Содержание

Моделирование

Для того, чтобы начать решать любую из задач идентификации, предложенных в работе: будь то задача идентификации параметров дорожного покрытия или параметров автомобиля, необходимо создать виртуальную среду, моделирующую поведение соответствующих физических сущностей. Также необходимо правильно подготовить набор данных для дальнейшего тестирования гипотез.

Моделирование профиля дорожного покрытия

Была разработана специальная процедура, позволяющая создавать горизонтальный профиль дорожного покрытия со случайным набором дорожных событий (ям или выпуклостей) с прямоугольными или синосоидальными профилями. Высота дорожных событий, их типы и удаление друг от друга определяются случайным образом с дополнительной настройкой. Чтобы учесть пятно контакта и его взаимодействие с профилем дороги, к смоделированному профилю дорожного покрытия применяется фильтр скользящего среднего.

На текущем этапе исследований предполагается работать с обособленными прямоугольными ямами (удаленными друг от друга на расстоянии, превышающем 2 корпуса автомобиля), глубиной до -20 см. Окно сглаживающего фильтра установлено равным 0.1 м. Предполагаемая скорость движения автомобиля - 10 км/ч. Сгенерируем такой профиль дороги во временном измерении, содержащий в себе 100 дорожных событий:

```
%% Preparing road
```

```
vehicle_speed = 10; % 10[km/h]
points_cnt=500; % [Hz]

road = zeros(10,1);
road_markers = zeros(10,1);

road = [road; spb_gen(5,-0.1,0,vehicle_speed,points_cnt)];
```

```
'spb_gen' is not found in the current folder or on the MATLAB path, but exists in:
    /Users/pauladm/MATLAB-Drive/RUPAS/ML/wf_spec_cl

Change the MATLAB current folder or add its folder to the MATLAB path.
```

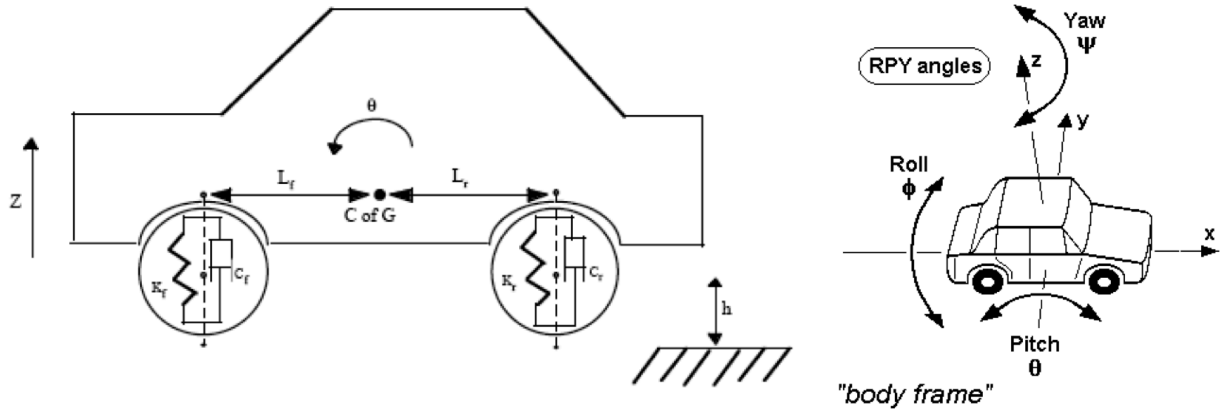
```
road_markers = [road_markers; zeros(length(spb_gen(5,-0.1,0,vehicle_speed,points_cnt)),

for k = 1:500%bumps_cnt
    bump_width = 0.05 + 1.95*rand(1,1); %from 0.05m to 2m width
    bump_height = -0.05 -0.15*rand(1,1); %from -0.1m to 0.1m
    %bump_cat = 1; %only rectangle
    bump_cat = 1 + round(rand(1,1)*1); % 1 2

    if bump_cat == 2 % if speed bump, then positive height
        bump_height = 0.1;
    end

    if bump_cat == 0
        bump_width=bump_width +  k/50; %increasing free space between bumps
    end

    road = [road; spb_gen(bump_width,bump_height, bump_cat, vehicle_speed, points_cnt)]
    road_markers = [road_markers; bump_cat*ones(length(spb_gen(bump_width,bump_height,


    zero_width = 50;%*rand(1,1); % % zero spaces with width = 50 m * rand(1,1)
    road = [road; spb_gen(zero_width,-0.1,0,vehicle_speed,points_cnt)];
    road_markers = [road_markers; zeros(length(spb_gen(zero_width,-0.1,0,vehicle_speed,


end

B = 0.1; % пятно контакта шины [m]
enable_tire_filter = true;
B_points =  round(B * points_cnt / (vehicle_speed*1000/3600)); %  smoothing window [poi
t = (1:length(road))/points_cnt;
road_TS = timeseries(smooth(road, B_points),t);
%road_M_TS = timeseries(road_markers,t);
clear B B_points bump_cat bump_height bump_width enable_tire_filter ...
    k points_cnt road t;


figure;
subplot(2,1,1)
plot(road_TS)
xlim([0.0 26.6])
title('Фрагмент профиля дорожного покрытия')
```

```
xlabel('Время [сек]')
ylabel('Высота профиля дороги [м]')


%subplot(2,1,2)
%plot(road_M_TS)
%xlim([0.0 26.6])
%title('Фрагмент маркировки профиля дорожного покрытия')
%xlabel('Время [сек]')
%ylabel('Маркер (0 или 1 или 2)')
```

Моделирование автомобиля

Велосипедная модель автомобиля

Следующим этапом в среде Simulink была реализована двухколесная велосипедная модель автомобиля модель автомобиля:



Математическое описание

Такая модель описывается двумя основными законами - вторым законом Ньютона и законом сохранения момента импульса:

$$\begin{cases} m_b\ddot{z} = F_{front} + F_{rear} - m_b g \\ I_{yy}\ddot{\theta} = M_{front} + M_{rear} + M_y \end{cases},$$

$$\begin{cases} F_{front} = 2K_f(L_f\theta - z) + 2C_f(L_f\dot{\theta} - \dot{z}) \\ F_{rear} = -2K_r(L_r\theta + z) - 2C_r(L_r\dot{\theta} + \dot{z}) \\ M_{front} = -L_f F_{front} \\ M_{rear} = L_r F_{rear} \end{cases},$$

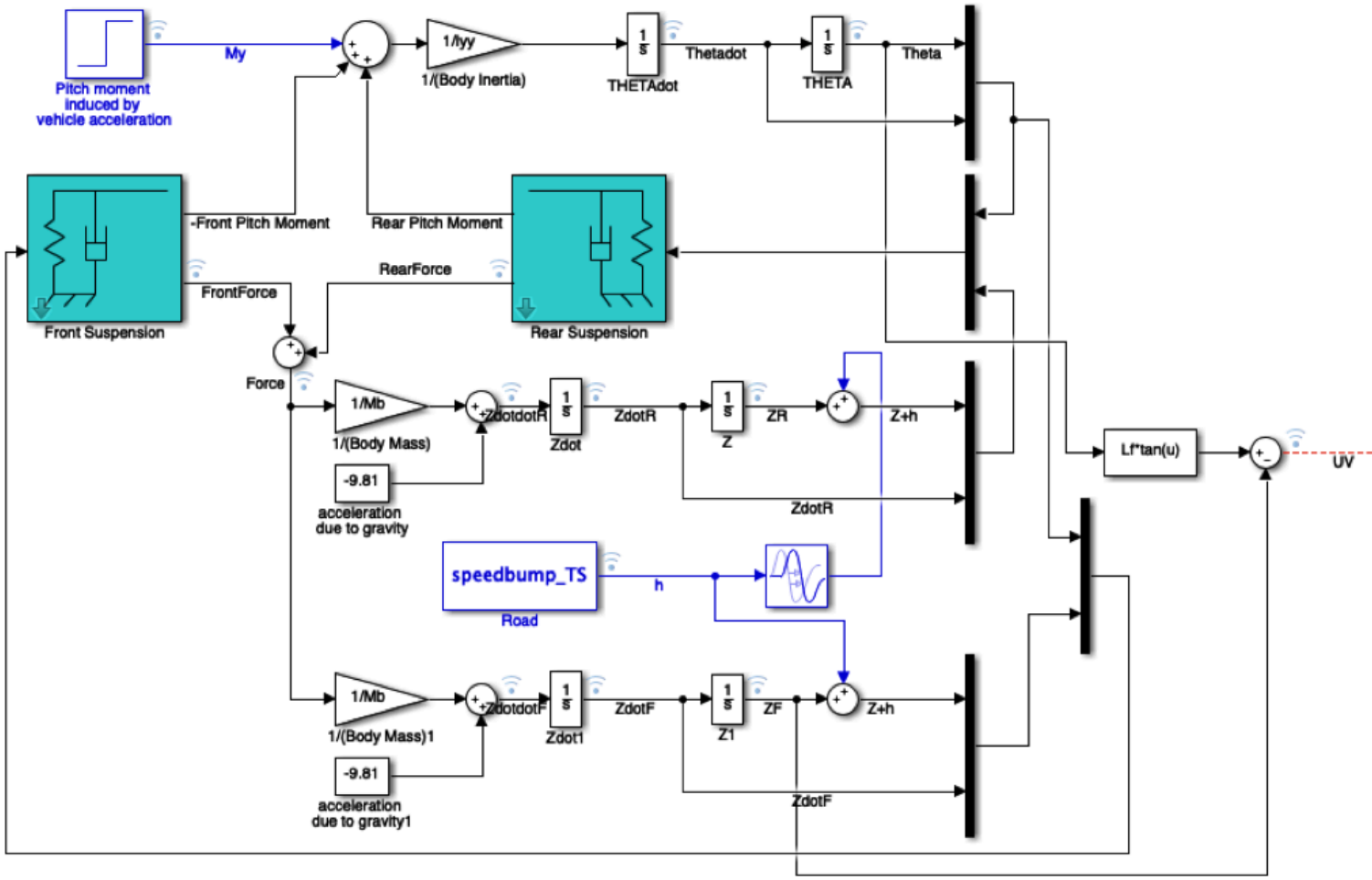
где F_{front}, F_{rear} – силы, направленные вверх и действующие на раму от передней и задней подвески соответственно; K_f, K_r – константы упругости передней и задней подвески; C_f, C_r – константы демпфирования передней и задней подвески; L_f, L_r – горизонтальное расстояние от центра масс рамы до передней и задней оси; $\theta, \dot{\theta}$ – угол тангажа (pitch angle) и его скорость изменения; $z, \dot{z}$ – высота

