AYRIK ZAMANLI SİSTEMLERDE KONVOLÜSYON İŞLEMİ - ÖDEV RAPORU

Berkay Gözübüyük

21011044

Erkan Uslu

1. Amaç

Bu çalışmada, ayrık zamanlı iki işaretin konvolüsyon toplamı hesaplanmıştır. Konvolüsyon işlemi hem kullanıcı tarafından yazılan manuel bir fonksiyonla (myConv) hem de Python'un hazır fonksiyonlarıyla gerçekleştirilmiştir. Ayrıca ses verisi kaydedilerek, belirli sistemlerin (ör. FIR filtre) konvolüsyon etkisi analiz edilmiştir.

2. Kullanılan Araçlar

- **Programlama dili:** Python 3.11
- **Kütüphaneler:** numpy, matplotlib, sounddevice, scipy.io

3. Yöntem ve Adımlar

3.1 Konvolüsyon Fonksiyonunun Yazılması (myConv)

Kendi yazdığımız myConv fonksiyonu, klasik konvolüsyon formülüne uygun olarak iki ayrık zamanlı sinyali toplam yöntemiyle işler. Daha verimli çalışması için i + j endekslemeli optimize edilmiş versiyon kullanılmıştır:

```
def myConv(x, h):
    Lx = len(x)
    Lh = len(h)
    Lz = Lx + Lh - 1
    z = np.zeros(Lz)
    for i in range(Lx):
        for j in range(Lh):
            z[i + j] += x[i] * h[j]
    return z
```

3.2 Manuel ve Hazır Fonksiyon Sonuçlarının Karşılaştırılması

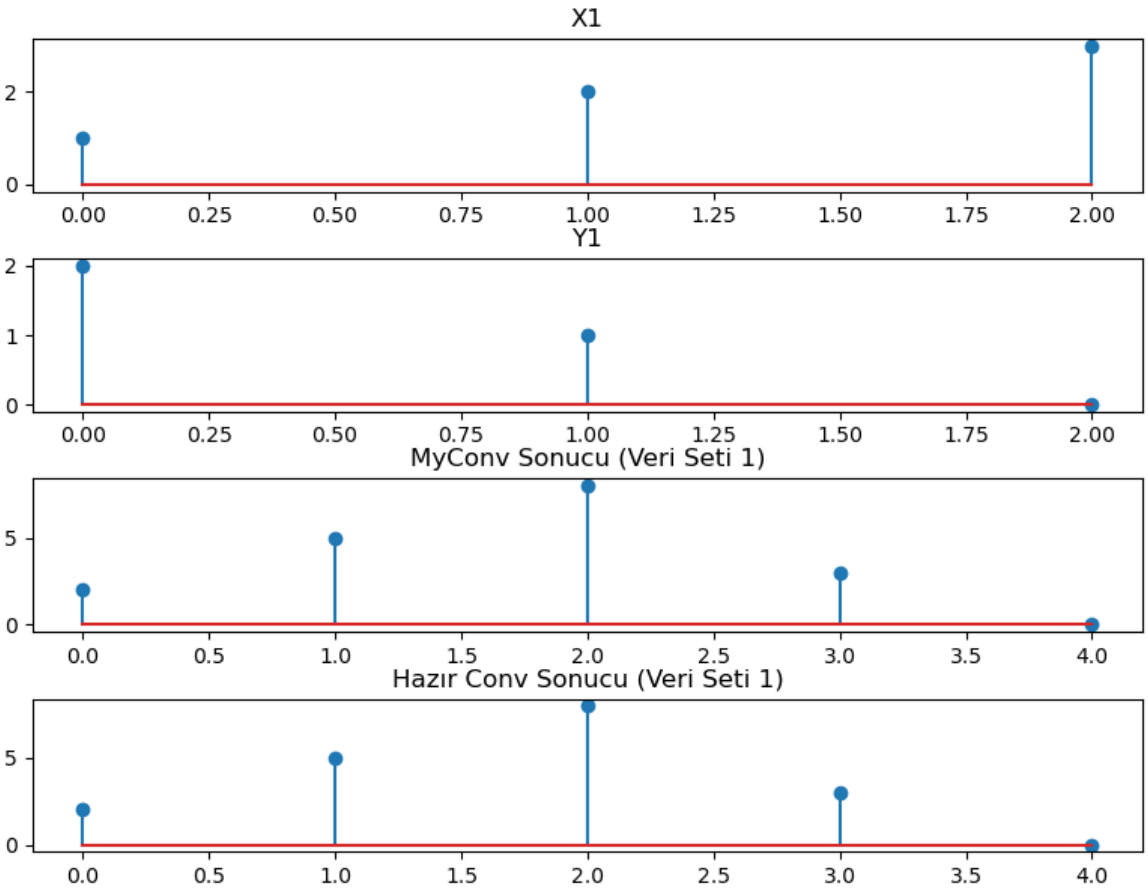
İki kısa işaret için hem myConv hem de np.convolve fonksiyonları uygulanmıştır. Sonuçlar hem sayısal hem de grafiksel olarak karşılaştırılmıştır. Çıktılar birebir örtüşmektedir.

