**Code:**

% Implementing the moving average using a simple for loop

windowSize = 4; % Kích thước cửa sổ trung bình động

Raw\_Sig = randn(1, 100); % Thay "your PPG signal" bằng tín hiệu PPG (một tín hiệu ngẫu nhiên được tạo)

figure, plot(Raw\_Sig); % Vẽ tín hiệu ban đầu

xlabel('Samples');

ylabel('Amplitude');

title('Raw Signal');

% Lọc trung bình động

Filtered\_Sig = zeros(1, length(Raw\_Sig) - windowSize + 1); % Khởi tạo tín hiệu lọc

for i = 1:(length(Raw\_Sig) - windowSize + 1)

Filtered\_Sig(i) = mean(Raw\_Sig(i:i + windowSize - 1)); % Lấy trung bình trong cửa sổ

end

% Vẽ tín hiệu đã lọc

figure, plot(Filtered\_Sig);

xlabel('Samples');

ylabel('Amplitude');

title('Moving Average');

**Mô tả:**

* **windowSize = 4;**
* Đặt kích thước cửa sổ cho bộ lọc trung bình động là 4 mẫu.
* **Raw\_Sig = randn(1, 100);**
* Tạo một tín hiệu ngẫu nhiên có 100 mẫu làm tín hiệu thô ban đầu.
* **figure, plot(Raw\_Sig);**
* Vẽ tín hiệu thô để so sánh với tín hiệu sau khi làm mượt.
* **Filtered\_Sig = zeros(1, length(Raw\_Sig) - windowSize + 1);**
* Khởi tạo mảng Filtered\_Sig với kích thước phù hợp để chứa tín hiệu đã lọc.
* Độ dài của tín hiệu đã lọc giảm đi do cửa sổ trung bình động.
* **for i = 1:(length(Raw\_Sig) - windowSize + 1)**
* Vòng lặp duyệt qua các mẫu của tín hiệu, mỗi lần lấy một cửa sổ có kích thước windowSize.
* **Filtered\_Sig(i) = mean(Raw\_Sig(i:i + windowSize - 1));**
* Lấy trung bình các giá trị trong cửa sổ hiện tại.
* **figure, plot(Filtered\_Sig);**
* Vẽ tín hiệu đã lọc để so sánh với tín hiệu ban đầu.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**Code:**

% Implementing the frequency response of Butterworth filter

% Sampling frequency in Hz

Fs=200;

Fc=6/(Fs/2);

m=6;

% Butterworth filter

[num dend]= butter(m,Fc);

% Logarithimc scale

L=logspace(0,2);

% call the Freqency response function

Z=freqz(num,dend,L,Fs);

% Compute and display the magnitude response

figure; semilogx(L,abs(Z),'K');

grid;

xlabel('Hz');

ylabel('Gain');

title('Butterworth')

**Mô tả:**

**Fs = 200;**

* Đặt tần số lấy mẫu là 200 Hz.

**Fc = 6 / (Fs / 2);**

* Tần số cắt được chuẩn hóa theo Nyquist (một nửa tần số lấy mẫu).

**m = 6;**

* Bậc của bộ lọc Butterworth (bộ lọc càng cao bậc, đáp ứng càng dốc tại tần số cắt).

**[num, dend] = butter(m, Fc);**

* Hàm butter thiết kế bộ lọc Butterworth bậc m với tần số cắt chuẩn hóa Fc.
* Trả về các hệ số của tử số (num) và mẫu số (dend) trong hàm truyền.

**L = logspace(0, log10(Fs/2), 500);**

* Tạo dải tần số từ 1 Hz đến Fs/2Fs/2Fs/2 (Nyquist), với 500 điểm trên thang log.

**[H, F] = freqz(num, dend, L, Fs);**

* Hàm freqz tính toán đáp ứng tần số của bộ lọc:
  + H: Đáp ứng tần số (hệ số phức).
  + F: Dải tần số tương ứng.

