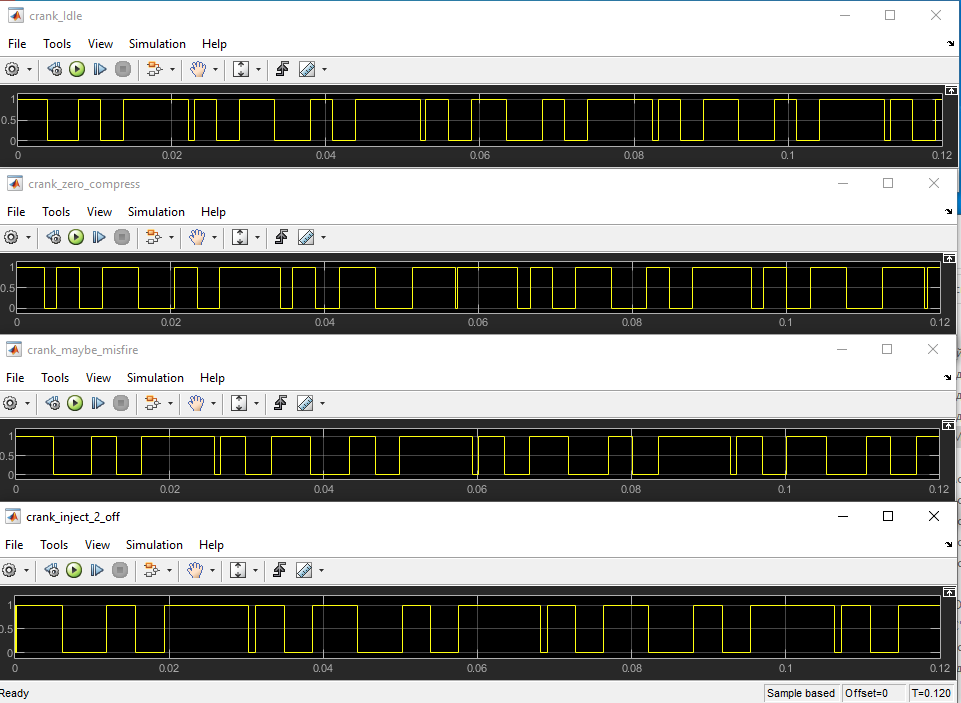
Сontent

[Part 1 Matlab Simulink processing 2](#_Toc80365918)

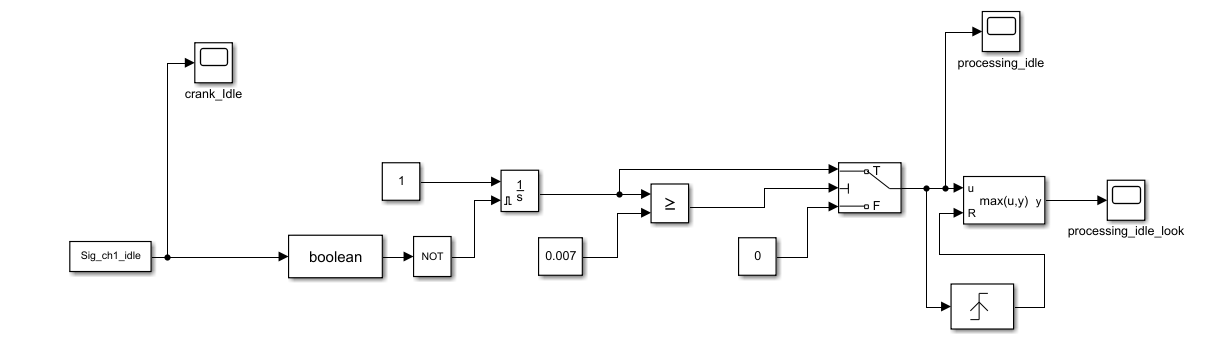
[Part 2 Python processing 6](#_Toc80365919)

[Part 3 Python code 10](#_Toc80365920)

