**Code:**

% TF = isoutlier(A,method) %specifies a method for

% detecting outliers. For example , isoutlier

% (A, 'mean ') returns true for all elements more

% than three standard deviations from the mean

% Simulating a segment of APG signal and the

% traget is detecting the

A = [0.57 0.59 0.60 0.1 0.59 0.58 0.57 0.58 0.3 0.61 0.62 0.60 0.62 0.58 0.57];

TF = isoutlier(A,'mean')

**Mô tả:**

Phát hiện **giá trị ngoại lai (outliers)** trong một đoạn tín hiệu APG được mô phỏng (A) bằng cách sử dụng hàm isoutlier trong MATLAB. Phương pháp được sử dụng là 'mean', xác định các giá trị ngoại lai nếu chúng cách xa trung bình hơn **3 lần độ lệch chuẩn**.

* **A**: Là một đoạn tín hiệu APG mô phỏng. APG là một tín hiệu liên quan đến sự thay đổi thể tích máu trong mô vi mạch.
* **isoutlier**: Hàm trong MATLAB để phát hiện giá trị ngoại lai dựa trên phương pháp do người dùng chỉ định.
* **Phương pháp 'mean'**: Đánh dấu các giá trị là ngoại lai nếu chúng vượt quá 3 lần độ lệch chuẩn từ giá trị trung bình của dãy dữ liệu.

**Kết quả:**

****

**Code:**

clear

function [error\_code,ppg\_feature] = ppg\_feature\_calculation(error\_code,sample\_time,PPG\_Loc,VPG\_Loc,APG\_Loc,ppg,vpg,apg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ppg\_feature\_calculation.m

% Defination and Calculation of PPG Features

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Input:

%error\_code————Extracted PPG feature points or not. ( 0 :YES, 1: NO)

%sample\_time————The sample time of PPG signal

%PPG\_Loc————The location of PPG feature points

%VPG\_Loc————The location of VPG feature points

%APG\_Loc————The location of APG feature points

%ppg————PPG waveform of two adjacent heartbeat cycles

%vpg————VPG waveform of two adjacent heartbeat cycles

%apg————APG waveform of two adjacent heartbeat cycles

%% Output:

%error\_code————Updated error code for calculation. ( 0 :YES, 1: NO)

%ppg\_feature————A Structure of PPG features

%% Dilive PPG feature point location tovariables

num\_O = PPG\_Loc(1);

num\_S = PPG\_Loc(2);

num\_N = PPG\_Loc(3);

num\_D = PPG\_Loc(4);

num\_O\_next = PPG\_Loc(5);

num\_w = VPG\_Loc(1);

num\_y = VPG\_Loc(2);

num\_z = VPG\_Loc(3);

num\_w\_next = VPG\_Loc(4);

num\_a = APG\_Loc(1);

num\_b = APG\_Loc(2);

num\_c = APG\_Loc(3);

num\_d = APG\_Loc(4);

num\_e = APG\_Loc(5);

num\_b2 = APG\_Loc(6);

ppg\_feature.Total = zeros(1,125);

if error\_code == 0

%% PPG Feature Type 1: Time Span

Tm\_Oa = (num\_a - num\_O)\* sample\_time;

Tm\_Ow = (num\_w - num\_O)\* sample\_time;

Tm\_Ob = (num\_b - num\_O)\* sample\_time;

Tm\_OS = (num\_S - num\_O)\* sample\_time;

Tm\_Oc = (num\_c - num\_O)\* sample\_time;

Tm\_Oy = (num\_y - num\_O)\* sample\_time;

Tm\_ON = (num\_N - num\_O)\* sample\_time;

Tm\_OD = (num\_D - num\_O)\* sample\_time;

Tss = (num\_w\_next - num\_w)\*sample\_time;

Tm\_Sc = (num\_c-num\_S)\* sample\_time;

Tm\_Sd = (num\_d-num\_S)\* sample\_time;