центра масс над землей и ее производная; $M_{\text{front}}, M_{\text{rear}}$ – момент тангажа, созданный передней/задней подвеской; M_y – момент тангажа благодаря измерению ускорения автомобиля; I_{yy} – момент инерции рамы относительно центра масс.

Параметры модели

```
Lf = 0.9;     % [m]
Lr = 1.2;     % [m]
Mb = 1200;    % [kg]
Iyy = 2100;   % [kg m^2]
kf = 28000;   % [N/m]
kr = 21000;   % [N/m]
cf = 2500;    % [N sec/m]
cr = 2000;    % [N sec/m]
```

Моделирование в Simulink



Симуляция езды велосипедной модели

Сформированный выше профиль дороги подается на вход модели Simulink и выполняется симуляция. Частота симуляции - 20 Гц.

```
Fs = 20; % current - 20 Hz simulation
sliding_window = Fs*10;

simres = sim('/Users/pauladm/MATLAB-Drive/RUPAS/SIM/vehicle_2dof_v3.slx',max(road_TS.Ti
```

```
Warning: The file containing block diagram 'vehicle_2dof_v3' is shadowed by a file of the same name
 higher on the MATLAB path. This can cause unexpected behavior. For more information see "Avoiding
 Problems with Shadowed Files" in the Simulink documentation.

The file containing the block diagram is: /Users/pauladm/MATLAB-Drive/RUPAS/SIM/vehicle_2dof_v3.slx.
The file higher on the MATLAB path is: /Users/pauladm/MATLAB-Drive/RUPAS/ML/wf_spec_cl/
vehicle_2dof_v3.slx
Warning: Unconnected output line found on 'vehicle_2dof_v3/Zdot2' (output port: 1)
Warning: Unconnected output line found on 'vehicle_2dof_v3/Derivative' (output port: 1)
Warning: Unconnected output line found on 'vehicle_2dof_v3/Sum4' (output port: 1)
```

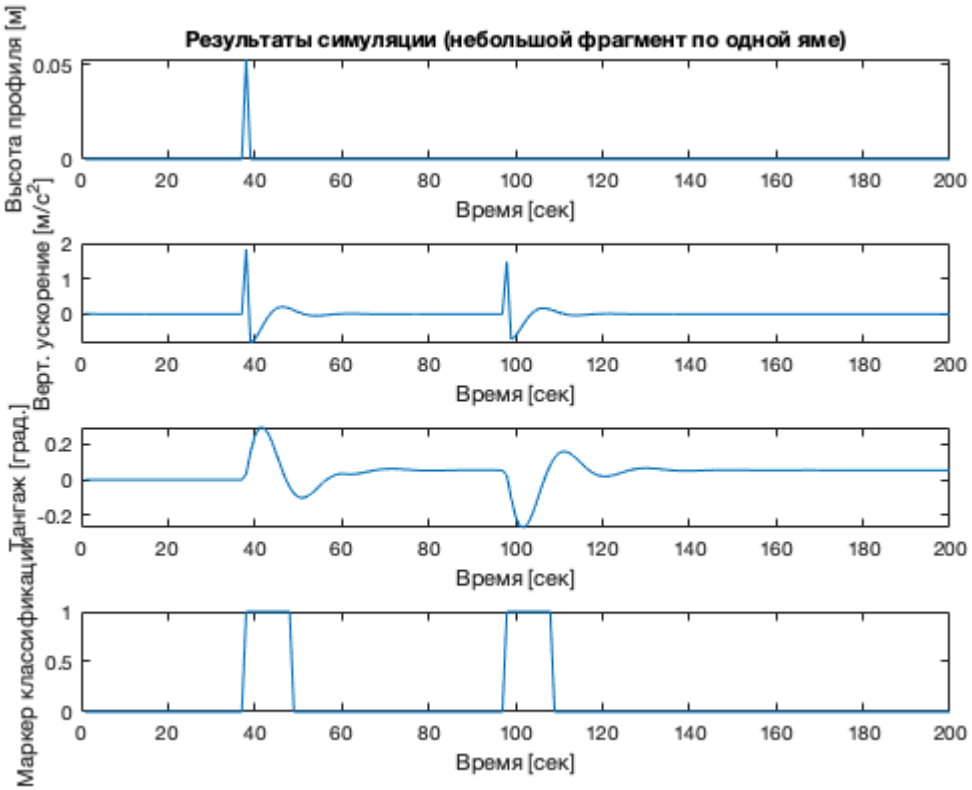
Результаты симуляции

На выходе модели выводятся 2 основных сигнала - сумма вертикальных сил, действущих на центр масс и угол тангажа автомобиля.

```
sliding_window = 200;
```

```
%h force theta timeseries result
h_y = h.Data; %[m]
h_x = h.Time*vehicle_speed*1000/3600; %[m]

h2_y = h2.Data; %[m]
h2_x = h2.Time*vehicle_speed*1000/3600; %[m]

force_markers = road_m.Data;


force_y = force.Data;
theta_y = theta.Data;



figure;
subplot(4,1,1)
plot(h_y(1:sliding_window))
title('Результаты симуляции (небольшой фрагмент по одной яме)')
xlabel('Время [сек]')
ylabel('Высота профиля [м]')

subplot(4,1,2)
plot(force_y(1:sliding_window))
%title('Subplot 1: sin(x)')
xlabel('Время [сек]')
ylabel('Верт. ускорение [м/с^2]')


subplot(4,1,3)
```

```
plot(theta_y(1:sliding_window))
%title('Subplot 1: sin(x)')
xlabel('Время [сек]')
ylabel('Тангаж [град.]')

subplot(4,1,4)
plot(force_markers(1:sliding_window))
%title('Subplot 1: sin(x)')
xlabel('Время [сек]')
ylabel('Маркер классификации')
```