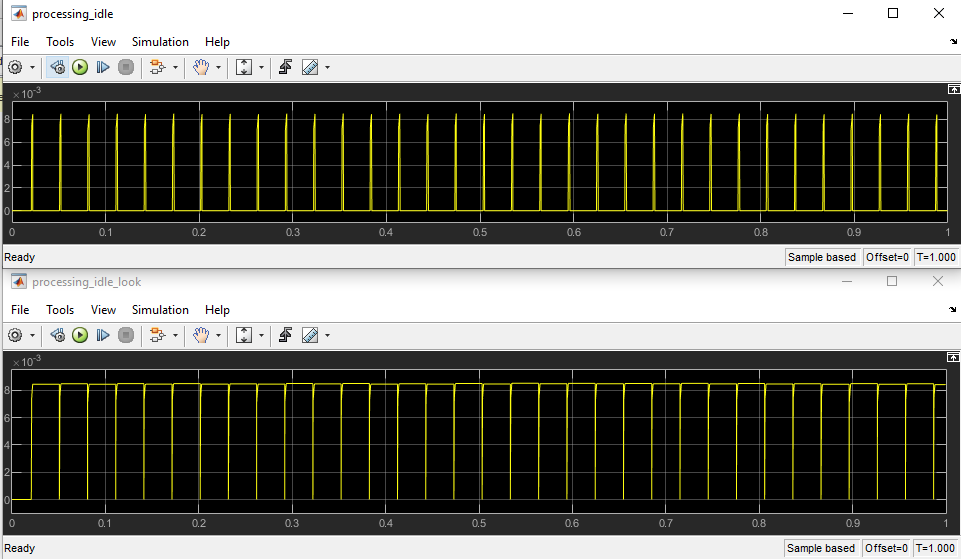
# Part 1 Matlab Simulink processing



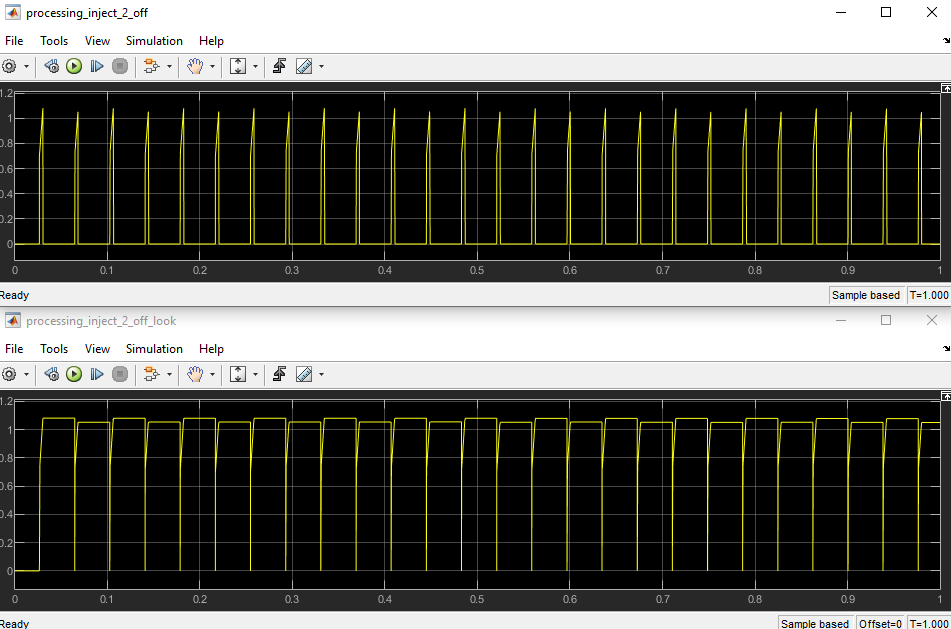
Raw\_signal\_of\_crank (matlab simulink), duration 0.12 s



