GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELM 368 SAYISAL İŞARET İŞLEME LABORATUVARI

ÖN HAZIRLIK ÇALIŞMASI

Laboratuvar 7

1 AMAÇ

- DZD sistemlerin dönüşümlerinin analizinin öğrenilmesi,
- Frekans uzayında sistem analizinin öğrenilmesi,
- Genlik, faz cevabı ve grup gecikmesi kavramları ve bunların *python* ile pratikte kullanımı,
- Lineer faz kavramı ve lineer fazlı FIR sistemlerin öğrenilmesi,
- Filtre tasarımı
- *Pyfda* filtre tasarım arayüzünün incelenmesi ve filtre tasarımı için kullanılması.

2 DZD SİSTEMLERDE DÖNÜŞÜM ANALİZİ

Daha önce, DZD sistemlerin analizinde yaygın olarak kullanılan iki temel matematiksel yöntem (i) Fourier dönüşümü ve (ii) Z-dönüşümü görmüştük. Fourier dönüşümünde işaret/sistem frekans uzayına taşınarak (genellikle) genlik cevabı üzerinden analiz ediliyordu. Şimdi ise, buna ek olarak faz cevabı da analiz edilerek, sistemin/işaretin bütünü hakkında fikir sahibi olunacaktır.

Bu amaçla, önce bir sistemin faz cevabı ve grup gecikmesi kavramlarından bahsedilecek, daha sonra bunlar üzerinden minimum fazlı ve tüm geçiren sistemler ve bu sistemlerin hangi amaçla kullanılabildiği üzerinde durulacaktır. Ardından bunlarla ilgili örnek bilgisayar uygulamaları verilecektir. Lineer faz kaz kavramı ve lineer faza sahip FIR filtrelerden bahsedilecektir.

Son olarak, IIR ve FIR filtre tasarımı tartışılacak ve *pyfda* arayüzü ile filtre tasarımı örnekleri sunulacaktır.

Bu çalışmanın sonunda, öğrencinin, verilen bir sistemi frekans uzayında analiz edebilme, belirli bir amaca yönelik farklı yollarla filtre tasarımı yapabilme, verilen mevcut sistemleri belirli bir amaç doğrultusunda modifiye edebilme kabiliyetleri kazanması beklenmektedir. Ayrıca, bu amaçlar doğrultusunda, filtre tipi, kesim frekansı, grup gecikmesi gibi kavramlara aşinalık kazanması beklenmektedir.

Bu lab çalışmasında işlenecek konuların özeti için **1-Giriş** videosunu izleyiniz. Dökümanın ilgili yerlerinde, paylaşılan videolara atıfta bulunulacaktır, dolayısıyla bu deney föyünü ve video derslerini beraber takip etmeniz gerekmektedir. Farklı olarak, videolarda **MATLAB** ile verilen örnekler, bu dökümanın ekinde ***python*** diline dönüştürülerek sizinle paylaşılmıştır. Dolayısıyla, videolarda gördüğünüz sonuçları örnek *python* kodlarıyla tekrar oluşturabilirsiniz.

2.1 Faz cevabı ve grup gecikmesi

Ayrık zamanlı bir sistemin/işaretin frekans uzayı ifadesi açısal frekans değişkenine bağlı kompleks bir fonksiyondur. Bu kompleks fonksiyon iki bileşen ile ifade edilmektedir: (i) genlik cevabı ve (ii) faz cevabı. Bu frekans uzayı ifadesi aşağıda verilen eşitlik ile gösterilebilir:

$$H\left(e^{j\omega}\right) = H(\omega) = \underbrace{|H(\omega)|}_{genlik\ terimi} \cdot \underbrace{\angle H(\omega)}_{faz\ terimi}. \tag{2.1}$$

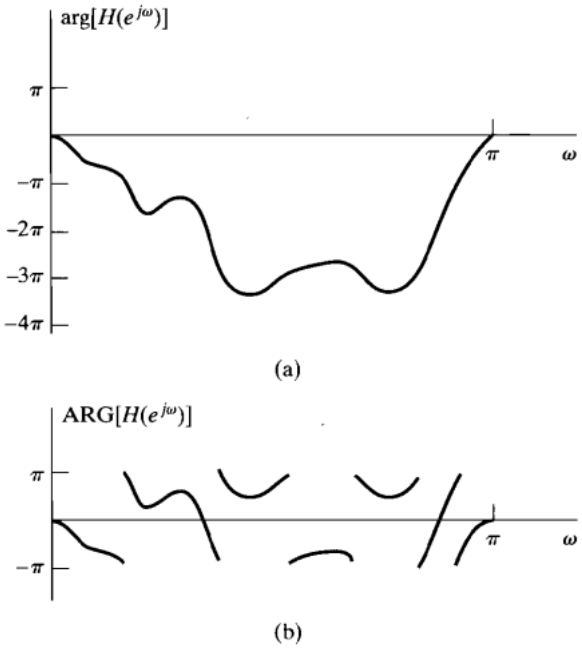
Bir kompleks sayının faz açısı benzersiz bir değere sahip **değildir**. Bu yüzden, özellikle ters tanjant ile nümerik olarak hesaplanırken genellikle "temel değeri" (principal value) elde edilir. Bu temel değer, şu eşitsizlikle verilmektedir:

