```
%Compare the results for simple operations (std, rms and mean) using your
own code and the functions provided by Matlab

SZ = 1000;
noisy_signal = randn(1,SZ);
%%%%%%%%
a = 0
```

```
a =
0
```

```
n=0
```

```
n =
0
```

```
for i = 1:length(noisy_signal);
    a = a + noisy_signal(i);
end
mean_written= a/SZ
```

```
mean_written =
-0.0326
```

```
mean_function = mean(noisy_signal)
```

```
mean_function =
-0.0326
```

```
%
% % %standard deviation
% b = 0
% for i = 1:length(noisy_signal);
%     b = (noisy_signal(i) - mean_written)^2 + b;
% end
%
% std_function = std(noisy_signal)
% std_written = sqrt(b/SZ)
%
%rms
c=0
```

```
c =
0
```

```
for i = 1:length(noisy_signal);
    c = (noisy_signal(i)^2) +c;
end
rms_written = sqrt((c) / SZ)
```

```
rms_written =
0.9990
```

```
rms_function = rms(noisy_signal)
```

```
rms_function =
0.9990
```

```
%Compute the max and min values of the signals. Repeat this procedure but
identify the max and min values between 1 and 2 seconds.
```

```
sz =
1000
```

```
f= 500;
a= 5;

%index/sampling rate = seconds

sz = 1000 %1000 samples
t = linspace(0,5,sz);

%regular signal
base_signal = a * sin(2 * pi * f * t);

%create noise
rng(1, 'twister');
noise = randn(1, sz);
s = rng;

noisy_signal = noise +base_signal;


% max and minimum of whole signal
[max_y, index] = max(noisy_signal)
```

```
max_y =
7.2277
index =
712
```

```
x_at_max_y = t(index)
```

```
x_at_max_y =
3.5586
```

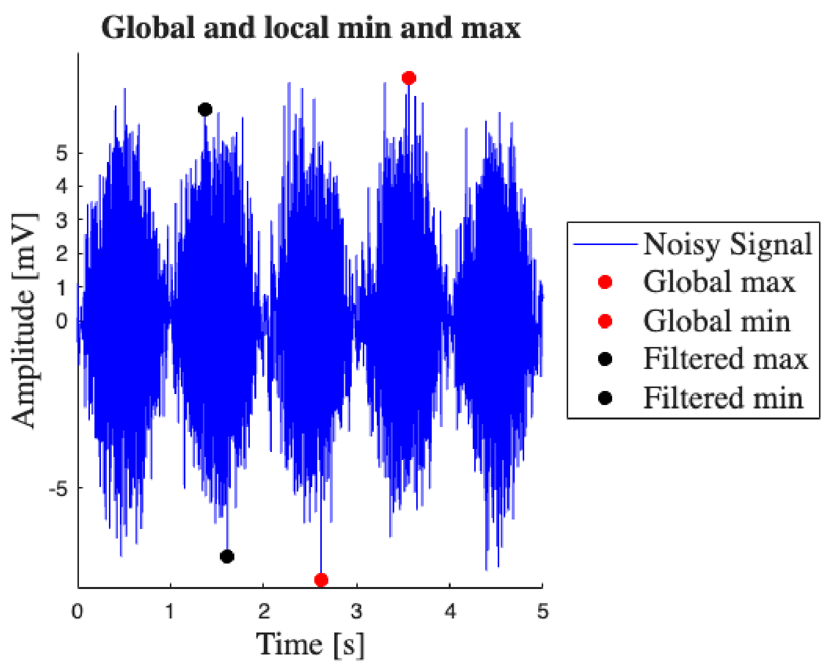
```
[min_y, index_min] = min(noisy_signal)
```

```
min_y =
-7.7277
index_min =
524
```

```
x_at_min_y = t(index_min)
```

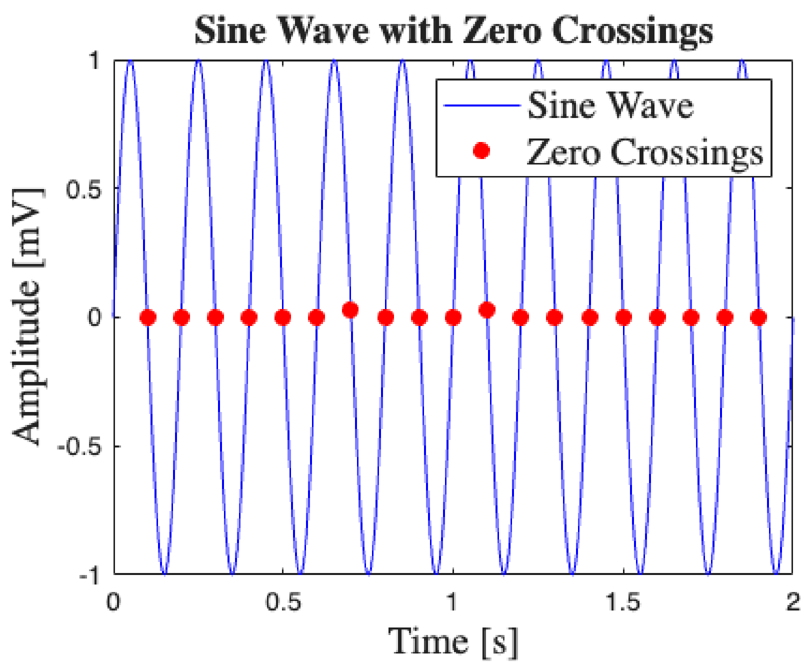
```
x_at_min_y =
2.6176
```

```
%max and min within 1-2 seconds
start_time = 1;
end_time = 2;
idx = t >= start_time & t <= end_time;

[max_in_range, local_max_idx] = max(noisy_signal(idx));
global_indices = find(idx);   % gives the actual indices in full vector
x_at_max_y_filtered = t(global_indices(local_max_idx));

[min_in_range, local_min_idx] = min(noisy_signal(idx));
x_at_min_y_filtered = t(global_indices(local_min_idx));



% [minA,maxA] = bounds(noisy_signal,"all") %computes the minimum and maximum
values over all elements of A.
% disp([minA,maxA])

% min_value = t,-8.5435
% max_value = 7.1983
%
figure;
hold on;
plot(t,noisy_signal,"b")  %signal
hold on;
plot(x_at_max_y, max_y, 'r.', 'MarkerSize', 20); % Plot a red circle at the
maximum point
hold on;
plot(x_at_min_y, min_y, 'r.', 'MarkerSize', 20);% Plot a red circle at the
maximum point
hold on;
plot(x_at_max_y_filtered , max_in_range, 'k.', 'MarkerSize', 20);
hold on;
plot(x_at_min_y_filtered , min_in_range, 'k.', 'MarkerSize', 20);
hold off;


xlabel("Time [s]", "FontSize",15, "FontName","Times");
ylabel("Amplitude [mV]", "FontSize",15, "FontName","Times");
title("Global and local min and max","FontSize",15, "FontName","Times");
yticks([-5 0 1 2 3 4 5])
