearing data using a moving average filter, then visualize the smoothed features for `fc`, `rmsf`, and `rvf` over 2104 time steps.

```
bearing_freq_features_mv = bearing_freq_features;
for i = 1:width(bearing_freq_features)
    bearing_freq_features_mv{:, i} = movmean(bearing_freq_features{:, i},
windowSize);
end

feature_names = {'fc', 'rmsf', 'rvf'};
x = 1:2104;
figure;
hold on;
for i = 1:3
    subplot(3,1,i)
    plot(x, table2array(bearing_freq_features_mv(:, i)))
    title(feature_names{i});
end
hold off;
```



6. Feature Postprocessing

Feature Smoothing & Datetime Adding

This code is focused on postprocessing the features by combining both time-domain and frequency-domain features, applying a moving average filter, and creating a timetable for visualization and further analysis.

```
bearing_features = horzcat(bearing_freq_features, bearing_time_features);

windowSize = 5;

bearing_features_mv = bearing_features;
for i = 1:width(bearing_features)
    bearing_features_mv{:, i} = movmean(bearing_features{:, i}, windowSize);
end

tm = datetime(2024, 4, 14) + days(1:59);
t = table;
i = 1;
for k = 1:59
    t(k,:) = bearing_features_mv(i,:);
    i  = i + 24;
end
t.time = tm'
```

t = 59×18 table

...

	fc	rmsf	rvf	mean	std	rms	sra	aav
1	0.0042	0.0111	0.0099	-4.2318e-05	1.4216	1.4215	0.9641	1.1362
2	0.0042	0.0103	0.0089	-1.5625e-05	1.3221	1.3221	0.8940	1.0552
3	0.0042	0.0100	0.0086	-9.5215e-06	1.2818	1.2817	0.8651	1.0220
4	0.0041	0.0102	0.0089	-2.8809e-05	1.3104	1.3104	0.8811	1.0424
5	0.0033	0.0083	0.0073	-1.3379e-05	1.0630	1.0629	0.7188	0.8488
6	0.0040	0.0099	0.0086	8.0566e-06	1.2694	1.2693	0.8526	1.0096
7	0.0042	0.0106	0.0092	-2.6855e-06	1.3521	1.3520	0.9103	1.0770
8	0.0042	0.0102	0.0088	-1.4893e-05	1.3046	1.3045	0.8820	1.0412
9	0.0043	0.0108	0.0094	-5.8594e-06	1.3828	1.3828	0.9358	1.1045
10	0.0044	0.0108	0.0094	-9.2773e-06	1.3813	1.3812	0.9265	1.0971
11	0.0045	0.0108	0.0093	-5.6152e-06	1.3816	1.3816	0.9276	1.0981
12	0.0043	0.0108	0.0095	-1.0742e-05	1.3885	1.3884	0.9374	1.1067
13	0.0044	0.0108	0.0093	2.1484e-05	1.3771	1.3770	0.9291	1.0975
14	0.0039	0.0096	0.0083	4.6387e-06	1.2254	1.2253	0.8276	0.9775
15	0.0040	0.0100	0.0087	-7.5684e-06	1.2841	1.2840	0.8633	1.0217
16	0.0043	0.0103	0.0088	-1.7578e-05	1.3176	1.3175	0.8882	1.0495
17	0.0040	0.0098	0.0085	4.8828e-06	1.2516	1.2515	0.8460	0.9992
18	0.0040	0.0099	0.0086	3.6621e-06	1.2622	1.2621	0.8514	1.0064
19	0.0041	0.0101	0.0088	-1.8066e-05	1.2936	1.2935	0.8716	1.0303
20	0.0041	0.0097	0.0083	2.0752e-05	1.2424	1.2423	0.8402	0.9917
21	0.0041	0.0099	0.0085	1.3916e-05	1.2651	1.2650	0.8499	1.0062
22	0.0041	0.0101	0.0088	-1.6846e-05	1.2940	1.2939	0.8747	1.0328
23	0.0044	0.0108	0.0093	7.5684e-06	1.3768	1.3767	0.9207	1.0906
24	0.0042	0.0105	0.0091	2.9297e-06	1.3447	1.3446	0.9037	1.0692
25	0.0044	0.0108	0.0094	1.2207e-05	1.3856	1.3856	0.9201	1.0930
26	0.0045	0.0108	0.0093	1.8066e-05	1.3873	1.3872	0.9260	1.0990
27	0.0043	0.0107	0.0093	6.1035e-06	1.3731	1.3730	0.9209	1.0902
28	0.0044	0.0107	0.0092	1.8311e-05	1.3724	1.3723	0.9215	1.0901
29	0.0047	0.0130	0.0116	-1.3916e-05	1.6623	1.6622	1.1187	1.3219
30	0.0051	0.0135	0.0119	-8.3008e-06	1.7243	1.7242	1.1617	1.3729
31	0.0054	0.0136	0.0119	-2.4414e-07	1.7401	1.7400	1.1724	1.3845
32	0.0055	0.0138	0.0120	7.3242e-07	1.7643	1.7642	1.1889	1.4051

	fc	rmsf	rvf	mean	std	rms	sra	aav
33	0.0059	0.0154	0.0136	-1.8799e-05	1.9687	1.9686	1.3238	1.5641
34	0.0054	0.0147	0.0131	1.5869e-05	1.8780	1.8779	1.2666	1.4968
35	0.0057	0.0154	0.0138	6.1035e-06	1.9729	1.9728	1.3329	1.5743
36	0.0044	0.0127	0.0115	4.5654e-06	1.6217	1.6216	1.1028	1.2982
37	0.0058	0.0166	0.0150	2.4414e-05	2.1305	2.1303	1.4482	1.7060
38	0.0062	0.0173	0.0155	-2.9297e-06	2.2099	2.2098	1.5004	1.7695
39	0.0060	0.0177	0.0161	7.8125e-06	2.2678	2.2677	1.5437	1.8175
40	0.0051	0.0152	0.0138	-2.0996e-06	1.9405	1.9403	1.3166	1.5519
41	0.0054	0.0163	0.0149	-1.6602e-06	2.0812	2.0810	1.4165	1.6671
42	0.0073	0.0223	0.0204	-1.5625e-05	2.8490	2.8489	1.9496	2.2913
43	0.0074	0.0256	0.0239	8.0566e-06	3.2747	3.2745	2.2210	2.6200
44	0.0078	0.0247	0.0228	1.0254e-05	3.1604	3.1602	2.1506	2.5335
45	0.0079	0.0240	0.0220	1.4648e-06	3.0758	3.0756	2.0899	2.4629
46	0.0078	0.0225	0.0204	5.6152e-06	2.8824	2.8822	1.9623	2.3099
47	0.0076	0.0235	0.0216	1.6846e-05	3.0112	3.0110	2.0425	2.4077
48	0.0070	0.0192	0.0172	-1.4404e-05	2.4537	2.4535	1.6838	1.9760
49	0.0067	0.0191	0.0173	1.7822e-05	2.4495	2.4494	1.6529	1.9527
50	0.0056	0.0151	0.0134	-6.7627e-06	1.9274	1.9273	1.2761	1.5181
51	0.0083	0.0215	0.0190	-9.7656e-07	2.7544	2.7542	1.8346	2.1738
52	0.0100	0.0327	0.0303	-2.1484e-05	4.1898	4.1895	2.6669	3.2091
53	0.0132	0.0452	0.0422	-4.8828e-07	5.7895	5.7891	3.6980	4.4607
54	0.0143	0.0562	0.0534	1.3428e-05	7.1918	7.1914	4.5192	5.4843
55	0.0135	0.0524	0.0497	-1.2207e-05	6.7064	6.7060	4.3763	5.2337
56	0.0146	0.0552	0.0522	5.1270e-06	7.0708	7.0704	4.8566	5.7106
57	0.0146	0.0524	0.0492	1.0254e-05	6.7064	6.7059	4.6752	5.4492
58	0.0148	0.0535	0.0503	-3.9062e-06	6.8500	6.8495	4.6218	5.4710
59	0.0143	0.0506	0.0475	-1.4893e-05	6.4830	6.4826	4.5066	5.2646

```
timeTable = table2timetable(t, "RowTimes","time")
```

```
timeTable = 59×17 timetable
```

