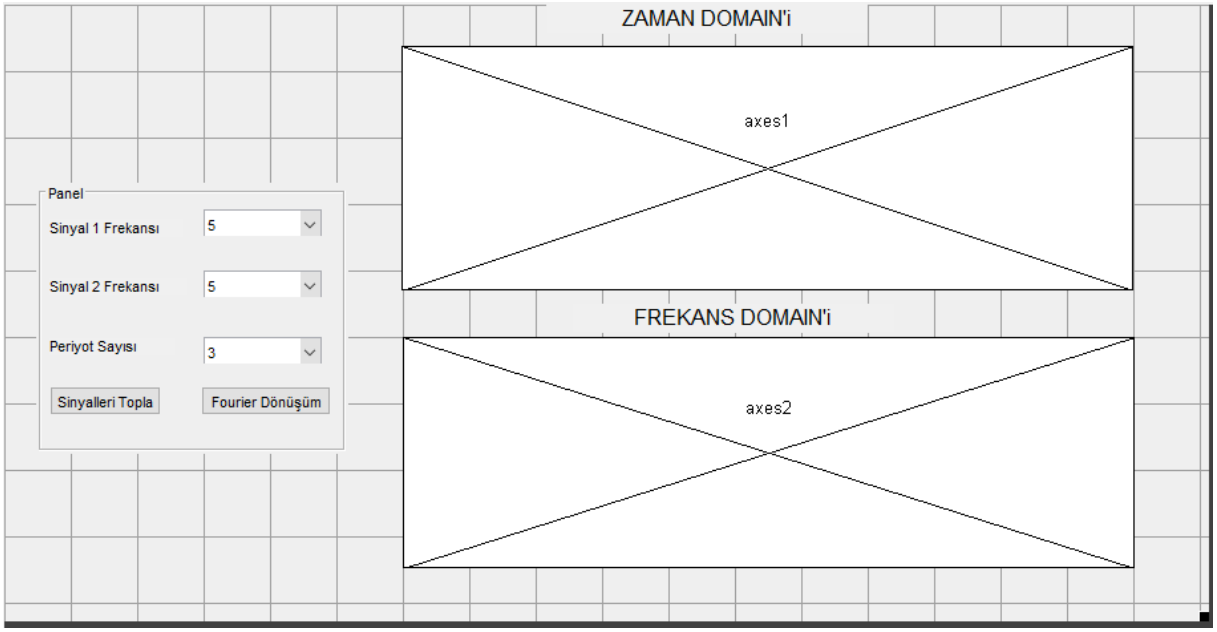
AKÜ TEKNOLOJİ FAKÜLTESİ MEKATRONİK MÜHENDİSLİĞİ

VTIY DERS-10 YARDIMCI NOTLARI -2018

TOPLANMIŞ İKİ SİNYALİN ZAMAN VE FREKANS DOMENLERİNDE GÖRÜNTÜLEYEN MATLAB ARA YÜZ TASARIMI:

Zaman domeni:

Aşağıdaki ara yüz tasarımında iki adet farklı frekanstaki sinüs sinyali toplanıyor. Toplanm sinyalin zaman domenindeki gösterimi ekranın üstünde bulunan axes üzerinde gösteriliyor. Frekans bilgileri popupMenu'lerden alınıyor. PopupMenu'lere 5, 25, 50 Hz. Seçenekleri girilmiştir. Periyot sayısını belirleyen popupMenu ile axes üzerinde toplam sinyalin kaç periyot boyunca görüntüleneceği belirlenir.



"Sinyalleri Topla" butonu ile toplama ve grafik çizdirme fonksiyonu başlatılmış olur. Tüm programın "Sinyalleri Topla" butonu ile tetiklenen kısmı aşağıda verilmiştir.

```
% --- Executes on button press in btnSinyalOlustur.
function btnSinyalOlustur_Callback(hObject, eventdata, handles)
clear fs f;cla;
axes(handles.axes1);%grafik axes1'e çizilecek.
SinyalFrekans1=get(handles.popFrekans1,'Value');%frekans1'i al.
switch SinyalFrekans1
    case 1
        SinyalFrekans1=5;
    case 2
        SinyalFrekans1=25;
    case 3
        SinyalFrekans1=50;
end
SinyalFrekans2=get(handles.popFrekans2,'Value');%frekans2'yi al.
switch SinyalFrekans2
    case 1
        SinyalFrekans2=5;
    case 2
```

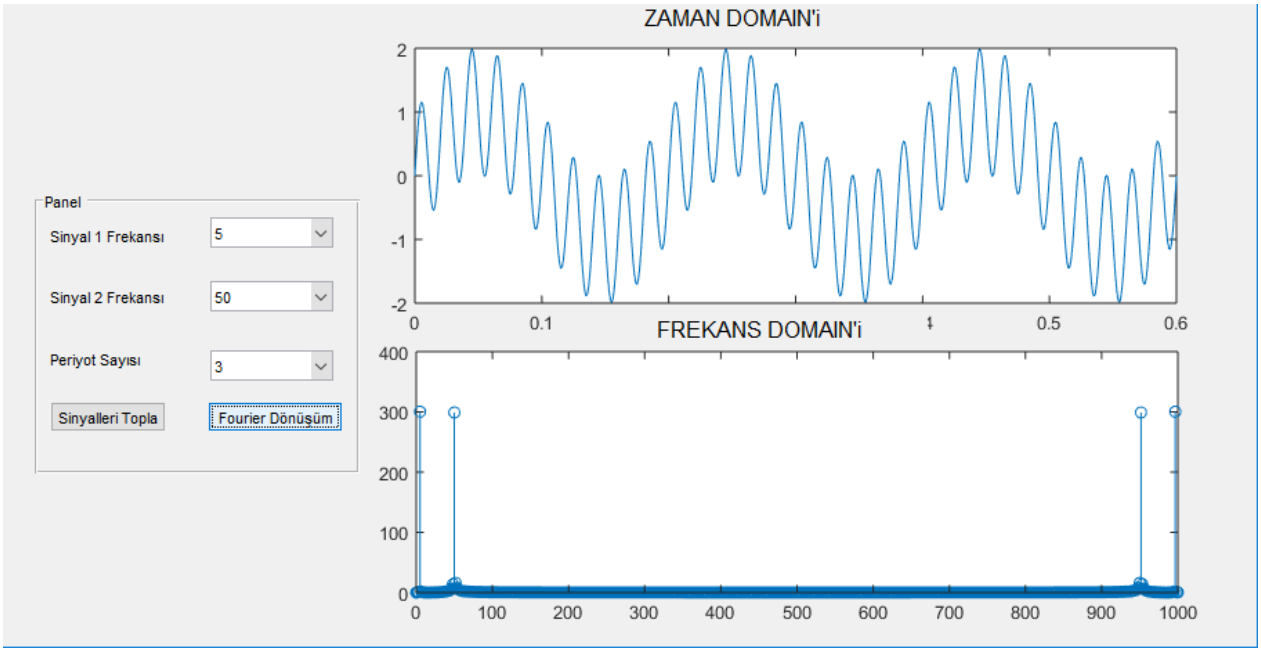
```
        SinyalFrekans2=25;
    case 3
        SinyalFrekans2=50;
end
Periyot=get(handles.popPeriyotSay,'Value');%periyot sayısını al.
switch Periyot
    case 1
        Periyot=3;
    case 2
        Periyot=5;
    case 3
        Periyot=10;
end
%Burada örnekleme frekansı (fs) belirleniyor. Hangi frekans büyükse o
%frekansın 20 katı olarak belirleniyor. Nyquist'e göre 2 kat olsa
%yeterli.
if   SinyalFrekans1>SinyalFrekans2
    fs=SinyalFrekans1*20;
else
    fs=SinyalFrekans2*20;
end

A=1;                    % genlik 1
ts=1/fs;                % örnekleme periyodu. Yani kaç sn'de bir örnek
alacak.
t=0:ts:Periyot/f1;      % Kaç periyot boyunca örnekleme yapıyoruz.

x=A*sin(2*pi*f1*t);     % x birinci sinyal, y ikinci sinyal
y=A*sin(2*pi*f2*t);

handles.z=x+y;          %z iki sinyalin toplamı. Başka bir fonksiyon
içinde kullanılacağı için handle edildi.
handles.fs=fs;
plot(t,handles.z);
guidata(hObject, handles);%handle edilen değişkenlerin tüm programda
tanımlı olabilmeleri için gereklidir.
```