Örnek:

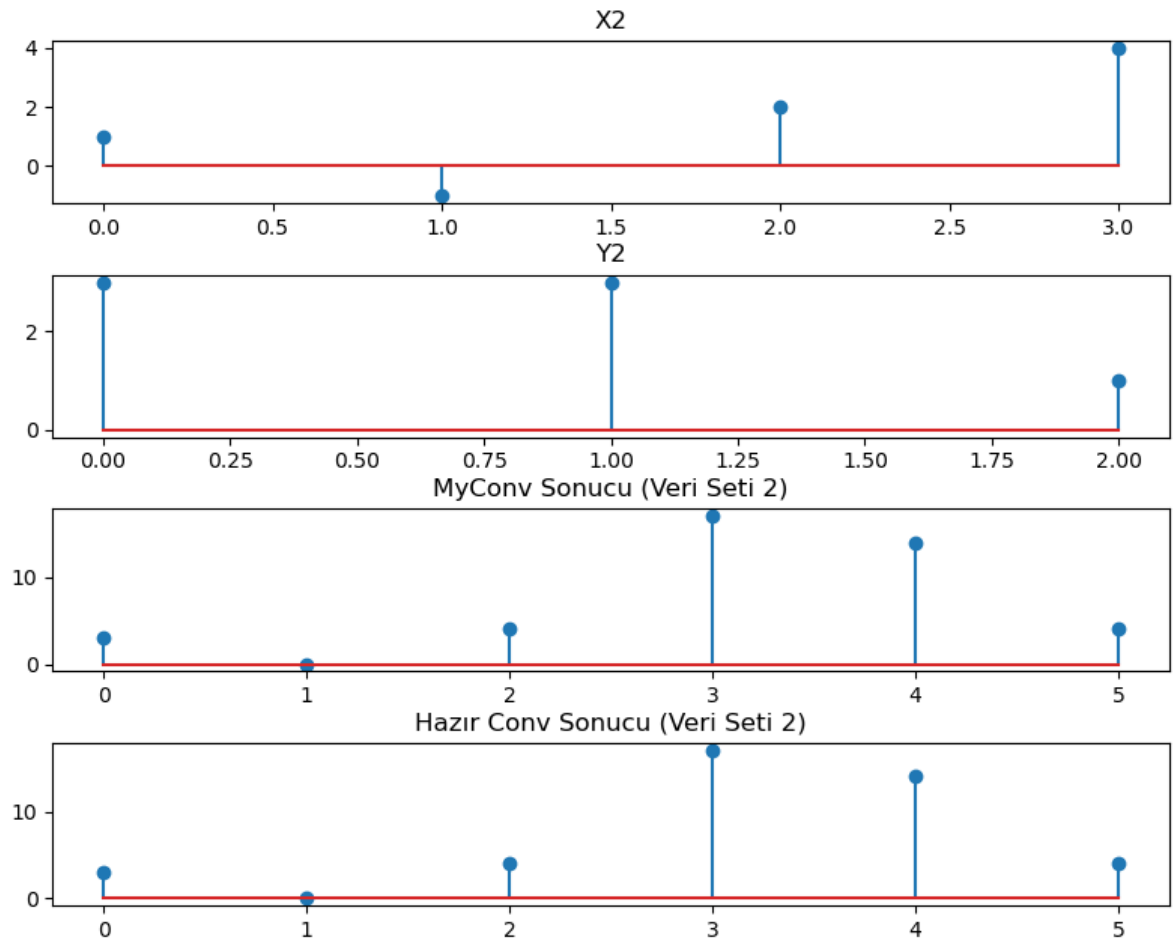
- x[n] = [1, 2, 3]
- y[n] = [2, 1, 0]
- Sonuç: [2, 5, 8, 3, 0]