...

	time	fc	rmsf	rvf	mean	std	rms	sra
1	2024-04-15	0.0042	0.0111	0.0099	-4.2318e-05	1.4216	1.4215	0.9641
2	2024-04-16	0.0042	0.0103	0.0089	-1.5625e-05	1.3221	1.3221	0.8940

	time	fc	rmsf	rvf	mean	std	rms	sra
3	2024-04-17	0.0042	0.0100	0.0086	-9.5215e-06	1.2818	1.2817	0.8651
4	2024-04-18	0.0041	0.0102	0.0089	-2.8809e-05	1.3104	1.3104	0.8811
5	2024-04-19	0.0033	0.0083	0.0073	-1.3379e-05	1.0630	1.0629	0.7188
6	2024-04-20	0.0040	0.0099	0.0086	8.0566e-06	1.2694	1.2693	0.8526
7	2024-04-21	0.0042	0.0106	0.0092	-2.6855e-06	1.3521	1.3520	0.9103
8	2024-04-22	0.0042	0.0102	0.0088	-1.4893e-05	1.3046	1.3045	0.8820
9	2024-04-23	0.0043	0.0108	0.0094	-5.8594e-06	1.3828	1.3828	0.9358
10	2024-04-24	0.0044	0.0108	0.0094	-9.2773e-06	1.3813	1.3812	0.9265
11	2024-04-25	0.0045	0.0108	0.0093	-5.6152e-06	1.3816	1.3816	0.9276
12	2024-04-26	0.0043	0.0108	0.0095	-1.0742e-05	1.3885	1.3884	0.9374
13	2024-04-27	0.0044	0.0108	0.0093	2.1484e-05	1.3771	1.3770	0.9291
14	2024-04-28	0.0039	0.0096	0.0083	4.6387e-06	1.2254	1.2253	0.8276
15	2024-04-29	0.0040	0.0100	0.0087	-7.5684e-06	1.2841	1.2840	0.8633
16	2024-04-30	0.0043	0.0103	0.0088	-1.7578e-05	1.3176	1.3175	0.8882
17	2024-05-01	0.0040	0.0098	0.0085	4.8828e-06	1.2516	1.2515	0.8460
18	2024-05-02	0.0040	0.0099	0.0086	3.6621e-06	1.2622	1.2621	0.8514
19	2024-05-03	0.0041	0.0101	0.0088	-1.8066e-05	1.2936	1.2935	0.8716
20	2024-05-04	0.0041	0.0097	0.0083	2.0752e-05	1.2424	1.2423	0.8402
21	2024-05-05	0.0041	0.0099	0.0085	1.3916e-05	1.2651	1.2650	0.8499
22	2024-05-06	0.0041	0.0101	0.0088	-1.6846e-05	1.2940	1.2939	0.8747
23	2024-05-07	0.0044	0.0108	0.0093	7.5684e-06	1.3768	1.3767	0.9207
24	2024-05-08	0.0042	0.0105	0.0091	2.9297e-06	1.3447	1.3446	0.9037
25	2024-05-09	0.0044	0.0108	0.0094	1.2207e-05	1.3856	1.3856	0.9201
26	2024-05-10	0.0045	0.0108	0.0093	1.8066e-05	1.3873	1.3872	0.9260
27	2024-05-11	0.0043	0.0107	0.0093	6.1035e-06	1.3731	1.3730	0.9209
28	2024-05-12	0.0044	0.0107	0.0092	1.8311e-05	1.3724	1.3723	0.9215
29	2024-05-13	0.0047	0.0130	0.0116	-1.3916e-05	1.6623	1.6622	1.1187
30	2024-05-14	0.0051	0.0135	0.0119	-8.3008e-06	1.7243	1.7242	1.1617
31	2024-05-15	0.0054	0.0136	0.0119	-2.4414e-07	1.7401	1.7400	1.1724
32	2024-05-16	0.0055	0.0138	0.0120	7.3242e-07	1.7643	1.7642	1.1889
33	2024-05-17	0.0059	0.0154	0.0136	-1.8799e-05	1.9687	1.9686	1.3238
34	2024-05-18	0.0054	0.0147	0.0131	1.5869e-05	1.8780	1.8779	1.2666
35	2024-05-19	0.0057	0.0154	0.0138	6.1035e-06	1.9729	1.9728	1.3329

	time	fc	rmsf	rvf	mean	std	rms	sra
36	2024-05-20	0.0044	0.0127	0.0115	4.5654e-06	1.6217	1.6216	1.1028
37	2024-05-21	0.0058	0.0166	0.0150	2.4414e-05	2.1305	2.1303	1.4482
38	2024-05-22	0.0062	0.0173	0.0155	-2.9297e-06	2.2099	2.2098	1.5004
39	2024-05-23	0.0060	0.0177	0.0161	7.8125e-06	2.2678	2.2677	1.5437
40	2024-05-24	0.0051	0.0152	0.0138	-2.0996e-06	1.9405	1.9403	1.3166
41	2024-05-25	0.0054	0.0163	0.0149	-1.6602e-06	2.0812	2.0810	1.4165
42	2024-05-26	0.0073	0.0223	0.0204	-1.5625e-05	2.8490	2.8489	1.9496
43	2024-05-27	0.0074	0.0256	0.0239	8.0566e-06	3.2747	3.2745	2.2210
44	2024-05-28	0.0078	0.0247	0.0228	1.0254e-05	3.1604	3.1602	2.1506
45	2024-05-29	0.0079	0.0240	0.0220	1.4648e-06	3.0758	3.0756	2.0899
46	2024-05-30	0.0078	0.0225	0.0204	5.6152e-06	2.8824	2.8822	1.9623
47	2024-05-31	0.0076	0.0235	0.0216	1.6846e-05	3.0112	3.0110	2.0425
48	2024-06-01	0.0070	0.0192	0.0172	-1.4404e-05	2.4537	2.4535	1.6838
49	2024-06-02	0.0067	0.0191	0.0173	1.7822e-05	2.4495	2.4494	1.6529
50	2024-06-03	0.0056	0.0151	0.0134	-6.7627e-06	1.9274	1.9273	1.2761
51	2024-06-04	0.0083	0.0215	0.0190	-9.7656e-07	2.7544	2.7542	1.8346
52	2024-06-05	0.0100	0.0327	0.0303	-2.1484e-05	4.1898	4.1895	2.6669
53	2024-06-06	0.0132	0.0452	0.0422	-4.8828e-07	5.7895	5.7891	3.6980
54	2024-06-07	0.0143	0.0562	0.0534	1.3428e-05	7.1918	7.1914	4.5192
55	2024-06-08	0.0135	0.0524	0.0497	-1.2207e-05	6.7064	6.7060	4.3763
56	2024-06-09	0.0146	0.0552	0.0522	5.1270e-06	7.0708	7.0704	4.8566
57	2024-06-10	0.0146	0.0524	0.0492	1.0254e-05	6.7064	6.7059	4.6752
58	2024-06-11	0.0148	0.0535	0.0503	-3.9062e-06	6.8500	6.8495	4.6218
59	2024-06-12	0.0143	0.0506	0.0475	-1.4893e-05	6.4830	6.4826	4.5066

Monotonicity Analyze (Feature Reduction)

This code performs a **Monotonicity Analysis** for feature reduction, aiming to select the most important features based on their monotonicity score. Here's the step-by-step explanation of each section:

```
breaktime = datetime(2024, 5, 7);
breakpoint = find(timeTable.time < breaktime, 1, 'last');
trainData = timeTable(1:breakpoint, :);
```