legend("Noisy Signal", "Global max", "Global min", "Filtered max", "Filtered
min", "FontSize",15, "FontName","Times",'location','eastoutside');
```




```
%

% plot(index_max,max_y, 'r.', 'MarkerSize', 20);
% % hold off;
```

```
%zero crossings

fs = 1000;                    % sampling frequency (Hz)
t = 0:1/fs:2;                 % time vector, 2 seconds long
f = 5;                        % frequency of sine wave (Hz)
y = sin(2*pi*f*t);            % sine wave

% --- Find zero crossings ---
zc_idx = find(y(1:end-1) .* y(2:end) < 0);   % indices where sign changes
zc_time = t(zc_idx);                          % times of zero crossings
zc_val = y(zc_idx);                           % values at zero crossings
(near 0)

% --- Plot ---
figure;
plot(t, y, 'b'); hold on;
plot(zc_time, zc_val, 'ro', 'MarkerFaceColor','r');
xlabel('Time [s]',  "FontSize",15, "FontName","Times");
ylabel('Amplitude [mV]',  "FontSize",15, "FontName","Times");
title('Sine Wave with Zero Crossings',  "FontSize",15, "FontName","Times");
legend('Sine Wave', 'Zero Crossings',  "FontSize",15, "FontName","Times");
```

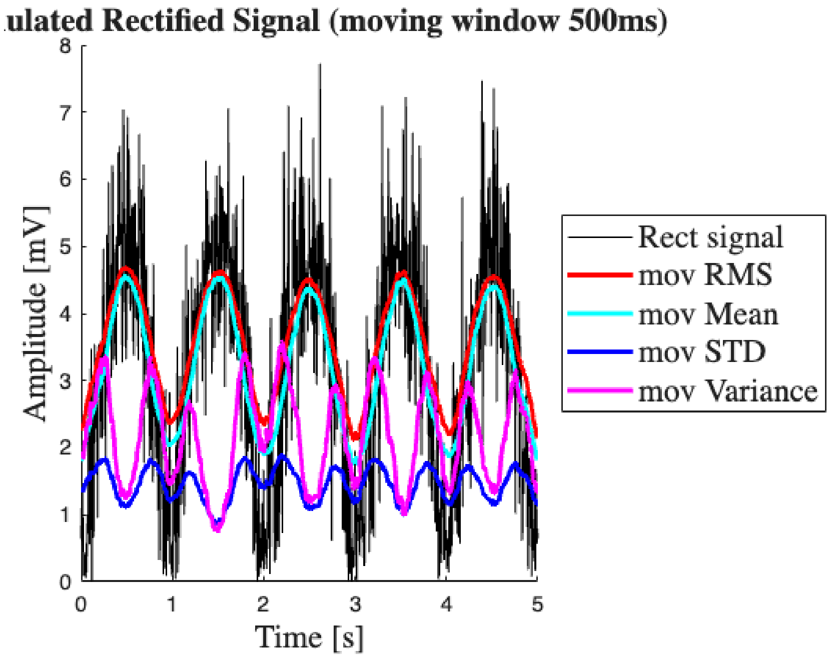



```
frequency = 1 / (2 / (20/2))
```

```
frequency =
5
```

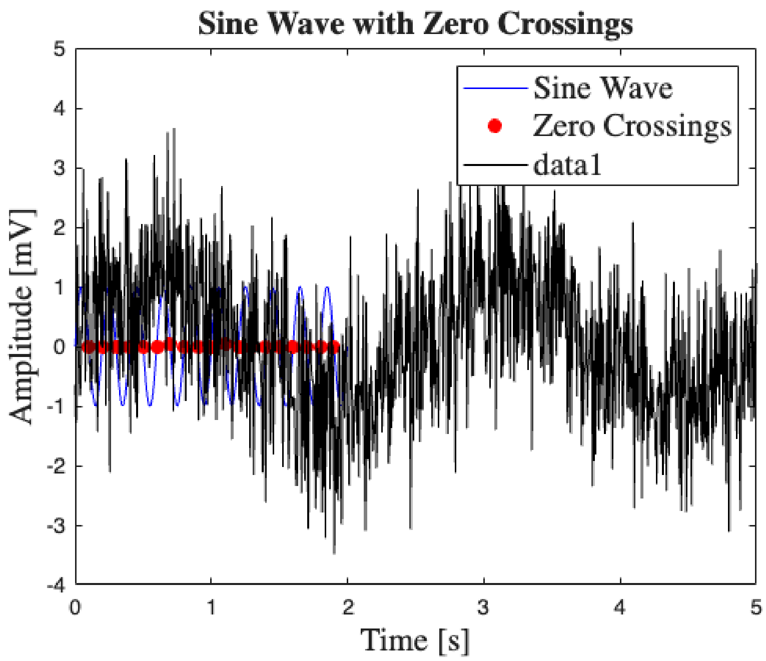
```
%rectify mean, then take moving mean
rectified_signal= abs(noisy_signal);

moving_mean = movmean(rectified_signal,100); %this is a 500ms moving average

rms_base =sqrt(movmean(rectified_signal.^2, 100));

%moving standard deviation