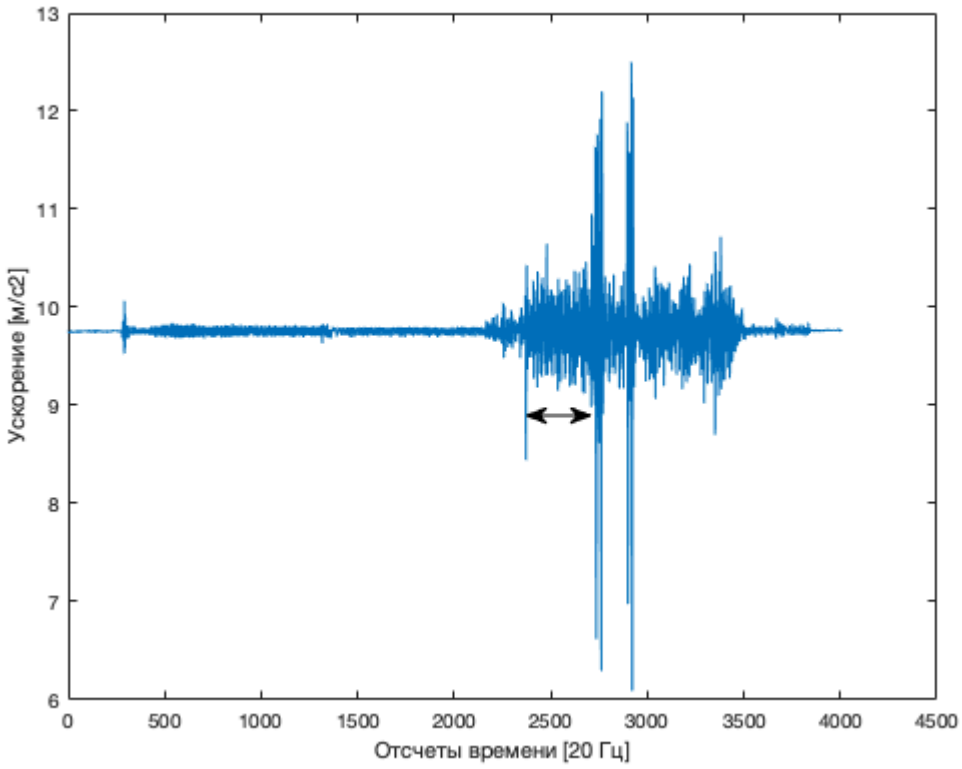


Квазиреальные данные

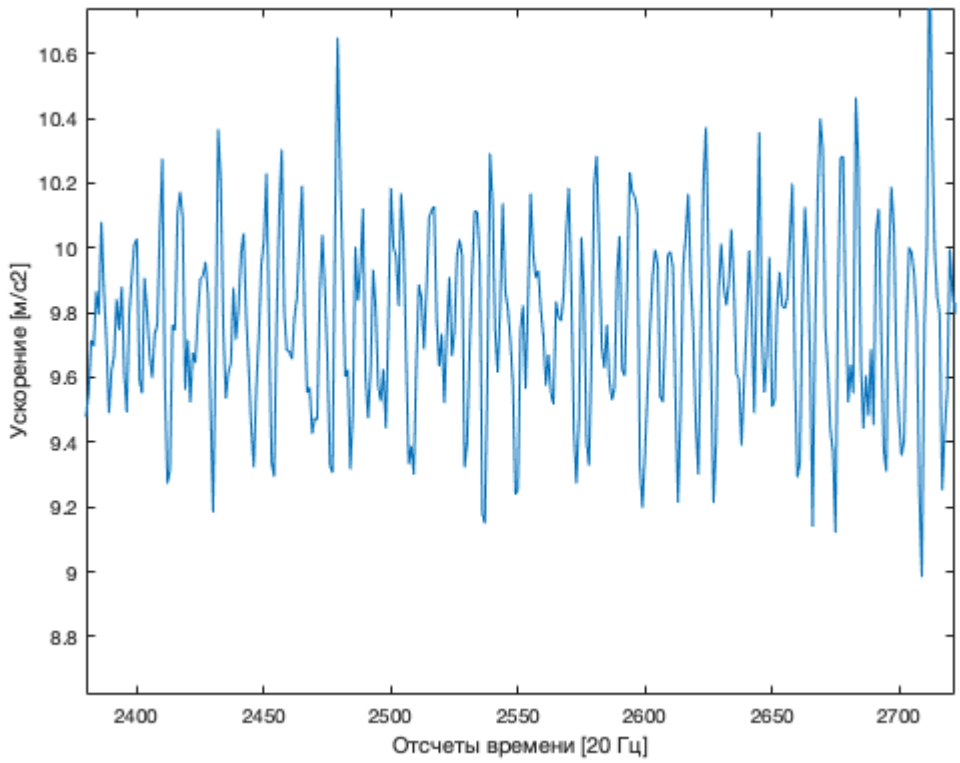
Для того, чтобы результаты симуляции выглядили более реалистично, они были объединены с реальным шумом, возникающим при движении автомобиля по ровной поверхности на скорости 10 км/ч. Такой хитрый трюк позволит учесть множество побочных эффектов, которые пока не моделируются: например, вибрацию двигателя автомобиля или небольшие шероховатости асфальтового покрытия. Фрагмент реальных данных была записан на автомобиле Nissan Qashqai при помощи смартфона:

```
load('/Users/pauladm/MATLAB-Drive/MobileSensorData/sensorlog_20190428_173242 qashqai.ma
figure;
plot(Acceleration.Z)
xlim('auto')
ylim('auto')
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
```

```
annotation('doublearrow',[0.5393 0.5964],[0.4476 0.4476])
```



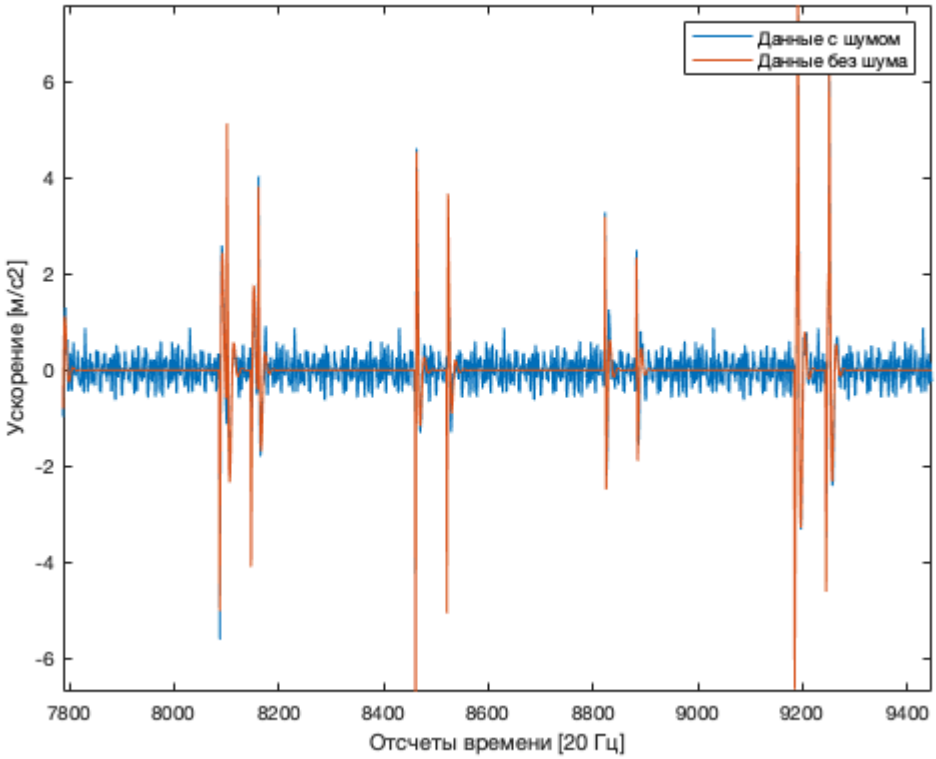

```
figure;
plot(Acceleration.Z)
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
xlim([2380 2722])
ylim([8.62 10.74])
```



Далее из шума вычитается среднее значение и выполняется процедура формирования семпла данных, сопоставимого по длинне с выходным сигналом по ускорению из симуляции. Затем оба сигнала складываются.

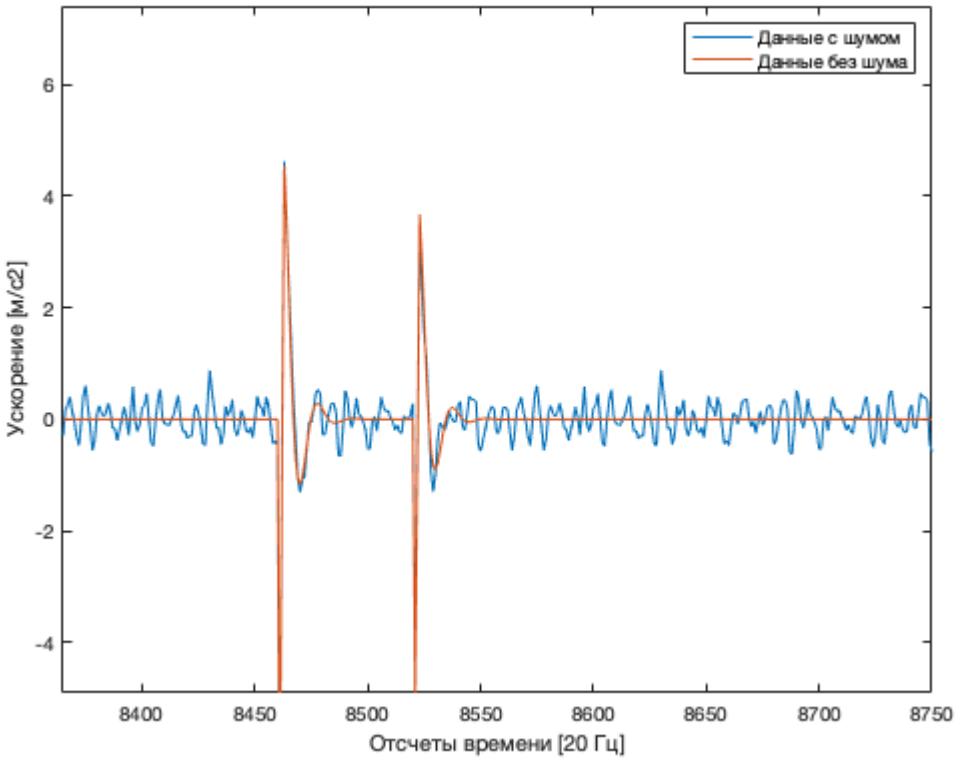
```
load('road_noise.mat');
road_acc_noise_10sec = acc_noise(1:200) - mean(acc_noise(1:200));


req_noise_length = length(force_y);

result_noise = zeros((round((req_noise_length/200)) + 1)*200,1);


for i = 1:(round((req_noise_length/200)) + 1)
    result_noise(200*(i-1)+1:200*i) = road_acc_noise_10sec;
end


result_noise = result_noise(1:req_noise_length);


force_y_noise = force_y + result_noise;

figure;
plot(force_y + result_noise)
```

```
hold on
plot(force_y)
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
xlim([7788 9447])
ylim([-6.7 7.6])
legend({'Данные с шумом','Данные без шума'})
hold off;
```