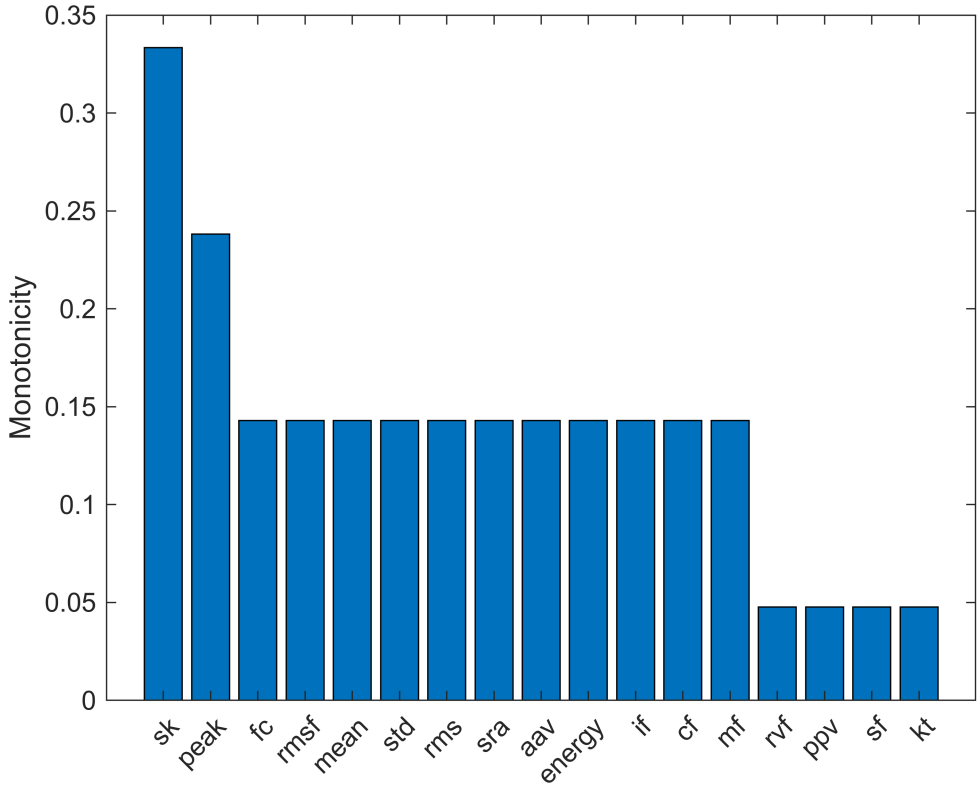
```
featureImportance = monotonicity(trainData, 'WindowSize', 0);
sum(featureImportance)
```

```
ans = 1×17 table
```

...

	fc	rmsf	rvf	mean	std	rms	sra	aav
1	0.1429	0.1429	0.0476	0.1429	0.1429	0.1429	0.1429	0.1429

```
helperSortedBarPlot(featureImportance, 'Monotonicity');
```




```
trainDataSelected = trainData(:, featureImportance{:,:}>0.2);
featureSelected = timeTable(:, featureImportance{:,:}>0.2);
```

7. Selection of Features

PCA

Performs feature selection and dimensionality reduction using Principal Component Analysis (PCA). First, it normalizes the selected training data (`trainDataSelected`) by calculating the mean and standard deviation. After normalization, PCA is applied to extract the primary components (`PCA1` and `PCA2`). The code then visualizes the data in the space of the first two principal components using a scatter plot, with a color bar indicating the time progression. Additionally, it plots `PCA1` and `PCA2` over time and designates `PCA1` as the health indicator, which is also plotted to track the machine's condition over time.

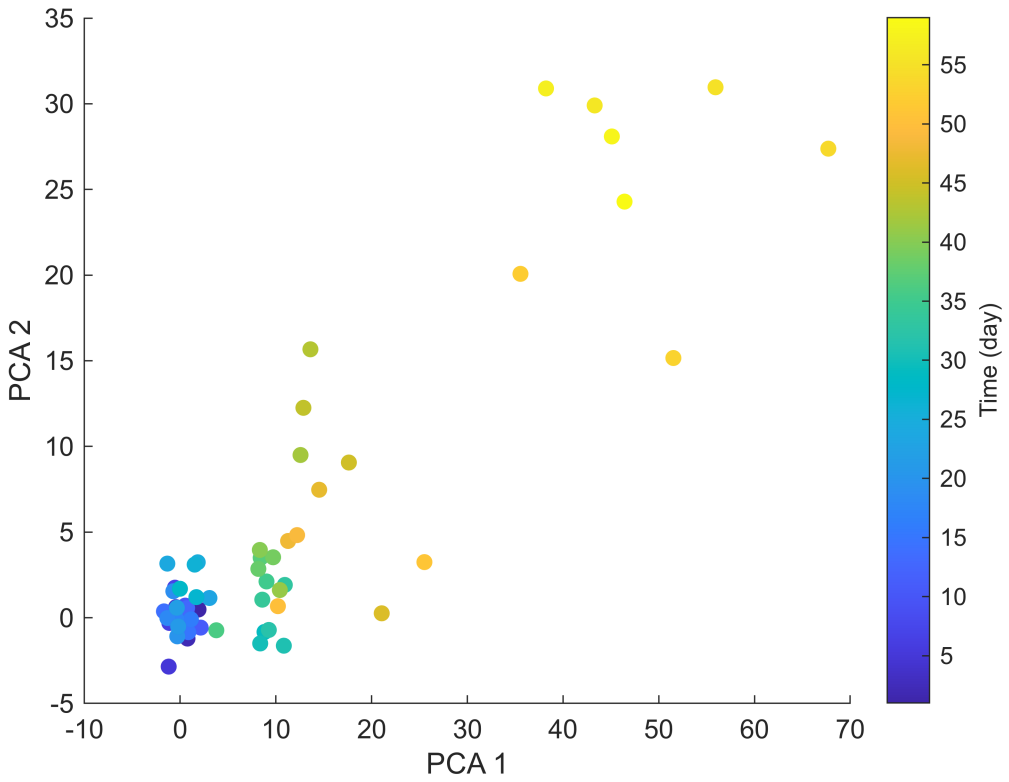
```
meanTrain = mean(trainDataSelected{:,:});
sdTrain = std(trainDataSelected{:,:});
trainDataNormalized = (trainDataSelected{:,:} - meanTrain)./sdTrain;
coef = pca(trainDataNormalized);
```

The mean, standard deviation and PCA coefficients are used to process the entire data set.

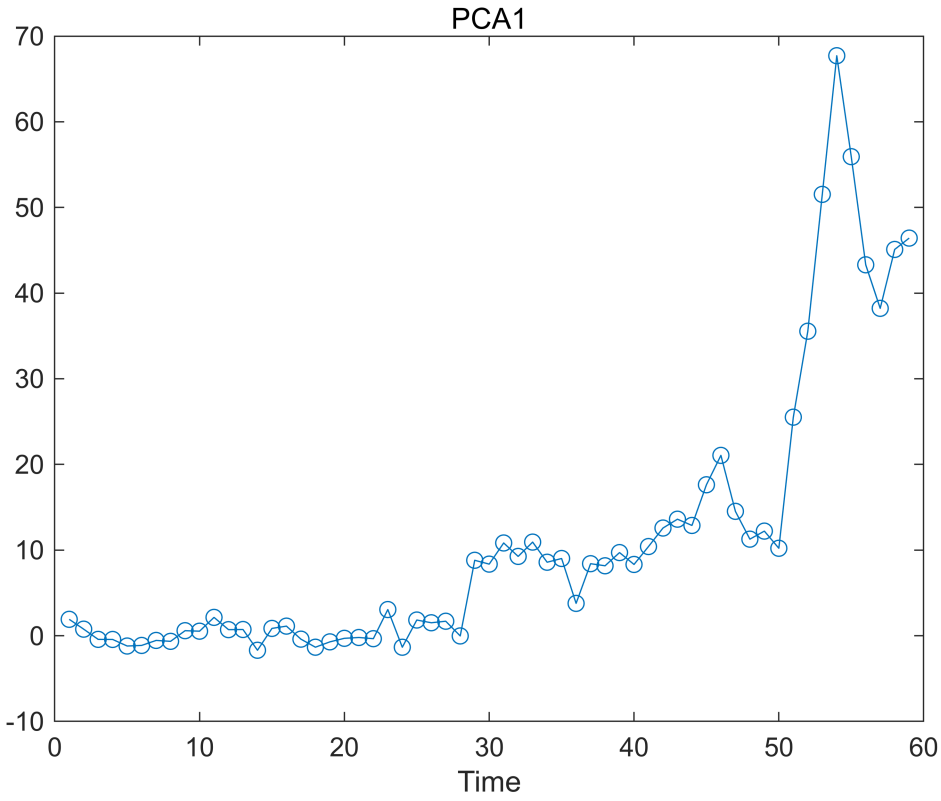
```
PCA1 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 1);
PCA2 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 2);
```

Visualize the data in the space of the first two principal components.