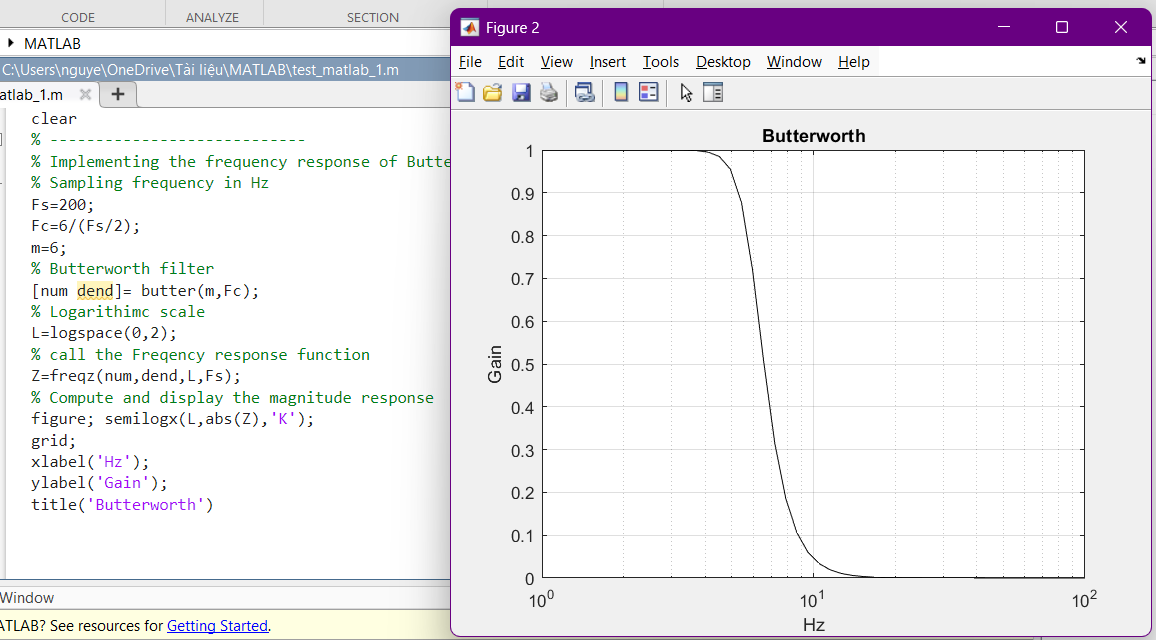
**semilogx(F, abs(H), 'k');**

* Vẽ biên độ đáp ứng tần số trên thang log:
  + Trục x: Tần số (Hz, trên thang log).
  + Trục y: Biên độ (Gain).

**grid on; xlabel('Frequency (Hz)'); ylabel('Magnitude (Gain)'); title('Butterworth Filter Frequency Response');**

* Thêm lưới, nhãn trục và tiêu đề cho biểu đồ để dễ hiểu hơn.

**Kết quả:**

****

**Code:**

% Implementing the frequency response of Chebyshev I filter

% Sampling frequency in Hz

Fs = 200; % Tần số lấy mẫu

Fc = 6 / (Fs / 2); % Tần số cắt, chuẩn hóa (0 <= Fc <= 1)

m = 6; % Bậc của bộ lọc

Rs = 1; % Gợn sóng tối đa trong dải thông (dB)

% Chebyshev Type I filter

[num, dend] = cheby1(m, Rs, Fc); % Thiết kế bộ lọc Chebyshev loại 1

% Logarithmic scale for frequency

L = logspace(0, log10(Fs/2), 500); % Tạo dải tần số trên thang log

% Frequency response

[H, F] = freqz(num, dend, L, Fs); % Tính đáp ứng tần số

% Compute and display the magnitude response

figure;

semilogx(F, abs(H), 'k'); % Vẽ biểu đồ đáp ứng trên thang log

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude (Gain)');

title('Chebyshev Type I Filter Frequency Response');

**Mô tả:**

**Fs = 200;**

* Tần số lấy mẫu của tín hiệu (Hz).

**Fc = 6 / (Fs / 2);**

* Tần số cắt được chuẩn hóa dựa trên Nyquist.

**m = 6;**

* Bậc của bộ lọc Chebyshev loại 1.

**Rs = 1;**

* Gợn sóng tối đa trong dải thông (ripple), tính bằng decibel (dB). Thay đổi giá trị này để kiểm soát độ gợn.

**[num, dend] = cheby1(m, Rs, Fc);**

* Hàm cheby1 thiết kế bộ lọc Chebyshev loại 1:
  + num: Hệ số của tử số.
  + dend: Hệ số của mẫu số.

