Step 1: Problem Identification and Statement

This assignment aims to analyze the relationship between the GSR signal and the physiological response to fear. The GSR data is stored in the CSV format. The data contains a total of 2 recordings: one for a fearful experience (stimulated using a fearful VR simulation) and one for a fearless baseline recording. The code will inform the Baseline and Fear Index Value and will show graphs of each of the analysis made.

Step 2: Gathering of Information and Input/Output Description

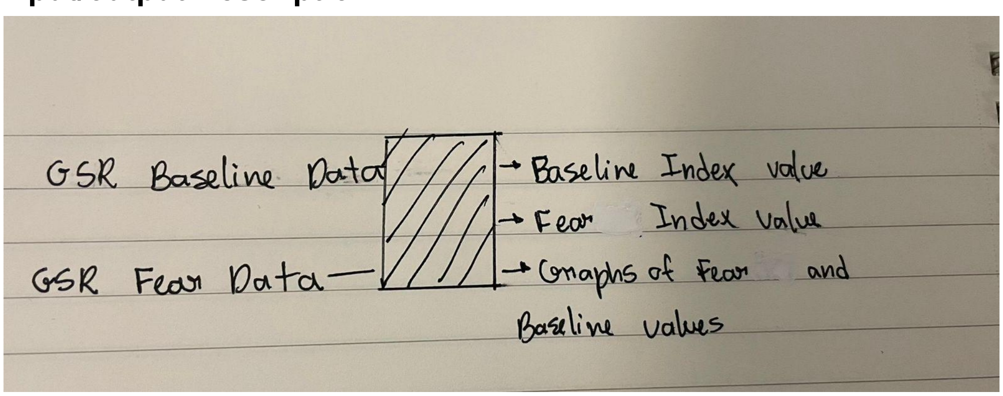
Relevant information:

The human body's sweating is regulated by the autonomic nervous system. In particular, if the sympathetic branch of the autonomic nervous system is highly aroused, then sweat gland activity also increases, which in turn raises the skin conductance, and vice versa. The galvanic skin response (also called electrodermal response, skin conductance, or psychogalvanic reflex) is the measure of the conductance caused by the variation of the human body sweating.

A GSR sensor allows us to measure sweat gland activity, which is related to emotional arousal. So, to measure GSR, we take advantage of the electrical properties of the skin. Specifically, how the skin resistance changes with sweat gland activity, i.e., the greater sweat gland activity, the more perspiration, and thus, less skin resistance. GSR activity is typically measured in “micro-Siemens (uS)” or “micro-Mho (uM)”, mirroring the conductance of a certain material.

GSR is not only used to monitor your emotional arousal. In psychology, the data from the GSR sensor can be analyze to determine whether a person is lying or not. In fact, the GSR sensor has been widely used in our lives, especially in some medical facilities, psychology, polygraphs, etc.

Input/output Description:



The program requires 2 inputs, the GSR Baseline Data and the GSR Fear Data, to be compared. The Output is the Baseline Index value and the Fear Index Value is calculated on the code. Also, one of the outputs is the Graphs of Fear and Baseline Values with each of the functions applied.

Step 3: Design of the algorithm and test cases

Test Case 1: Baseline and Fear Index Value provided by the professor

```
>> MainScript
Success! the GSR data was read
Applied Median Filter!
Filter Low Pass Applied!
Applied Point Average filter!
Applied Cuts!
Success!Data normalized!
Fear Index (Fear):
          5570.46817231074

Fear Index (Baseline):
          6047.59537678532
```

Algorithm design:

```
Analyze and Visualize Galvanic Skin Response (GSR) Data
```

Load Data:

```
   Set GSR_baseline_filename to 'GSR_Baseline.csv'
   Set GSR_fear_filename to 'GSR_FEAR.csv'
   Load baseline data from GSR_baseline_filename into baseline_matrix
   Load fear data from GSR_fear_filename into fear_data
   Display message "Success! The GSR data was read"
```

Calculate Sampling Frequency:

```
   Calculate the average difference between timestamps in fear_data
   Convert this average to seconds
   Calculate Functions as the rounded inverse of this average time in
seconds
```

Plot Fear Data GSR (Galvanic Skin Response):

```
   Create a subplot grid (3 rows, 4 columns, position 1)
   Plot fear_data timestamps against GSR numbers
   Set plot title to 'GSR Values (Fear)'
   Label x-axis as 'Time' and y-axis as 'GSR Value'
   Enable grid on the plot
```

Plot Baseline Data GSR:

Create a subplot grid (3 rows, 4 columns, position 3)
Plot baseline_matrix timestamps against GSR numbers
Set plot title to 'GSR Values (Baseline)'
Label x-axis and y-axis as before
Enable grid on the plot

Apply Median Filter:

Apply a third-order median filter to GSR numbers in fear_data
Apply the same filter to GSR numbers in baseline_matrix
Display message "Applied Median Filter!"

Plot Median Filtered Data:

Create subplots in positions 2 and 4 for fear_data and baseline_matrix respectively
Set appropriate titles: 'Median filtered GSR (Fear)' and 'Median filtered GSR (Baseline)'
Label axes and enable grid as before

Apply Low Pass Filter:

Apply a low pass filter to GSR numbers in both fear_data and baseline_matrix with a cutoff frequency of 20 and using Functions
Display message "Filter Low Pass Applied!"

Plot Low Pass Filtered Data:

Create subplots in positions 5 and 7 for fear_data and baseline_matrix respectively
Set appropriate titles and labels as before

Apply Ten Point Average Filter:

Apply a ten-point average filter to GSR numbers in both datasets
Display message "Applied Point Average filter!"

Plot Point Average Filtered Data:

Create subplots in positions 6 and 8 for fear_data and baseline_matrix respectively
Set appropriate titles and labels as before

Data Trimming:

Trim the first 2 seconds of data from both fear_data and baseline_matrix
Display message "Applied Cuts!"

Normalize Data:

Map GSR numbers in both datasets to a range of 0 to 100
Display message "Success! Data normalized!"

Visualize Normalized Baseline GSR Data:
Create a subplot for baseline data visualization
Plot GSR values against time for baseline data
Title the plot 'Normalized GSR (Baseline)'
Label x-axis as 'Time' and y-axis as 'GSR Value'
Enable grid on the plot

Find Local Extrema:
Find local maxima and minima in fear data
Find local maxima and minima in baseline data

Plot Fear Data with Local Extrema:
Create a subplot for fear data visualization
Plot GSR values against time for fear data
Overlay scatter plots of local maxima (in green) and minima (in red) on the fear data plot
Title the plot 'Processed GSR Values with Local Min and Max (Fear)'
Label axes and enable grid

Plot Baseline Data with Local Extrema:
Create a subplot for baseline data visualization
Plot GSR values against time for baseline data
Overlay scatter plots of local maxima (in green) and minima (in red) on the baseline data plot
Title the plot 'Processed GSR Values with Local Min and Max (Baseline)'
Label axes and enable grid

Analyze Data:
Perform analysis on fear data and baseline data to calculate various metrics

Calculate Fear Indices:
Calculate Fear Index for fear data and baseline data using a specific formula
Print Fear Index values for both fear and baseline data

Auxiliary Functions:
read_data(file_name): Reads and formats GSR data from a file
third_order_medianfilter(input_array): Applies a third-order median filter to an array

findlocal_extrema(data): Identifies local maxima and minima in GSR data
analyze(data, maxima, minima): Performs detailed analysis on GSR data, calculating various metrics
calculate_FearIndex(parameters): Calculates a Fear Index based on multiple input parameters

Apply Ten-Point Average Filter:
Function: ten_point_aver_filter(input_array)
Create a filter coefficient for a 10-point moving average
Apply this filter to the input array
Return the filtered array