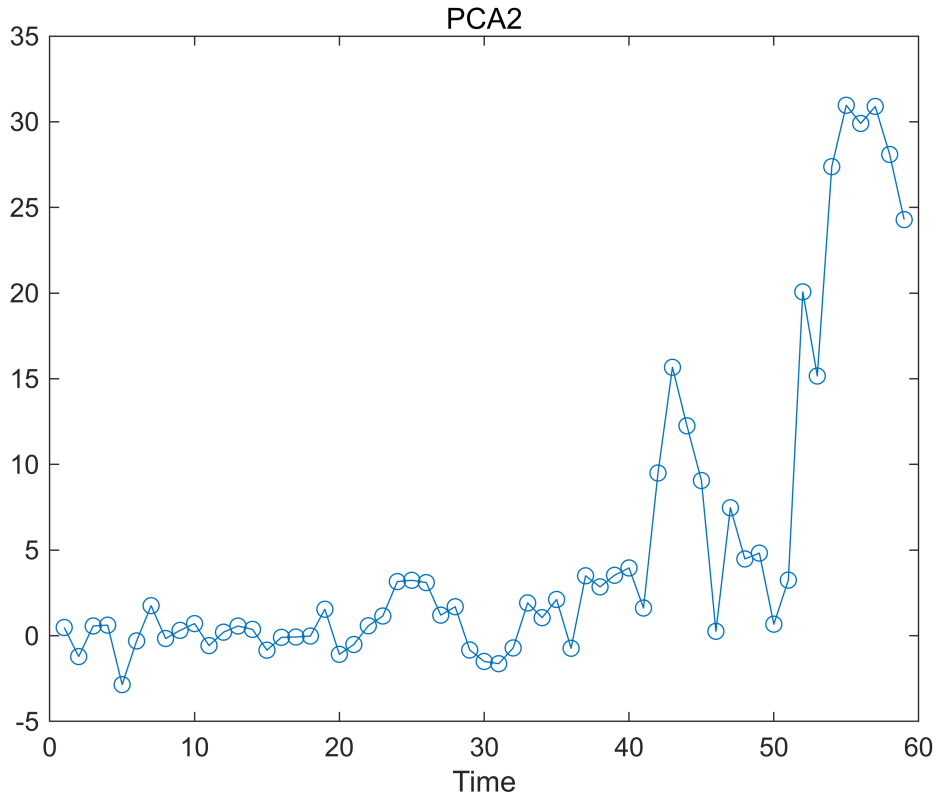
```
figure
numData = size(timeTable, 1);
scatter(PCA1, PCA2, [], 1:numData, 'filled')
xlabel('PCA 1')
ylabel('PCA 2')
cbar = colorbar;
timeUnit = 'day';
ylabel(cbar, ['Time (' timeUnit ')'])
```




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



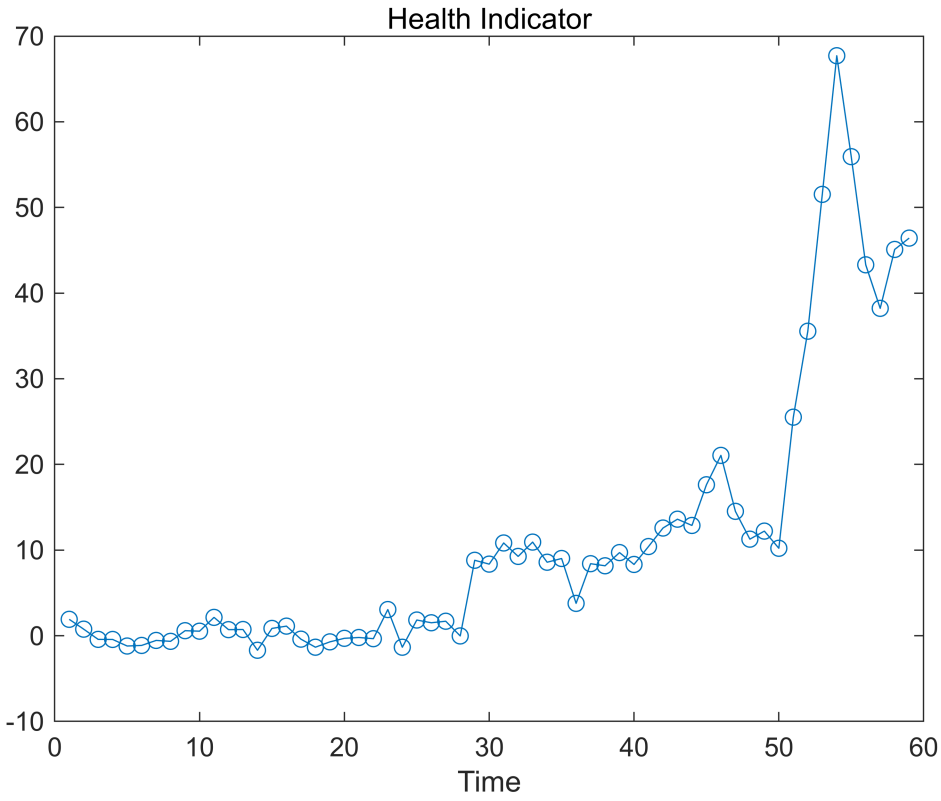

```
figure
plot(PCA2, '-o')
xlabel('Time')
title('PCA2')
```




```
healthIndicator = PCA1;
```

Visualize the health indicator.

```
figure
plot(healthIndicator, '-o')
xlabel('Time')
title('Health Indicator')
```



8. Fault Monitoring and Warning

Exponential degradation model is defined as

$$h(t) = \phi + \theta \exp\left(\beta t + \epsilon - \frac{\sigma^2}{2}\right)$$

where $h(t)$ is the health indicator as a function of time. ϕ is the intercept term considered as a constant. θ and β are random parameters determining the slope of the model, where θ is lognormal-distributed and β is Gaussian-distributed. At each time step t, the distribution of θ and β is updated to the posterior based on the latest observation of $h(t)$. ϵ is a Gaussian white noise yielding to $N(0, \sigma^2)$. The $-\frac{\sigma^2}{2}$ term in the exponential is to make the expectation of $h(t)$ satisfy

$$E[h(t)|\theta, \beta] = \phi + \theta \exp(\beta t).$$

Here an Exponential Degradation Model is fit to the health indicator extracted in the last section, and the performances is evaluated in the next section.

First shift the health indicator so that it starts from 0.

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

The selection of threshold is usually based on the historical records of the machine or some domain-specific knowledge. Since no historical data is available in this dataset, the last value of the health indicator is chosen as the threshold. It is recommended to choose the threshold based on the smoothed (historical) data so that the delay effect of smoothing will be partially mitigated.

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

If historical data is available, use `fit` method provided by **`exponentialDegradationModel`** to estimate the priors and intercept. However, historical data is not available for this wind turbine bearing dataset.

The prior of the slope parameters are chosen arbitrarily with large variances $(E(\theta) = 1,\ \mathrm{Var}(\theta)\ = 10^6, E(\beta) = 1, \mathrm{Var}(\beta) = 10^6)$ so that the model is mostly relying on the observed data. Based on $E[h(0)] = \phi + E(\theta)$, intercept ϕ is set to -1 so that the model will start from 0 as well.

The relationship between the variation of health indicator and the variation of noise can be derived as

$$\Delta h(t) \approx (h(t) - \phi\ )\Delta\epsilon(t)$$

Here the standard deviation of the noise is assumed to cause 10% of variation of the health indicator when it is near the threshold. Therefore, the standard deviation of the noise can be represented as $\frac{10\% \cdot \mathrm{threshold}}{\mathrm{threshold} - \phi}$.