Tm\_Se = (num\_e-num\_S)\* sample\_time;

Tm\_SD = (num\_D - num\_S)\* sample\_time;

Tm\_ND = (num\_D - num\_N)\* sample\_time;

Tm\_bb2 = (num\_b2 - num\_b)\* sample\_time;

Tm\_bc = (num\_c-num\_b)\* sample\_time;

Tm\_bd = (num\_d-num\_b)\* sample\_time;

Tm\_wb = (num\_b-num\_w)\* sample\_time;

Tm\_wS = (num\_S-num\_w)\* sample\_time;

Tm\_wc = (num\_c-num\_w)\* sample\_time;

Tm\_wd = (num\_d-num\_w)\* sample\_time;

Tm\_wz = (num\_z - num\_w)\* sample\_time;

Tm\_ac = (num\_c -num\_a)\* sample\_time;

ppg\_feature.TimeSpan(1) = Tm\_Oa;

ppg\_feature.TimeSpan(2) = Tm\_Ow;

ppg\_feature.TimeSpan(3) = Tm\_Ob;

ppg\_feature.TimeSpan(4) = Tm\_OS;

ppg\_feature.TimeSpan(5) = Tm\_Oc;

ppg\_feature.TimeSpan(6) = Tm\_Oy;

ppg\_feature.TimeSpan(7) = Tm\_ON;

ppg\_feature.TimeSpan(8) = Tm\_OD;

ppg\_feature.TimeSpan(9) = Tss;

ppg\_feature.TimeSpan(10) = Tm\_Sc;

ppg\_feature.TimeSpan(11) = Tm\_Sd;

ppg\_feature.TimeSpan(12) = Tm\_Se;

ppg\_feature.TimeSpan(13) = Tm\_SD;

ppg\_feature.TimeSpan(14) = Tm\_ND;

ppg\_feature.TimeSpan(15) = Tm\_bb2;

ppg\_feature.TimeSpan(16) = Tm\_bc;

ppg\_feature.TimeSpan(17) = Tm\_bd;

ppg\_feature.TimeSpan(18) = Tm\_wb;

ppg\_feature.TimeSpan(19) = Tm\_wS;

ppg\_feature.TimeSpan(20) = Tm\_wc;

ppg\_feature.TimeSpan(21) = Tm\_wd;

ppg\_feature.TimeSpan(22) = Tm\_wz;

ppg\_feature.TimeSpan(23) = Tm\_ac;

%% PPG Feature Type 2: Features of PPG Amplitude

AMS = ppg(num\_S) - ppg(num\_O);

Am\_Oa = ppg(num\_a) - ppg(num\_O);

Am\_Ow = ppg(num\_w) - ppg(num\_O);

Am\_Ob = ppg(num\_b) - ppg(num\_O);

Am\_Oc = ppg(num\_c) - ppg(num\_O);

Am\_Oy = ppg(num\_y) - ppg(num\_O);

Am\_OO2 = ppg(num\_O\_next) - ppg(num\_O);

Am\_OD = ppg(num\_D) - ppg(num\_O);

Am\_ON = ppg(num\_N) - ppg(num\_O);

Am\_NS = ppg(num\_S) - ppg(num\_N);

AI\_ON\_AMS = Am\_ON/AMS;

AI\_OD\_AMS = Am\_OD/AMS;

AI\_NS\_AMS = Am\_NS/AMS;