Более крупно - одна из ям:

```
figure;
plot(force_y + result_noise)
hold on
plot(force_y)
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
xlim([8365 8750])
ylim([-4.9 7.4])
legend({'Данные с шумом','Данные без шума'})
hold off;
```



Машинное обучение

Классификация инерциальных данных при помощи нейронных сетей

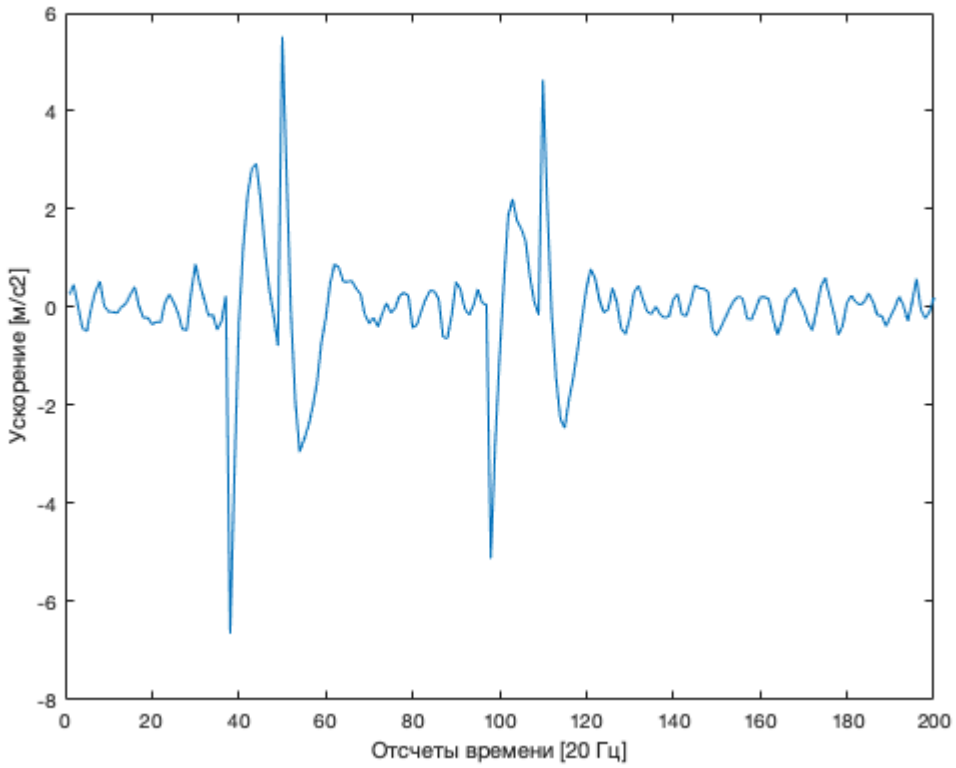
Постановка задачи

Поставим задачу так: разбить все отсчеты заданного окна инерциальных данных по ускорению автомобиля на два класса:

- отсчеты, при которых автомобиль проезжал по определенному дорожному событию (выпуклости или яме)
- отсчеты, при которых автомобиль ехал по ровной дороге

Кстати, далее будем рассматривать окна длинной 200 отсчетов. При частоте дискретизации 20 Гц такой длины заведомо хватит для того, чтобы обе оси автомобиля проехали по обособленной яме. Например, такое окно, в котором автомобиль проехал по лежачему полицейскому на скорости 2м/с:

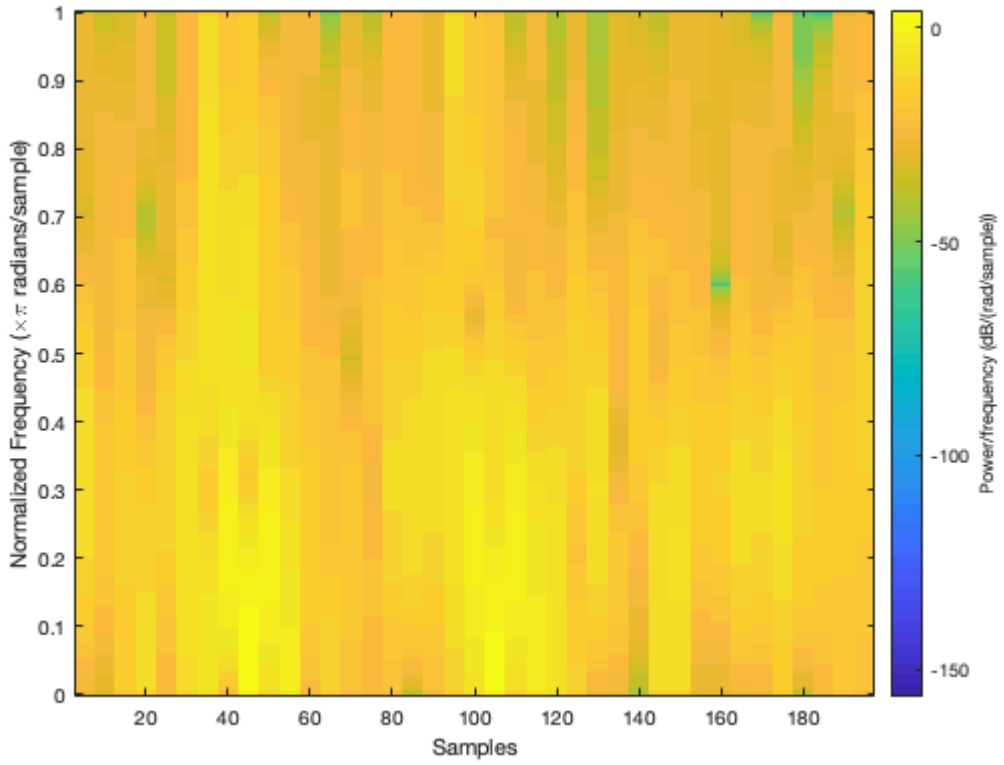
```
acc_200 = force_y_noise(1:200);

figure;
plot(acc_200)
xlim('auto')
ylim('auto')
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
```



Спектрограммы

Очевидно, что в случае такой постановки задачи нужна какая-либо инвариантная метрика, работающая для любого наперед заданного окна данных. Такой метрикой может быть спектрограмма:

```
spectrogram(acc_200,10,'yaxis')
```



Идея обучения нейронной сети

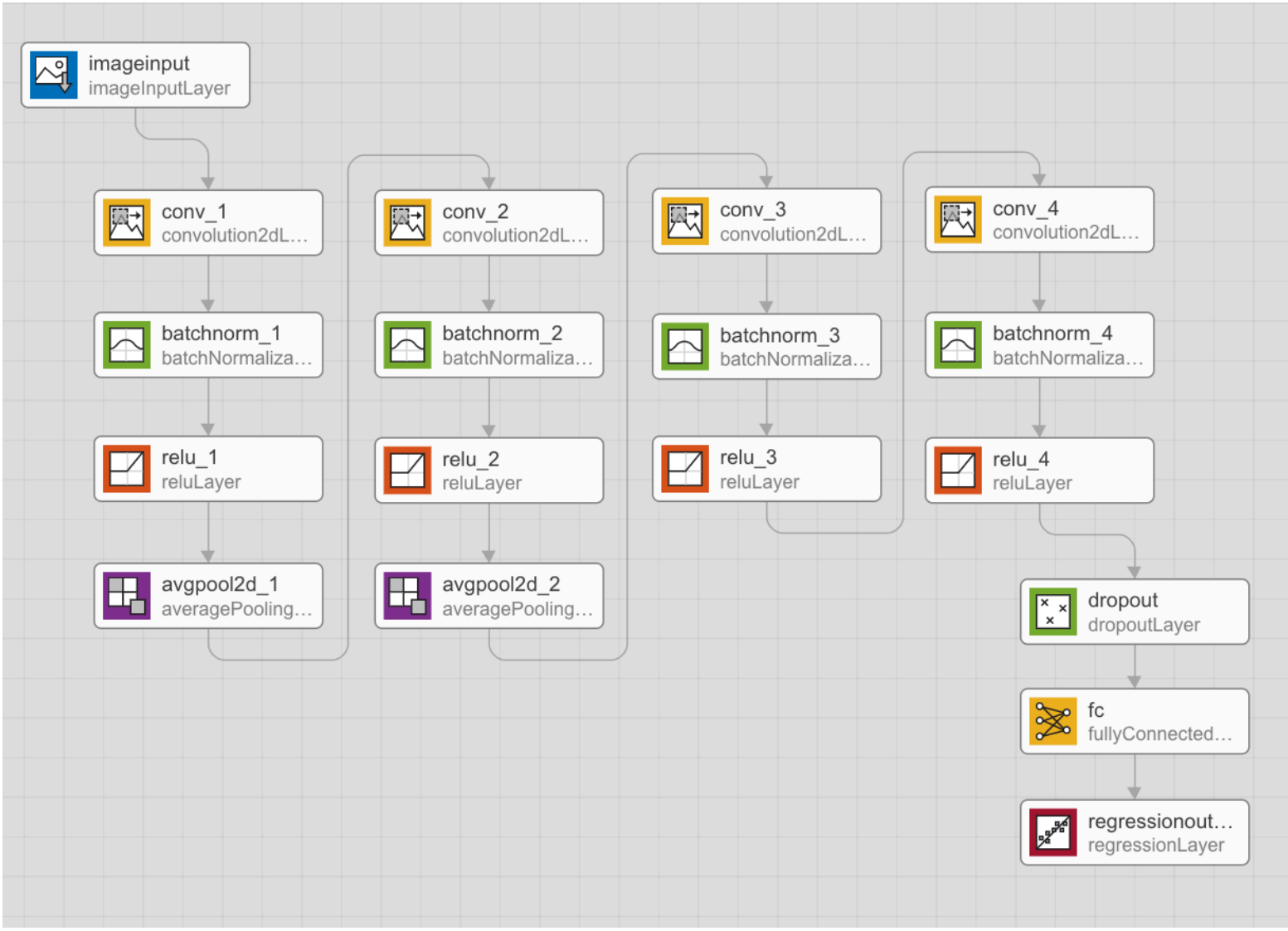
Подавать нейронной сети на вход черно-белое изображение спектрограммы окна данных и учить ее находить точки, соответствующие "наиболее желтым" вертикальным полосам спектрограммы.