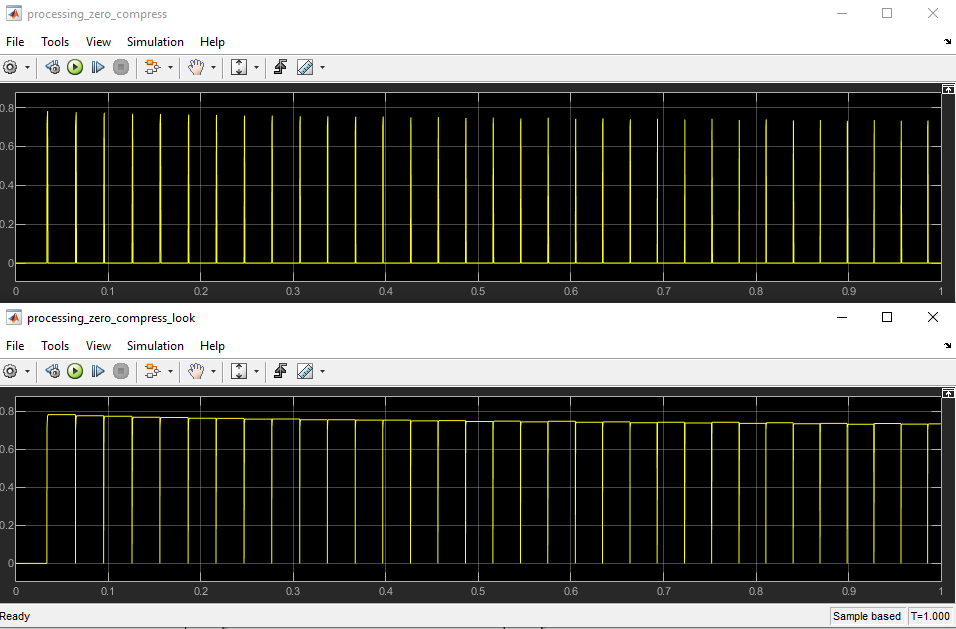
Simulink model



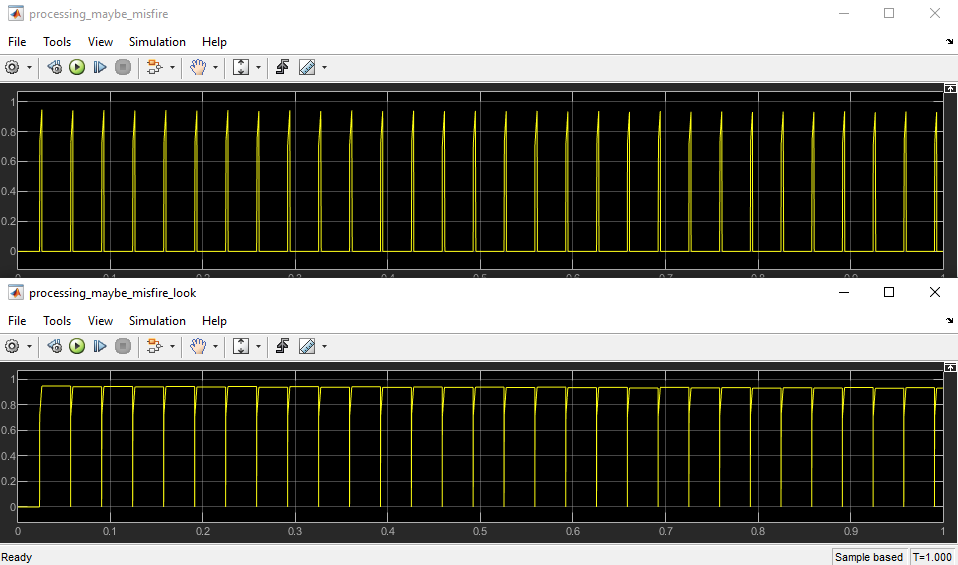
Result idle signal processing, duration 1 s