AI\_DS\_AMS = (ppg(num\_S)-ppg(num\_D))/AMS;

ppg\_feature.Amplitude(1) = AMS;

ppg\_feature.Amplitude(2) = Am\_Oa;

ppg\_feature.Amplitude(3) = Am\_Ow;

ppg\_feature.Amplitude(4) = Am\_Ob;

ppg\_feature.Amplitude(5) = Am\_Oc;

ppg\_feature.Amplitude(6) = Am\_Oy;

ppg\_feature.Amplitude(7) = Am\_OO2;

ppg\_feature.Amplitude(8) = Am\_OD;

ppg\_feature.Amplitude(9) = Am\_ON;

ppg\_feature.Amplitude(10) = Am\_NS;

ppg\_feature.Amplitude(11) = AI\_ON\_AMS;

ppg\_feature.Amplitude(12) = AI\_OD\_AMS;

ppg\_feature.Amplitude(13) = AI\_NS\_AMS;

ppg\_feature.Amplitude(14) = AI\_DS\_AMS;

%% PPG Feature Type 3: Features of VPG andAPG

w = vpg(num\_w);

y = vpg(num\_y);

z = vpg(num\_z);

a = apg(num\_a);

b = apg(num\_b);

c = apg(num\_c);

d = apg(num\_d);

e = apg(num\_e);

cc = vpg(num\_c);

dd = vpg(num\_d);

r\_z\_w = z/w;

r\_y\_w = y/w;

r\_cc\_w = cc/w;

r\_dd\_w = dd/w;

r\_b\_a = b/a;

r\_c\_a = c/a;

r\_d\_a = d/a;

r\_e\_a = e/a;

r\_bcde\_a =(b-c-d-e)/a;

r\_bcd\_a = (b-c-d)/a;

ppg\_feature.VpgApg(1) = w;

ppg\_feature.VpgApg(2) = y;

ppg\_feature.VpgApg(3) = z;

ppg\_feature.VpgApg(4) = a;

ppg\_feature.VpgApg(5) = b;

ppg\_feature.VpgApg(6) = c;

ppg\_feature.VpgApg(7) = d;

ppg\_feature.VpgApg(8) = e;

ppg\_feature.VpgApg(9) = cc;

ppg\_feature.VpgApg(10) = dd;

ppg\_feature.VpgApg(11) = r\_z\_w;

ppg\_feature.VpgApg(12) = r\_y\_w;

ppg\_feature.VpgApg(13) = r\_cc\_w;

ppg\_feature.VpgApg(14) = r\_dd\_w;

ppg\_feature.VpgApg(15) = r\_b\_a;

ppg\_feature.VpgApg(16) = r\_c\_a;

ppg\_feature.VpgApg(17) = r\_d\_a;

ppg\_feature.VpgApg(18) = r\_e\_a;

ppg\_feature.VpgApg(19) = r\_bcde\_a;

ppg\_feature.VpgApg(20) = r\_bcd\_a;

%% PPG Feature Type 4: Waveform Area

S\_OO = area\_calculate(ppg(num\_O),num\_O,num\_O\_next,ppg);

S\_OS = area\_calculate(ppg(num\_O),num\_O,num\_S,ppg);

S\_Oc = area\_calculate(ppg(num\_O),num\_O,num\_c,ppg);

S\_ON = area\_calculate(ppg(num\_O),num\_O,num\_N,ppg);

ppg\_feature.WavefromArea(1) = S\_OO;

ppg\_feature.WavefromArea(2) = S\_OS;

ppg\_feature.WavefromArea(3) = S\_Oc;

ppg\_feature.WavefromArea(4) = S\_ON;

%% PPG Feature Type 5: Power Area

power\_OS\_ppg = power\_area\_calculate(ppg(num\_O),num\_O, num\_S,ppg);

power\_wS\_ppg = power\_area\_calculate(ppg(num\_O),num\_w, num\_S,ppg);

power\_Sc\_ppg = power\_area\_calculate(ppg(num\_O),num\_S, num\_c,ppg);

power\_Sd\_ppg = power\_area\_calculate(ppg(num\_O),num\_S, num\_d,ppg);

power\_OS\_vpg = power\_area\_calculate(0,num\_O,num\_S,vpg);

power\_wS\_vpg = power\_area\_calculate(0,num\_w,num\_S,vpg);