Подготовка набора данных для обучения на базе спектрограмм

```
%% 0) load hft array


% training array H regression
spectrogram_window = 10;


test_image_bw_arr = force2gray_spectrogram(force_y_noise(i:i+sliding_window),spectrogra
test_size = size(test_image_bw_arr);
row_px = test_size(1,1);
col_px = test_size(1,2);

train_cnt = round(length(h_y)*0.8);

XTrain = zeros(row_px, col_px, 1, train_cnt); %features
YTrain = zeros(train_cnt,sliding_window); % experience

XValidation = zeros(row_px, col_px, 1, length(h_y)-sliding_window - (train_cnt+1)); %fe
YValidation = zeros(length(h_y)-sliding_window - (train_cnt+1),sliding_window); % exper
```

```
%Y_f = zeros(length(h_y),1);

%%
for i = 1:train_cnt
    XTrain(:, :,1, i) = force2gray_spectrogram(force_y_noise(i:i+sliding_window),spectr

    f_window = force_y(i:i+sliding_window);
    f_window_plus = abs(f_window) > 0.01;

    YTrain(i,:) = round(smooth(f_window_plus(1:200)));
end

for i = 1:length(h_y)-sliding_window - (train_cnt+1)
    i_corr = i+ train_cnt;
    XValidation(:, :,1, i) = force2gray_spectrogram(force_y_noise(i:i+sliding_window),s

    f_window = force_y(i:i+sliding_window);
    f_window_plus = abs(f_window) > 0.01;

    YValidation(i,:) =round(smooth(f_window_plus(1:200)));
end
```

Архитектура нейронной сети




```
layers = [
    imageInputLayer([129 39 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(3,32,'Padding','same')
```

```
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)
    fullyConnectedLayer(200)
    regressionLayer];

%%

miniBatchSize  = 128;
validationFrequency = floor(numel(YTrain)/miniBatchSize);
```

```
Undefined function or variable 'YTrain'.
```

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',30, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',validationFrequency, ...
    'Plots','training-progress', ...
    'Verbose',false);
```

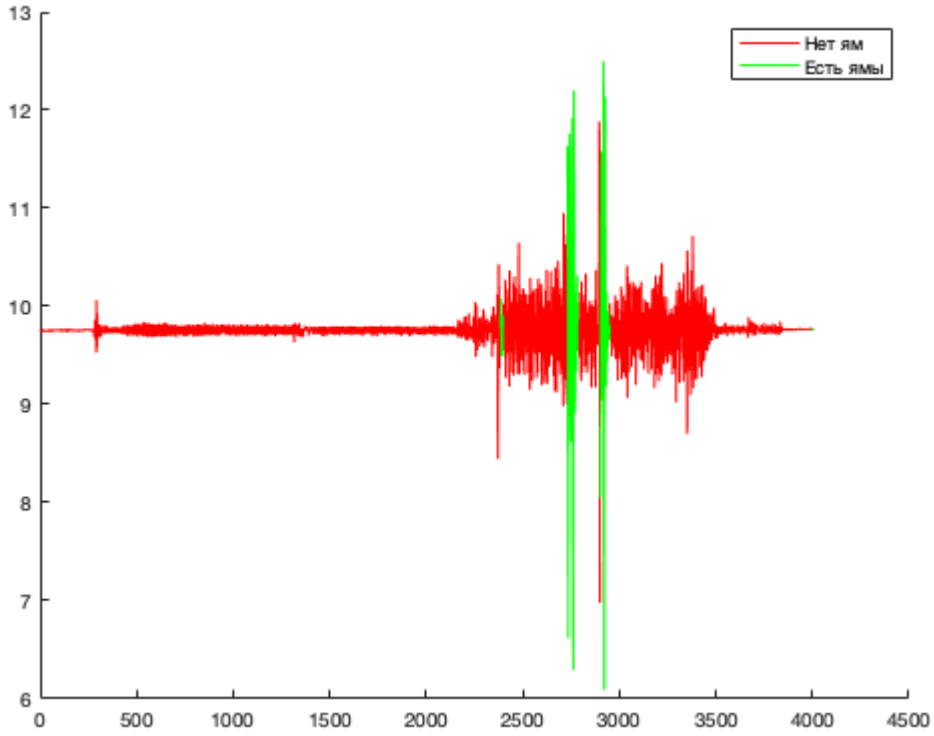
Обучение

```
net_noise = trainNetwork(XTrain,YTrain,layers,options);
```

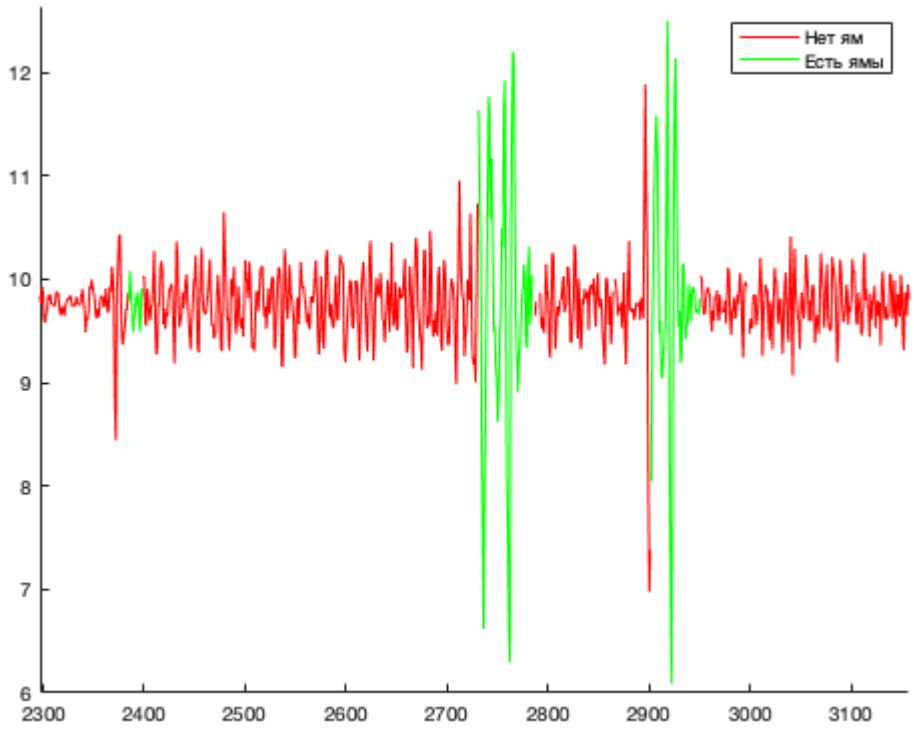
Предсказание

```
y = Acceleration.Z;
x = 1:length(y);


% Level for Color Change
lev = 0.2;
% Find points above the level
aboveLine = predictForceCl(net_noise,Acceleration.Z) == 1;%(round(predict(net_noise, fo
% Create 2 copies of y
bottomLine = y;
topLine = y;
% Set the values you don't want to get drawn to nan
bottomLine(aboveLine) = NaN;
topLine(~aboveLine) = NaN;

figure;
hold on
```

```
plot(x,bottomLine,'r',x,topLine,'g');
%plot(upsample(Position.speed,20));
legend({'Нет ям','Есть ямы'})
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
hold off
```