The exponential degradation model also provides a functionality to evaluate the significance of the slope. Once a significant slope of the health indicator is detected, the model will forget the previous observations and restart the estimation based on the original priors. The sensitivity of the detection algorithm can be tuned by specifying `SlopeDetectionLevel`. If p value is less than `SlopeDetectionLevel`, the slope is declared to be detected. Here `SlopeDetectionLevel` is set to 0.05.

Now create an exponential degradation model with the parameters discussed above.

```
mdl = exponentialDegradationModel(...
    'Theta', 1, ...
    'ThetaVariance', 1e6, ...
    'Beta', 1, ...
    'BetaVariance', 1e6, ...
    'Phi', -1, ...
    'NoiseVariance', (0.1*threshold/(threshold + 1))^2, ...
    'SlopeDetectionLevel', 0.05);
```

9. RUL Estimation

You need to explain the process cleary and analyze the results

Use `predictRUL` and `update` methods to predict the RUL and update the parameter distribution in real time.

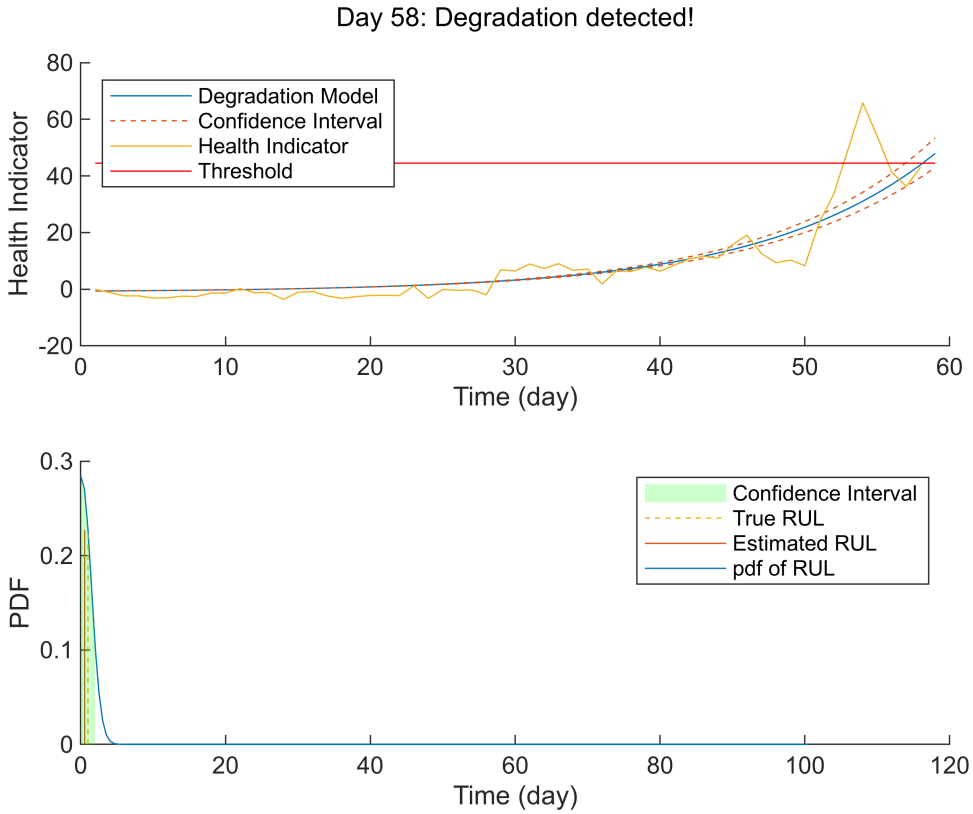
```
% Keep records at each iteration
totalDay = length(healthIndicator) - 1;
estRULs = zeros(totalDay, 1);
```

```
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
```

```
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해
보십시오.
```

경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.
경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.

경고: 데이터가 지수 모델 가정을 충족하지 않습니다. 모델의 사전 파라미터를 확인하거나 다른 성능 저하 모델을 사용해 보십시오.



Here is the animation of the real-time RUL estimation.

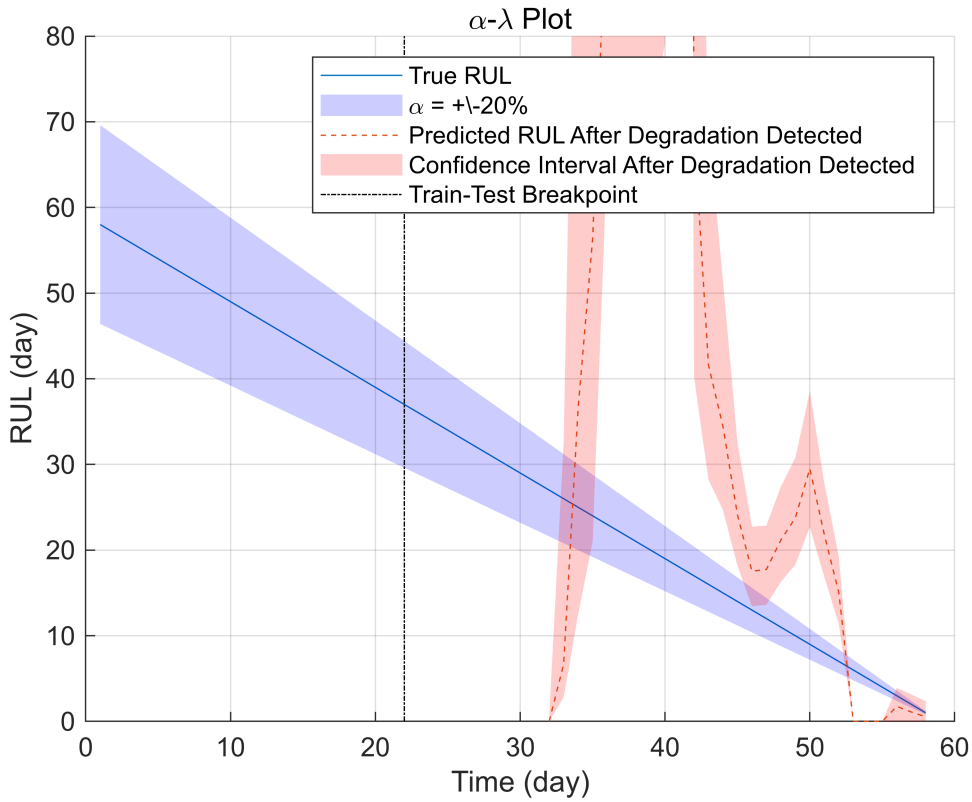
10. Performance Analysis

α-λ plot is used for prognostic performance analysis [5], where α bound is set to 20%. The probability that the estimated RUL is between the α bound of the true RUL is calculated as a performance metric of the model:

$$\Pr(r^*(t) - \alpha r^*(t) < r(t) < r^*(t) + \alpha r^*(t) | \Theta(t))$$

where $r(t)$ is the estimated RUL at time t, $r^*(t)$ is the true RUL at time t, $\Theta(t)$ is the estimated model parameters at time t.