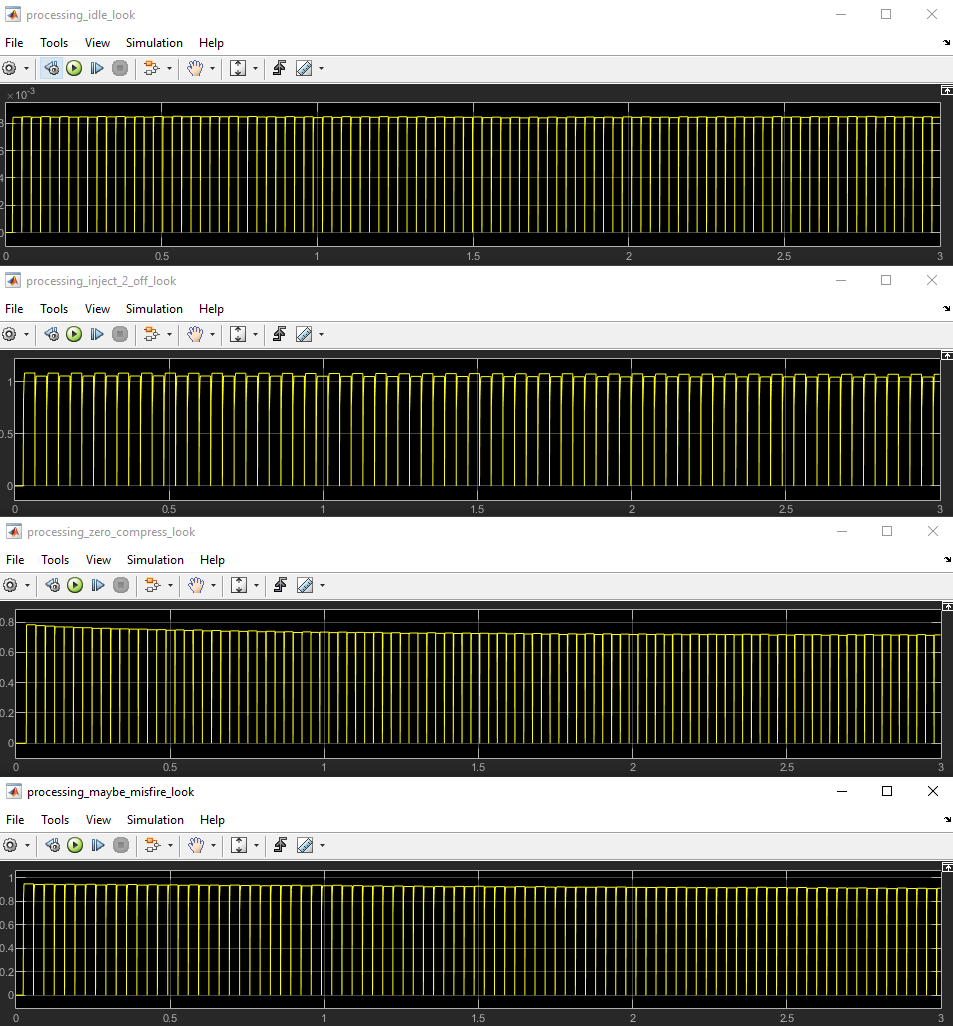
Result inject\_2 off signal processing, duration 1 s



Result zero compressing signal processing, duration 1 s



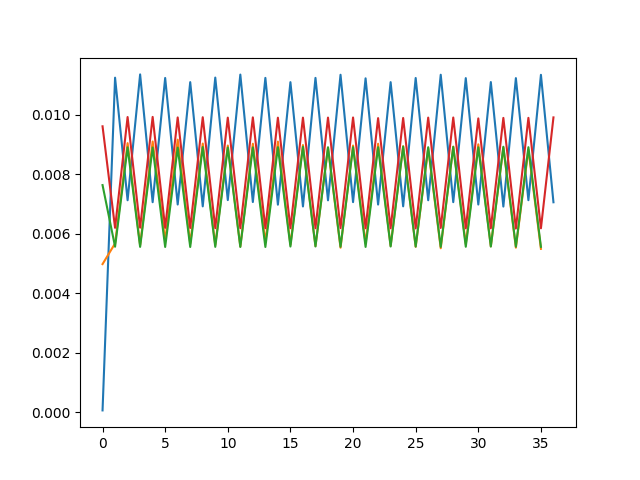
Result maybe misfire signal processing, duration 1 s

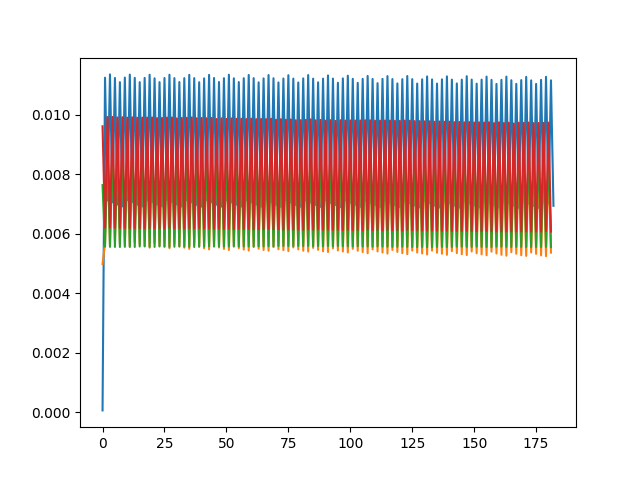


Signals processing, duration 3 s

# Part 2 Python processing

Period calc by crank signal and time series data:

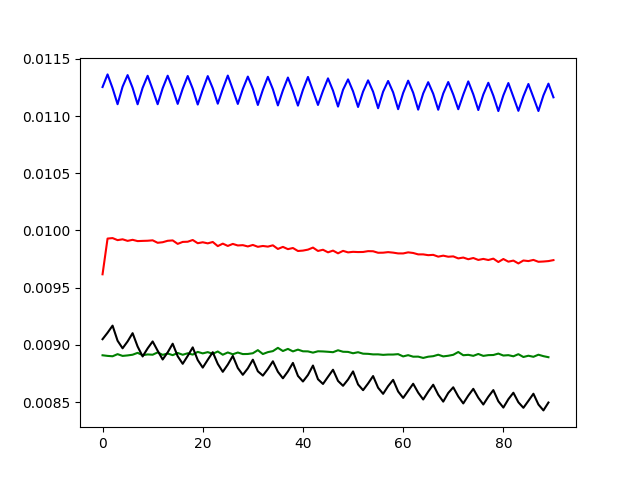
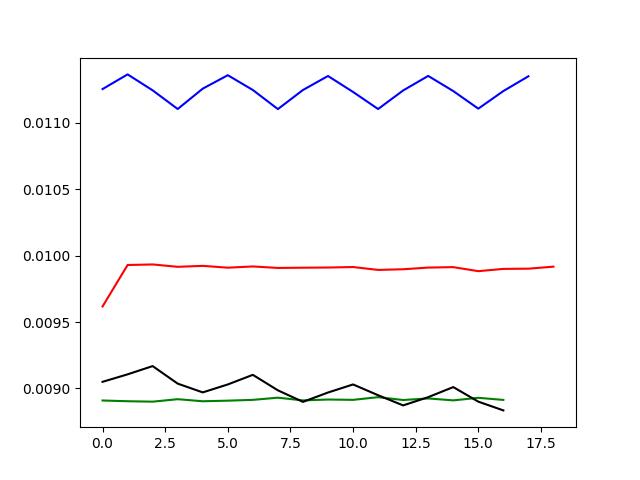