```
figure;
hold on
plot(x,bottomLine,'r',x,topLine,'g');
%plot(upsample(Position.speed,20));
legend({'Нет ям','Есть ямы'})
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
xlim([2297 3157])
ylim([6.00 12.64])
hold off
```



Дальнейшие задачи

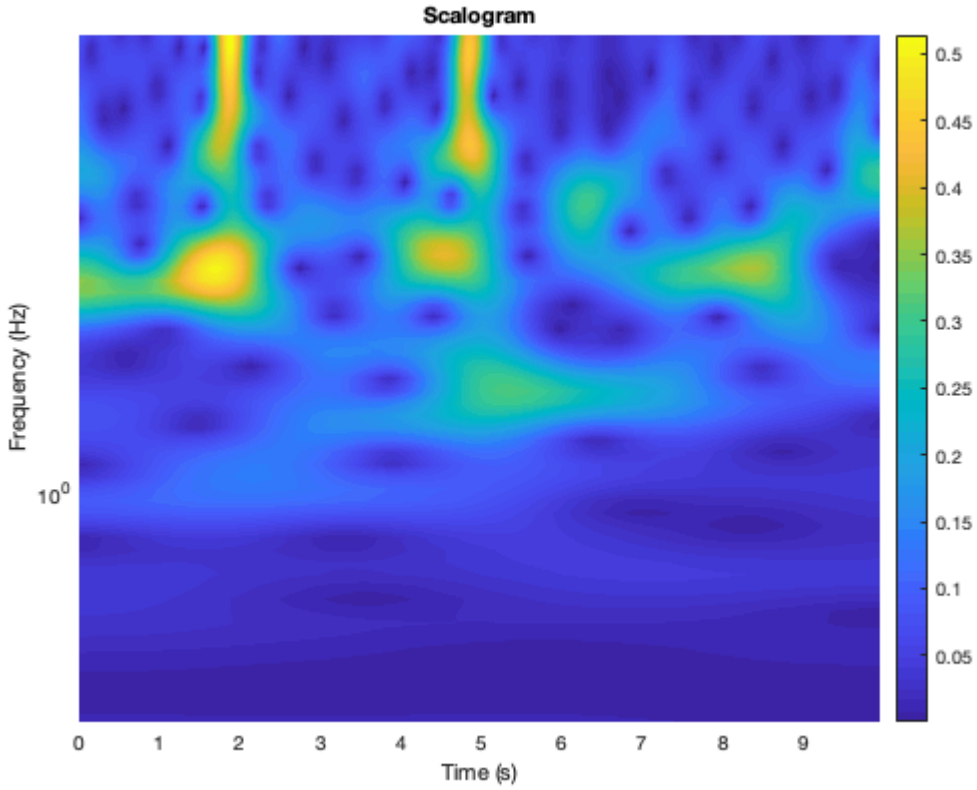
- Классифицировать на два класса: лежачие полицейские и прочие ямы
- Проверить обученную сеть на реальных данных (торможения, ускорения, нестандартные случаи)
- Проварьировать модели автомобиля и скорости и обучить на всех параметрах (скорость, параметры амортизаторов и тд) Классы автомобилей какие-то известные и общепринятые, несколько классов.
- Попробовать скалограммы для сравнения
- Выделить непрерывные фрагменты классов ям и построить по ним регрессию для идентификации ям
-

Скалограммы

Очевидно, что в случае такой постановки задачи нужна какая-либо инвариантная метрика, работающая для любого наперед заданного окна данных. Такой метрикой может быть скалограмма:

```
%scalogram(acc_200,10)



fb = cwtfilterbank('SignalLength',sliding_window,...
    'SamplingFrequency',Fs,...
```

```
    'VoicesPerOctave',12);
sig = force_y_noise(1:sliding_window);%Acceleration.Z(1:5000);%ECGData.Data(1,1:1000);
[cfs,frq] = wt(fb,sig);
t = (0:sliding_window-1)/Fs;

figure;
pcolor(t,frq,abs(cfs))

set(gca,'yscale','log');shading interp;axis tight;
colorbar
title('Scalogram');xlabel('Time (s)');ylabel('Frequency (Hz)')
```



Рассчитаем максимальную плотность мощности по всем смоделированным данным для нормировки дальнейших изображений

```
%scalogram(acc_200,10)

fb = cwtfilterbank('SignalLength',length(force_y_noise),...
    'SamplingFrequency',Fs,...
    'VoicesPerOctave',12);
sig = force_y_noise(:);%Acceleration.Z(1:5000);%ECGData.Data(1,1:1000);
[cfs,frq] = wt(fb,sig);
%t = (0:14999)/Fs;figure;pcolor(t,frq,abs(cfs))

max_scalogram_psd = max(max(abs(cfs)))
```

```
max_scalogram_psd = 2.5427
```

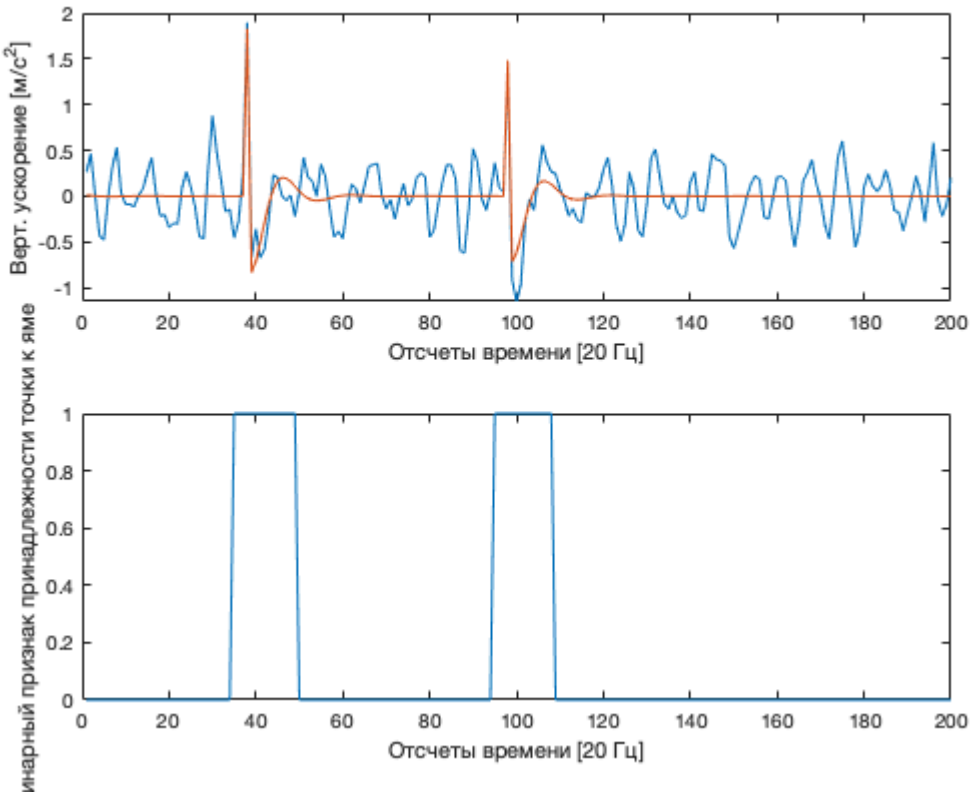
Подготовка набора данных для обучения на базе скалограмм

```
%% 0) load hft array


% training array H regression
%sliding_window = 200;

i=1;
test_image_bw_arr = force2gray_scalogram(force_y_noise(i:i+sliding_window-1),20, max_sc
test_size = size(test_image_bw_arr);
row_px = test_size(1,1);
col_px = test_size(1,2);

total_cnt = round(length(h_y)/sliding_window -0.5)-1;
train_cnt = round(0.8 * total_cnt);
validation_cnt = total_cnt - train_cnt;

XTrain = zeros(row_px, col_px, 1, train_cnt); %features
YTrain = zeros(train_cnt,sliding_window); % experience

XValidation = zeros(row_px, col_px, 1, validation_cnt); %features
YValidation = zeros(validation_cnt,sliding_window); % experience

figure;
imshow(test_image_bw_arr)
```

Зададим порог для обучения

```
accLearnThreshold = 0.1;

i = 1;
XTrain(:, :, 1, i) = force2gray_scalogram(force_y_noise(i:i+sliding_window-1),20, max_s

f_window = force_y(i:i+sliding_window);
f_window_plus = abs(f_window) > accLearnThreshold;
f_window_plus = max(f_window_plus,[0 0 0 0 0 0 f_window_plus(10:190)' 0 0 0 0 0 0 0 0 0


YTrain(i,:) =   round(smooth(f_window_plus(1:200)));
%YTrain(i,:) = round(force_markers(i:i+sliding_window-1)');
```

```
figure;
subplot(2,1,1)
plot(force_y_noise(i:i+sliding_window-1))
hold on
plot(force_y(i:i+sliding_window-1))
hold off
%title('Subplot 1: sin(x)')
xlabel('Отсчеты времени [20 Гц]')
ylabel('Верт. ускорение [м/с^2]')
xlim([0 200])

%annotation('line',[0 1],...
%    [accLearnThreshold accLearnThreshold]);


subplot(2,1,2)
plot(YTrain(i,:))
%title('Результаты симуляции (небольшой фрагмент по одной яме)')
xlabel('Отсчеты времени [20 Гц]')
ylabel('Бинарный признак принадлежности точки к яме')
xlim([0 200])
```



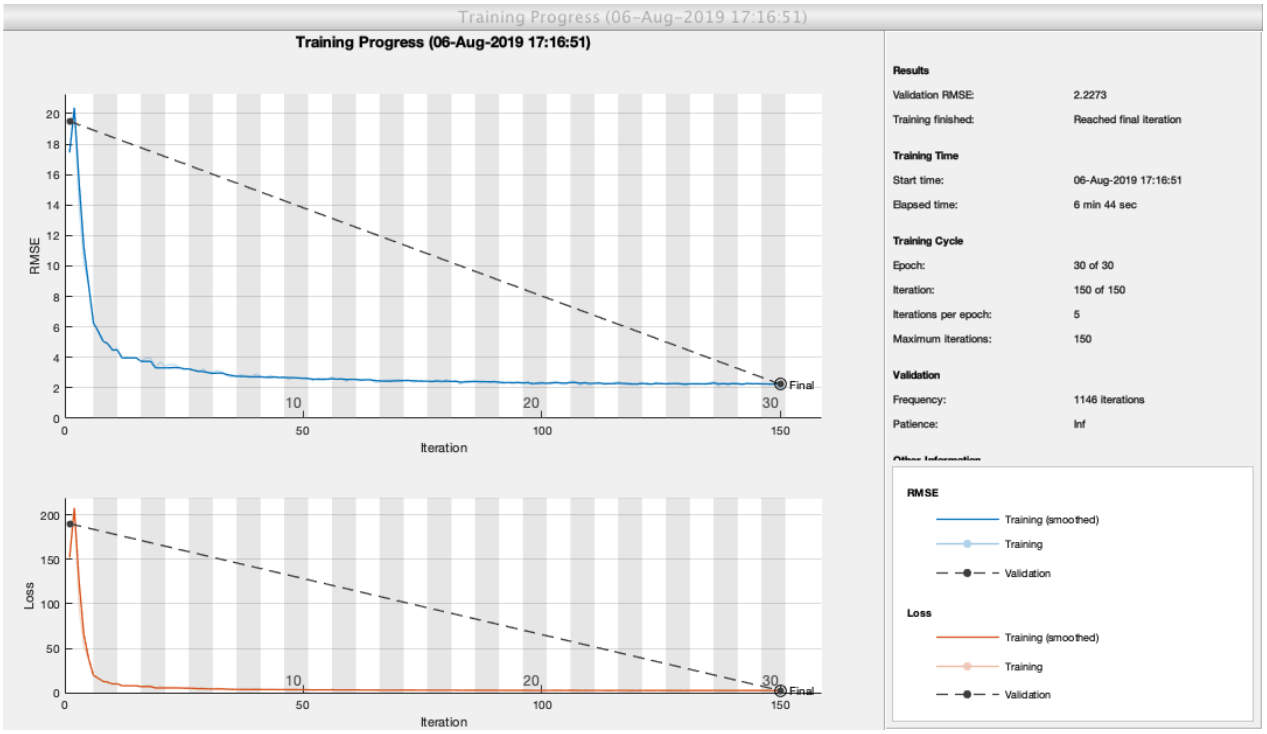

```
%%
sliding_window_iterator = 1;
for i = 1:sliding_window:train_cnt*sliding_window
```

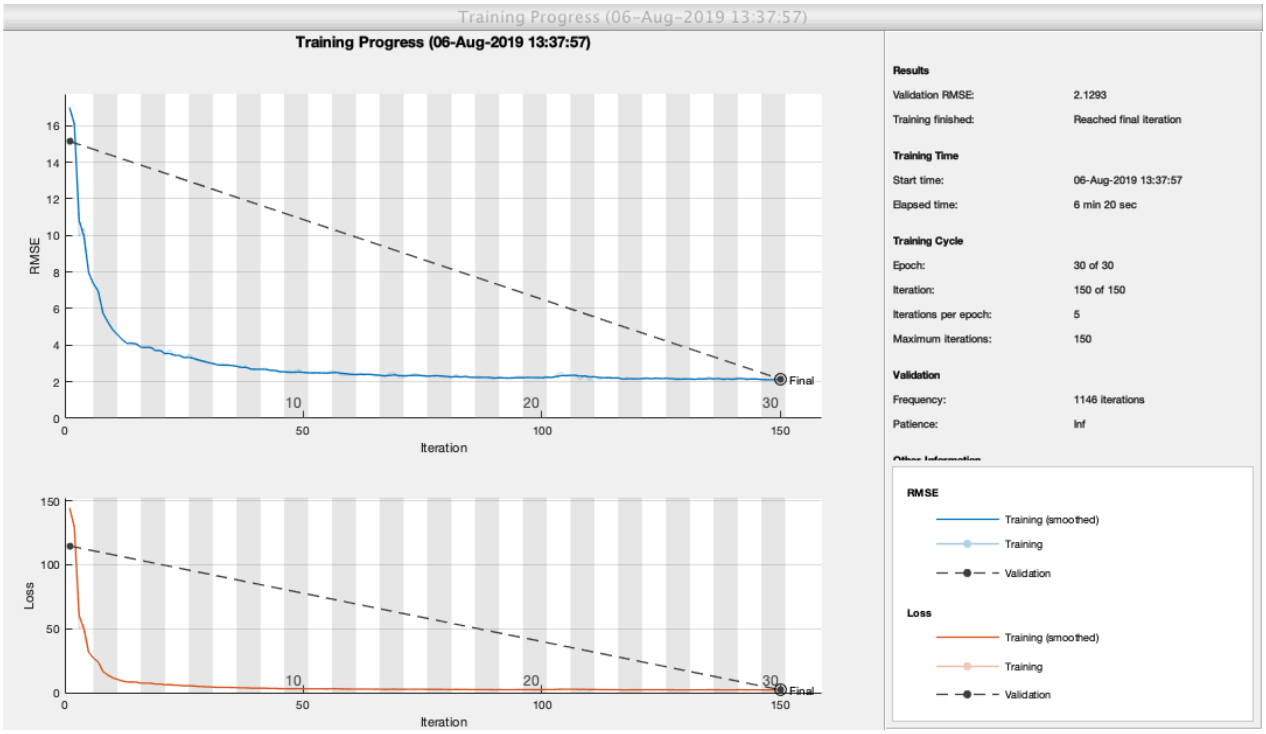
```
    XTrain(:, :, 1, sliding_window_iterator) = force2gray_scalogram(force_y_noise(i:i+s

    f_window = force_y(i:i+sliding_window);
    f_window_plus = abs(f_window) > accLearnThreshold;
    f_window_plus = max(f_window_plus,[0 0 0 0 0 0 f_window_plus(10:190)' 0 0 0 0 0 0 0
    YTrain(sliding_window_iterator,:) = round(smooth(f_window_plus(1:200)));

    sliding_window_iterator = sliding_window_iterator + 1;
    %YTrain(i,:) = round(force_markers(i:i+sliding_window-1)');
end

sliding_window_iterator = 1;
for i = (train_cnt+1)*sliding_window:sliding_window:(train_cnt + validation_cnt)*slidin
    %i_corr = i+ train_cnt;

    XValidation(:, :,1, sliding_window_iterator) = force2gray_scalogram(force_y_noise(i

    f_window = force_y(i:i+sliding_window);
    f_window_plus = abs(f_window) > accLearnThreshold;
    f_window_plus = max(f_window_plus,[0 0 0 0 0 0 f_window_plus(10:190)' 0 0 0 0 0 0 0
    YValidation(sliding_window_iterator,:) = round(smooth(f_window_plus(1:200)));


    sliding_window_iterator = sliding_window_iterator + 1;
    %YValidation(i,:) = round(force_markers(i:i+sliding_window-1)');
end
```

Обучение

```
layers = [
    imageInputLayer([row_px col_px 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)
```

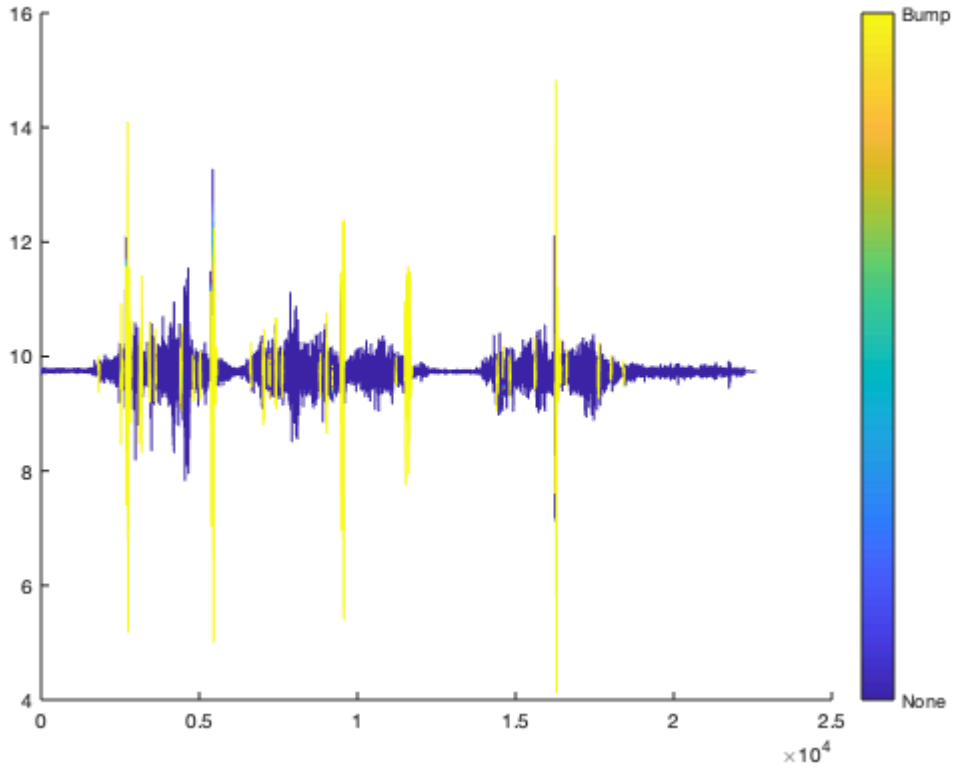
```
    fullyConnectedLayer(200)
    regressionLayer];

%%

miniBatchSize  = 128;
validationFrequency = floor(numel(YTrain)/miniBatchSize);

options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',30, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',validationFrequency, ...
    'Plots','training-progress', ...
    'Verbose',false);

net_noise_scal = trainNetwork(XTrain,YTrain,layers,options);
```





Предсказание

```
load('/Users/pauladm/MATLAB-Drive/RUPAS/ML/wf_spec_cl/spec_cnn_wf_100H_10V.mat', 'max_s
load('/Users/pauladm/MATLAB-Drive/RUPAS/ML/wf_spec_cl/spec_cnn_wf_100H_10V.mat', 'net_n
y = Acceleration.Z'; % force_y_noise';
x = 1:length(y);

%x = 0:.05:2*pi;
%y = sin(x);
z = predictForceClScal(net_noise_scal, y, max_scalogram_psd)';
col = round(smooth(round(z))');  % This is the color, vary with x in this case.
%col = z;

figure;
surface([x(1:length(z));x(1:length(z))],[y(1:length(z));y(1:length(z))],[z(1:length(z))
        'facecol','no',...
        'edgecol','interp',...
        'linew',1);
%colormap hot;
colorbar('Ticks',[0,1],...
         'TickLabels',{'None','Bump'})
```

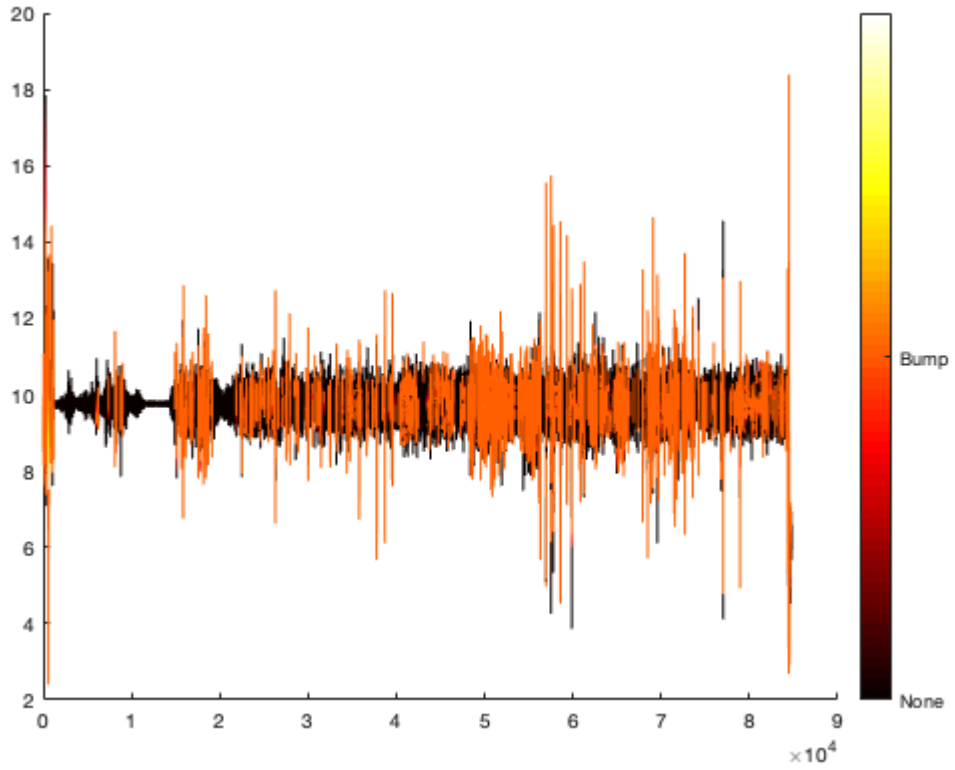



```
previous_marker = 0;
event_start = 0;
event_end = 0;
events = cell(1,5);
events_count = 0;


for i = 1:length(col)
    if previous_marker == 0 && col(i) == 1
        event_start = i;
    elseif previous_marker == 1 && col(i) == 0
        event_end = i;
    end
    previous_marker = col(i);

    if event_end > event_start
        events_count = events_count + 1;
        events{events_count,1} = event_start;
        events{events_count,2} = event_end;
        events{events_count,3} = event_end-event_start;
        events{events_count,4} = y(event_start:event_end);


        F = fft(y(event_start:event_end));
        pow = F.*conj(F);
```

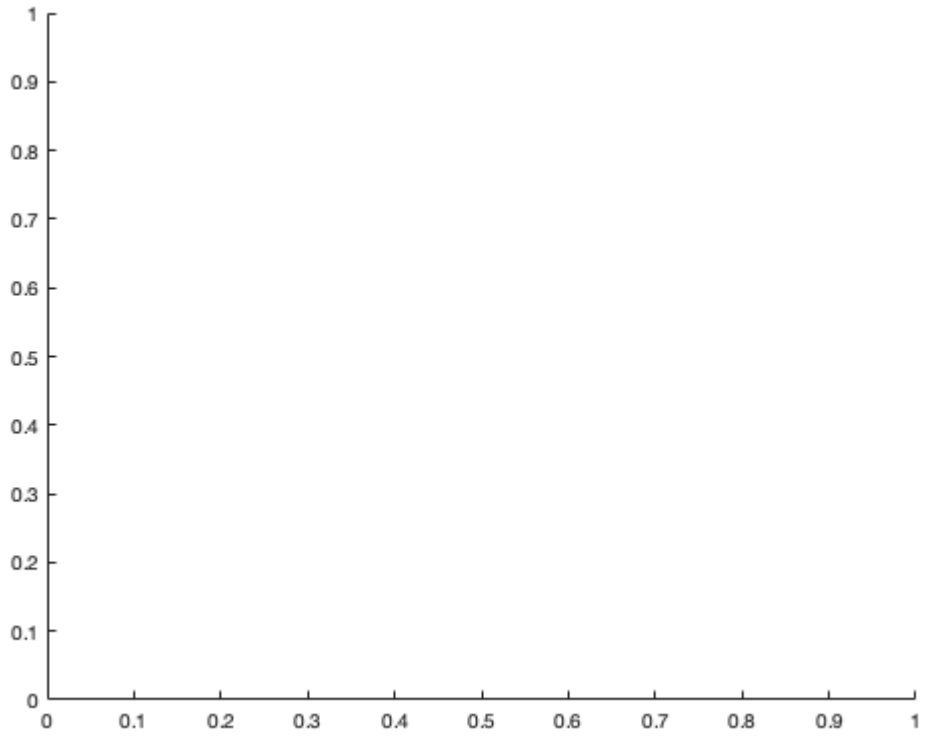
```
        events{events_count,5} = sum(pow);

        event_start = 0;
        event_end = 0;

    end



end
```

```

```

```
%load('sensorlog_20190326_224758.mat')
y = Acceleration.Z';
x = 1:length(y);

%x = 0:.05:2*pi;
%y = sin(x);
z = predictForceClScal(net_noise_scal, Acceleration.Z, 1.2)';
col = round(smooth(round(z))');  % This is the color, vary with x in this case.
%col = z;

figure;
surface([x(1:length(z));x(1:length(z))],[y(1:length(z));y(1:length(z))],[z(1:length(z))
        'facecol','no',...
        'edgecol','interp',...
        'linew',1);
colormap hot;
colorbar('Ticks',[0,1],...
         'TickLabels',{'None','Bump'})
```




```
 %%

% Level for Color Change
lev = 0.2;
% Find points above the level
aboveLine = predictForceClScal(net_noise_scal2, Acceleration.Z, max_scalogram_psd) == 1
bumpLine = predictForceClScal(net_noise_scal2, Acceleration.Z, max_scalogram_psd) == 2;
%aboveLine =  (round(predict(net_noise_scal, force2gray_scalogram(y-mean(y),20, max_sca
% Create 2 copies of y
bottomLine = y;
topLine = y;
% Set the values you don't want to get drawn to nan
bottomLine(aboveLine) = NaN;
topLine(~aboveLine) = NaN;

figure;
hold on
```




```
plot(x,bottomLine,'r',x,topLine,'g',x, bumpLine,'b');
```

```
Error using plot
Vectors must be the same length.
```

```
%plot(upsample(Position.speed,20));
legend({'Нет ям','Есть ямы'})
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
hold off
figure;
hold on
plot(x,bottomLine,'r',x,topLine,'g');
%plot(upsample(Position.speed,20));
legend({'Нет ям','Есть ямы'})
xlabel('Отсчеты времени [20 Гц]')
ylabel('Ускорение [м/с2]')
xlim([2297 3157])
ylim([6.00 12.64])
hold off
```

Для сравнения, вспомним, каков результат был при аналогичном процессе с использованием спектрограмм: