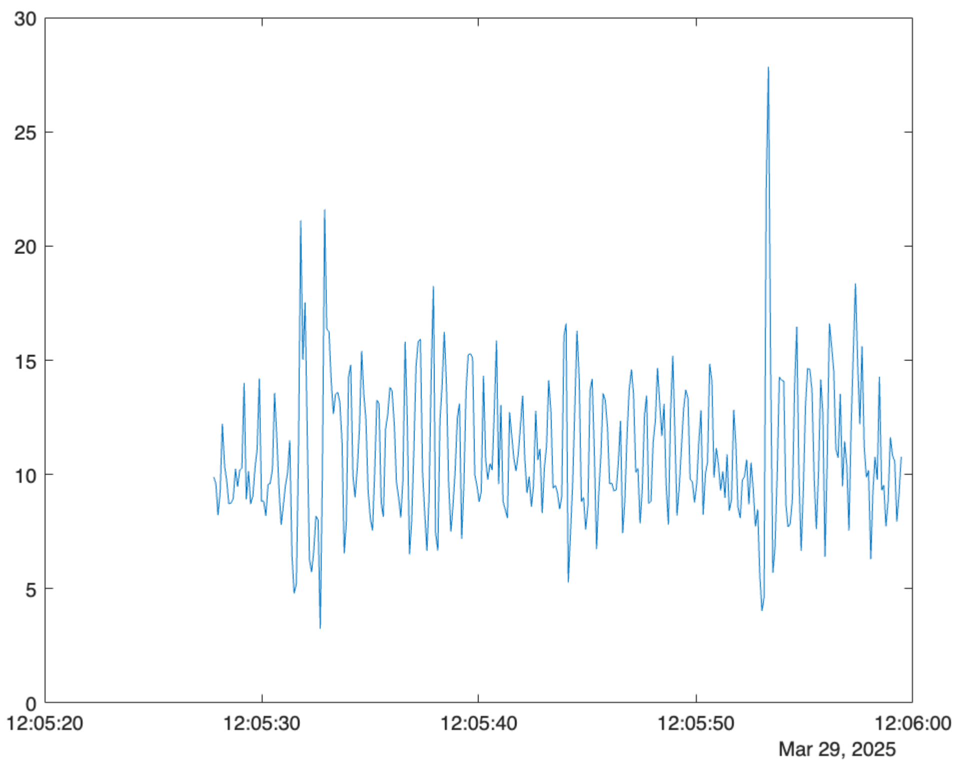
Implementing a Walking Step Counter using Acceleromter Sensor

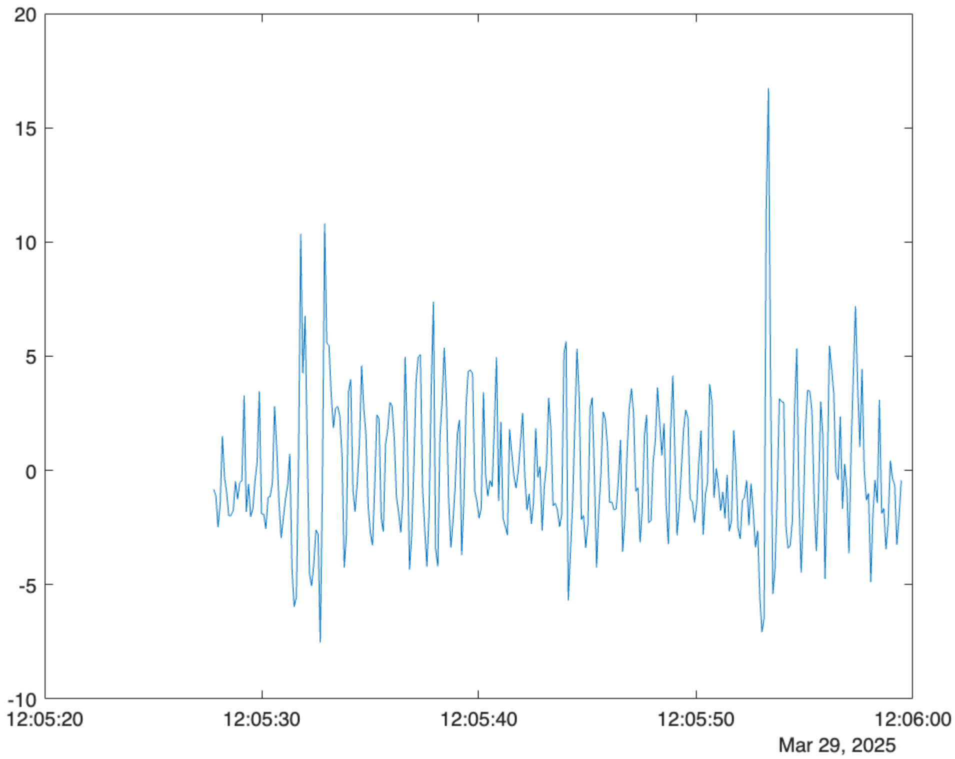
Step-1: Load Step signal and plot signal data

```
% Load Data (If not already loaded)
load steps.mat
% Extract X, Y, and Z columns
ax = Acceleration.X;
ay = Acceleration.Y;
az = Acceleration.Z;

% Compute Acceleration Magnitude
acc_magnitude = sqrt(ax.^2 + ay.^2 + az.^2);
fs=10;
% Add the computed magnitude to the timetable
Acceleration.Magnitude = acc_magnitude;

plot(Acceleration.Timestamp,Acceleration.Magnitude)
```