Frekans domeni:



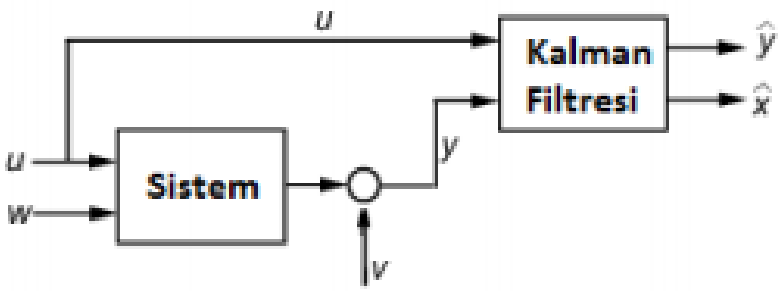
"Fourier Dönüşüm" butonuna tıklandığında ilk bölümde oluşturulan toplam sinyal Hızlı Fourier Dönüşümü kullanılarak frekans domenine aktarılmaktadır. Bu dönüşüm neticesinde toplam sinyal frekanslarına bölünür. Ekranın alt kısmındaki axes2 de bulunan grafikte toplam sinyali oluşturan 5 Hz ve 50 Hz. lik sinyallerin frekansları baskın sinyaller olarak açıkça görülmektedir. "Fourier Dönüşüm" butonuna basıldığında tetiklenen fonksiyon kodları aşağıda verilmiştir. Frekans ekseninin oluşturulması ile ilgili bir önceki derste verilen kod açıklamalarına bakınız.

```
% --- Executes on button press in btnFourierDonusumu.
function btnFourierDonusumu_Callback(hObject, eventdata, handles)

axes(handles.axes2);% grafik axes2 ye çizilecek.
fft0=fft(handles.z);
fft1=abs(fft0);
%frekans ekseni (fv) belirleniyor. En yüksek frekans fs'dir.
fv=0:(handles.fs/(length(handles.z)-1)):handles.fs;
stem(fv,fft1);
```

CALMAN FİLTRESİ OLUŞTURMA

Calman filtreleme ile ilgili ayrıntılı bilgi için:
http://www.ibrahimcayiroglu.com/Dokumanlar/Makale_BilgiPaylasim/%281-2012%29-Kalman_Filtresi_Ve_Bir_Programlama_Ornegi-Ibrahim_CAYIROGLU.pdf sitesinden bilgi alınabilir. Ancak kısaca bilgi vermek gerekirse;
Kalman filtresi sisteme verdiğimiz girişe (u) ve ölçülen gürültülü çıkışa (y) bakarak sistemin gerçek çıkışını ve durumunu tahmin etmeye çalışır. Bu durum şematik olarak aşağıdaki gibi ifade edilebilir:



Burada (w)'ya proses gürültüsü denir ve sistem modelinde yukarıda sayılan nedenlerden dolayı meydana gelen idealden sapmaları temsil eder. Ayrıca (v)'ye ölçüm gürültüsü denir ve ölçümde meydana gelen sıkıntıları temsil eder. Kalman filtresinde bu gürültülerin sisteme aşağıdaki gibi etki ettiği varsayılır.

$$\dot{x} = Ax + Bu + Gw$$

$$y = Cx + Du + Hw + v$$

Kalman filtresinin sistemin gerçek çıkışı için yaptığı tahmin $\hat{y}$ ile, ve sistemin durumu için yaptığı tahmin $\hat{x}$ ile gösterilir.

(https://kasnakoglu.files.wordpress.com/2014/01/ders7_ck01.pdf)

Kalman filtresi ile ilgili önemli kısımları işaretlenmiş makale ders notlarının bulunduğu linkten indirilebilir. Kalman filtresi bir filtreden daha çok bir tahmin (kestirim) mekanizmasıdır. Dolayısıyla veri okurken araya karışabilecek ani gürültülerin sinyali aşırı bir şekilde etkilemesini engeller. Sinyalin okunamadığı noktalarda sinyalin geçmiş karakteristiğine göre faydalı tahminlerde bulunur.

Kalman Filtre Uygulaması:

Programdaki formülasyon kod içindeki yorum satırlarında açıklanmıştır.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bu program dış ortamdan alınan verileri Kalman Filtresinden geçirerek
% verilere güvenilemediği durumlarda tahminler yapmaktadır.
% Tarih     : 09.05.2017
% Yer       : AKÜ TEKNOLOJİ FAKÜLTESİ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
close all;
%Başlangıç değerleri:
hata_kovaryansi =1.5;
kalman_katsayisi = 0.7;
onceki_kalman = 0;
```

```
figure
%%
%COM5 de kurulu olan Uno arduino sınıfını "a" değişkenine atadık.
a=arduino('COM3','Uno');
%Formülasyon:
%http://www.ibrahimcayiroglu.com/Dokumanlar/Makale_BilgiPaylasim/(1-
2012)-Kalman_Filtresi_Ve_Bir_Programlama_Ornegi-Ibrahim_CAYIROGLU.pdf
alınıştır.
%K(k): Kalman kazancının o andaki değeri
%K(k-1): Kalman kazancının bir önceki değeri
%P(k): Hata kovaryansının o andaki değeri
%P(k-1): Hata kovaryansının bir önceki değeri
%R: Kalman katsayısı
%X(k)=Sinyalin o andaki hesaplanan değeri , Kalman tarafından tahmin
edilen
%değer.
%X(k-1): Sinyalin bir önceki hesaplanmış değeri
%Z(k): Sinyalin o andaki ölçülen değeri (-ki biz doğruluğundan emin
değiliz!)