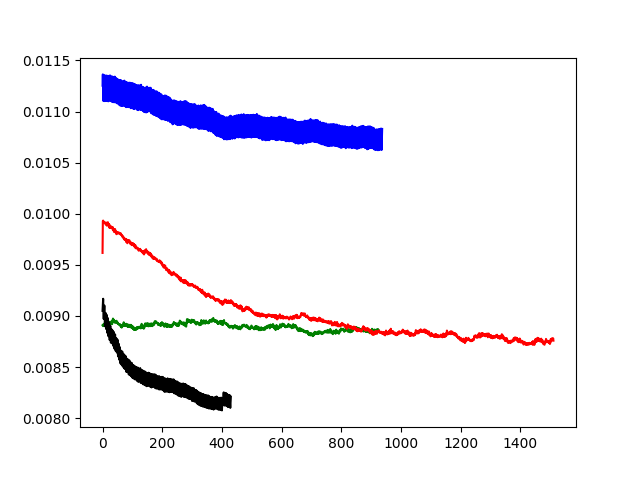
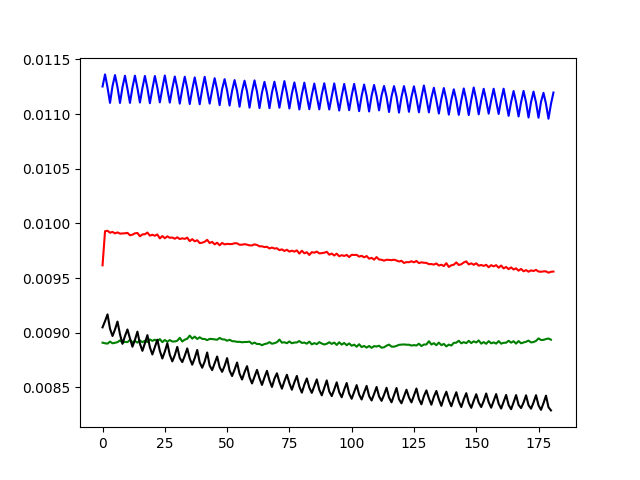


green – idle, blue – inject 2 off, yolo (bed look on picture..) – zero compression, red – suspected bad sparkplug

About Y axis – calc period, About X axis – number of data

Filtered by level (0.007 s) period data:

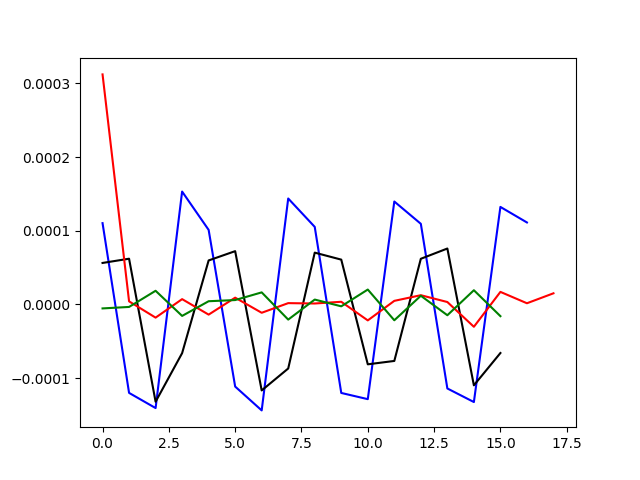


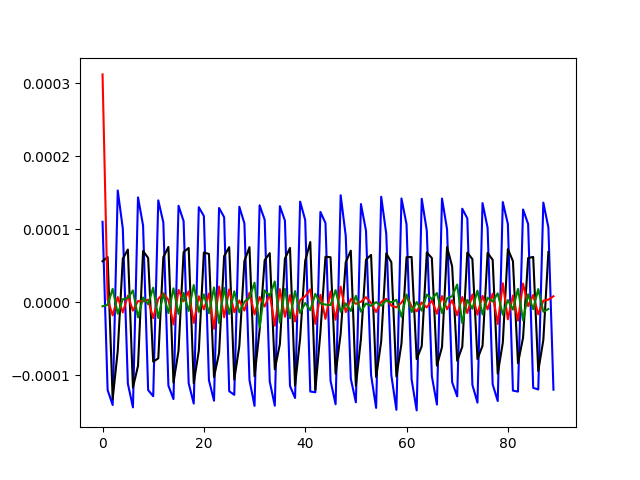


green – idle, blue – inject 2 off, black – zero compression, red – suspected bad sparkplug