Normalize Data to Range 0 to 100:
Function: map_to_0_100(data)
Find the minimum and maximum values in the data
Normalize the data to a range between 0 and 100
Return the normalized data

Find Local Extrema in GSR Data:
Function: findlocal_extrema(fear)
Set a minimum prominence level for identifying extrema
Identify local maxima and minima in the GSR data
Return arrays of indices for local maxima and minima

Cut/Trim Data Based on Time Duration:
Function: cut(fear, time)
Define a duration to cut from the data
Keep only the data points beyond this duration
Return the trimmed data

Analyze GSR Data:
Function: analyze(data, maxima, minima)
Calculate various statistics and parameters from GSR data, such as mean, variance, total amplitude, rise time, total energy, maximum and minimum difference, time difference between max and min, number of maxima, power, and GSR bandwidth
These calculations involve iterating over the GSR values and applying specific formulas to determine each parameter
Return these calculated values

Calculate Fear Index:
Function: calculate_FearIndex(F1, F2, F3, F4, F5, F6, F7, F8, F9, F10)

```
   Use a specific formula (provided in the assignment) to calculate
the Fear Index from the given parameters
   Return the calculated Fear Index
```

Step 4: Implementation

```
/**********************************************************
%Author: Pedro Felix Fernandes
%Date Created: December 15, 2023
%Description:
%Assignment 4 - Electrical/Bioengineering Case Study – Fear Analysis
using GSR data


GSR_baseline = 'GSR_Baseline.csv';
GSR_fear = 'GSR_FEAR.csv';
baseline_mtrx = read_data(GSR_baseline);
fear = read_data(GSR_fear);
disp('Success! the GSR data was read');

%Define Sampling Frequency
data = seconds(mean(diff(fear.Timestamp)));
Functions = round(1/data);

subplot(3, 4, 1);
plot(fear.Timestamp, fear.GSRNumb);
title('GSR Values (Fear)');
xlabel('Time');
ylabel('GSR Value');
grid on;

subplot(3, 4, 3);
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
title('GSR Values (Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;

fear.GSRNumb = third_order_medianfilter(fear.GSRNumb);
baseline_mtrx.GSRNumb =
third_order_medianfilter(baseline_mtrx.GSRNumb);
disp('Applied Median Filter!');
```

```
subplot(3, 4, 2);
plot(fear.Timestamp, fear.GSRNumb);
title('Median filtered GSR(Fear)');
xlabel('Time');
ylabel('GSR Value');
grid on;

subplot(3, 4, 4);
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
title('Median filtered GSR(Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;

fear.GSRNumb = lowpass(fear.GSRNumb, 20, Functions);
baseline_mtrx.GSRNumb = lowpass(baseline_mtrx.GSRNumb, 20,
Functions);
disp('Filter Low Pass Applied!');

subplot(3, 4, 5);
plot(fear.Timestamp, fear.GSRNumb);
title('Low Pass filtered GSR(Fear)');
xlabel('Time');
ylabel('GSR Value');
grid on;

subplot(3, 4, 7);
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
title('Low Pass filtered GSR(Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;

fear.GSRNumb = ten_point_aver_filter(fear.GSRNumb);
baseline_mtrx.GSRNumb = ten_point_aver_filter(baseline_mtrx.GSRNumb);
disp('Applied Point Average filter!');

subplot(3, 4, 6);
plot(fear.Timestamp, fear.GSRNumb);
title('Point Average filtered GSR(Fear)');
xlabel('Time');
ylabel('GSR Value');
grid on;

subplot(3, 4, 8);
```

```
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
title('Point Average filtered GSR(Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;

%Due to filters, we need to cut out a few seconds of data
fear = cut(fear, 2);
baseline_mtrx = cut(baseline_mtrx, 2);
disp('Applied Cuts!');

fear.GSRNumb = map_to_0_100(fear.GSRNumb);
baseline_mtrx.GSRNumb = map_to_0_100(baseline_mtrx.GSRNumb);
disp('Success!Data normalized!');

subplot(3, 4, 9);
plot(fear.Timestamp, fear.GSRNumb);
title('Normalized GSR(Fear)');
xlabel('Time');
ylabel('GSR Value');
grid on;

subplot(3, 4, 11);
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
title('Normalized GSR(Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;


%%%%
[fear_max, fear_min] = findlocal_extrema(fear);
[baseline_maxima, baseline_minima] =
findlocal_extrema(baseline_mtrx);

% Plot Fear data with local maxima and minima
subplot(3, 4, 10);
plot(fear.Timestamp, fear.GSRNumb);
hold on;
scatter(fear.Timestamp(fear_max), fear.GSRNumb(fear_max), 'g',
'filled');
scatter(fear.Timestamp(fear_min), fear.GSRNumb(fear_min), 'r',
'filled');
hold off;
title('Processed GSR Values with Local Min and Max (Fear)');
```

```
xlabel('Time');
ylabel('GSR Value');
grid on;

% Plot Baseline data with local maxima and minima
subplot(3, 4, 12);
plot(baseline_mtrx.Timestamp, baseline_mtrx.GSRNumb);
hold on;
scatter(baseline_mtrx.Timestamp(baseline_maxima),
baseline_mtrx.GSRNumb(baseline_maxima), 'g', 'filled');
scatter(baseline_mtrx.Timestamp(baseline_minima),
baseline_mtrx.GSRNumb(baseline_minima), 'r', 'filled');
hold off;
title('Processed GSR Values with Local Min and Max(Baseline)');
xlabel('Time');
ylabel('GSR Value');
grid on;


[F1_F, F2_F, F3_F, F4_F, F5_F, F6_F, F7_F, F8_F, F9_F, F10_F] =
analyze(fear, fear_max, fear_min);
[F1_B, F2_B, F3_B, F4_B, F5_B, F6_B, F7_B, F8_B, F9_B, F10_B] =
analyze(baseline_mtrx, baseline_maxima, baseline_minima);

FearIndex_F = calculate_FearIndex(F1_F, F2_F, F3_F, F4_F, F5_F, F6_F,
F7_F, F8_F, F9_F, F10_F);
FearIndex_B = calculate_FearIndex(F1_B, F2_B, F3_B, F4_B, F5_B, F6_B,
F7_B, F8_B, F9_B, F10_B);
format longG;
disp('Fear Index (Fear): ');
disp(FearIndex_F);
disp('Fear Index (Baseline): ');
disp(FearIndex_B);



%This function read the data and define the time format for the
column
%called Time Stamp (following the assingnment of provided by the
professor)
function data_table = read_data(file_name)
timeFormat = 'mm:ss:SSS';

opts = delimitedTextImportOptions("NumVariables", 2); %Define the
options of the formating
```

```
opts.DataLines = [2, Inf];
opts.Delimiter = ",";
opts.VariableNames = ["Timestamp", "GSRNumb"];
opts.VariableTypes = ["string", "double"];

% Construct file path
file_path = fullfile(pwd, file_name);

% Check the existence of the file, if error says file not found
if exist(file_path, 'file') ~= 2
    error('File not found: %s', file_path);
end

% Put the Comma Separated value into a table
data_table = readtable(file_path, opts);

% Convert 'Timestamp' to datetime format (called as 'data' on this
code
data_table.Timestamp = datetime(data_table.Timestamp, 'Format',
timeFormat);
end

% this function get the size of the input array and initialize the
filtered array
function filtered_array = third_order_medianfilter(input_array)
N = length(input_array);
filtered_array = zeros(size(input_array));

% Apply the median filter third-order and extract the neighborhood of
the
% element, after extracting calculate the median and assign to the
filtered
% array.
for i = 2:N-1
    neighborhood = input_array(i-1:i+1);
    filtered_array(i) = median(neighborhood);
end
end

% this function define the coefficient as 10-point moving average
filter
function filtered_array = ten_point_aver_filter(input_array)
b = ones(1, 10) * 1/10;
a = 1;
```

```
% Apply the filter to the input array
filtered_array = filter(b, a, input_array);
end

% this function find the max and minimun values in the input data
function normalize_data = map_to_0_100(data)
min_val = min(data);
max_val = max(data);

% Normalize the data to the range 0 to 100
normalize_data = 100 * (data - min_val) / (max_val - min_val);
end

% this function set the minimum prominence for identifying extrema
(math vocabulary)
function [fear_max, fear_min] = findlocal_extrema(fear)
minProminence = 3;

% Identify local maxima and local minima based on GSR values
fear_max = islocalmax(fear.GSRNumb, 'MinProminence', minProminence);
fear_min = islocalmin(fear.GSRNumb, 'MinProminence', minProminence);
end

% this function "cut/convert" the time to duration for comparison
function [cut_fear] = cut(fear, time)
duration_to_cut = seconds(time);

% Determine points to keep based on the specified time
points_to_keep = fear.Timestamp >= fear.Timestamp(1) +
duration_to_cut;
cut_fear = fear(points_to_keep, :);% "cut/extract" the relevant
portion of the fear (the part that we want)
end

% this function calculates mean and variance to display on the graph
function [meanval, varianceval, totalamplit, total_risetime,
totalenergy, max_min_diff, time_difference_maxnmin, num_max, power,
gsr_bdwid] = analyze(data, maxima, minima)
gsr_values = data.GSRNumb;
sum_val = 0;
for i = 1:length(gsr_values)
    sum_val = sum_val + gsr_values(i);
end
meanval = sum_val / length(gsr_values);
```

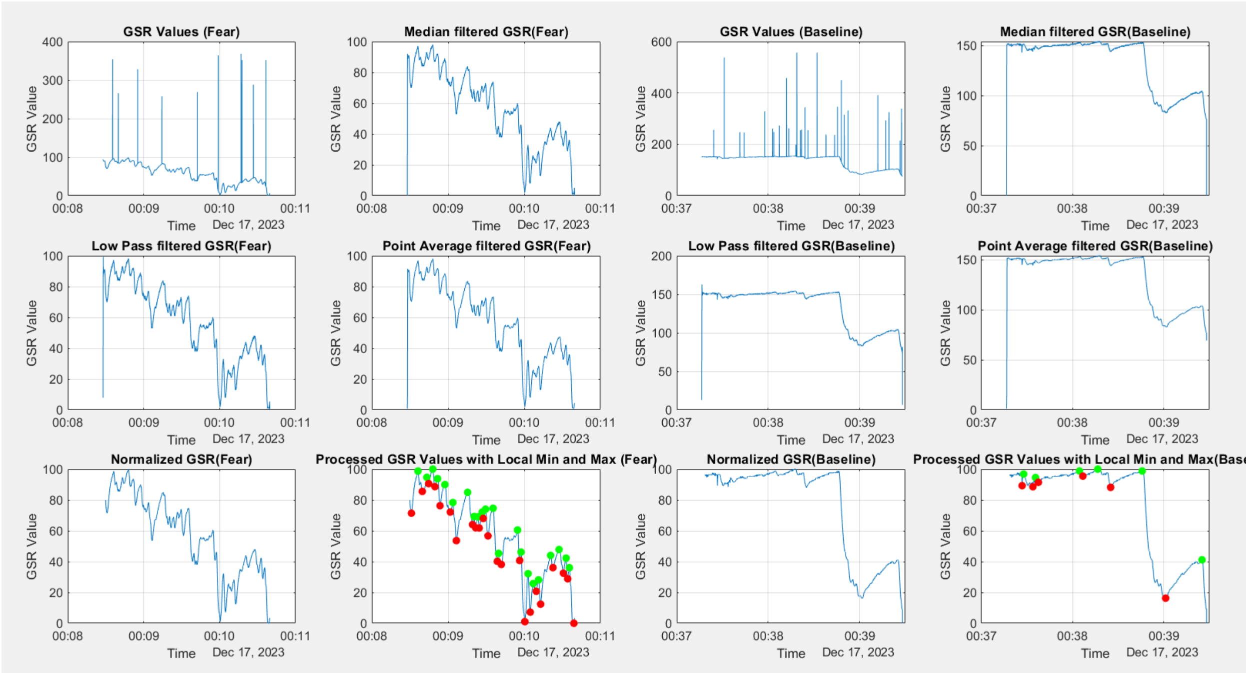
```
% Start Calculating Variance
sum_squared_diff = 0;

% Calculating Variance...
for i = 1:length(gsr_values)
    squared_diff = (gsr_values(i) - meanval)^2;
    sum_squared_diff = sum_squared_diff + squared_diff;
end
varianceval = sum_squared_diff / length(gsr_values);


% Analysis of the peak
max_points = find(maxima);
min_points = find(minima);
total_risetime = 0;
totalamplit = 0;
totalenergy = 0;
max_min_diff = 0;
time_difference_maxnmin = 0;
num_max = length(max_points);
gsr_bdwid_numrtr = 0;
gsr_bdwid_denomntr = 0;

for i = 1:length(max_points)
    current_max_index = max_points(i);
    preceding_min_index = max(min_points(min_points <
current_max_index)); % Find the nearest previous minimum index
    totalamplit = totalamplit + data.GSRNumb(current_max_index); %
Calculate the amplitude and add to the total
    rise_time = seconds(data.Timestamp(current_max_index) -
data.Timestamp(preceding_min_index)); % Calculate the rise time and
addit to the total
    total_risetime = total_risetime + rise_time;
    totalenergy = totalenergy + 0.5 * data.GSRNumb(current_max_index)
* rise_time; % Calculate the energy and add to the total
    current_difference = data.GSRNumb(current_max_index) -
data.GSRNumb(preceding_min_index); % Calculate max-min difference and
- if it's the highest until now - update it
    if current_difference > max_min_diff
        max_min_diff = current_difference;
        time_difference_maxnmin =
seconds(data.Timestamp(current_max_index) -
data.Timestamp(preceding_min_index));% Calculate the time difference
between max and min with the highest difference of the amplitude
    end
```

```
    % that if calculate numerator (numrtr) and denominator (denomntr)
for GSR bandwidth
    if i > 1
        gsr_bdwid_numrtr = gsr_bdwid_numrtr +
(data.GSRNumb(current_max_index) - data.GSRNumb(current_max_index -
1))^2;
    end
    gsr_bdwid_denomntr = gsr_bdwid_denomntr +
data.GSRNumb(current_max_index)^2;
end
power = totalenergy / seconds(data.Timestamp(end) -
data.Timestamp(1)); %Calculate the power
gsr_bdwid = (1 / (2 * pi)) * sqrt(gsr_bdwid_numrtr /
gsr_bdwid_denomntr); % Calculate the GSR bandwidth
end

%That functions calculates the Fear Index (formula given by the
assignment)
function FearIndex = calculate_FearIndex(F1, F2, F3, F4, F5, F6, F7,
F8, F9, F10)
FearIndex = F1 + 2 * F2 + F3 + 0.5 * F4 + F5 + 2 * F6 + F7 + 5 * F8 +
0.001 * F9 + 0.5 * F10;
end
```

Step 5: Software Testing and Verification

Test Case 1: Baseline and Fear Index Value provided by the professor

```
Success:Data normalized!
Fear Index (Fear):
          5570.46817231074

Fear Index (Baseline):
          6047.59537678532
```



User Guide

This program will help you analyze the relationship between the GSR signal and the physiological response to fear. It will also calculate the Baseline and Fear Index Value, as a plus, it will show the graph with all the functions applied to the baseline and fear values. The user just need to add the databases with the Fear and the Baseline values to the folder of the program and run it.