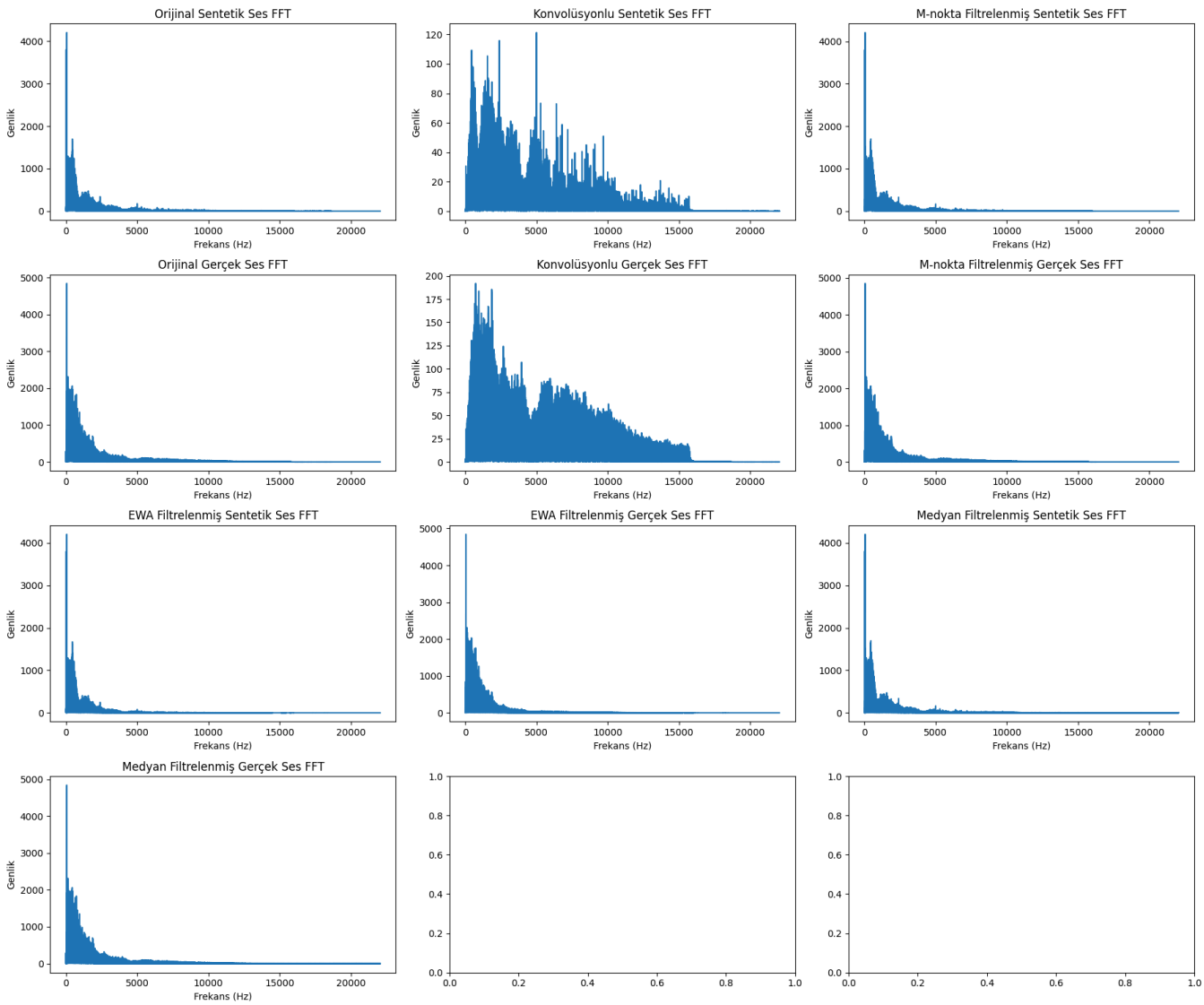
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
from scipy.signal import medfilt, fftconvolve


#def sentetik_dogal_ses_ayirt_et(ses_verisi, M, alpha):
""" Doğal ve sentetik ses ayırt etme işlemini yapmak için “sentetik_dogal_ses_ayirt_et” adında bir fonksiyonu tanımlanabilir.
Bu durumda aşağıda yazılan filtreme fonksiyonları, fonksiyon olarak değil, bu ayırt etme fonksiyonunun adımları şeklinde yazılır.
Bu fonksiyon; ses_verisi, M ve alpha olmak üzere 3 parametre alır.
M ve alpha değişkenlerine farklı değerler verilerek istenilen filtre etkisi elde edilir.
Her bir adımı ayrı fonksiyonlarla işlemek daha uygulanabilir olduğu için bu kodda ayrı fonksiyonlar şeklinde tanımlanmıştır.
Diğer versiyonu da başka bir proje dosyası olarak mevcuttur.

Aşağıda tanımlı filtrelerin sonuçlarını inceleyerek doğal ve sentetik ses arasındaki farklılıkları anlamak mümkündür. Detayları raporda a

# Sentetik ve doğal ses ayırt etmek için kullanılan Fonksiyon Tanımlamaları:

# M nokta kayan ortalama alıcı filtre:
def m_nokta_kayar_ortalama_filtre(ses_verisi, M):
    return np.convolve(ses_verisi, np.ones(M)/M, mode='valid')
    #convolve fonksiyonunu konvolüsyon almak için kullanılır.
    #Konvolüsyon işlemi, iki fonksiyonun integralinin her bir zaman noktasında çarpımının alınıp toplanmasıdır.
    #mode parametresi, konvolüsyon işlemi sırasında dizilerin kenarlarıyla nasıl davranılacağını kontrol eder.
    #'valid' modu, sadece tam olarak örtüşen kısımları içerir.

# Üstel Ağırlıklı Ortalama (EWA) alıcı filtre:
def üssel_ağırlıklı_ortalama_filtre(ses_verisi, alpha):
    ewa_sonucu = np.zeros(len(ses_verisi)) #ses verisiyle aynı boyutta sonuçların tutulacağı bir dizi (ewa_sonucu)oluşturur.
    ewa_sonucu[0] = ses_verisi[0] #: Sonuç dizisinin ilk elemanı, giriş ses verisinin ilk elemanına eşitlenir

    for i in range(1, len(ses_verisi)):
        önceki_değer = ewa_sonucu[i-1] #Her adımda, bir önceki ağırlıklı ortalama değeri alınır.
        güncel_değer = alpha * ses_verisi[i] + (1 - alpha) * önceki_değer #Yeni bir değer hesaplanır. Bu, giriş ses verisinin mevcut elem
        ewa_sonucu[i] = güncel_değer # Hesaplanan yeni değer, sonuç dizisine eklenir.

    return ewa_sonucu #Fonksiyon, sonuç olarak üssel ağırlıklı ortalama filtrelenmiş ses verisini içeren bir dizi döndürür.

    #ses verisi uzunlugu kadar tekrar edecek bir for döngüsü tanımlanmıştır ve döngüyle her bir zaman noktası için bir ağırlık faktörü ol
    #Her eleman, alpha değerinin sırasıyla artan kuvvetlerine sahiptir.

# Medyan (Ortanca) filtre:
def medyan_filtre(ses_verisi, kernel_size):
    return  medfilt(ses_verisi, kernel_size)

    #medfilt fonksiyonunu, bir sinyale Medyan Filtresi uygulamak için kullanılan hazır fonksiyon.
    #kernel_size parametresi,  pencere boyutunu belirler. Pencere boyutu, pencere etrafındaki değerlerin kaç adet olduğunu belirler.
    #Küçük pencere boyutu seçmek program açısından daha verimlidir ve bu nedenle en çok kullanılan pencere boyutu olan 3 değeri seçilmişt

 # Fourier Dönüşümü (FT)
def fourier_dönüşümü_plot(ses_verisi, sr, title, ax):
  #Ses verisi, örnekleme frekansı(sr), grafik başlığı(title) ve çizilecek eksen (ax) olmak üzere 4 parametre alır.
  #Verilen ses verisinin FFT (Hızlı Fourier Dönüşümü) hesaplar ve bu dönüşümü belirtilen çizim yüzeyine (axes) çizer.

    fft_spektrumu = np.abs(np.fft.fft(ses_verisi)) #np.fft.fft fonksiyonu, giriş ses verisinin FFT (Hızlı Fourier Dönüşümü) değerini hesa
    frekans = np.linspace(0, sr, len(fft_spektrumu)) #frekans değerlerini içeren bir dizi oluşturulur (frekans[]).
    ax.plot(frekans[:len(fft_spektrumu)//2], fft_spektrumu[:len(fft_spektrumu)//2])   #sadece pozitif frekansları içerir
    ax.set_title(title) #grafiğin çağrıldığı yerdeki başlığı(title) alır.
    ax.set_xlabel('Frekans (Hz)')
    ax.set_ylabel('Genlik')
    # ax.plot fonksiyonu, belirtilen x ve y değerleri üzerinde bir çizgi grafiği oluşturur.
    #x ekseni: frekans dizisi, y ekseni:fft_spektrumu (genlik)
    #FFT sonucu simetriktir ve negatif frekanslar pozitif frekanslarla aynıdır bu nedenle çizim sadece pozitif frekansları içerir.

# Örnek ses verilerinin (gerçek/sentetik) yüklenmesi:
gerçek_ses, sr= librosa.load('/content/linus-original-DEMO.mp3', sr=None) #kaggle platformundan elde edilmiştir.
sentetik_ses, _ = librosa.load('/content/linus-to-musk-DEMO.mp3', sr=sr)  # kaggle platformundan elde edilmiştir (Gerçek veri ile aynı ör
#Fonksiyon, yüklenen ses verisini ve örnekleme frekansını (sr) iki ayrı değişkene atar.
#Eğer sr=None olarak belirlenirse, orijinal örnekleme frekansı kullanılır.
#2. satırda, ses verisi, sentetik_ses değişkenine, örnekleme frekansı ise _ değişkenine atanır.
#Önceki satırdan gelen sr değeri kullanılarak, aynı örnekleme frekansında yükleme yapılır.

# Konvolüsyon işlemi gerçekleştime (örnek çekirdek: [1, -1])
çekirdek = np.array([1, -1])  # Bir fark çekirdeği oluşturulur.

konvolüsyonlu_gerçek_ses = fftconvolve(gerçek_ses, çekirdek, mode='same')
konvolüsyonlu_sentetik_ses = fftconvolve(sentetik_ses, çekirdek, mode='same')
#çekirdek: Fark çekirdeği olarak isimlendirilir.
#Bir sinyalde her örnek ile bir önceki örnek arasındaki farkı hesaplar. Kısaca, bir sinyalin türevini değişim oranını hesaplar.
#mode='same': Konvolüsyon işlemi sonucu elde edilen çıktı uzunluğunun, giriş sinyalinin uzunluğuyla aynı olmasını sağlar.
```

```
# Filtre Parametreleri (filtreleme etkisine göre istenildiği gibi ayarlanabilir)
M = 2  # M-nokta kayar ortalama için pencere boyutu
alpha = 0.3  # Üssel ağırlıklı ortalama için ağırlık
kernel_size = 3  # Medyan filtresi için çekirdek boyutu

""" başka değerler denenebilir:
M = 10
alpha = 0.9
kernel_size = 3
"""

# Filtrelerin gerçek ve sentetik ses verisine uygulanması:
#tanımlanmış fonksiyonlar burada çağrılır.
filtreli_sentetik_m_nokta = m_nokta_kayar_ortalama_filtre(sentetik_ses, M)
filtreli_gerçek_m_nokta = m_nokta_kayar_ortalama_filtre(gerçek_ses, M)

filtreli_sentetik_ewa = üssel_ağırlıklı_ortalama_filtre(sentetik_ses, alpha)
filtreli_gerçek_ewa = üssel_ağırlıklı_ortalama_filtre(gerçek_ses, alpha)

filtreli_sentetik_medyan = medyan_filtre(sentetik_ses, kernel_size)
filtreli_gerçek_medyan = medyan_filtre(gerçek_ses, kernel_size)

# Görselleştirme
fig, axs = plt.subplots(4, 3, figsize=(18, 15))
# Sonuçların çizileceği martisi ifade eder. 3 satır ve 3 sütundan oluşan bir alt grafik düzeni (3x3 matris) oluşturur.
#Bu, 3x3 bir matris içinde toplam 9 alt grafik (subplot) bulunacağı anlamına gelir.
#Sırasıyla aşağıdaki sinyallere ait grafikleri oluşturur. 10 tane grafik oluşturulacağı için 3x3 matris boyutu yetersiz kalır,
# bu nedenle kodda 4x3 olarak tanımlanmıştır. Son iki eleman için grafik olmadığı için boş grafik çizdirir.

# Orijinal ve Konvolüsyonlu Sinyaller
fourier_dönüşümü_plot(sentetik_ses, sr, "Orijinal Sentetik Ses FFT", axs[0, 0])
fourier_dönüşümü_plot(konvolüsyonlu_sentetik_ses, sr, "Konvolüsyonlu Sentetik Ses FFT", axs[0, 1])
fourier_dönüşümü_plot(gerçek_ses, sr, "Orijinal Gerçek Ses FFT", axs[1, 0])
fourier_dönüşümü_plot(konvolüsyonlu_gerçek_ses, sr, "Konvolüsyonlu Gerçek Ses FFT", axs[1, 1])

# M-nokta Filtrelenmiş Sinyaller
fourier_dönüşümü_plot(filtreli_sentetik_m_nokta, sr, "M-nokta Filtrelenmiş Sentetik Ses FFT", axs[0, 2])
fourier_dönüşümü_plot(filtreli_gerçek_m_nokta, sr, "M-nokta Filtrelenmiş Gerçek Ses FFT", axs[1, 2])

# EWA Filtrelenmiş Sinyaller
fourier_dönüşümü_plot(filtreli_sentetik_ewa, sr, "EWA Filtrelenmiş Sentetik Ses FFT", axs[2, 0])
fourier_dönüşümü_plot(filtreli_gerçek_ewa, sr, "EWA Filtrelenmiş Gerçek Ses FFT", axs[2, 1])

# Medyan Filtrelenmiş Sinyaller
fourier_dönüşümü_plot(filtreli_sentetik_medyan, sr, "Medyan Filtrelenmiş Sentetik Ses FFT", axs[2, 2])
fourier_dönüşümü_plot(filtreli_gerçek_medyan, sr, "Medyan Filtrelenmiş Gerçek Ses FFT", axs[3, 0])

plt.tight_layout()
plt.show()
#Bu iki komut genellikle bir arada kullanılır.
#plt.tight_layout() önce çağrılarak grafik düzeni ayarlanır, ardından plt.show() komutuyla grafikler ekranda görüntülenir.
```


Orijinal Sentetik Ses FFT
Konvolüsyonlu Sentetik Ses FFT
M-nokta Filtrelenmiş Sentetik Ses FFT
Orijinal Gerçek Ses FFT
Konvolüsyonlu Gerçek Ses FFT
M-nokta Filtrelenmiş Gerçek Ses FFT
EWA Filtrelenmiş Sentetik Ses FFT
EWA Filtrelenmiş Gerçek Ses FFT
Medyan Filtrelenmiş Sentetik Ses FFT
Medyan Filtrelenmiş Gerçek Ses FFT
Genlik
Frekans (Hz)