for i = 1:300
olculen_veri(i)=readVoltage(a,'A0'); %  A0 potansiyometre
%Kalman kazancının yeni değeri bulunuyor.
%K(k)=P(k-1)/(P(k-1)+ R) R: Kalman katsayısı
kalman_kazanci = hata_kovaryansi/(hata_kovaryansi+ kalman_katsayisi);
%Yeni kalman çıkış değeri hesaplanıyor.
%X(k)=X(k-1) + (K(k)*(Z(k)-X(k-1)))
kalman_tahmin  = onceki_kalman + (kalman_kazanci*(olculen_veri(i)-
onceki_kalman));
% hata kovaryansının yeni değeri bulunuyor.
% P(k)=(1-K(k))*P(k-1)
hata_kovaryansi = (1-kalman_kazanci)*hata_kovaryansi;
%X(k-1) burada oluşturuluyor.
%X(k-1): onceki_kalman, X(k): kalman_tahmin
onceki_kalman = kalman_tahmin;
%X(k) değeri kalman_tahmin_sakla dizisine kayıt ediliyor.
kalman_tahmin_sakla(i) = kalman_tahmin;
plot(olculen_veri,'b');
hold on
plot(kalman_tahmin_sakla,'r');
drawnow
end
hold off;
```

MEDYAN FİLTRESİ OLUŞTURMA

Görüntü ve sinyal işleme konularında, gürültü temizlemek için kullanılan yöntemlerden birisidir. Amaç belirli bir pencere aralığındaki sayıların ortancasını (median) alarak aşırı büyük atlamaları kaldırmaktır. Yani filtre uygulandıktan sonra sinyalde bulunan ve normal değerlerden belirgin şekilde ayrılan değerlerin tespit edilerek temizlenmesi sağlanır.

Basit bir ortanca filtresinin nasıl çalıştığını inceleyelim. Örneğin aşağıdaki sayılar için kenar tekrarlı (edge repeating) ortanca filitresi uygulayalım (pencere genişliği 3 olarak alınmıştır.)

Yani medfilt1(x,3) ya da medfilt1(x) yeterlidir.

g = [2 32 2 1 2 8 9]

ç [0] = [2 2 32] = [2 2 32] => 2 // burada ilk sayıyı tekrar ettik çünkü katar 3 olmalı ve şayet ilk sayıyı 2 kere almazsak ilk sayı için 3 adet sayımız olmayacaktır.
ç [1] = [2 32 2] = [2 2 32] => 2 // ilk dizi giriş dizisinin ilk 3 sayısıdır (katar 3 olduğu için) ikinci dizi ise sıralanmış halidir. ve sonuç olarak ortanca değer 2 bulunur.
ç [2] = [32 2 1] = [1 2 32] => 2
ç [3] = [2 1 2] = [1 2 2] => 2
ç [4] = [1 2 8] = [1 2 8] => 2
ç [5] = [2 8 9] = [2 8 9] => 8
ç [6] = [8 9 9] = [8 8 9] => 9

ç = [2 2 2 2 2 8 9]
yukarıdaki örnekte g giriş dizisi, ç ise çıkış dizisi olarak kabul edilmiştir.
Görüldüğü üzere yukarıdaki örnekte ortanca filtresi uygulandıktan sonra çok büyük bir sayı olan ve yakınlarında başka benzer sayı bulunmayan 32 sayısı elenmiştir. Benzer şekilde yakınlarında benzeri bulunmayan 1 sayısı da çok fazla 2 sayısı arasında kalıp bu sayılar tarafından boğulmuştur.

(http://bilgisayarkavramlari.sadievrenseker.com/2007/11/26/ortanca-filitresi-median-filter/)

Medyan Filtresi Uygulaması

Bu uygulamada sinyali kendimiz oluşturduk. Ancak dışarıdan alabileceğimiz sinyallerde de kullanabiliriz. Kod aralarındaki açıklamalara dikkat ediniz.

```
%önce sinyalimizi oluşturuyoruz.
clear;clc;close all;
fs = 100;
t = 0:1/fs:1;
%aşağıda iki sinyalin toplamı var. Frekanslar ve genlikler arasındaki
farka dikkat edelim. İlk sinyalin frekansı düşük (3 hz) diğer ise daha
büyük (40 hz). Bu durumda ikinci sinyal birincinin üzerine gürültü gibi
eklenecektir.
x = sin(2*pi*t*3)+0.25*sin(2*pi*t*40);
plot(t,x);hold on
legend('Orjinal sinyal')
%%
%Medyan, sıralanmış bir sayı listesinde ortadaki sayıdır.
%örnek:{13, 23, 11, 16, 15, 10, 26} sayılarının medyanı:
% önce sayılar sıralanır: {10, 11, 13, 15, 16, 23, 26}
```

```
%medyan değeri 15'dir.

%medfilt1(x,n) olarak kullanılır. n ortancası bulunacak aralığı
%belirtir. Bu örnekte 11 olarak alınmıştır.
%verilmez ise varsayılan olarak değeri 3'dür.
y = medfilt1(x,11);
plot(t,y,'r','linewidth',2)
legend('filtrelenmiş');
% Eğer n = 11, ise y(k) x(k-5:k+5) aralığının ortancasıdır.
% Eğer n = 12, ise y(k) x(k-6:k+5) aralığının ortancasıdır.
```

Programın Çıktısı: