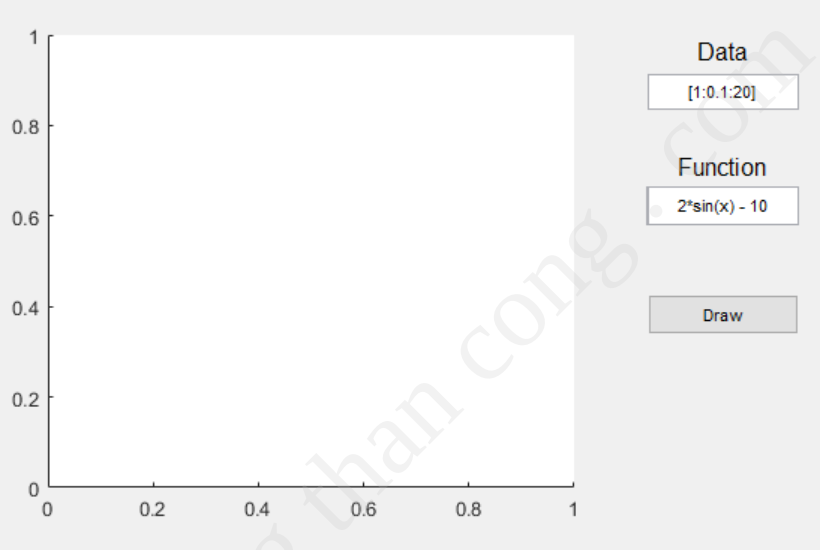
Bài 5: MATLAB GUIDE

Mục đích:

- Thiết kế giao diện đồ họa người dùng trên Matlab sử dụng GUIDE.
- Tìm hiểu phương pháp thiết kế giao diện và xử lý dữ liệu cung cấp từ người dùng.
- Sử dụng khối lệnh try-catch để kiểm soát và thông báo lỗi tới người dùng.

Bài 1: Sử dụng GUIDE để xây dựng giao diện của chương trình vẽ đồ thị của hàm một biến như trong Hình 1. ***Chương trình không được xuất bất cứ dòng lỗi nào ra Command Window của Matlab trong toàn bộ bài làm.***



Hình 1: Chương trình vẽ đồ thị của hàm một biến.

```
function plotButton_Callback(hObject, eventdata, handles)
try
  x = get(handles.dataEdit,'String');
  x = str2num(x);

  fx = get(handles.functionEdit,'String');
  fx = inline(fx);
  y = fx(x);
  plot(x,y);
catch e
  switch(e.identifier)
    case 'MATLAB:Inline:subsref:tooFewInputs'
      msgbox('Chi nhap ham mot bien','Error','error');
    otherwise
      msgbox({e.identifier e.message},'Error','error');
  end
end
```

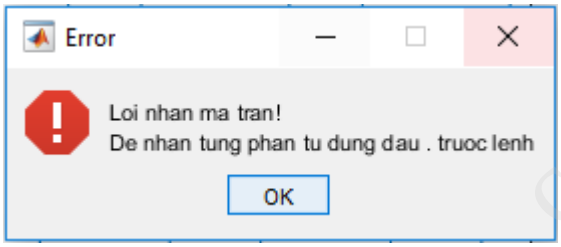
Trong đó:

- Đặt tên **Tag** của Edit Text cho **Data** và **Function** lần lượt là **dataEdit** và **functionEdit**.
- Dùng khối lệnh try-catch để viết hàm callback tên plotButton_Callback cho nút **Draw** như đoạn code ở trên.

a. Tiến hành chạy chương trình để vẽ đồ thị với **Data** thay đổi từ 1 tới 20 với bước nhảy là 0.1 và **Function** = 2*sin(x) - 10. Các đoạn code trong khối lệnh catch có được thực thi hay không? .....

b. Với dữ liệu tương tự câu a, lần lượt chạy lại chương trình với **Function** là 2*sin(x) - y và @2*sin(x) - 10. Các lệnh ở dòng thứ 8 và 9 trong plotButton_Callback có được thực thi hay không? .....

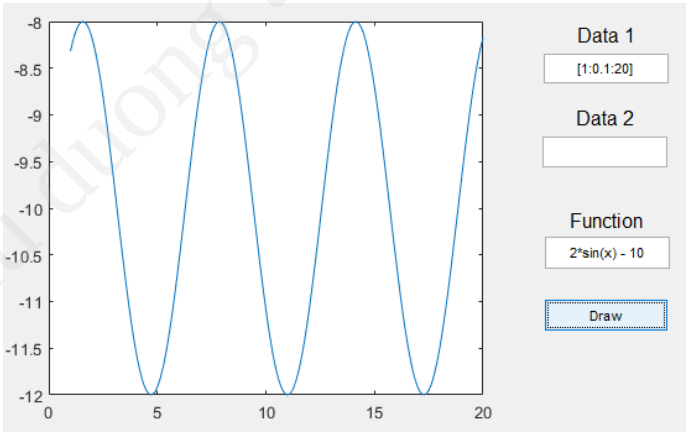
c. Chạy trương trình với **Function** = 2*sin(x) - 10 + x^3. Sau đó, thêm vào khối lệnh switch trường hợp e.identifier == 'MATLAB:inlineeval:InlineExprError' và nhắc người dùng chuyển từ nhân ma trận sang nhân từng phẩn tử trong thông báo lỗi. Kết quả cuối cùng phải có dạng như Hình 2.



Hình 2: Thông báo lỗi do nhân ma trận.

d. Sửa lại giao diện (Hình 3) và hàm plotButton_Callback để có thể vẽ hàm một biến hoặc 2 biến dựa vào dữ liệu nhập vào của người dùng. Tiến hành chạy chương trình và kết quả phải như các hình sau:

- Một biến:



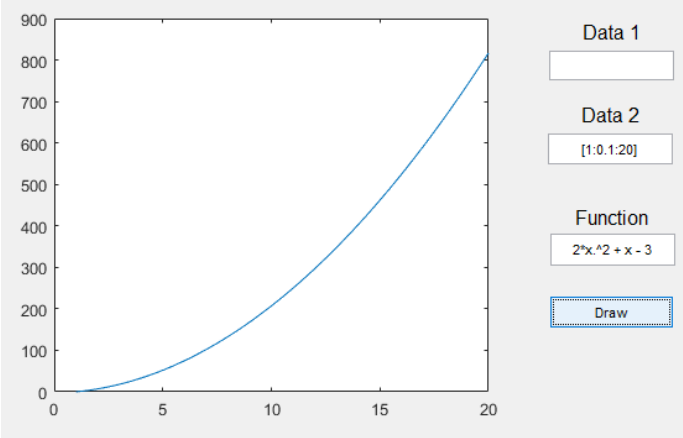
Hình 3: Hàm 2*sin(x) - 10 với dữ liệu ở **Data 1**.

- Hai biến:

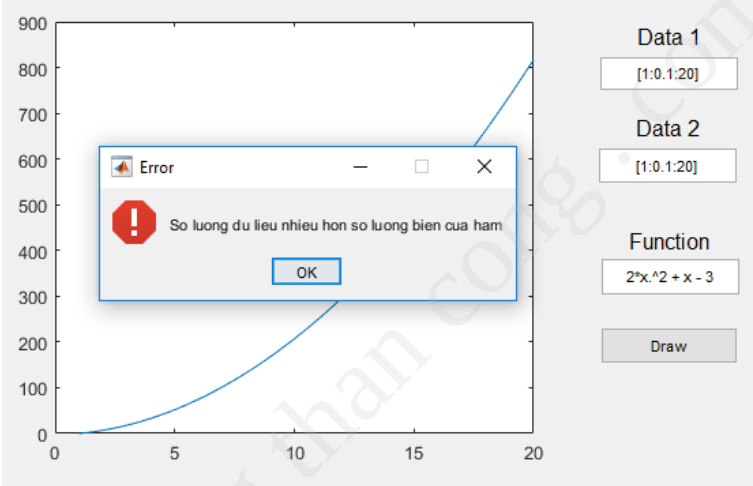
e. Sử dụng Toolbar Editor (Tools) để thêm nút save vào trong giao diện. Viết hàm callback cho nút này để cho phép người dùng chọn folder và lưu lại đồ thị đã vẽ.

Gợi ý:

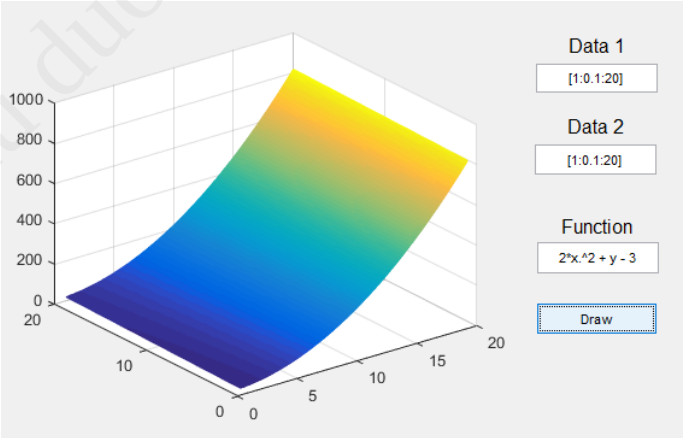
- Sử dụng lệnh fileName = uiputfile('../*.jpg') để cho phép người dùng chọn vị trí và tên file cần lưu.
- Lưu lại đồ thị dưới dạng .jpg bằng ba lệnh sau:
  frame = getframe(handles.AxesName); Image = frame2im(frame);
  imwrite(Image, fileName)



Hình 4: Hàm 2*x.^2 + x - 3 với dữ liệu ở **Data 2**.



Hình 5: Hàm 2*x.^2 + x - 3 với hai dữ liệu vào.

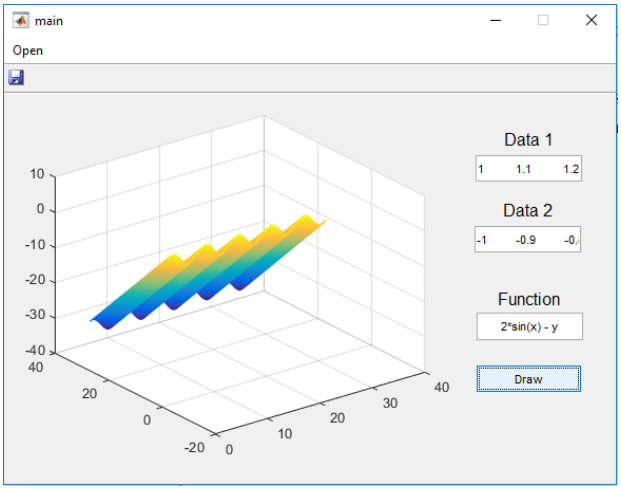


Hình 6: Đồ thị hàm 2*x.^2 + y - 3 sử dụng hàm vẽ mesh.

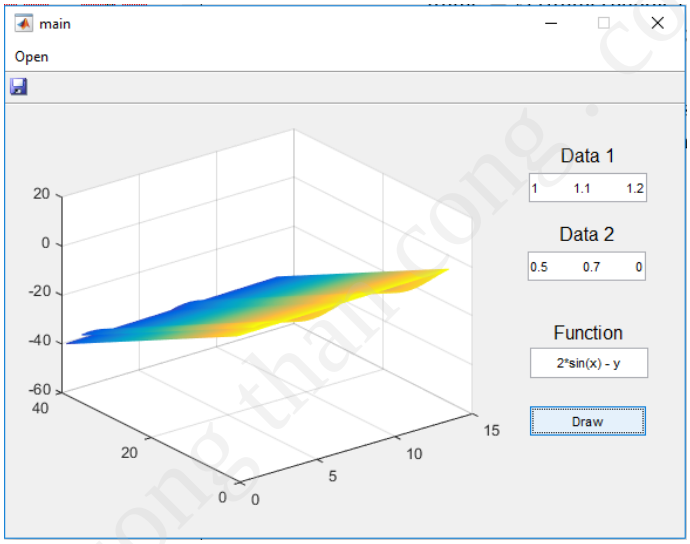
f. Dùng Menu Editor tạo một Menu Item tên **Open**. Sửa lại hàm callback của **Open** để cho phép người dùng chọn file .csv. Sau đó, chương trình sẽ tự động đọc nội dung của file và gán giá trị lên **Data 1** và **Data 2**. Lần lượt đọc các file Bai1f1.csv, Bai1f2.csv, và Bai1f3.csv. Kết quả như các hình bên dưới:

Gợi ý:

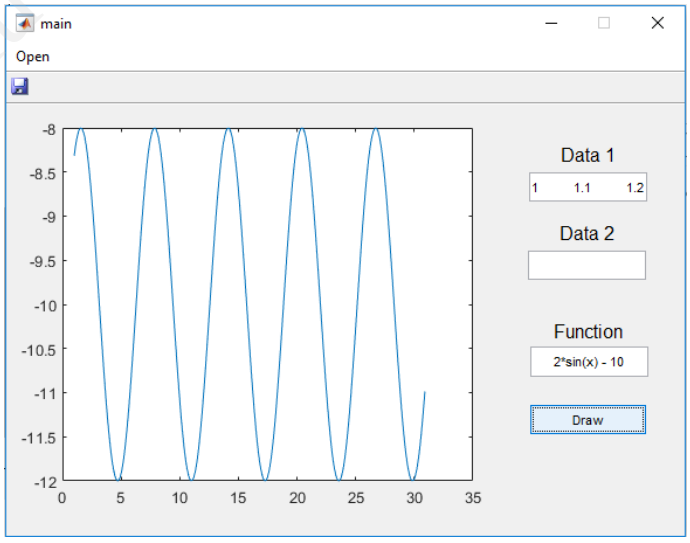
- Chọn file: fileName = uigetfile('../*.csv');



Hình 7: Đồ thị hàm 2*sin(x) - y với dữ liệu từ file Bai1f1.csv



Hình 8: Đồ thị hàm 2*sin(x) - y với dữ liệu từ file Bai1f2.csv

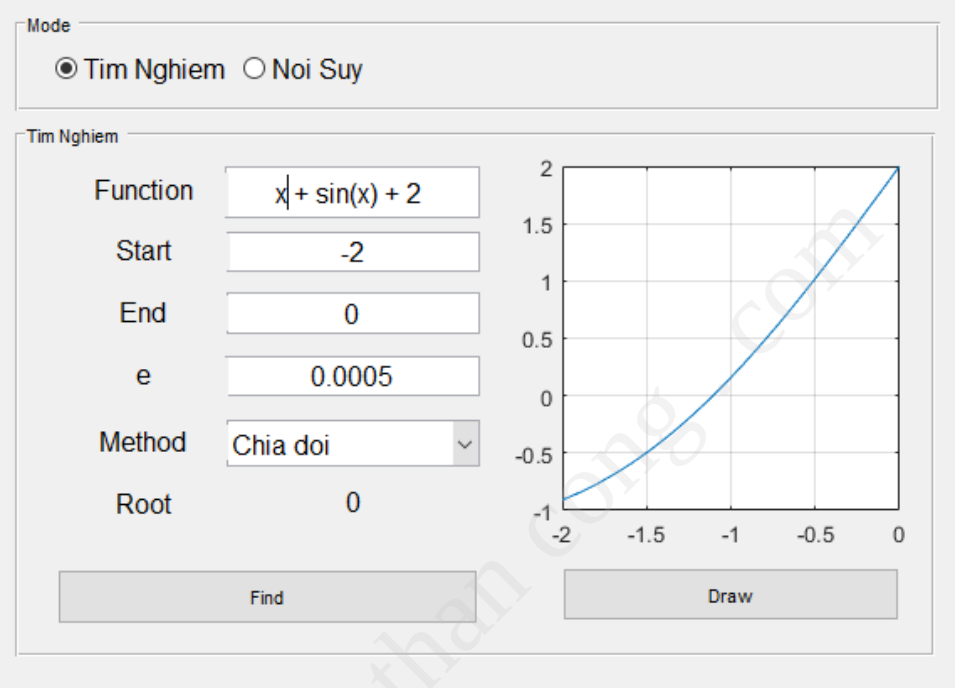


Hình 9: Đồ thị hàm 2*sin(x) - 10 với dữ liệu từ file Bai1f3.

- Đọc file csv: data = csvread(fileName);

Bài 2: Xây dựng một chương trình cho phép người dùng chọn 2 chế độ hoạt động như sau:

a. Tìm nghiệm gần đúng cho hàm một biến. Trong đó cho phép nhập vào hàm, giá trị đầu, giá trị cuối, sai số cho phép và cho phép lựa một trong 3 phương pháp: chia đôi, dây cung, tiếp tuyến. Đồng thời người dùng cũng có thể vẽ đồ thị của hàm theo giá trị đầu và cuối. Kết quả như Hình 10.

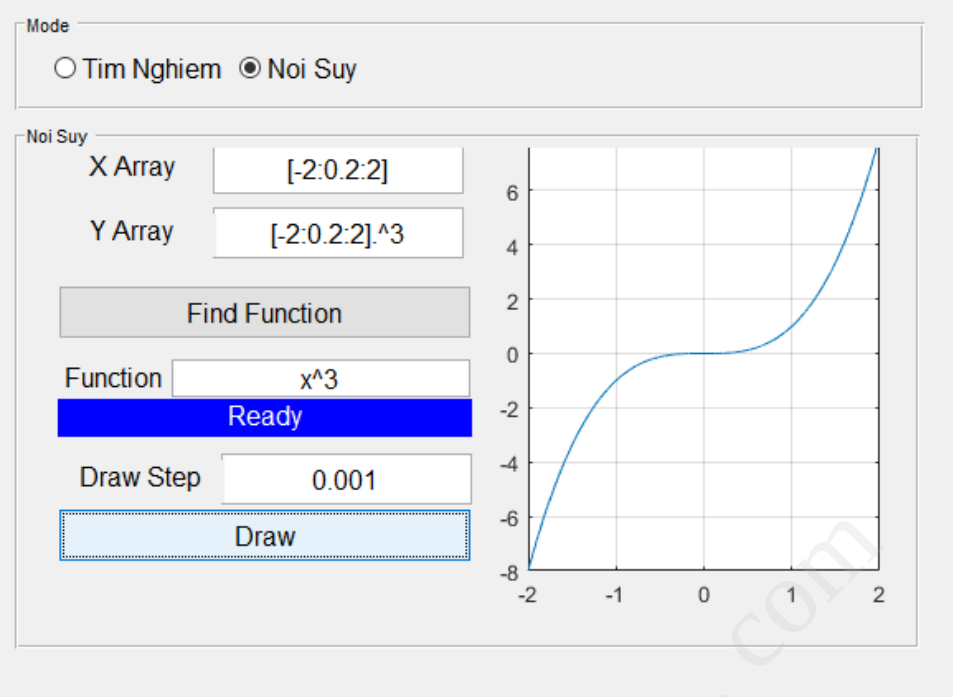


Hình 10: Chế độ tìm nghiệm.

b. Tìm hàm nội suy dựa theo Lagrange. Trong đó cho phép nhập vào mảng giá trị của X và Y, một thanh trạng thái để báo quá trình hoạt động của chương trình (Ready, Running, và Error). Đồng thời người dùng cũng có thể vẽ đồ thị của hàm theo giá trị đầu và cuối của X cùng với bước nhảy cho phép. Kết quả như Hình 11.

Gợi ý:

- Sử dụng Panel để nhóm chung các phần tử của mỗi chế độ. Không để các panel trùng lên nhau. Để noiSuyPanel có cùng vị trí với timNghiemPanel khi chương trình thực thi, đặt 2 câu lệnh sau vào hàm TenProject_OpeningFunc:
  `pos = getpixelposition(handles.timNghiemPanel);`
  `setpixelposition(handles.noiSuyPanel,pos);`
- Sử dụng Radio Button để chuyển đổi chế độ. **Lưu ý:** nhóm chung các Radio Button vào Button Group. Trong hàm Callback của Radio Button sử dụng lệnh sau để tắt/mở panel:
  `set(handles.timNghiemPanel,'Visible','off'); %or 'on'`
- Thanh trạng thái của chế độ nội suy sẽ hiển thị Ready và nền xanh dương khi chương trình đã hoàn thành việc tìm hàm. Hiển thị Running và nền xanh lá khi chương trình đang tìm hàm từ X và Y. Hiển thị Error và nền đỏ khi gặp lỗi. Lưu ý, mỗi khi đổi màu nền cần dùng lệnh `pause(.25);` để Matlab có thể cập nhật màu trước khi tiếp tục thực thi lệnh.



Hình 11: Chế độ nội suy.

Tiến hành kiểm tra chương trình với các hàm và chế độ làm việc khác nhau. ***Bảo đảm chương trình không được xuất bất cứ dòng lỗi hoặc cảnh báo nào ra Command Window của Matlab dưới mọi trường hợp.***

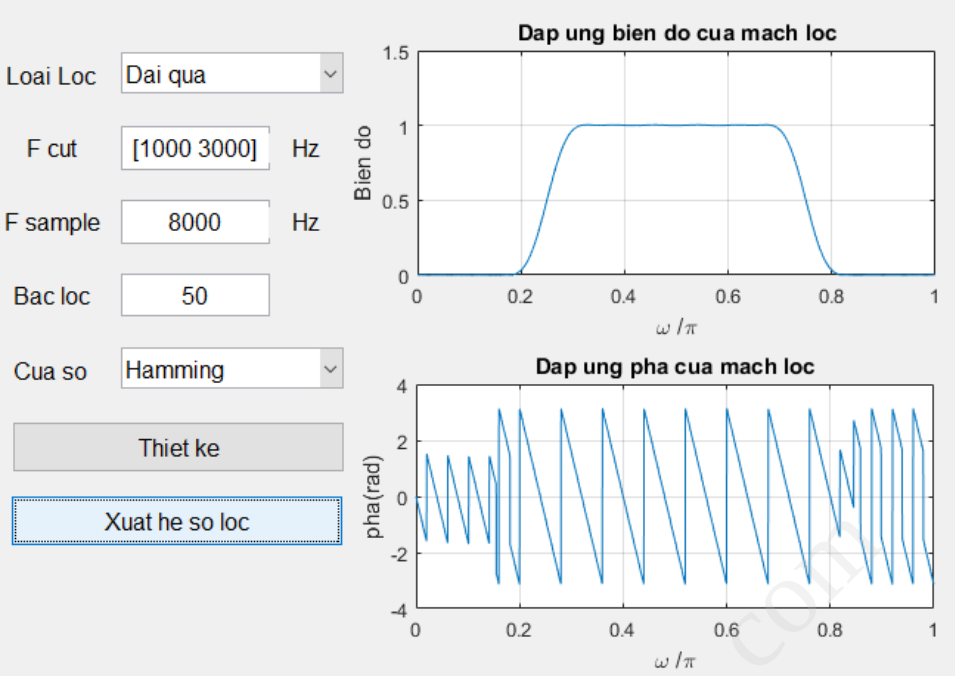
Bài 3: Xây dựng một chương trình có 3 nút ấn gồm **FIR Designer, IIR Designer** và **Sound Filter** (Hình 12). Mỗi lần ấn vào một nút thì một cửa sổ chương trình mới sẽ hiện ra. ***Bảo đảm chương trình không được xuất bất cứ dòng lỗi hoặc cảnh báo nào ra Command Window của Matlab dưới mọi trường hợp.***

Hình 12: Giao diện chương trình chính.

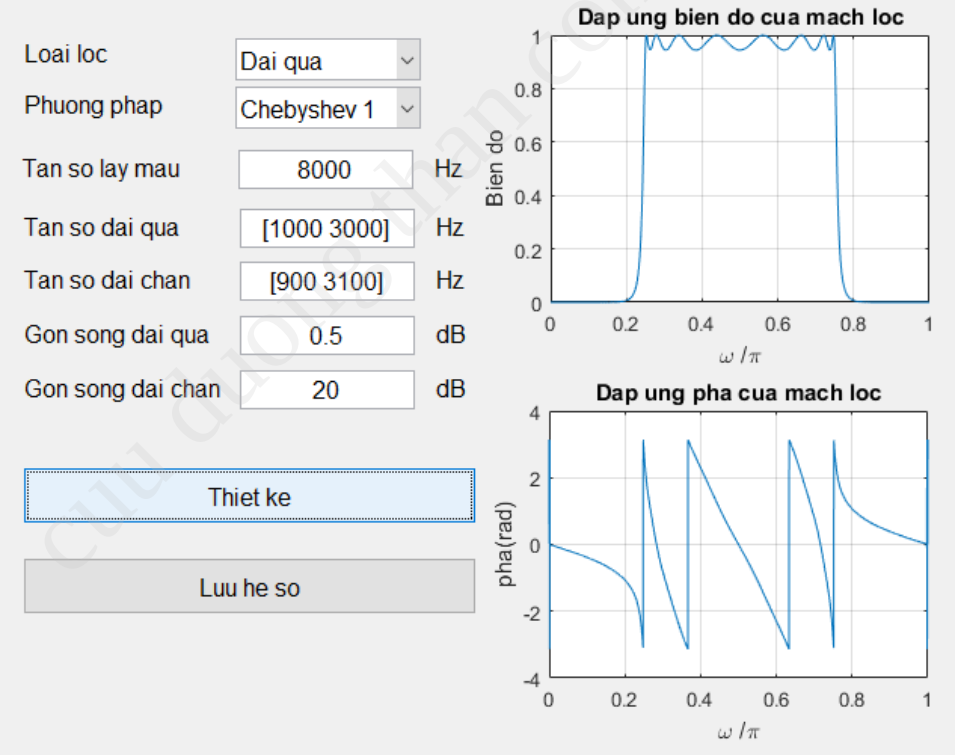
a. FIR Designer (Hình 13): Cho phép người dùng chọn loại lọc (thấp qua, cao qua, dải qua, và dải chặn), bậc lọc, loại cửa sổ (chữ nhật, tam giác, và Hamming), tần số lấy mẫu, và tần số cắt. Khi ấn nút **Thiet ke**, chương trình sẽ vẽ đáp ứng biên độ và đáp ứng pha của mạch lọc được thiết kế. **Xuat he so loc** cho phép người dùng lưu các hệ số lọc ra file .csv.

b. IIR Designer (Hình 14): Cho phép người dùng chọn loại lọc (thấp qua, cao qua, dải qua, và dải chặn), phương pháp (Butterworth, Chebyshev 1, Chebyshev 2, và Elliptic), tần số và gợn sóng dải qua/dải chặn, và tần số lấy mẫu. Khi ấn nút **Thiet ke**, chương trình sẽ vẽ đáp ứng biên độ và đáp ứng pha của mạch lọc được thiết kế. **Luu he so** cho phép người dùng lưu hệ số tử và mẫu ra file .csv.

c. Sound Filter (Hình 15):

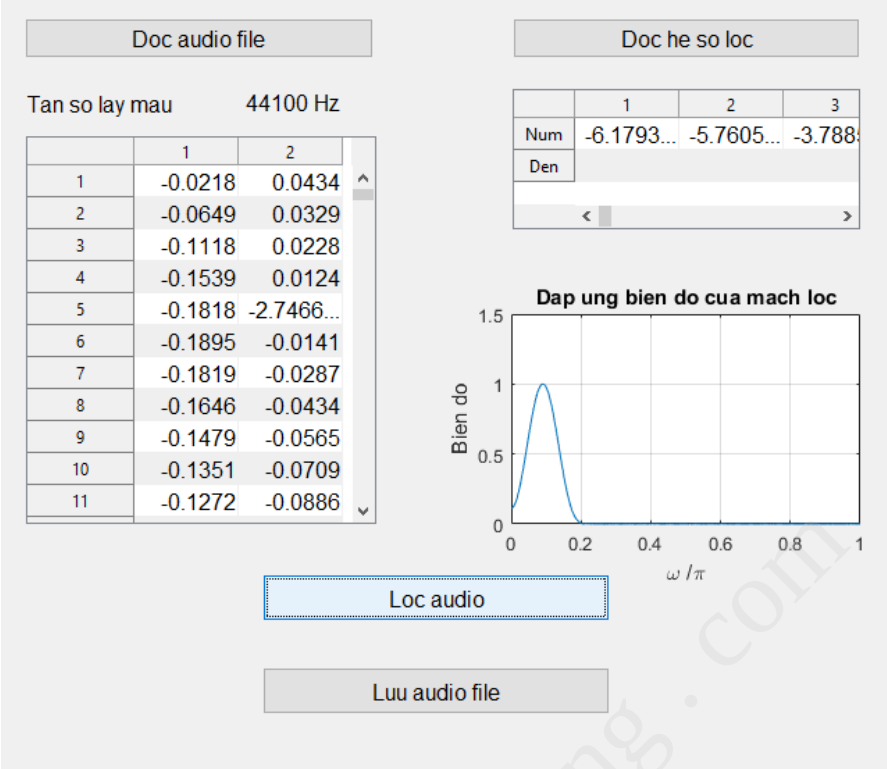


Hình 13: FIR Designer.



Hình 14: IIR Designer.

- **Doc audio file**: Cho phép đọc dữ liệu từ file audio (audioread(fileName, 'double') hoặc wavread ở version cũ) vào table (handles.soundDataTable.Data = data;) và hiển thị tần số lấy mẫu của tín hiệu.
- **Doc he so loc**: Cho phép đọc dữ liệu từ file .csv chứa các hệ số của mạch lọc được thiết kế từ FIR Designer hoặc IIR Designer và hiển thị trong table (Đặt tên cho hàng handles.hesoLocTable.RowName = {'Num' 'Den'};). Đồng thời vẽ lại đáp ứng biên độ của mạch lọc.



Hình 15: Sound Filter.

- **Loc audio**: Tiến hành lọc tín hiệu (cả 2 kênh trái phải) và vẽ đáp ứng tần số tín hiệu trước và sau khi lọc bằng hàm plotFFT ở dưới.
- **Luu audio file**: Cho phép lưu lại tín hiệu sau khi lọc dưới định dạng .wav.

Tiến hành kiểm tra Sound Filter như sau:

i. Đọc file 'hootie.wav' vào Sound Filter.
ii. Thiết kế mạch lọc dải qua 1000Hz-3000Hz FIR từ FIR Designer với tần số lấy mẫu tương ứng với tần số của file audio. Bậc lọc là 50 và sử dụng cửa sổ Hamming. Lưu lại hệ số vào file .csv.
iii. Đọc file .csv vào Sound Filter.
iv. Lọc tín hiệu và nhận xét đáp ứng tần số trước và sau khi lọc.
v. Lưu lại dữ liệu sau khi lọc vào file .wav.

```
function []=plotFFT(wave,fs,titleName)
wave = wave';
N  = 10240; % Number of points
l=length(wave);
waves(1,:)=wave(1,(l-N+1):l);
win=hanning(N);
waves=win.*waves';
wavefft=fft(waves,N);
wave_db=20*log10(abs(wavefft));
figure
stem([0:N/2-1].*fs/N,wave_db(1:N/2));
axis([0 5000 -10 40]);
grid on;
title(titleName);
xlabel('Tan so (Hz)');
ylabel('Bien do (dB)');
end
```