**L = logspace(0, log10(Fs/2), 500);**

* Tạo dải tần số từ 1 Hz đến Fs/2Fs/2Fs/2 (Nyquist) với 500 điểm trên thang log.

**[H, F] = freqz(num, dend, L, Fs);**

* Hàm freqz tính đáp ứng tần số:
  + H: Đáp ứng phức.
  + F: Dải tần số tương ứng.

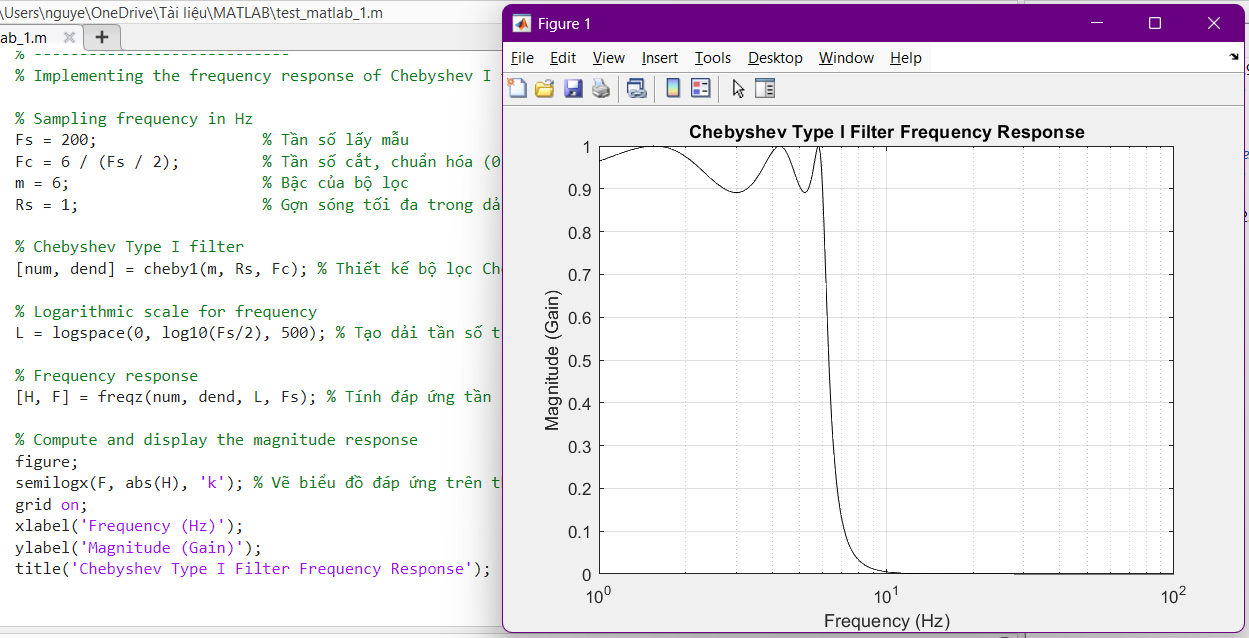
**semilogx(F, abs(H), 'k');**

* Vẽ biểu đồ biên độ đáp ứng trên thang log:
  + Trục x: Tần số (Hz).
  + Trục y: Biên độ (Gain).

**grid on; xlabel('Frequency (Hz)'); ylabel('Magnitude (Gain)'); title('Chebyshev Type I Filter Frequency Response');**

* Thêm lưới, nhãn trục và tiêu đề cho biểu đồ để dễ hiểu hơn.

**Kết quả:**



**Code:**

% Implementing the frequency response of Chebyshev II filter

% Sampling frequency in Hz

Fs = 200; % Tần số lấy mẫu

Fc = 6 / (Fs / 2); % Tần số cắt, chuẩn hóa (0 <= Fc <= 1)

m = 6; % Bậc của bộ lọc

Rs = 18; % Độ suy giảm tối thiểu trong dải chắn (dB)

% Chebyshev Type II filter

[num, dend] = cheby2(m, Rs, Fc); % Thiết kế bộ lọc Chebyshev loại 2

% Logarithmic scale for frequency

L = logspace(0, log10(Fs/2), 500); % Tạo dải tần số trên thang log

% Frequency response

[H, F] = freqz(num, dend, L, Fs); % Tính đáp ứng tần số

% Compute and display the magnitude response

figure;

semilogx(F, abs(H), 'k'); % Vẽ biểu đồ đáp ứng trên thang log

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude (Gain)');

title('Chebyshev Type II Filter Frequency Response');

**Mô tả:**

* **Fs = 200;**
* Đặt tần số lấy mẫu của tín hiệu.
* **Fc = 6 / (Fs / 2);**
* Tần số cắt được chuẩn hóa dựa trên Nyquist (FcFcFc phải nằm trong khoảng 0 đến 1).
* **m = 6;**
* Bậc của bộ lọc Chebyshev loại 2.
* **Rs = 18;**
* Độ suy giảm tối thiểu trong dải chắn (stopband), tính bằng decibel (dB). Tăng giá trị này để làm cho dải chắn suy giảm mạnh hơn.
* **[num, dend] = cheby2(m, Rs, Fc);**
* Hàm cheby2 thiết kế bộ lọc Chebyshev loại 2:
  + num: Hệ số của tử số.
  + dend: Hệ số của mẫu số.
* **L = logspace(0, log10(Fs/2), 500);**
* Tạo dải tần số từ 1 Hz đến Fs/2Fs/2Fs/2 (Nyquist) với 500 điểm trên thang log.
* **[H, F] = freqz(num, dend, L, Fs);**
* Hàm freqz tính toán đáp ứng tần số:
  + H: Đáp ứng phức của bộ lọc.
  + F: Dải tần số tương ứng.
* **semilogx(F, abs(H), 'k');**
* Vẽ biểu đồ đáp ứng tần số trên thang log:
  + Trục x: Tần số (Hz, trên thang log).
  + Trục y: Biên độ (Gain).
* **xlabel('Frequency (Hz)'); ylabel('Magnitude (Gain)'); title('Chebyshev Type II Filter Frequency Response');**
* Thêm nhãn trục và tiêu đề cho biểu đồ.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**Code:**

% Implementing the frequency response of Elliptic filter

% Sampling frequency in Hz

Fs = 200; % Tần số lấy mẫu

Fc = 6 / (Fs / 2); % Tần số cắt, chuẩn hóa (0 <= Fc <= 1)

m = 6; % Bậc của bộ lọc

Rp = 0.5; % Gợn sóng tối đa trong dải thông (dB)

Rc = 20; % Độ suy giảm tối thiểu trong dải chắn (dB)

% Elliptic filter

[num, dend] = ellip(m, Rp, Rc, Fc); % Thiết kế bộ lọc Elliptic

% Logarithmic scale for frequency

L = logspace(0, log10(Fs/2), 500); % Tạo dải tần số trên thang log

% Frequency response

[H, F] = freqz(num, dend, L, Fs); % Tính đáp ứng tần số

% Compute and display the magnitude response

figure;

semilogx(F, abs(H), 'k'); % Vẽ biểu đồ đáp ứng trên thang log

grid on;

xlabel('Frequency (Hz)');

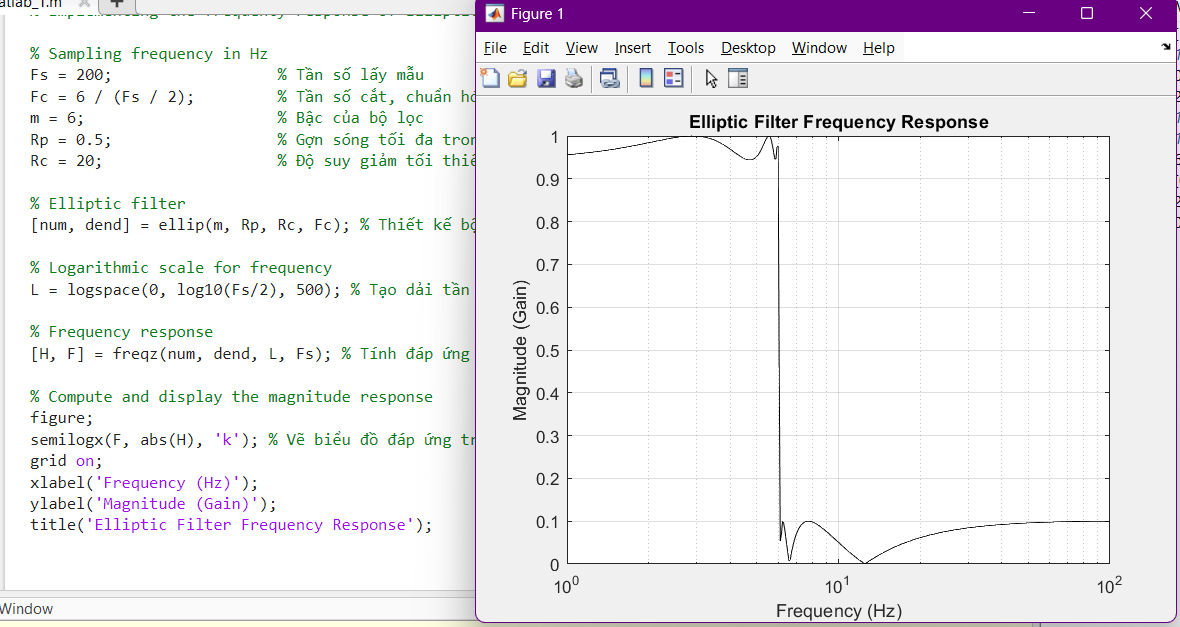
ylabel('Magnitude (Gain)');

title('Elliptic Filter Frequency Response');

**Mô tả:**

* **Fs = 200;**
* Đặt tần số lấy mẫu của tín hiệu.
* **Fc = 6 / (Fs / 2);**
* Tần số cắt chuẩn hóa đối với Nyquist.
* **m = 6;**
* Bậc của bộ lọc Elliptic (số lượng các thành phần bậc cao).
* **Rp = 0.5;**
* Gợn sóng tối đa trong dải thông (passband) được cho bằng 0.5 dB.
* **Rc = 20;**
* Độ suy giảm tối thiểu trong dải chắn (stopband) tính bằng decibel (20 dB).
* **[num, dend] = ellip(m, Rp, Rc, Fc);**
* Hàm ellip thiết kế bộ lọc Elliptic với các tham số:
  + num: Hệ số của tử số.
  + dend: Hệ số của mẫu số.
* **L = logspace(0, log10(Fs/2), 500);**
* Tạo dải tần số từ 1 Hz đến Fs/2Fs/2Fs/2 (Nyquist) với 500 điểm trên thang log.
* **[H, F] = freqz(num, dend, L, Fs);**
* Hàm freqz tính toán đáp ứng tần số của bộ lọc Elliptic:
  + H: Đáp ứng phức.
  + F: Dải tần số tương ứng.
* **semilogx(F, abs(H), 'k');**
* Vẽ biểu đồ biên độ đáp ứng tần số trên thang log:
  + Trục x: Tần số (Hz).
  + Trục y: Biên độ (Gain).
* **xlabel('Frequency (Hz)'); ylabel('Magnitude (Gain)'); title('Elliptic Filter Frequency Response');**
* Thêm nhãn trục và tiêu đề cho biểu đồ.

**Kết quả:**

****

**Code:**

%% Parameters:

% filter\_type ———— lọc theo loại bộ lọc (1: Butterworth, 2: Chebyshev 1, 3: Chebyshev 2, 4: Elliptic, 5: FIR, 6: FIR thiết kế theo phương pháp least squares, 7: Smooth, 8: Median Filter, 9: Wavelet)

% order —————— độ bậc của bộ lọc (cấp độ cho các bộ lọc như Wavelet hoặc bộ lọc FIR)

% raw\_data ———— tín hiệu PPG chưa qua xử lý

% Fs ———————— tần số lấy mẫu

% fc ———————— tần số cắt

function [filtered\_data] = PPG\_Lowpass(raw\_data, filter\_type, order, Fs, fc)

Fn = Fs / 2; % Tần số Nyquist (nửa tần số lấy mẫu)

% Kiểm tra kiểu bộ lọc và xử lý tín hiệu

switch filter\_type

case 1

% Bộ lọc Butterworth

[A, B, C, D] = butter(order, fc / Fn, 'low'); % Thiết kế bộ lọc Butterworth

[filter\_SOS, g] = ss2sos(A, B, C, D); % Chuyển đổi hệ thống trạng thái sang dạng SOS (Second-Order Sections)

filtered\_data = filtfilt(filter\_SOS, g, raw\_data); % Lọc tín hiệu

case 2

% Bộ lọc Chebyshev loại 1

[A, B, C, D] = cheby1(order, 0.1, fc / Fn, 'low'); % Thiết kế bộ lọc Chebyshev loại 1

[filter\_SOS, g] = ss2sos(A, B, C, D);

filtered\_data = filtfilt(filter\_SOS, g, raw\_data);

case 3

% Bộ lọc Chebyshev loại 2

[A, B, C, D] = cheby2(order, 20, fc / Fn, 'low'); % Thiết kế bộ lọc Chebyshev loại 2

[filter\_SOS, g] = ss2sos(A, B, C, D);

filtered\_data = filtfilt(filter\_SOS, g, raw\_data);

case 4

% Bộ lọc Elliptic

[A, B, C, D] = ellip(order, 0.1, 30, fc / Fn, 'low'); % Thiết kế bộ lọc Elliptic

[filter\_SOS, g] = ss2sos(A, B, C, D);

filtered\_data = filtfilt(filter\_SOS, g, raw\_data);

case 5

% Bộ lọc FIR

d = fir1(order, fc / Fn, 'low'); % Thiết kế bộ lọc FIR

filtered\_data = filtfilt(d, 1, raw\_data); % Lọc tín hiệu

case 6

% Bộ lọc FIR theo phương pháp least squares

d = designfilt('lowpassfir', 'FilterOrder', order, 'PassbandFrequency', fc, 'StopbandFrequency', fc + 0.2, 'DesignMethod', 'ls', 'SampleRate', Fs);

filtered\_data = filtfilt(d, raw\_data); % Lọc tín hiệu

case 7

% Phương pháp smoothing (làm mịn tín hiệu)

filtered\_data = smooth(raw\_data, order); % Làm mịn tín hiệu

case 8

% Phương pháp median filtering

filtered\_data = medfilt1(raw\_data, order); % Lọc trung bình theo phương pháp median

case 9

% Phương pháp Wavelet

filtered\_data = wden(raw\_data, 'modwtsqtwolog', 's', 'mln', order, 'db2'); % Lọc bằng sóng con (Wavelet)

end

end

**Mô tả:**

1. **raw\_data**: Dữ liệu tín hiệu PPG chưa qua xử lý, sẽ được lọc bằng các bộ lọc khác nhau.
2. **filter\_type**: Xác định loại bộ lọc sử dụng. Có 9 lựa chọn:
   * 1: Bộ lọc Butterworth.
   * 2: Bộ lọc Chebyshev loại 1.
   * 3: Bộ lọc Chebyshev loại 2.
   * 4: Bộ lọc Elliptic.
   * 5: Bộ lọc FIR (Finite Impulse Response).
   * 6: FIR theo phương pháp least squares.
   * 7: Làm mịn (smoothing).
   * 8: Lọc trung bình (Median filtering).
   * 9: Lọc bằng sóng con (Wavelet filtering).
3. **order**: Độ bậc của bộ lọc (chỉ định mức độ phức tạp của bộ lọc).
4. **Fs**: Tần số lấy mẫu của tín hiệu.
5. **fc**: Tần số cắt của bộ lọc.

**Các phương pháp lọc**

* **Bộ lọc Butterworth**: Bộ lọc có đáp ứng tần số mượt mà, không có gợn sóng trong dải thông.
* **Bộ lọc Chebyshev**: Có gợn sóng trong dải thông nhưng suy giảm nhanh trong dải chắn.
* **Bộ lọc Elliptic**: Đưa ra sự suy giảm nhanh chóng trong cả dải thông và dải chắn, với một số gợn sóng trong cả hai dải.
* **FIR Filter**: Bộ lọc đáp ứng xung hữu hạn, dễ dàng thiết kế và sử dụng.
* **Smoothing**: Làm mịn tín hiệu để giảm các biến động ngắn hạn.
* **Median Filtering**: Lọc trung bình theo phương pháp median, giúp loại bỏ nhiễu.
* **Wavelet Filtering**: Sử dụng sóng con để phân tích tín hiệu và loại bỏ các tần số không mong muốn.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**Code:**

% Tạo tín hiệu giả cho 'template' và 'signal'

Fs = 200; % Tần số lấy mẫu (Hz)

t = 0:1/Fs:2; % Thời gian mẫu từ 0 đến 2 giây

% Tạo template PPG mẫu (tín hiệu mẫu)

f = sin(2\*pi\*1\*t) + 0.5\*sin(2\*pi\*0.5\*t); % Sóng sin với tần số 1Hz và 0.5Hz

% Tạo tín hiệu PPG cần xử lý (signal) (có chút thay đổi so với template)

g = sin(2\*pi\*1.05\*t) + 0.5\*sin(2\*pi\*0.55\*t) + 0.1\*randn(size(t)); % Tín hiệu gần giống, có nhiễu

% Resample tín hiệu g sao cho độ dài của nó bằng với độ dài của f

g = resample(g, length(f), length(g));

% Tính toán cross-correlation giữa f và g

xcorr\_f\_g = xcorr(f, g);

xcorr\_g\_f = xcorr(g, f);

% Vẽ đồ thị

figure;

% Đồ thị 1: Template (f)

subplot(4,1,1);

plot(f, 'k--'); % Dòng màu đen đứt quãng cho template

title('f: Template');

% Đồ thị 2: Raw signal (g)

subplot(4,1,2);

plot(g, 'r--'); % Dòng màu đỏ đứt quãng cho tín hiệu raw

title('g: Raw signal');

% Đồ thị 3: Cross-correlation (f \* g)

subplot(4,1,3);

plot(xcorr\_f\_g); % Vẽ đồ thị cross-correlation giữa f và g

title('f \* g');

% Đồ thị 4: Cross-correlation (g \* f)

subplot(4,1,4);

plot(xcorr\_g\_f); % Vẽ đồ thị cross-correlation giữa g và f

title('g \* f');

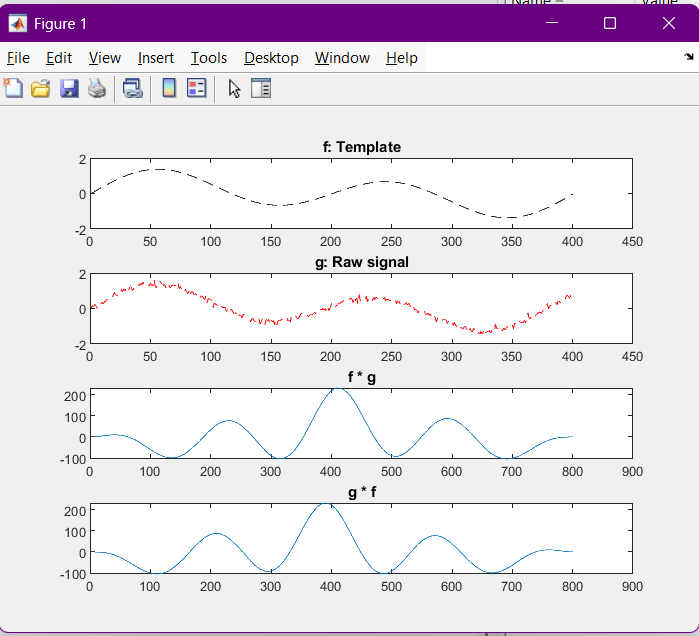
% Tiêu đề chung cho toàn bộ đồ thị

suptitle('Cross correlation');

**Mô tả:**

* **Tạo tín hiệu giả**:
* f (template): Là một tín hiệu PPG mẫu được tạo ra từ tổng của hai sóng sin với tần số 1Hz và 0.5Hz.
* g (signal): Là một tín hiệu gần giống f, nhưng có sự thay đổi tần số và thêm nhiễu ngẫu nhiên để mô phỏng tín hiệu thực tế (bằng cách thêm 0.1\*randn(size(t))).
* **Resample**:
* Tín hiệu g được thay đổi độ dài sao cho phù hợp với f bằng hàm resample(). Điều này cần thiết để tính toán cross-correlation.
* **Tính toán cross-correlation**:
* Sử dụng hàm xcorr() để tính toán độ tương quan giữa f và g. Bạn sẽ có hai độ tương quan chéo: f \* g và g \* f, với các lệnh xcorr\_f\_g và xcorr\_g\_f.
* **Vẽ đồ thị**:
* Đoạn mã sử dụng subplot() để vẽ 4 đồ thị trong một cửa sổ figure.
* Đồ thị đầu tiên là tín hiệu mẫu (f), đồ thị thứ hai là tín hiệu g, đồ thị thứ ba là cross-correlation f \* g, và đồ thị thứ tư là cross-correlation g \* f.

**Kết quả:**

****