stdev = movstd(rectified_signal, 100);

%moving variance
varience = movvar(rectified_signal, 100);



figure;
hold on;
plot(t,rectified_signal,"k", "LineWidth", 0.5);  %signal
hold on;
plot(t,rms_base, "r",  "LineWidth", 2);
hold on;
plot(t,moving_mean, "c",  "LineWidth", 2);
```

```
hold on;
plot(t,stdev, "b",  "LineWidth", 2);
hold on;
plot(t,varience, "m",  "LineWidth", 2);
hold off;

xlabel("Time [s]", "FontSize",15, "FontName","Times");
ylabel("Amplitude [mV]", "FontSize",15, "FontName","Times");
title("Simulated Rectified Signal (moving window 500ms)","FontSize",15,
"FontName","Times");
legend("Rect signal", "mov RMS", "mov Mean", "mov STD", "mov Variance",
"FontSize",15, "FontName","Times",'location','eastoutside');
```



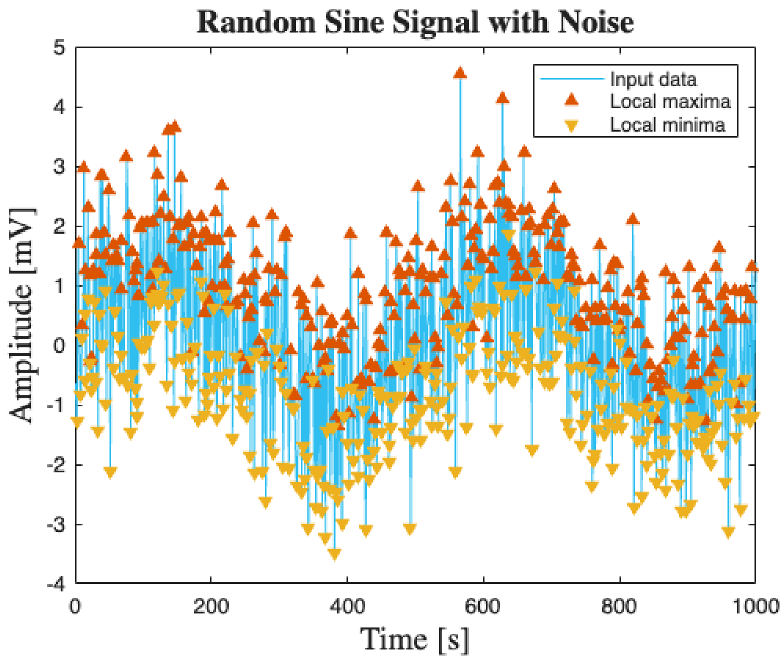

```
%detect local maxima and local minima

sz = 1000;
rng(1, 'twister');
noise_trials = randn(sz);
s = rng;


SZ = 1:sz;
SZ = SZ / (sz/2);

S = sin(2*pi*SZ);

t = linspace(0,5,sz);
```

```
for i = 1:sz   % loop over number of trials
    noise_trials(i,:) = noise_trials(i,:) + S;
end

average = sum(noise_trials,1) / sz;
signal = (noise_trials(1,:));

plot(t,noise_trials(1,:),"k");
```




```
% Find local maxima and minima
maxIndices = islocalmax(signal);
minIndices = islocalmin(signal);

% Display results
figure
plot(signal,SeriesIndex=6,DisplayName="Input data")
hold on

% Plot local maxima
scatter(find(maxIndices),signal(maxIndices),"^","filled",SeriesIndex=2, ...