$$-\pi < ARG[H(\omega)] \leq \pi. \tag{2.2}$$

Aynı kompleks sayıyı veren başka herhangi bir açı şu eşitlikle elde edilebilir:

$$\angle H(\omega) = ARG[H(\omega)] + 2\pi r(\omega). \tag{2.3}$$

Burada, $r(\omega)$ pozitif veya negatif bir **tamsayıdır**, dolayısıyla faz açısı 2π ile periyodiktir. Anlatım boyunca, $ARG[H(\omega)]$ ifadesi **sarılı faz (wrapped phase)** olarak anılacaktır. (2.3) ise **sürekli faz fonksiyonu** (**unwrapped**) olarak anılacaktır. Bu iki ifade arasındaki fark, aşağıdaki şekilde gösterilmektedir. Detaylı bilgi için ders kitabında 5.1.1 bölümüne başvurunuz.



Şekil 1 Sürekli (a) ve Sarılı (b) faz fonksiyonları.

Şekil 1-(a)'da faz fonksiyonunun sürekli hali (**unwrapped**), (b)'ye baktığımızda ise katlanmış (**wrapped**) faz fonksiyonunun ELM368 dersini alan öğrencilerin yüz ifadesini andırdığını görüyoruz.

Faz cevabı, sistemin giriş işaretine her bir frekans değerinde ne kadarlık **zaman gecikmesi** olarak etki edeceğini söylemektedir ve **birimi radyan** olarak tanımlıdır. Ancak, sistemden kaynaklı bu gecikme miktarı, faz cevabında doğrudan görünmemektedir. Sistemden kaynaklanan bu zaman gecikmesi, faz cevabı üzerinden hesaplanan ve **grup gecikmesi** olarak adlandırılan bir reel değerli fonksiyon ile gösterilmektedir. Bu reel değerli fonksiyonun **birimi örnek**tir. Grup gecikmesi fonksiyonu aşağıdaki eşitlikte verildiği gibi hesaplanmaktadır:

$$\tau(\omega) = grd[H(\omega)] = -\frac{d}{d\omega}\angle H(\omega). \tag{2.4}$$

Burada kullanılan açı, **sürekli faz fonksiyonudur** (**unwrapped phase**). Yani, sistemin faz cevabının açısal frekansa göre türevi, grup gecikmesini vermektedir.

Örnek-1: dürtü cevabı $h[n] = \delta[n - n_0]$ olan ideal DZD sistemi ele alalım. Bu sistem için, frekans cevabını hesaplayıp, faz cevabını ve grup gecikmesini hesaplayalım. Bu sistemin frekans cevabı,

$$H(\omega) = e^{-j\omega n_0}$$

Şeklindedir. Veya genlik ve faz formunda aşağıdaki gibi gösterilir:

$$|H(\omega)| = 1$$
$$\angle H(\omega) = -\omega n_0$$

Buradan, faz fonksiyonunun açısal frekansa göre türevi alınırsa, grup gecikmesi $\tau(\omega) = n_0$ (örnek/sample) ifadesine ulaşılır. Bundan elde edilecek sonuç ise, verilen bu sistemin "**sabit grup gecikmesi**" veya "**lineer fazlı**" bir sistem olduğudur. Grup gecikmesinin sabit olması ise şu anlamı taşımaktadır: **bu sistem, girişine uygulanan işaretin tüm frekans bileşenleri üzerinde aynı n_0 miktarda zaman gecikmesine neden olmaktadır**.

Örnek-2: Bu ifadeyi daha iyi anlayabilmek için, ders kitabının 5.1.2 bölümünde anlatılan örnek *python* üzerinde gerçekleştirilecektir. **2-Faz ve grup gecikmesi** isimli videoyu izleyiniz ve ***LAB7_Ornek2.ipynb*** inceleyiniz.

2.2 Lineer Fazlı FIR Filtreler

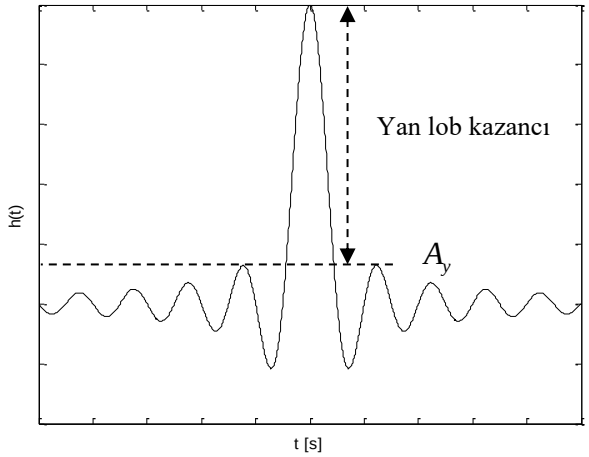
Lineer faz, bir önceki örnekte de görüldüğü üzere, grup gecikmesi açısından, DZD sistemlerin önemli bir özelliğidir. Her DZD sistem, lineer faza sahip olmayabilir. Doğrusal faza sahip olması istenen IIR sistemler/filtreler büyük özenle tasarlanmalıdır. Buna karşın, doğrusal faza sahip FIR sistemler/filtreler spesifik bazı formlarda bulunmaktadır. Bu FIR filtre formları, sahip oldukları terim sayısı ve katsayıların simetrikliği açısından Tip-1, Tip-2, Tip-3 ve Tip-4 FIR filtreler olarak adlandırılmaktadır. Bu dört tipten birine ait FIR filtreler **lineer faz** garantisi vermektedir. Ancak her bir filtre tipi ile sadece spesifik türde filtreler (alçak geçiren, bant geçiren, vs) tasarlanabilmektedir. **3-FIR filtre tipleri** isimli videoyu izleyiniz.

Örnek-3: **4-Grup gecikmesi** ve **5-Grup gecikmesi-2** isimli videoları izleyiniz ve ***LAB7_Ornek3.ipynb*** inceleyiniz.

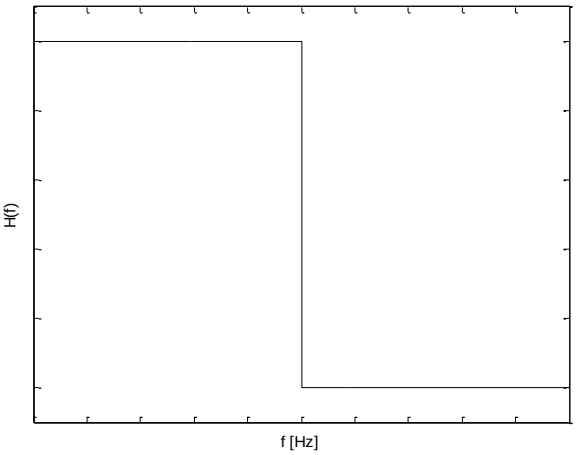
3 FİLTRE TASARIMI

3.1 İdeal Olmayan Filtreler

Temel olarak filtreler, belirli banttaki sinyalleri geçiren, diğer banttaki sinyalleri sönümlendiren sistemlerdir. Şekil 2 ve Şekil 3'te ideal bir alçak geçiren filtrenin sırasıyla dürtü ve genlik cevabı görülmektedir.



Şekil 2. İdeal AGF dürtü cevabı.



Şekil 3. İdeal AGF genlik cevabı.

Şekil 3'de görüldüğü üzere, ideal bir alçak geçiren filtrenin genlik cevabı dikdörtgen fonksiyonundan oluşmaktadır.

$$H_{AGF}(f) = \Pi\left(\frac{f}{2f_c}\right) \tag{1}$$

Burada f_c filtrenin kesim frekansını göstermektedir. Eşitlik (1)'in ters Fourier dönüşümü alınırsa, ideal bir alçak geçiren filtrenin dürtü cevabı,

$$h_{AGF}(t) = \mathrm{sinc}\left(2f_c t\right) \tag{2}$$

ifadesine sahiptir. Diğer tüm ideal filtreler (yüksek geçiren, bant geçiren, vb.), farklı kesim frekanslarına sahip alçak geçiren filtrelerin lineer kombinasyonundan oluşmaktadır.

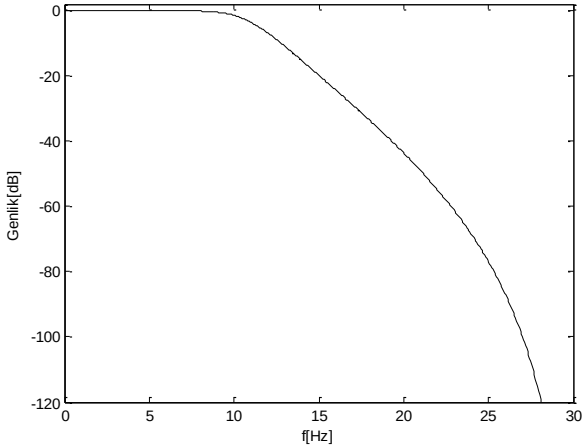
$$H(f) = \sum_{i=1}^{I} a_i \Pi\left(\frac{f}{2f_{c,i}}\right) \tag{3}$$

Burada I AGF sayısını, a_i ise her bir AGF'nin katsayısını göstermektedir. Dolayısıyla, tüm ideal filtrelerin dürtü cevabı sinc fonksiyonlarının lineer kombinasyonundan oluşmaktadır.

$$h(t) = \sum_{i=1}^{I} a_i \mathrm{sinc}\left(2f_c t\right) \tag{4}$$

Şekil 2'deki grafikten anlaşılacağı üzere, dürtü cevabının *sinc()* fonksiyonuna sahip olması, dürtü cevabında ana lob ve yan lobların varlığını beraberinde getirmektedir. Ana lobun genliği $A = 0$ dB iken en yüksek genlikli yan lobun kazancı $(A_y, \text{desibel biriminde})$, bu filtrenin yan lob kazancını vermektedir. Frekans uzayında yapılan filtreleme işleminin zaman uzayında meydana getireceği gürültülerin miktarı, dürtü cevabının yan lob kazancı ile ilgilidir. Daha düşük yan lob seviyesi, zaman uzayında daha az gürültü bileşeni meydana getirmektedir. Bunu sağlamak için, frekans uzayında

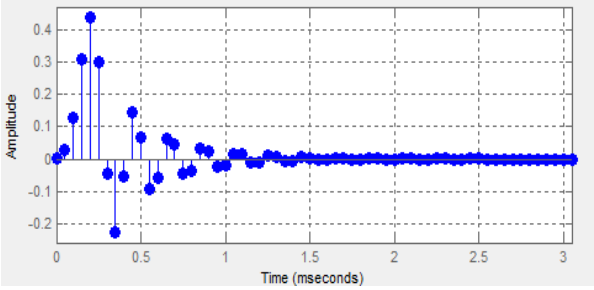
dikdörtgen fonksiyonu ile filtreleme yapmak yerine, Şekil 4'teki gibi, köşelerindeki sivrilikleri azaltılmış, bu bölgelerde daha düşük türeve sahip fonksiyonlar kullanılmaktadır.



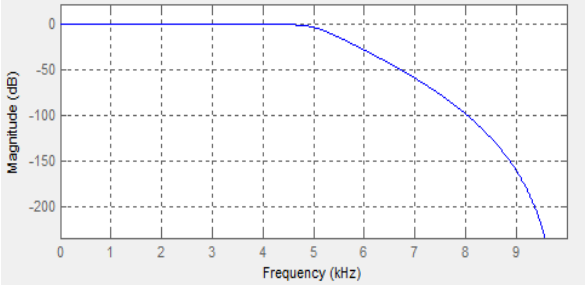
Şekil 4. Butterworth filtresi genlik cevabı.

Daha düşük değerli türevin varlığı, dürtü cevabında daha düşük yan lob seviyesini beraberinde getirmektedir. Bu sayede, zaman uzayında yan lob seviyesinden kaynaklanan gürültüler azaltılmaktadır. Buna karşılık, frekans uzayında daha az keskinliğe sahip filtreleme işlemi yapılmaktadır. Şekil 3 incelenirse, ideal AGF'lerde kesim frekansından daha yüksek banttaki sinyaller 0 ile çarpılmaktadır. Fakat, Şekil 4'deki gibi bir ideal olmayan filtrenin kullanımı, yüksek frekans bileşenlerinin de çıkış sinyalinde görülmesine sebep olmaktadır.

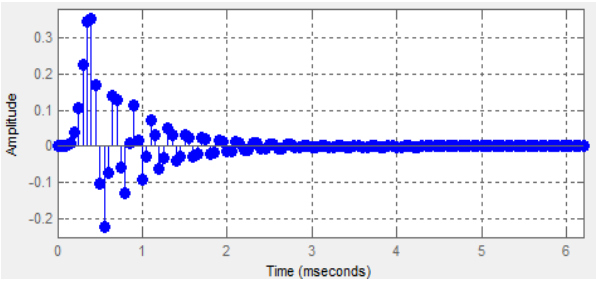
İdeal olmayan filtrelerin dürtü cevabındaki yan lob seviyesi, filtre türüne ve derecesine bağlıdır. Genel olarak, daha yüksek keskinliğe sahip bir filtreleme yapmak için filtre derecesini arttırmak, filtrenin genlik cevabını ideale yaklaştırmaktadır. Buna karşılık, dürtü cevabındaki yan lobların seviyesi artmaktadır. Şekil 5 ile Şekil 7 arasındaki grafiklerde, kesim frekansı $f_c = 5$ kHz ve örnekleme frekansı $f_s = 20$ kHz olan farklı mertebeden Butterworth filtrelerin dürtü ve genlik cevapları görülmektedir.



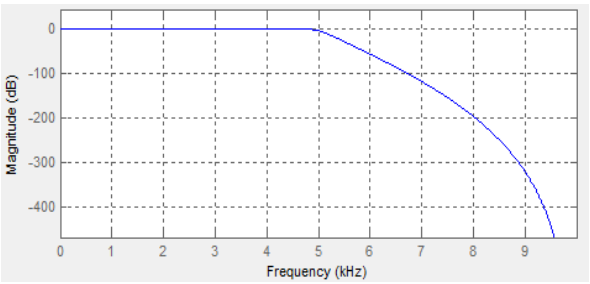
Şekil 5. 10. dereceden bir Butterworth filtresinin dürtü cevabı.



Şekil 6. 10. dereceden bir Butterworth filtresinin genlik cevabı.

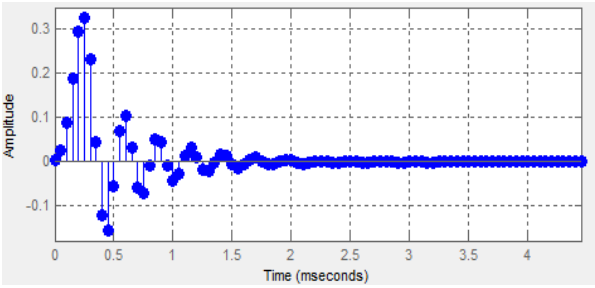


Şekil 7. 20. dereceden bir Butterworth filtresinin dürtü cevabı.

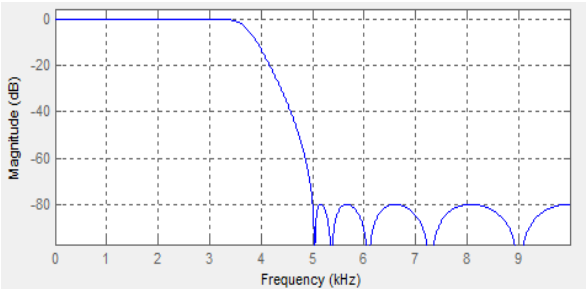


Şekil 8. 20. dereceden bir Butterworth filtresinin genlik cevabı.

Dürtü cevabındaki yan lob kazancını etkileyen faktörlerden biri de filtre fonksiyonudur. Aşağıdaki grafikler Şekil 5 ve Şekil 6 ile karşılaştırıldığında, aynı filtre mertebesi kullanılmasına rağmen Chebyshev II filtresinin Butterworth filtresine göre daha keskin filtreleme imkanı sunduğu görülmektedir. Buna karşılık, Şekil 5 ve Şekil 9 karşılaştırıldığında, dürtü cevabında Chebyshev II filtresinin Butterworth filtresine göre daha yüksek yan lob kazancına sahip olduğu görülmektedir.

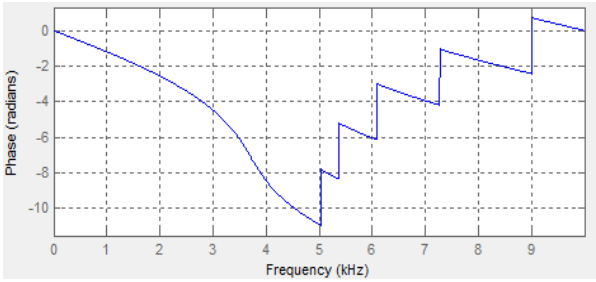


Şekil 9. 10. dereceden bir Chebyshev II filtresinin dürtü cevabı.