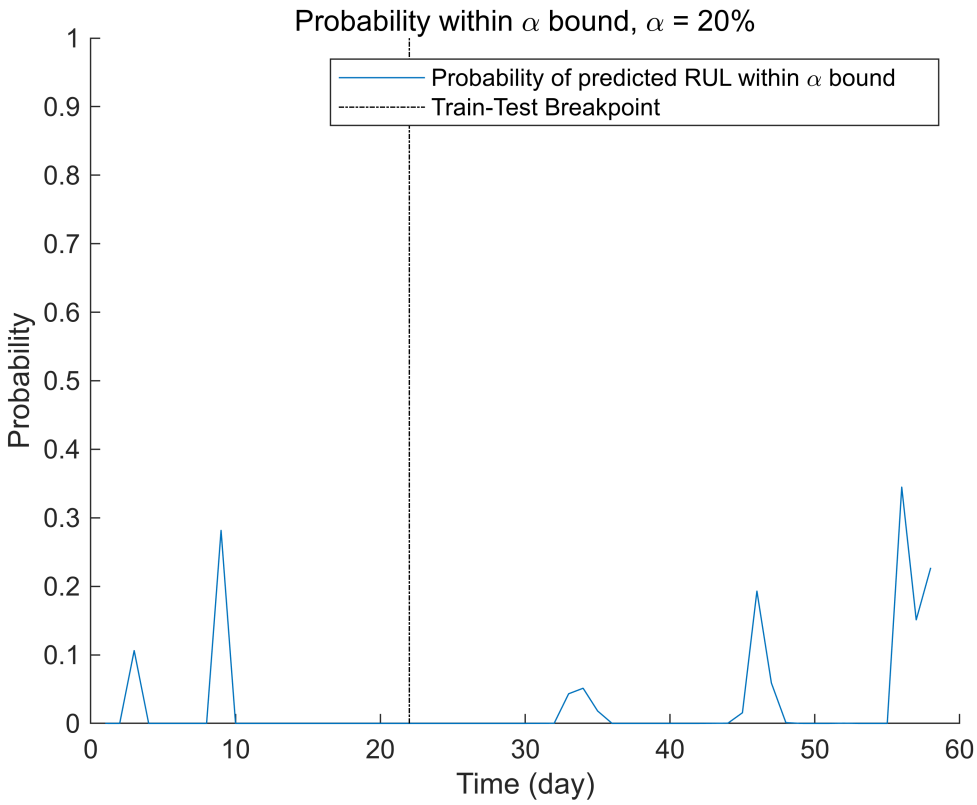
```
alpha = 0.2;
detectTime = mdl.SlopeDetectionInstant;
prob = helperAlphaLambdaPlot(alpha, trueRULs, estRULs, CIRULs, ...
    pdfRULs, detectTime, breakpoint, timeUnit);
title('\alpha-\lambda Plot')
```



Since the preset prior does not reflect the true prior, the model usually need a few time steps to adjust to a proper parameter distribution. The prediction becomes more accurate as more data points are available.

Visualize the probability of the predicted RUL within the α bound.

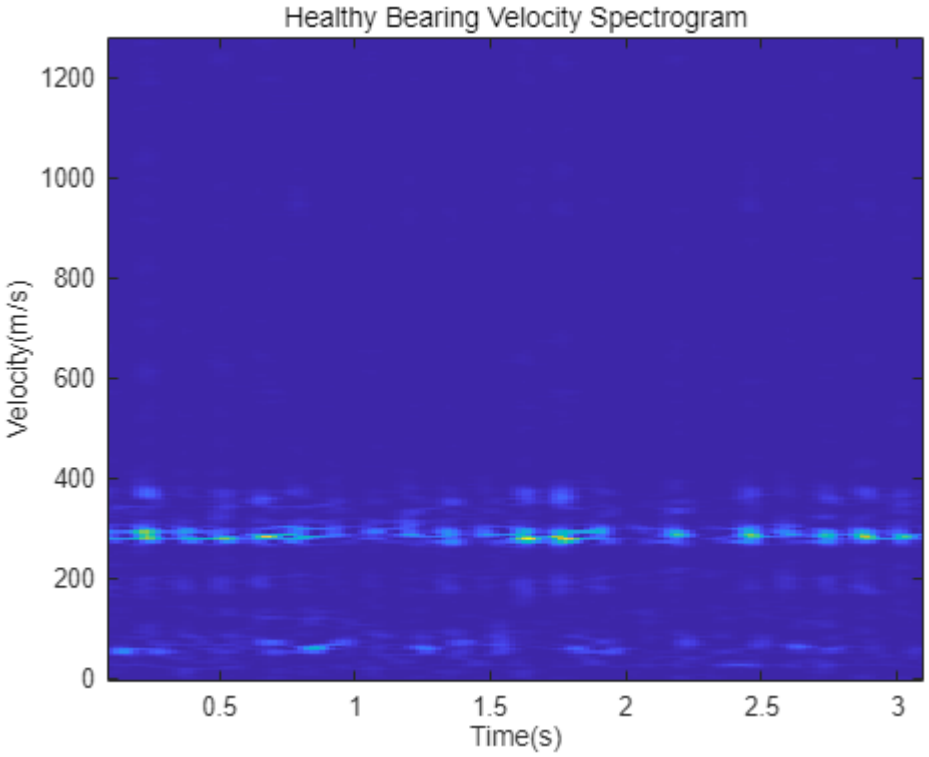
```
figure
t = 1:totalDay;
hold on
plot(t, prob)
plot([breakpoint breakpoint], [0 1], 'k-.')
hold off
xlabel(['Time (' timeUnit ')'])
ylabel('Probability')
legend('Probability of predicted RUL within \alpha bound', 'Train-Test Breakpoint')
title(['Probability within \alpha bound, \alpha = ' num2str(alpha*100)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

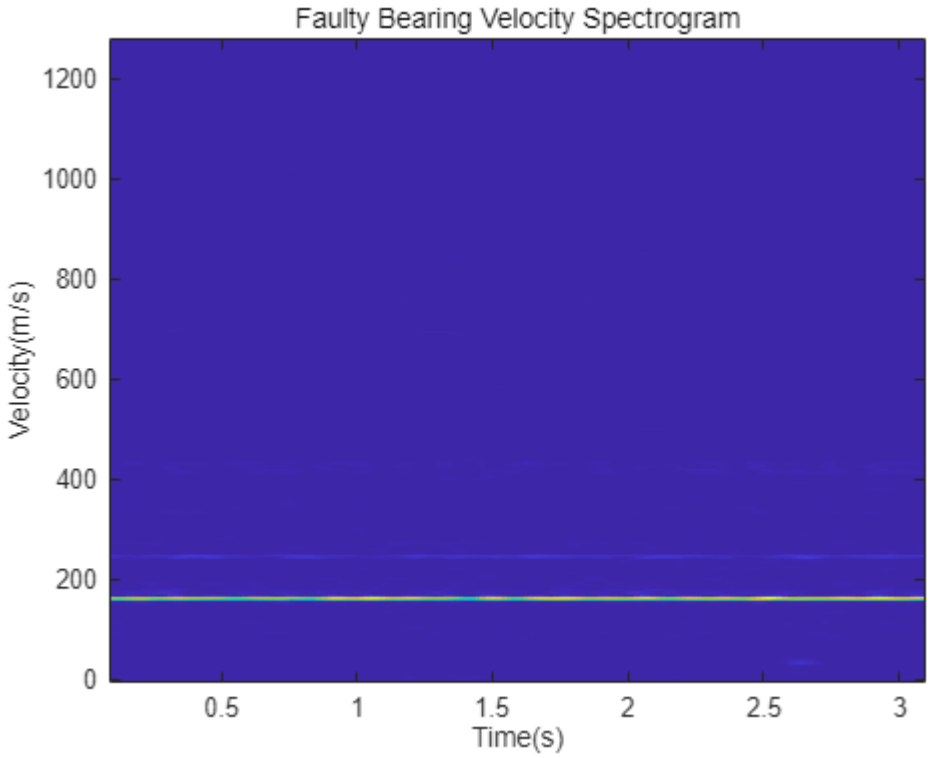


11. Discussion

11.1. Data Analysis

In this study, various techniques were employed to monitor bearing fault progression and estimate the Remaining Useful Life (RUL) using a comprehensive approach to data analysis. The process began with time-domain feature extraction, where statistical metrics such as mean, standard deviation, root mean square (RMS), and spectral kurtosis were calculated from the bearing dataset. These features enabled an initial visualization of the bearing's operational condition over time, allowing the identification of key points that could signify the onset of failure.

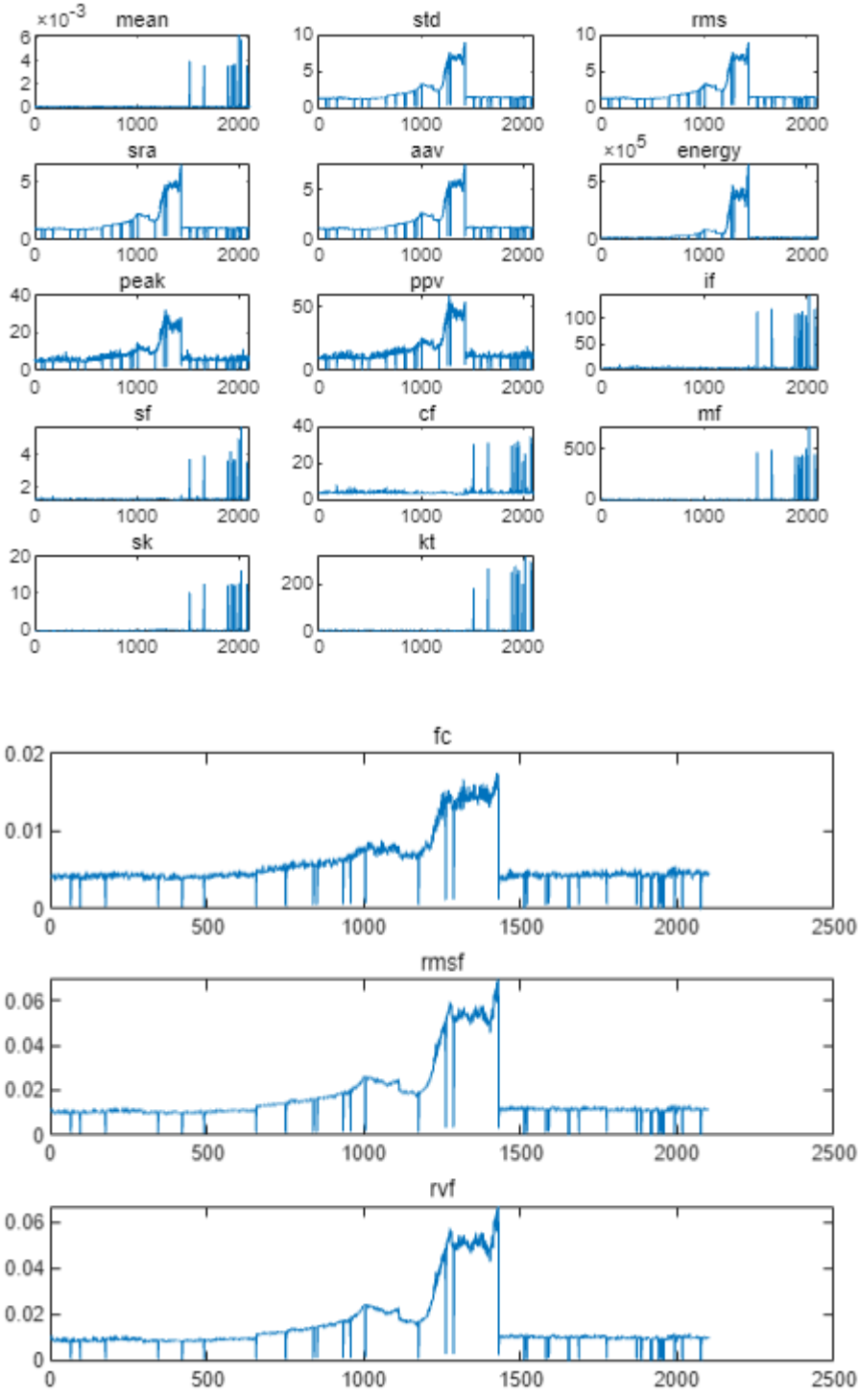




When analyzing the spectrogram, it was observed that before the bearing failure and after the repair, the velocity in the time domain ranged between 200 and 400. However, at the point of failure, the velocity dropped below 200.

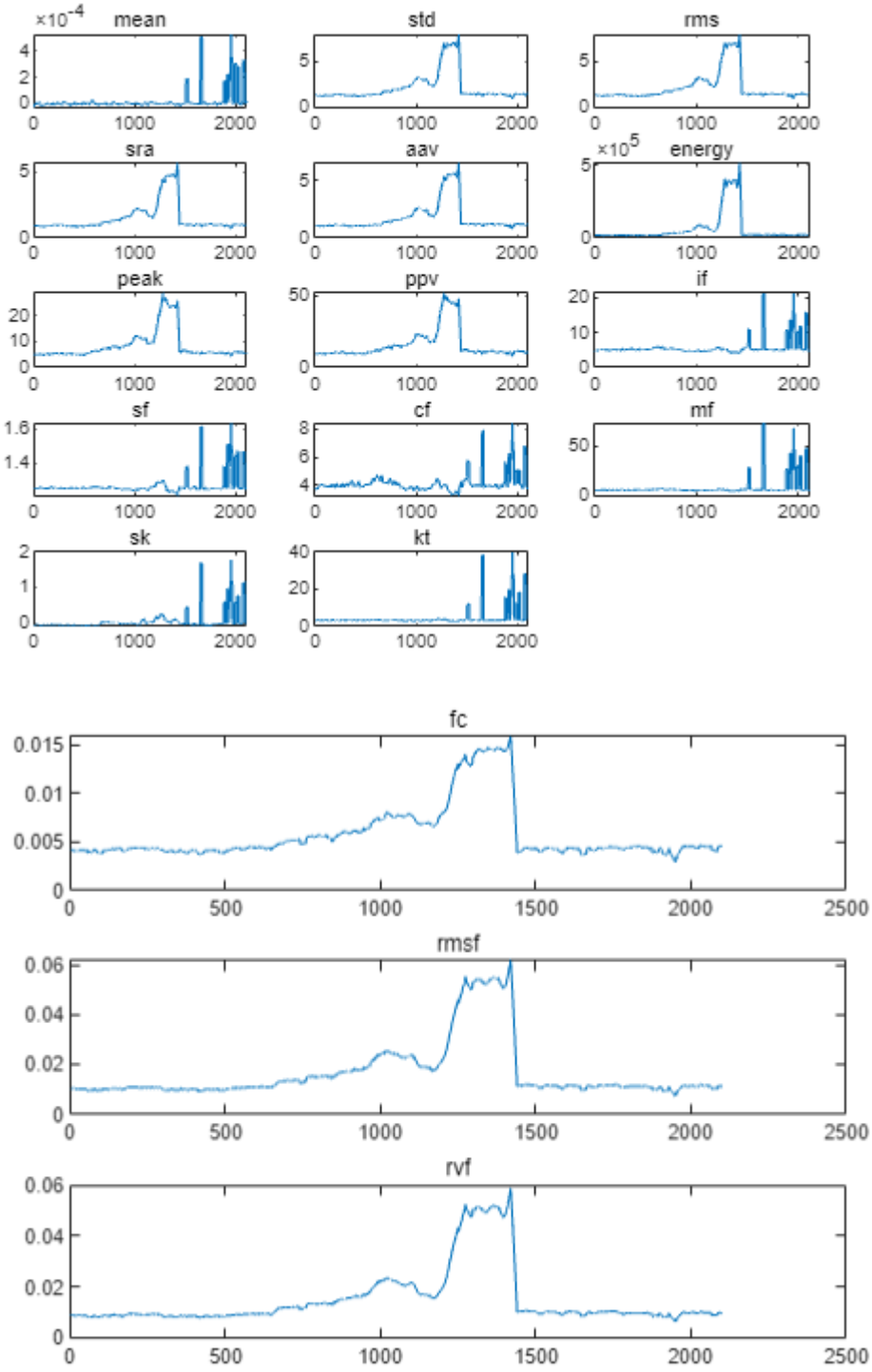
11.2. Feature Extraction & Smoothing

Following this, frequency-domain features were extracted using the Fast Fourier Transform (FFT), capturing the vibrational characteristics of the bearings. Features such as frequency center, RMS frequency, and root variance frequency provided a more detailed perspective on the dynamic behavior of the machine, supplementing the insights gained from the time-domain analysis.



When extracting and plotting the time-domain and frequency-domain features, the presence of outliers was identified. These outliers could negatively affect prognosis, so it was necessary to remove the noise.

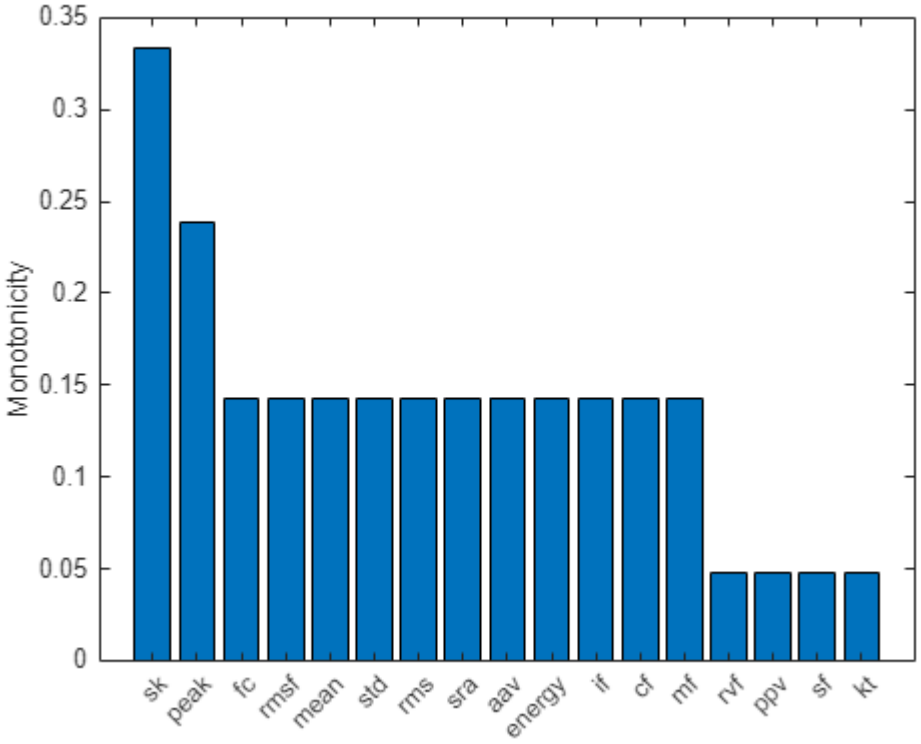
To address the presence of noise in the data, which can negatively impact RUL predictions, a moving average filter was applied. This filtering step was critical in smoothing the extracted features, reducing short-term fluctuations and ensuring that long-term trends, essential for accurate health monitoring, were preserved.



After applying the moving average filter, a significant reduction in noise was observed.

11.3. Feature Selection (Monotonicity)

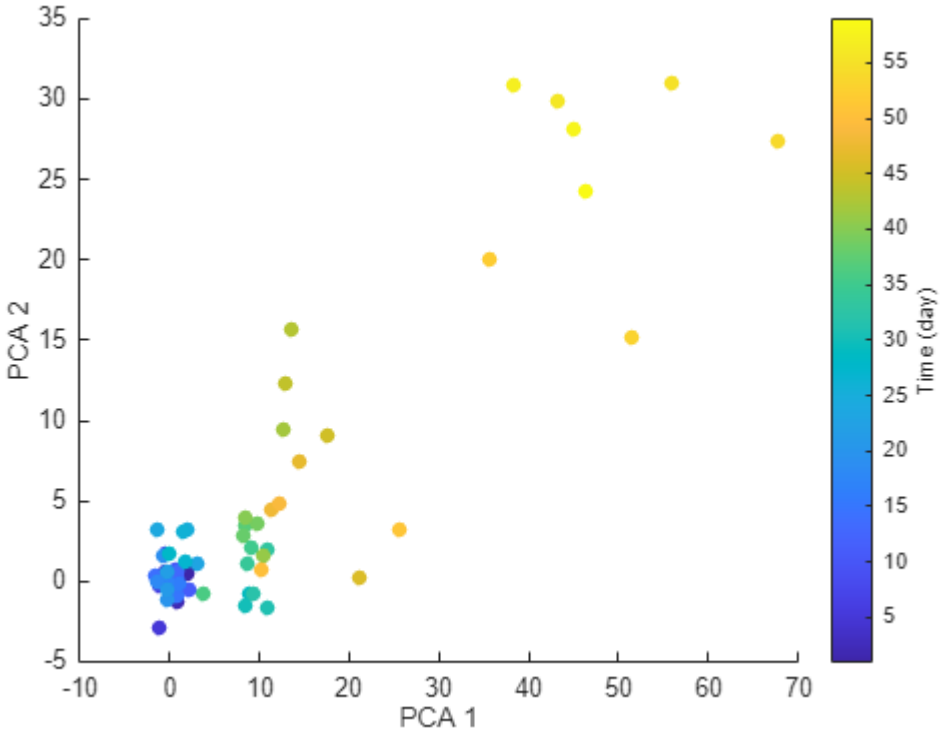
Feature selection was performed using the monotonicity criterion, prioritizing features that demonstrated a consistent trend over time as the machine approached failure. This step was instrumental in reducing the complexity of the model and focusing on the most informative features for RUL estimation.

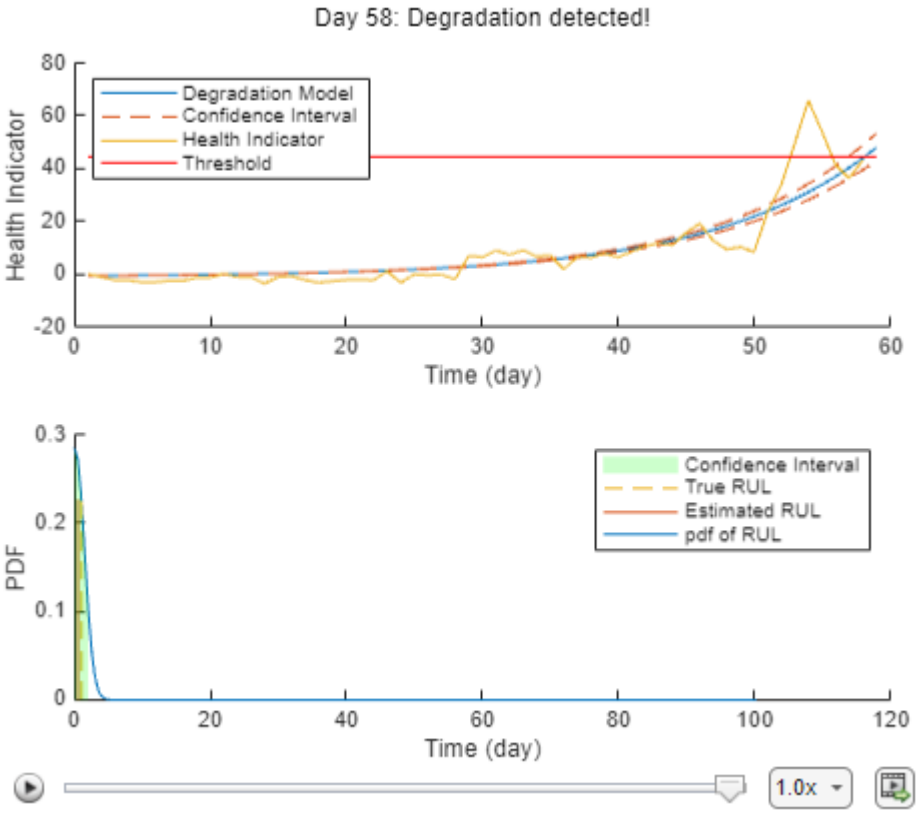


Upon examining monotonicity, it was found that the skewness and peak values showed the most consistent trends. Among these, skewness was selected for RUL estimation.

11.4. Feature Reduction (PCA) & Bearing Fault Progonosis

Subsequently, Principal Component Analysis (PCA) was utilized to further reduce the dimensionality of the feature set, retaining the most significant variance within the data. The first principal component (PCA1) emerged as a key health indicator, with its progression over time correlating strongly with the degradation of the bearing. The use of PCA1 as a health indicator provided a robust means of visualizing the machine's overall condition and identifying trends indicative of impending failure.





Based on the PCA analysis, PCA 1 was used as the health indicator, and it was successfully applied as a metric for predicting bearing failure. As a result, the bearing failure prediction was carried out successfully.

Reference