    DisplayName="Local maxima")

% Plot local minima
scatter(find(minIndices),signal(minIndices),"v","filled",SeriesIndex=3, ...
    DisplayName="Local minima")
hold off
legend
xlabel("Time [s]", "FontSize",15, "FontName","Times");
ylabel("Amplitude [mV]", "FontSize",15, "FontName","Times");
```

```
title("Random Sine Signal with Noise","FontSize",15, "FontName","Times");
```




```
%Detect all data points that are more (+ and -) than 1SD from the mean.

signal_mean = mean(signal)
```

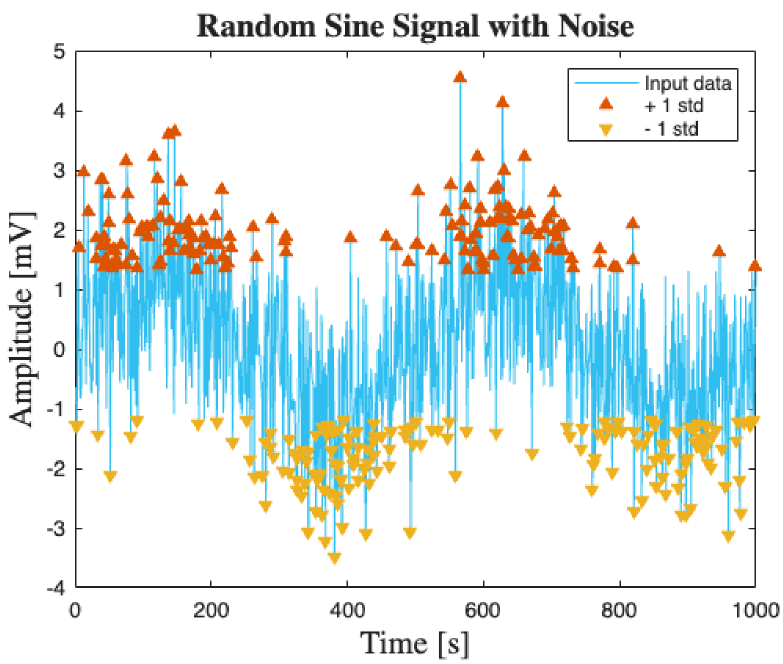
```
signal_mean =
0.0625
```

```
stdev_signal = std(signal)
```

```
stdev_signal =
1.2561
```

```
%detect upper and lower boiunds
lower_bound = signal_mean-stdev_signal;
upper_bound = signal_mean+stdev_signal;

data_upper = signal > upper_bound;
data_lower = signal < lower_bound;

% Display results
figure
plot(signal,SeriesIndex=6,DisplayName="Input data")
hold on

% Plot local maxima
scatter(find(data_upper),signal(data_upper),"^","filled",SeriesIndex=2, ...
```

```
    DisplayName="+ 1 std")

% Plot local minima
scatter(find(data_lower),signal(data_lower),"v","filled",SeriesIndex=3, ...
    DisplayName="- 1 std")
title("Number of extrema: " + (nnz(maxIndices)+nnz(minIndices)))
hold off
legend
xlabel("Time [s]", "FontSize",15, "FontName","Times");
ylabel("Amplitude [mV]", "FontSize",15, "FontName","Times");
title("Random Sine Signal with Noise","FontSize",15, "FontName","Times");
```




```
%zero crossing = 1/(t/(zc/2))
%t= time

%make signal mean an array for subtraction

no_mean_signal = zeros(1);

for i= 1:length(signal)
    no_mean_signal(end+1) = i - signal_mean;
end


% no_mean_signal = signal.- signal_mean
```

```
% Append to a numeric array
myArray = []; % Initialize an empty numeric array

for i = 1:5
    myArray(end + 1) = i * 10; % Append new elements
end

disp(myArray);
```

```
    10    20    30    40    50
```

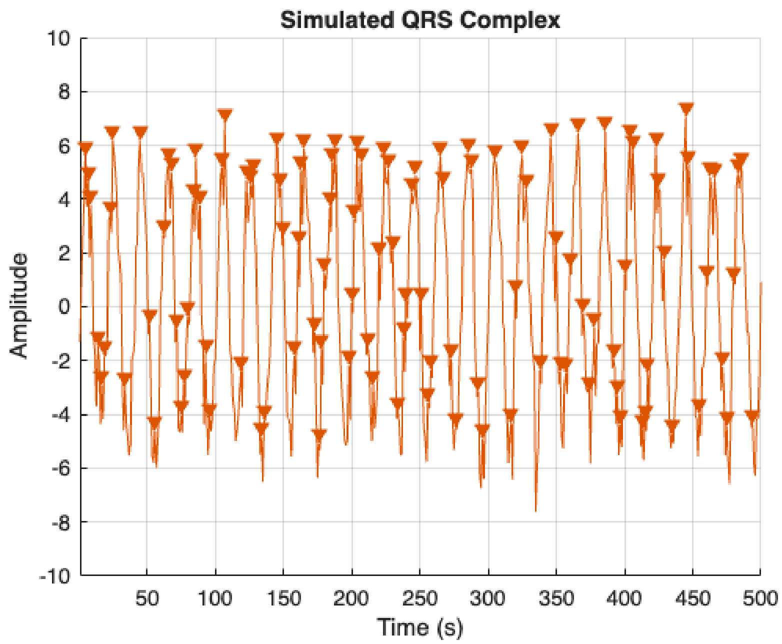
```
%create a normal signal
f= 25;
a= 5;
phase = 3/pi';

sz = 500 %500 samples
```

```
sz =
500
```

```
t = linspace(0,1,sz);

%regular signal
base_signal = a * sin(2 * pi * f * t);

%create noise
noise = randn(1, sz);

%noisy signal
noisy_signal = noise +base_signal;


figure;
hold on;
plot(t,noisy_signal,"b")
%plot(t,base_signal, "r")

%find peaks
findpeaks(noisy_signal)

xlabel('Time (s)')
ylabel('Amplitude')
title('Simulated QRS Complex')
grid on

rms = sqrt(mean((noisy_signal.^2)))
```

```
rms =
3.6703
```

```
plot(t, noisy_signal)
```




```
% Calculate the Signal-to-Noise Ratio (SNR)
snrValue = 20 * log10(rms / std(noise));
disp(['Signal-to-Noise Ratio (SNR): ', num2str(snrValue), ' dB']);
```

```
Signal-to-Noise Ratio (SNR): 10.7355 dB
```

```
% Parameters
Fs = 1000;              % Sampling frequency (Hz)
T = 1;                  % Duration of one heartbeat (s)
t = 0:1/Fs:T;           % Time vector

% QRS-like waveform using Gaussian pulses
QRS = zeros(size(t));

% Define Q, R, S positions (in seconds)
t_Q = 0.4;
t_R = 0.45;
t_S = 0.5;

% Amplitudes
A_Q = -0.5;
A_R = 1;
```

```
A_S = -0.75;

% Standard deviation of each pulse
sigma = 0.01;  % width of the spikes

% Create Gaussian pulses for Q, R, S
QRS = A_Q * exp(-((t - t_Q).^2)/(2*sigma^2)) + ...
      A_R * exp(-((t - t_R).^2)/(2*sigma^2)) + ...
      A_S * exp(-((t - t_S).^2)/(2*sigma^2));

% Optional: add small baseline noise
QRS = QRS + 0.05*randn(size(QRS));

% Plot
plot(t, QRS, 'LineWidth', 2)
xlabel('Time (s)')
ylabel('Amplitude')
title('Simulated QRS Complex')
grid on
```