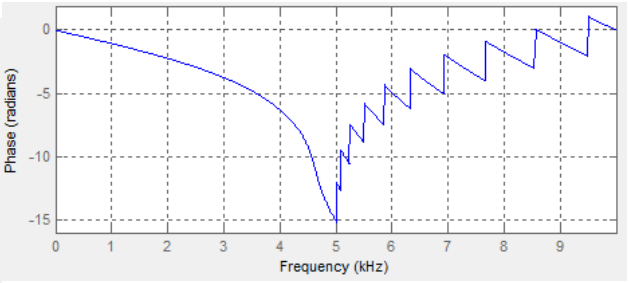


Şekil 10. 10. dereceden bir Chebyshev II filtresinin genlik cevabı.

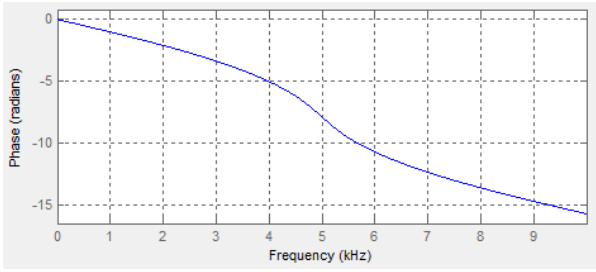
İdeal olmayan tüm filtrelerin faz gecikmesi vardır. Bu gecikme filtre türüne, mertebesine ve kesim frekansına bağlıdır. Filtre mertebesinin arttırılması faz gecikmesini de arttırmaktadır. Dolayısıyla, filtre mertebesi ve kesim frekansları seçilirken zaman uzayında meydana getireceği faz gecikmesi de dikkate alınmalıdır.



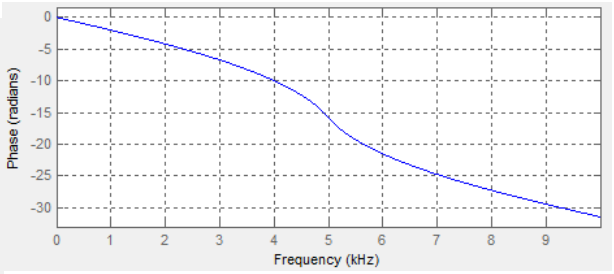
Şekil 11. 10. dereceden bir Chebyshev II filtresinin faz cevabı.



Şekil 12. 20. dereceden bir Chebyshev II filtresinin faz cevabı.



Şekil 13. 10. dereceden bir Butterworth filtresinin faz cevabı.



Şekil 14. 20. dereceden bir Butterworth filtresinin faz cevabı.

Şekil 11 ile Şekil 12 ve Şekil 13 ile Şekil 14 karşılaştırıldığında, filtre mertebesinin arttırılmasının faz gecikmesini arttırdığı görülmektedir. Aynı zamanda, Şekil 11 ve Şekil 13 karşılaştırıldığında, farklı filtre fonksiyonlarının farklı faz gecikmelerine sahip olduğu da görülmektedir.

İdeal olmayan filtreleri birbirinden ayıran özelliklerden birisi de grup gecikmesidir. İdeal olmayan bir filtrenin girişine gelen sinyal,

$$x(t) = A\cos(2\pi f_0 t), \tag{5}$$

formunda olması halinde filtrenin çıkışından görülen sinyal,

$$y(t) = A\left|H(f_0)\right|\cos(2\pi f_0 t + \angle H(f_0)), \tag{6}$$

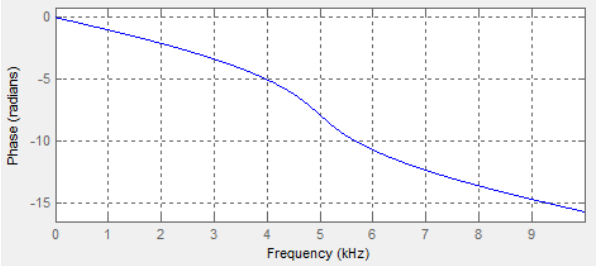
ifadesine sahiptir. Bu ifade zaman gecikmesi cinsinden yazılırsa,

$$y(t) = A\left|H(f_0)\right|\cos\left(2\pi f_0 t + 2\pi f_0 \frac{\angle H(f_0)}{2\pi f_0}\right) = A\left|H(f_0)\right|\cos\left(2\pi f_0\left(t + \overbrace{\frac{\angle H(f_0)}{2\pi f_0}}^{t_0}\right)\right), \tag{7}$$

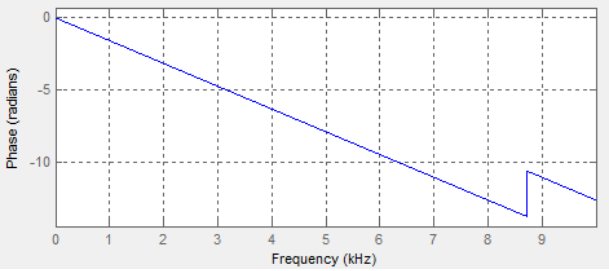
elde edilmektedir. Giriş sinyalinde birden çok frekans bileşeninin olması durumunda bozulmasız iletimin gerçekleşebilmesi için, t_0 zaman gecikmesinin f_0 frekansına bağlı olmaması gerekmektedir. Bu sebeple,

$$\frac{\partial}{\partial f_0}\left(\frac{\angle H(f_0)}{2\pi f_0}\right)=0, \quad \frac{\angle H(f_0)}{2\pi f_0}=c \tag{8}$$

olması gerekmektedir. Burada c sabit bir sayıdır. Filtrenin nedensel olması için, çıkış sinyalinin $y(t)=A|H(f_0)|\cos(2\pi f_0(t-t_0)),\ t_0>0$ formunda olması gerekmektedir. Bu sebeple $c<0$ şartı sağlanmalıdır. Dolayısıyla, **filtrenin geçirme bölgesinde** faz cevabı negatif eğimli rampa fonksiyonu olmalıdır. FIR filtreleri IIR filtrelerden ayıran özelliklerden birisi faz cevabıdır. FIR filtreler lineer faza sahipken, IIR filtrelerin faz cevabı lineer değildir.

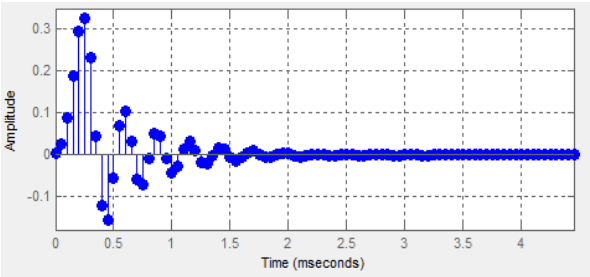


Şekil 15. 10. dereceden bir Butterworth filtresinin (IIR) faz cevabı.

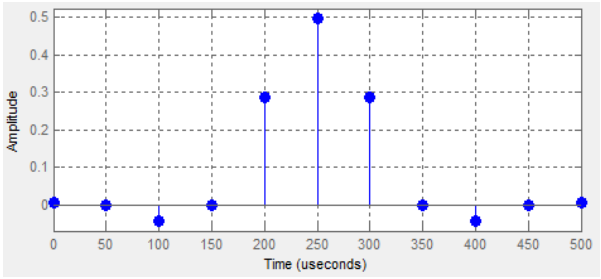


Şekil 16. 10. dereceden bir Hamming penceresinin (FIR) faz cevabı.

FIR ve IIR filtreleri birbirinden ayıran özelliklerden biri de dürtü cevabı genişliğidir. FIR filtrelerin dürtü cevabı sonlu elemana sahipken, IIR filtrelerinki sonsuzdur.



Şekil 17. 10. dereceden bir Butterworth filtresinin (IIR) faz cevabı.



Şekil 18. 10. dereceden bir Hamming penceresinin (FIR) faz cevabı.

FIR filtrelerin dürtü cevabının IIR filtrelerden daha dar olması, FIR filtrelerin genlik cevabının IIR filtrelerden daha geniş olmasını beraberinde getirmektedir. Şekil 6 ve Şekil 10 bu durumu özetlemektedir.

3.2 PYFDA ile Filtre Tasarımı

Python ile filtre tasarımı, *pyfda* isimli bir araç üzerinden yapılacaktır. Bu aracın nasıl kurulacağı ve kullanılacağı ile ilgili olarak **6-anaconda ve pyfda 1** ve **7-pyfda ve filtre tasarımı 2** videolarını izleyiniz. **NOT: videoların aksine, filtre katsayıları kaydedilirken .csv yerine .mat tipi seçilmelidir.**

Örnek-4: Bu örnek, *pyfda* ile tasarlanan bir filtrenin kullanılmasını incelemektedir. *Pyfda* kullanarak bir IIR filtre tasarlanmıştır. Bu filtreye ait katsayılar ***butterworth.mat*** olarak kaydedilmiştir. ***LAB7_Ornek4.ipynb*** inceleyiniz. Örnek kodlarda, filtre katsayılarının **.mat** içinden nasıl okunacağı ve farklı amaçlarla nasıl kullanılabileceğine dair örnek kodlar mevcuttur. En sonda ise örnek bir sinyalin nasıl bu filtreye uygulanabileceği gösterilmiştir.

4 ÖN HAZIRLIK SORULARI

1. **3-FIR filtre tipleri** isimli videoda bahsedilen dört FIR filtre tipinden her birini dürtü cevapları ile *python*'da tanımlayınız. Her bir filtreyi **4-5 noktada, genlikleri 1 veya -1 (simetriklik ve anti-simetriklik durumlarını dikkate alarak) değerine sahip** olarak tanımlamanız yeterlidir. Her bir sistemin 100 noktada Fourier dönüşümlerini hesaplayınız ve **genlik, faz ve grup gecikmesi** grafiklerini $[0, 2\pi]$ açısal frekans aralığında çizdiriniz (eksen etiketlerine ve birimlerine dikkat ediniz). Bu filtrelerin ne iş yaptığını (alçak geçiren, yüksek geçiren, bant geçiren vs) **genlik cevabına bakarak** tahmin etmeye çalışınız. Vardığınız sonucu, filtre tipi ile karşılaştırarak teyit ediniz. Filtrelerin lineer faza, sabit grup gecikmesine sahip olup olmadığını söyleyiniz. Ardından, bu sistemlerin kutup-sıfır diyagramlarını çizdiriniz. Sıfır konumlarını inceleyerek filtrenin görevini tekrar söylemeye çalışınız (alçak geçiren, bant geçiren, vs).
2. ***Pyfda*** kullanarak, farklı filtreler tasarlayınız. Filtrelerden biri bant geçiren, diğeri alçak geçiren olacak şekilde **en az iki farklı** yöntemle **FIR** filtre tasarlayınız. Kesim frekanslarını, filtrelerin **bant genişlikleri** yaklaşık 0.3π olacak şekilde istediğiniz gibi seçebilirsiniz. Filtre derecesini istenen kesim frekanslarında **en az 3 dB zayıflama** olacak şekilde belirleyiniz. Filtrelerin genlik, faz cevaplarını ve grup gecikmelerini çizdirip inceleyiniz. **Gözlemlerinizi anlatınız.**
3. ***Pyfda*** kullanarak, farklı filtreler tasarlayınız. Filtrelerden biri yüksek geçiren, diğeri alçak geçiren olacak şekilde **en az iki farklı** yöntemle **IIR** filtre tasarlayınız. Kesim frekanslarını, filtrelerin **bant genişlikleri** yaklaşık 0.3π olacak şekilde istediğiniz gibi seçebilirsiniz. Filtre derecesini istenen kesim frekanslarında **en az 3 dB zayıflama** olacak şekilde belirleyiniz. Filtrelerin genlik, faz cevaplarını ve grup gecikmelerini çizdirip inceleyiniz. **Gözlemlerinizi anlatınız.**
4. ***Ornek-2***'de verilen işareti, tasarladığınız filtrelerin hepsine uygulayın ve elde ettiğiniz sonuçları yorumlayın. Filtreler beklediğiniz gibi çalıştı mı? Ne bekliyordunuz? Nasıl bir sonuçlar karşılaştınız? Bu sinyal üzerinde hangi filtre daha iyi çalıştı size göre? Filtrelerin performanslarını karşılaştırmak isteseniz, nasıl bir yol izlersiniz? Filtrelerin fazları (veya grup gecikmeleri) bu işareti nasıl etkiledi? Filtre derecelerinin etkisi sizce ne oldu?

5 TESLİM ŞEKLİ ve ZAMANI

Bu dokümanda ve/veya ekinde verilen kodları kendiniz bir Jupyter Notebook'ta yazarak sonuçları gözlemleyiniz. Ön Hazırlık Soruları bölümünde verilen soruları çözmek için Python kodu yazınız. Jupyter Notebook'ta yaptığınız çalışmaların tamamını tek bir dosya olarakOgrenciNo_Ad_Soyad_LAB5.ipynb formatına uygun bir isimle kaydedip Google Classroom'a yükleyiniz. Laboratuvar ön çalışmaları (ev ödevi), 4 Haziran 2020 sabah 05:00'a kadar sisteme yüklenmelidir. Sisteme geç yüklenen dosyalar kabul edilmeyecektir. Ön hazırlık çalışmasını yapmamış (sisteme ön çalışmasını yüklememiş) öğrenciler aynı yapılacak laboratuvar çalışmasına giremez. Ekte, örnek bir ödev çözümü şablonu verilmektedir (bknz: 101024099_AYSE_SEN_LAB1.ipynb). Jupyter Notebook'ta yapacağınız çözümler bu şablona göre hazırlanmalıdır.