Step-2:Eliminate the trend in signal data

```
designal=detrend(acc_magnitude);
plot(Acceleration.Timestamp,designal);
```

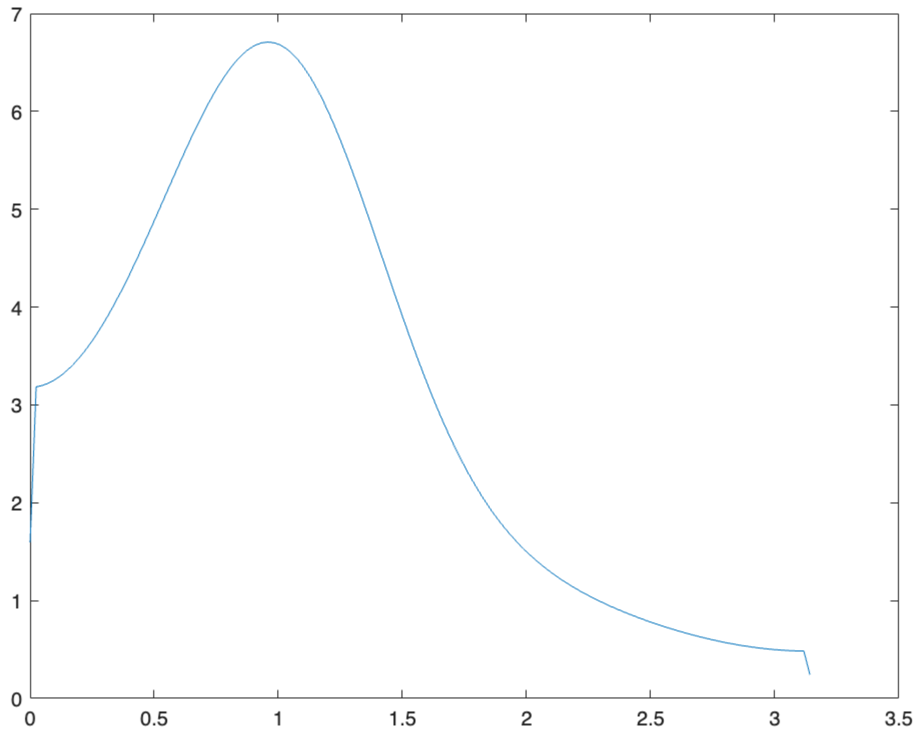


Step-3:Convert time domain to frequency domain

```
[pxx,fxx]=pwelch(designal,fs)
```

```
pxx = 129×1
    1.5900
    3.1846
    3.1988
    3.2223
    3.2551
    3.2971
    3.3481
    3.4078
    3.4762
    3.5529
      ⋮
fxx = 129×1
         0
    0.0245
    0.0491
    0.0736
    0.0982
    0.1227
    0.1473
    0.1718
    0.1963
    0.2209
      ⋮
```

```
plot(fxx,pxx)
```