power\_Sc\_vpg = power\_area\_calculate(0,num\_S,num\_c,vpg);

power\_Sd\_vpg = power\_area\_calculate(0,num\_S,num\_d,vpg);

power\_OS\_apg = power\_area\_calculate(0,num\_O,num\_S,apg);

power\_wS\_apg = power\_area\_calculate(0,num\_w,num\_S,apg);

power\_Sc\_apg = power\_area\_calculate(0,num\_S,num\_c,apg);

power\_Sd\_apg = power\_area\_calculate(0,num\_S,num\_d,apg);

power\_OO\_ppg = power\_area\_calculate(ppg(num\_O),num\_O,num\_O\_next,ppg);

power\_OO\_vpg = power\_area\_calculate(0,num\_O,num\_O\_next,vpg);

power\_OO\_apg = power\_area\_calculate(0,num\_O,num\_O\_next,apg);

ppg\_feature.PowerArea(1) = power\_OS\_ppg;

ppg\_feature.PowerArea(2) = power\_wS\_ppg;

ppg\_feature.PowerArea(3) = power\_Sc\_ppg;

ppg\_feature.PowerArea(4) = power\_Sd\_ppg;

ppg\_feature.PowerArea(5) = power\_OS\_vpg;

ppg\_feature.PowerArea(6) = power\_wS\_vpg;

ppg\_feature.PowerArea(7) = power\_Sc\_vpg;

ppg\_feature.PowerArea(8) = power\_Sd\_vpg;

ppg\_feature.PowerArea(9) = power\_OS\_apg;

ppg\_feature.PowerArea(10) = power\_wS\_apg;

ppg\_feature.PowerArea(11) = power\_Sc\_apg;

ppg\_feature.PowerArea(12) = power\_Sd\_apg;

ppg\_feature.PowerArea(13) = power\_OO\_ppg;

ppg\_feature.PowerArea(14) = power\_OO\_vpg;

ppg\_feature.PowerArea(15) = power\_OO\_apg;

%% PPG Feature Type 6: Ratio

r\_Tm\_Oa\_Tss = Tm\_Oa/Tss;

r\_Tm\_Ow\_Tss = Tm\_Ow/Tss;

r\_Tm\_Ob\_Tss = Tm\_Ob/Tss;

r\_Tm\_OS\_Tss = Tm\_OS/Tss;

r\_Tm\_Oc\_Tss = Tm\_Oc/Tss;

r\_Tm\_Oy\_Tss = Tm\_Oy/Tss;

r\_Tm\_ON\_Tss = Tm\_ON/Tss;

r\_Tm\_wz\_Tss = Tm\_wz/Tss;

r\_Tm\_SD\_Tss = Tm\_SD/Tss;

r\_Tm\_bb2\_Tss = Tm\_bb2/Tss;

r\_Oa\_AMS = Am\_Oa/AMS;

r\_Ow\_AMS = Am\_Ow/AMS;

r\_Ob\_AMS = Am\_Ob/AMS;

r\_Oc\_AMS = Am\_Oc/AMS;

r\_Oy\_AMS = Am\_Oy/AMS;

r\_OO2\_AMS = Am\_OO2/AMS;

S\_NO2 = area\_calculate(ppg(num\_O),num\_N,num\_O\_next,ppg);

IPA = S\_NO2/S\_ON;

PIR = ppg(num\_S)/ppg(num\_O);

r\_SOS\_SOO = S\_OS/S\_OO;

r\_SOc\_SOO = S\_Oc/S\_OO;

r\_SON\_SOO = S\_ON/S\_OO;

r\_OS\_OO\_ppg = power\_OS\_ppg/power\_OO\_ppg;

r\_wS\_OO\_ppg = power\_wS\_ppg/power\_OO\_ppg;

r\_Sc\_OO\_ppg = power\_Sc\_ppg/power\_OO\_ppg;

r\_Sd\_OO\_ppg = power\_Sd\_ppg/power\_OO\_ppg;