About Y axis – calc period, About X axis – number of data

Filtered and diff period data:

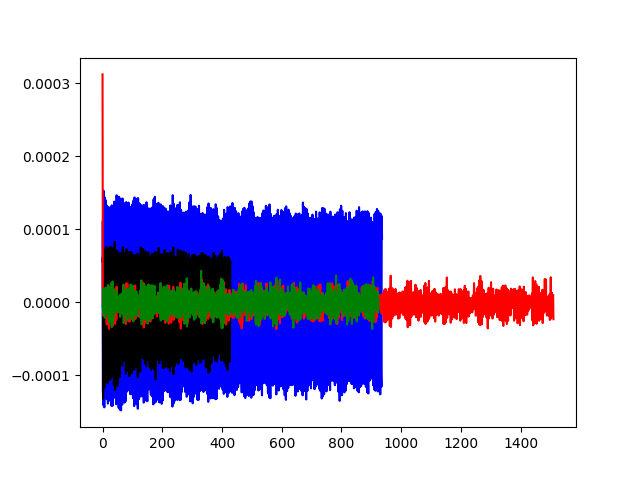
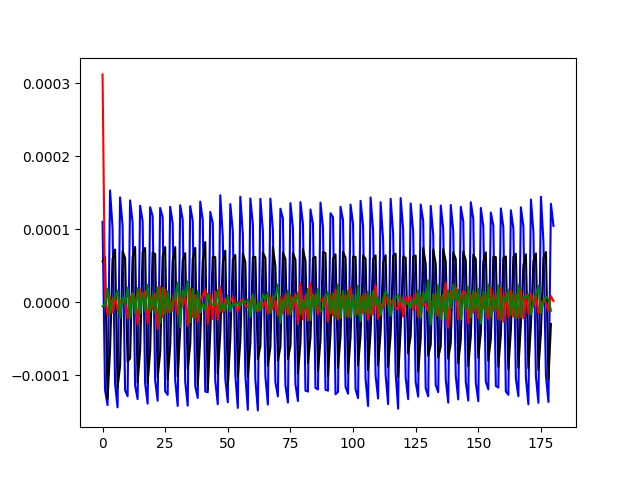




green – idle, blue – inject 2 off, black – zero compression,

red – suspected bad sparkplug

***(see next page)***



green – idle, blue – inject 2 off, black – zero compression,

red – suspected bad sparkplug

About Y axis – calc period, About X axis – number of data

# Part 3 Python code

import csv

import numpy as np

from matplotlib import pyplot as plt

path\_file\_idle = r'C:\Users\home\Documents\nb2-happy-warmup-idle.csv'

path\_file\_miss = r'C:\Users\home\Documents\nb2-idle-without-injector-2.csv'

path\_file\_zero = r'C:\Users\home\Documents\nb2-sparkplug-1-completely-removed-zero-compression-in-1.csv'

path\_file\_maby = r'C:\Users\home\Documents\nb2-suspected-bad-sparkplug-1.csv'

def parse\_data(path\_file\_idle):

data = np.array([[]])

time\_s = []

cam = []

crank = []

with open(path\_file\_idle) as csvfile:

reader = csv.DictReader(csvfile)

for row in reader:

cam.append(row['cam'])

time\_s.append(row['Time [s]'])

crank.append(row['crank'])

cam = np.array(cam)

time\_s = np.array(time\_s)

crank = np.array(crank)

cam = cam.astype(int)

time\_s = time\_s.astype(float)

crank = crank.astype(int)

#plt.plot(crank[:100])

data = np.vstack((crank, time\_s)).T

print(data.shape)

return data

data\_idle = parse\_data(path\_file\_idle)

data\_miss = parse\_data(path\_file\_miss)

data\_zero = parse\_data(path\_file\_zero)

data\_maby = parse\_data(path\_file\_maby)

def invert\_value(in\_val):

out\_val = []

for i in range(in\_val.shape[0]):

if in\_val[i] == 0:

out\_val.append(1)

else:

out\_val.append(0)

return out\_val

def calc\_period(data):

sig\_period = []

t\_0 = data[0, 1]

t\_1 = float()

print('perid in data len --> ', data.shape[0])

for i in range(data.shape[0]):# 100

if i == 0:

crank\_current = data[0, 0]

print('ferst\_data')

pass

crank\_mem = crank\_current

crank\_current = data[i, 0]

if crank\_current == crank\_mem:

pass

if crank\_current > crank\_mem:

t\_0 = data[i, 1]

if crank\_current < crank\_mem:

t\_1 = data[i, 1]

period = t\_1 - t\_0

sig\_period.append(period)

'''

if i == 2000:

break

'''

sig\_period = np.array(sig\_period)

sig\_period = sig\_period.astype(float)

print(sig\_period.shape)

return sig\_period

sig\_period\_idle = calc\_period(data\_idle)

sig\_period\_miss = calc\_period(data\_miss)

sig\_period\_zero = calc\_period(data\_zero)

sig\_period\_maby = calc\_period(data\_maby)

#print(sig\_1)

plt.figure('period')

plt.plot(sig\_period\_miss)

plt.plot(sig\_period\_zero)

plt.plot(sig\_period\_idle)

plt.plot(sig\_period\_maby)

level = 0.008

def filt\_level(level, data\_in):

data\_out = []

for i in range(data\_in.shape[0]):

if data\_in[i] >= level:

data\_out.append(data\_in[i])

else:

pass

return data\_out

filt\_level\_data\_idle = filt\_level(level, sig\_period\_idle)

filt\_level\_data\_miss = filt\_level(level, sig\_period\_miss)

filt\_level\_data\_zero = filt\_level(level, sig\_period\_zero)

filt\_level\_data\_maby = filt\_level(level, sig\_period\_maby)

plt.figure('filt')

plt.plot(filt\_level\_data\_idle, color='green')

plt.plot(filt\_level\_data\_miss, color='blue')

plt.plot(filt\_level\_data\_zero, color='black')

plt.plot(filt\_level\_data\_maby, color='red')

filt\_and\_diff\_level\_data\_idle = np.diff(filt\_level\_data\_idle)

filt\_and\_diff\_level\_data\_miss = np.diff(filt\_level\_data\_miss)

filt\_and\_diff\_level\_data\_zero = np.diff(filt\_level\_data\_zero)

filt\_and\_diff\_level\_data\_maby = np.diff(filt\_level\_data\_maby)

plt.figure('filt\_and\_diff')

plt.plot(filt\_and\_diff\_level\_data\_miss, color='blue')

plt.plot(filt\_and\_diff\_level\_data\_zero, color='black')

plt.plot(filt\_and\_diff\_level\_data\_maby, color='red')

plt.plot(filt\_and\_diff\_level\_data\_idle, color='green')

sensor\_level = 0.00007

def sensor(sensor\_level, data\_in):

miss = []

miss\_value = []

print('count\_cycle -->', data\_in.shape[0])

for i in range(data\_in.shape[0]):

if i == 0:

val\_current = data\_in[0]

pass

val\_mem = val\_current

val\_current = data\_in[i]

if abs(val\_current - val\_mem) >= sensor\_level:

miss.append(i)

miss\_value.append(abs(val\_current - val\_mem))

print('count miss --> ', len(miss))

if len(miss) <= 30:

print('numba\_of\_cycle\_miss -->', miss)

print('numba\_of\_cycle\_miss\_value -->', miss\_value)

return miss

miss\_idle = sensor(sensor\_level, filt\_and\_diff\_level\_data\_idle)

miss\_miss = sensor(sensor\_level, filt\_and\_diff\_level\_data\_miss)

miss\_miss = sensor(sensor\_level, filt\_and\_diff\_level\_data\_zero)

miss\_maby = sensor(sensor\_level, filt\_and\_diff\_level\_data\_maby)

def SKO():

pass

plt.show()

#print(cam\_idle)

#print(time\_idle)

#print(data\_idle)