Step 4-: Design low pass filter with automatic coding

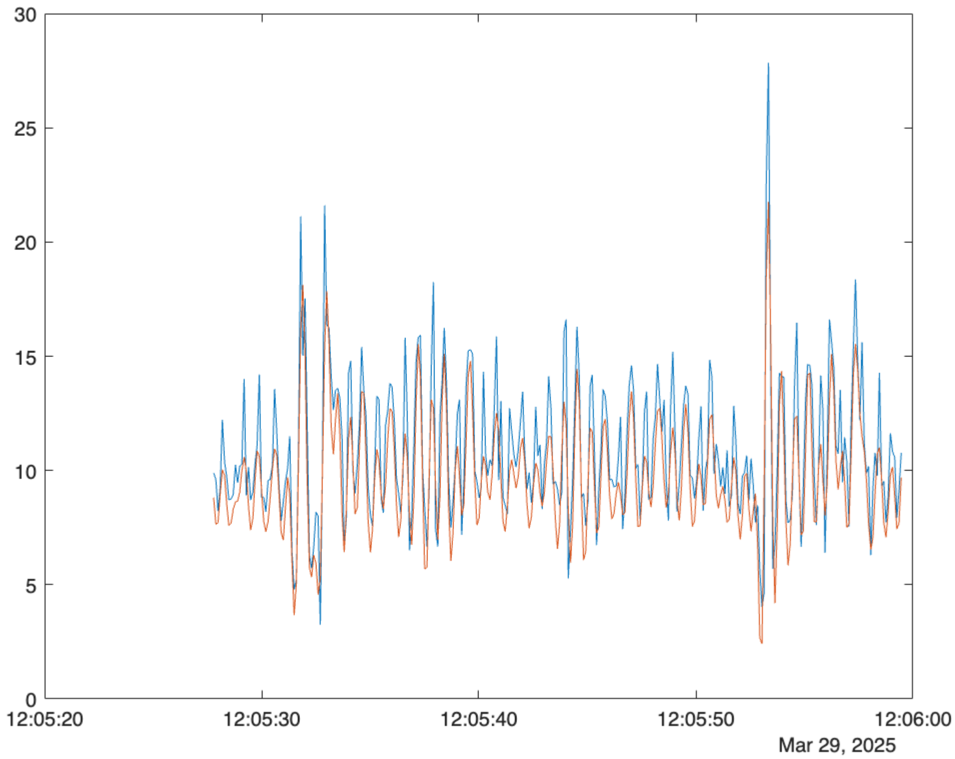
```
fs = 10;          % Sampling frequency (adjust if needed)
fc = 2;           % Cutoff frequency (Hz)
N = 4;            % Filter order
Rp = 0.5;         % Passband ripple (dB)

% Design Chebyshev Type I Low-Pass Filter
[b, a] = cheby1(N, Rp, fc/(fs/2), 'low');  % Normalize cutoff frequency
```

Apply Filter to the signal

```
filtered_data = filtfilt(b, a, acc_magnitude);  % Zero-phase filtering
figure;
```

Time Domain Visualisation

```
plot(Acceleration.Timestamp,Acceleration.Magnitude,Acceleration.Timestamp,fi
ltered_data)
```



Step 5:Peak finder

```
[peaks,locs]=findpeaks(filteredsignal)
```

```
peaks = 71×1
    1.5100
   -0.4753
    3.2783
   -0.5908
    3.4623
    2.8150
    0.7350
   10.3563
    6.7610
   -2.6008
       ⋮

locs = 71×1
     5
    11
    15
    17
    22
    29
    36
    41
    43
    48
     ⋮
```

Step 6:Total steps

```
xy=length(locs);
disp(xy)
```

```
71
```