r\_OS\_OO\_vpg = power\_OS\_vpg/power\_OO\_vpg;

r\_wS\_OO\_vpg = power\_wS\_vpg/power\_OO\_vpg;

r\_Sc\_OO\_vpg = power\_Sc\_vpg/power\_OO\_vpg;

r\_Sd\_OO\_vpg = power\_Sd\_vpg/power\_OO\_vpg;

r\_OS\_OO\_apg = power\_OS\_apg/power\_OO\_apg;

r\_wS\_OO\_apg = power\_wS\_apg/power\_OO\_apg;

r\_Sc\_OO\_apg = power\_Sc\_apg/power\_OO\_apg;

r\_Sd\_OO\_apg = power\_Sd\_apg/power\_OO\_apg;

ppg\_feature.Ratio(1) = r\_Tm\_Oa\_Tss;

ppg\_feature.Ratio(2) = r\_Tm\_Ow\_Tss;

ppg\_feature.Ratio(3) = r\_Tm\_Ob\_Tss;

ppg\_feature.Ratio(4) = r\_Tm\_OS\_Tss;

ppg\_feature.Ratio(5) = r\_Tm\_Oc\_Tss;

ppg\_feature.Ratio(6) = r\_Tm\_Oy\_Tss;

ppg\_feature.Ratio(7) = r\_Tm\_ON\_Tss;

ppg\_feature.Ratio(8) = r\_Tm\_wz\_Tss;

ppg\_feature.Ratio(9) = r\_Tm\_SD\_Tss;

ppg\_feature.Ratio(10) = r\_Tm\_bb2\_Tss;

ppg\_feature.Ratio(11) = r\_Oa\_AMS;

ppg\_feature.Ratio(12) = r\_Ow\_AMS;

ppg\_feature.Ratio(13) = r\_Ob\_AMS;

ppg\_feature.Ratio(14) = r\_Oc\_AMS;

ppg\_feature.Ratio(15) = r\_Oy\_AMS;

ppg\_feature.Ratio(16) = r\_OO2\_AMS;

ppg\_feature.Ratio(17) = IPA;

ppg\_feature.Ratio(18) = PIR;

ppg\_feature.Ratio(19) = r\_SOS\_SOO;

ppg\_feature.Ratio(20) = r\_SOc\_SOO;

ppg\_feature.Ratio(21) = r\_SON\_SOO;

ppg\_feature.Ratio(22) = r\_OS\_OO\_ppg;

ppg\_feature.Ratio(23) = r\_wS\_OO\_ppg;

ppg\_feature.Ratio(24) = r\_Sc\_OO\_ppg;

ppg\_feature.Ratio(25) = r\_Sd\_OO\_ppg;

ppg\_feature.Ratio(26) = r\_OS\_OO\_vpg;

ppg\_feature.Ratio(27) = r\_wS\_OO\_vpg;

ppg\_feature.Ratio(28) = r\_Sc\_OO\_vpg;

ppg\_feature.Ratio(29) = r\_Sd\_OO\_vpg;

ppg\_feature.Ratio(30) = r\_OS\_OO\_apg;

ppg\_feature.Ratio(31) = r\_wS\_OO\_apg;

ppg\_feature.Ratio(32) = r\_Sc\_OO\_apg;

ppg\_feature.Ratio(33) = r\_Sd\_OO\_apg;

%% PPG Feature Type 7: Slope

m\_Sc\_ppg = (ppg(num\_c)-ppg(num\_S))/((num\_c -num\_S)\* sample\_time);

m\_Sd\_ppg = (ppg(num\_d)-ppg(num\_S))/((num\_d -num\_S)\* sample\_time);

m\_bS\_ppg = (ppg(num\_S)-ppg(num\_b))/((num\_S -num\_b)\* sample\_time);

m\_bc\_ppg = (ppg(num\_c)-ppg(num\_b))/((num\_c -num\_b)\* sample\_time);

m\_bd\_ppg = (ppg(num\_d)-ppg(num\_b))/((num\_d -num\_b)\* sample\_time);

m\_wS\_ppg = (ppg(num\_S)-ppg(num\_w))/((num\_S -num\_w)\* sample\_time);

m\_OS\_ppg = (ppg(num\_S)-ppg(num\_O))/((num\_S -num\_O)\* sample\_time);

m\_ab\_ppg = (ppg(num\_b)-ppg(num\_a))/((num\_b -num\_a)\* sample\_time);

m\_ab\_apg = (apg(num\_b)-apg(num\_a))/((num\_b -num\_a)\* sample\_time);

m\_bS\_apg = (apg(num\_S)-apg(num\_b))/((num\_S -num\_b)\* sample\_time);

m\_bc\_apg = (apg(num\_c)-apg(num\_b))/((num\_c -num\_b)\* sample\_time);

m\_bd\_apg = (apg(num\_d)-apg(num\_b))/((num\_d -num\_b)\* sample\_time);

m\_be\_apg = (apg(num\_e)-apg(num\_b))/((num\_e -num\_b)\* sample\_time);

m\_Sc\_apg = (apg(num\_c)-apg(num\_S))/((num\_c -num\_S)\* sample\_time);

m\_wS\_apg = (apg(num\_S)-apg(num\_w))/((num\_S -num\_w)\* sample\_time);

m\_OS\_apg = (apg(num\_S)-apg(num\_O))/((num\_S -num\_O)\* sample\_time);

ppg\_feature.Slope(1) = m\_Sc\_ppg;

ppg\_feature.Slope(2) = m\_Sd\_ppg;

ppg\_feature.Slope(3) = m\_bS\_ppg;

ppg\_feature.Slope(4) = m\_bc\_ppg;

ppg\_feature.Slope(5) = m\_bd\_ppg;

ppg\_feature.Slope(6) = m\_wS\_ppg;

ppg\_feature.Slope(7) = m\_OS\_ppg;

ppg\_feature.Slope(8) = m\_ab\_ppg;

ppg\_feature.Slope(9) = m\_ab\_apg;

ppg\_feature.Slope(10) = m\_bS\_apg;

ppg\_feature.Slope(11) = m\_bc\_apg;

ppg\_feature.Slope(12) = m\_bd\_apg;

ppg\_feature.Slope(13) = m\_be\_apg;

ppg\_feature.Slope(14) = m\_Sc\_apg;

ppg\_feature.Slope(15) = m\_wS\_apg;

ppg\_feature.Slope(16) = m\_OS\_apg;

%% Data Exclusion Stardard :

%If the amplitude of ’O’ in next heartbeatcycle is more than 50% baseline , theerror code will be set to 1

if 0.50 < abs(r\_O2\_AMS)

error\_code = 1;

ppg\_feature.Total = zeros(1,125);

else

ppg\_feature.Total = [ppg\_feature.TimeSpan ppg\_feature.Amplitude ppg\_feature.VpgApg ppg\_feature.WavefromArea ppg\_feature.PowerArea ppg\_feature.Ratio ppg\_feature.Slope];

end

end

end

**Mô tả:**

**Kết quả:**

**Code:**

clear

% Tạo dữ liệu mẫu intervals

intervals = 0.8 + 0.05 \* randn(1, 100); % Các khoảng thời gian (s), trung bình 0.8s, có nhiễu nhỏ

% Tính toán các chỉ số

SDNN = std(intervals) \* 1000; % Đơn vị: ms

diffNN = diff(intervals) \* 1000; % Đơn vị: ms

RMSSD = rms(diffNN);

SDSD = std(diffNN);

NN50 = length(find(abs(diffNN) > 50)); % Đếm số khoảng thay đổi > 50 ms

pNN50 = NN50 / length(intervals); % Tỷ lệ % của NN50

NN20 = length(find(abs(diffNN) > 20)); % Đếm số khoảng thay đổi > 20 ms

pNN20 = NN20 / length(intervals); % Tỷ lệ % của NN20

% Hiển thị kết quả

fprintf('SDNN: %.2f ms\n', SDNN);

fprintf('RMSSD: %.2f ms\n', RMSSD);

fprintf('SDSD: %.2f ms\n', SDSD);

fprintf('NN50: %d, pNN50: %.2f%%\n', NN50, pNN50 \* 100);

fprintf('NN20: %d, pNN20: %.2f%%\n', NN20, pNN20 \* 100);

**Mô tả:**

* **SDNN**: Độ lệch chuẩn của các khoảng RR. Đây là thước đo mức độ thay đổi của nhịp tim.
* **diffNN**: Sự khác biệt giữa các khoảng RR liên tiếp (được nhân với 1000 để đổi sang đơn vị milliseconds).
* **RMSSD**: Căn bậc hai của trung bình bình phương các giá trị trong diffNN. Đây là một chỉ số đo sự thay đổi của nhịp tim liên tục.
* **SDSD**: Độ lệch chuẩn của diffNN. Tương tự RMSSD nhưng tính theo phương sai.
* **NN50**: Số lượng các sự khác biệt giữa các khoảng RR liên tiếp có độ lớn lớn hơn 50 ms.
* **pNN50**: Tỷ lệ phần trăm của NN50 so với tổng số khoảng RR.
* **NN20**: Số lượng các sự khác biệt giữa các khoảng RR liên tiếp có độ lớn lớn hơn 20 ms.
* **pNN20**: Tỷ lệ phần trăm của NN20 so với tổng số khoảng RR.

**Kết quả:**

**A white background with black text

Description automatically generated**

**Code:**

clear

% Tạo dữ liệu mẫu cho 'intervals' (khoảng thời gian nhịp tim)

intervals = 0.8 + 0.05 \* randn(1, 100); % Giá trị trung bình 0.8s, có nhiễu nhỏ

%% interplotion

f\_interplotion = 4;

t = zeros(1,length(intervals));

for j = 1:1:length(intervals)

t(j) = sum(intervals(1:j));

end

t2 = t(1):1/f\_interplotion:t(length(t)); %time values for interp.

y=interp1(t,intervals,t2','spline')'; %cubic spline interpolation

%% power spectral density

y1=y-mean(y); %remove mean

NFFT = max(256,2^nextpow2(length(y1))); % the number of FFT

[pxx,f] = pwelch(y1,length(t2),length(t2)/2,(NFFT\*2)-1,f\_interplotion); % using welch method to calculate the power spectrum

%% HRV

f\_VLF = [0 0.04]; % frequency range of VLF

f\_LF = [0.04 0.15]; % frequency range of LF

f\_HF = [0.15 0.4]; % frequency range of HF

VLF= trapz(f(f>=f\_VLF(1)&f<=f\_VLF(2)),pxx(f>=f\_VLF(1)&f<=f\_VLF(2))) \* 1e6; % the unit of VLF isms^2

LF = trapz(f(f>=f\_LF(1)&f<=f\_LF(2)),pxx(f>=f\_LF(1)&f<=f\_LF(2))) \*1e6; % the unit of LF is ms^2

HF = trapz(f(f>=f\_HF(1)&f<=f\_HF(2)),pxx(f>=f\_HF(1)&f<=f\_HF(2))) \*1e6; % the unit of HF is ms^2

ratio\_LF\_HF = LF/HF;

%% plot

figure;

plot(f(f>=f\_VLF(1)&f<=f\_VLF(2)),pxx(f>=f\_VLF(1)&f<=f\_VLF(2)),'color','k');

hold on;

plot(f(f>=f\_LF(1)&f<=f\_LF(2)),pxx(f>=f\_LF(1)&f<=f\_LF(2)),'color','r');

plot(f(f>=f\_HF(1)&f<=f\_HF(2)),pxx(f>=f\_HF(1)&f<=f\_HF(2)),'color','b');

set(gca,'XTick',[0, 0.04, 0.15,0.4]);

set(gca,'XTicklabel',{'0','0.04','0.15','0.4'});

**Mô tả:**

* **Nội suy tín hiệu**: Dữ liệu intervals được nội suy để tạo ra tín hiệu có tần số mẫu cao hơn.
* **Mật độ phổ công suất (PSD)**: Phương pháp Welch được sử dụng để tính phổ công suất của tín hiệu đã nội suy.
* **Chỉ số HRV**: Tính toán các thành phần tần số VLF, LF, HF từ phổ công suất và tỷ lệ LF/HF để đánh giá biến động nhịp tim.
* **Vẽ đồ thị**: Đồ thị PSD được vẽ với các vùng tần số VLF, LF, HF được đánh dấu bằng các màu khác nhau.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**Code:**

clear

clear

% Tạo dữ liệu mẫu cho 'intervals' (khoảng thời gian nhịp tim)

intervals = 0.8 + 0.05 \* randn(1, 100); % Giá trị trung bình 0.8s, có nhiễu nhỏ

% Tính toán SD1 và SD2

diffNN = diff(intervals);

SD1 = sqrt(0.5 \* std(diffNN)^2) \* 1000; % đơn vị: ms

SD2 = sqrt(2 \* (std(intervals)^2) - (0.5 \* std(diffNN)^2)) \* 1000; % đơn vị: ms

ratio\_SD2\_SD1 = SD2 / SD1;

%% Vẽ biểu đồ Poincaré

poincare\_x = intervals(1:end-1) \* 1000; % chuyển sang ms

poincare\_y = intervals(2:end) \* 1000; % chuyển sang ms

figure;

plot(poincare\_x, poincare\_y, '.');

hold on;

% Vẽ elip

phi = pi / 4; % hệ tọa độ mới tại góc 45 độ với trục bình thường

new\_x = poincare\_x ./ cos(phi); % chuyển x sang hệ tọa độ mới

center\_new\_x = mean(new\_x); % lấy trung tâm theo trục x mới

[cnx, cny] = deal(center\_new\_x \* cos(phi), center\_new\_x \* sin(phi)); % chuyển trung tâm sang x mới, y mới

ellipse\_width = SD2;

ellipse\_height = SD1;

theta = 0:0.01:2\*pi;

x1 = ellipse\_width \* cos(theta);

y1 = ellipse\_height \* sin(theta);

X = cos(phi) \* x1 - sin(phi) \* y1;

Y = sin(phi) \* x1 + cos(phi) \* y1;

X = X + cnx;

Y = Y + cny;

plot(X, Y, 'k-');

% Vẽ SD1 và SD2 trong elip

line\_SD1 = line([cnx cnx], [cny - ellipse\_height, cny + ellipse\_height], 'color', 'g');

rotate(line\_SD1, [0, 0, 1], 45, [cnx, cny, 0]);

line\_SD2 = line([cnx - ellipse\_width, cnx + ellipse\_width], [cny cny], 'color', 'm');

rotate(line\_SD2, [0, 0, 1], 45, [cnx, cny, 0]);

**Mô tả:**

* **SD1 và SD2** là hai chỉ số trong phân tích Poincaré dùng để đo lường biến động nhịp tim ngắn hạn và dài hạn.
* **Tỷ lệ SD2/SD1** cho biết sự phân bố năng lượng của các biến động ngắn hạn và dài hạn trong dữ liệu.
* **Biểu đồ Poincaré** được vẽ với các điểm nhịp tim và một elip mô tả các chỉ số SD1, SD2, cũng như các đường SD1 và SD2 trong elip.

**Kết quả:**

**A screenshot of a computer screen

Description automatically generated**