Grafiksel olarak

```
X1 = [1, 2, 3]
Y1 = [2, 1, 0]
```



```
X2 = [1, -1, 2, 4]
Y2 = [3, 3, 1]
```



3.3 Ses Kaydı ve İşaretin Elde Edilmesi

Python'daki sounddevice kütüphanesi ile mikrofon kullanılarak 5 saniyelik ses kaydı alınmıştır:

```
def record_audio(duration=5, fs=16000):
    recording = sd.rec(int(duration * fs), samplerate=fs, channels=1, dtype='float32')
    sd.wait()
    return recording[:, 0]
```

Bu kayıt, daha sonra konvolüsyon işlemlerine girdi olarak kullanılmıştır.

3.4 Sistemin Tanımlanması (h[n])

Sistemin dürtü yanıtı aşağıdaki gibi tanımlanmıştır:

$$h[n] = \delta[n] + A \cdot \delta[n-1] + A^2 \cdot \delta[n-2] + \cdots + A^{M-1} \cdot \delta[n-(M-1)]$$

Bu ifade Python'da dizi olarak create_h() fonksiyonu ile oluşturulmuştur:

def create_h(M, A=0.5):

return np.array([A**i for i in range(M)])

M değerleri olarak 3, 4 ve 5 seçilmiştir. A sabiti 0.5 alınmıştır.

3.5 Konvolüsyon Uygulaması (Ses + Sistem)

5 saniyelik ses sinyali, her M değeri için myConv ve np.convolve fonksiyonlarıyla h[n] dizisiyle konvolüsyonlanmıştır.

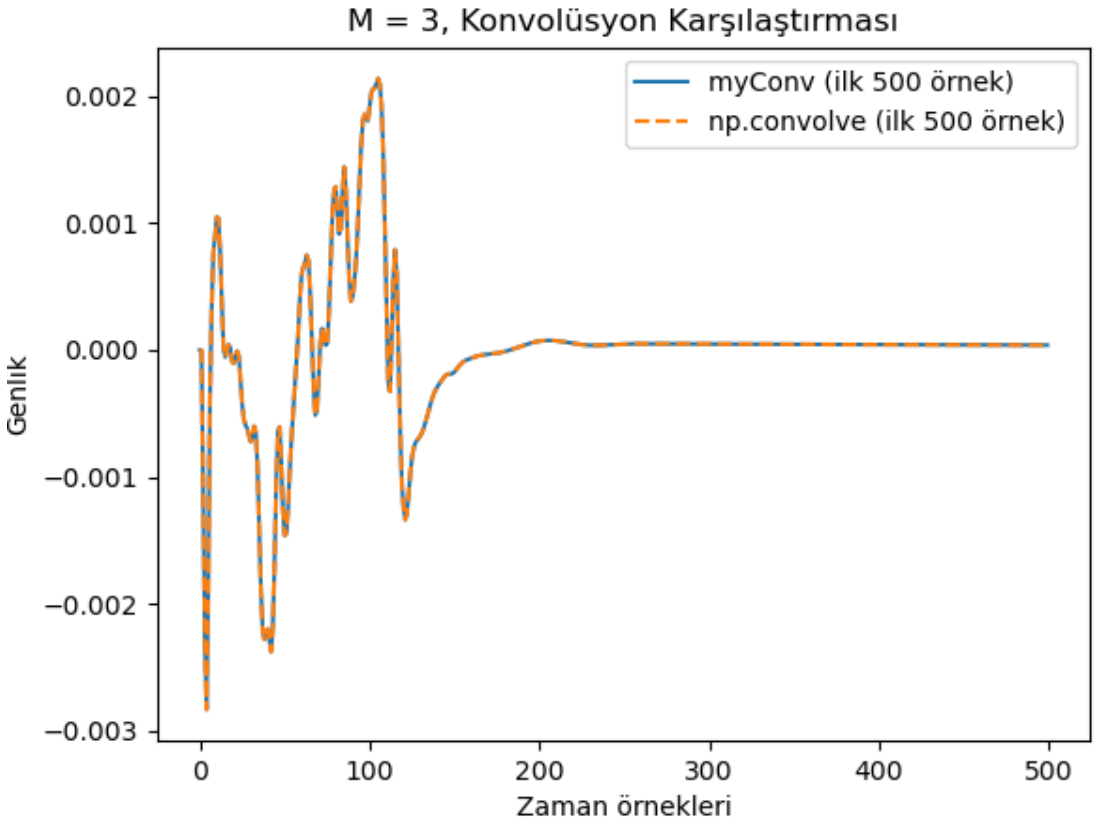
myY1 = myConv(X1, hM)

Y1 = np.convolve(X1, hM)

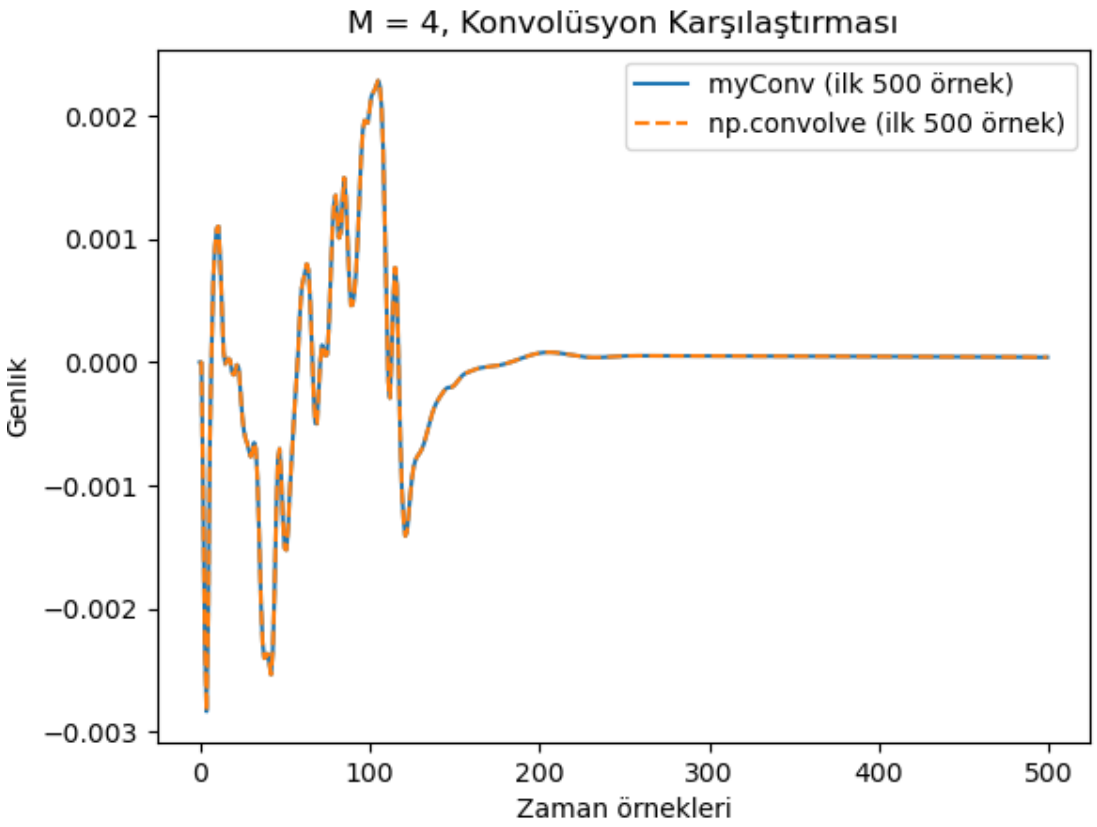
3.6 Sonuçların Karşılaştırılması

- Her iki konvolüsyon fonksiyonu aynı sonuçları üretmiştir.
- Sonuç sinyaller grafiksel olarak kıyaslanmıştır.
- Ek olarak, kullanıcı sesinin konvolüsyon sonrası sesini dinleyerek sistemin etkisi gözlemlenmiştir.
- M değeri arttıkça çıkış sinyali daha yumuşak ve "yankılı" bir hale gelmiştir. Bu, sistemin daha uzun süreli hafızaya sahip olduğunu gösterir.

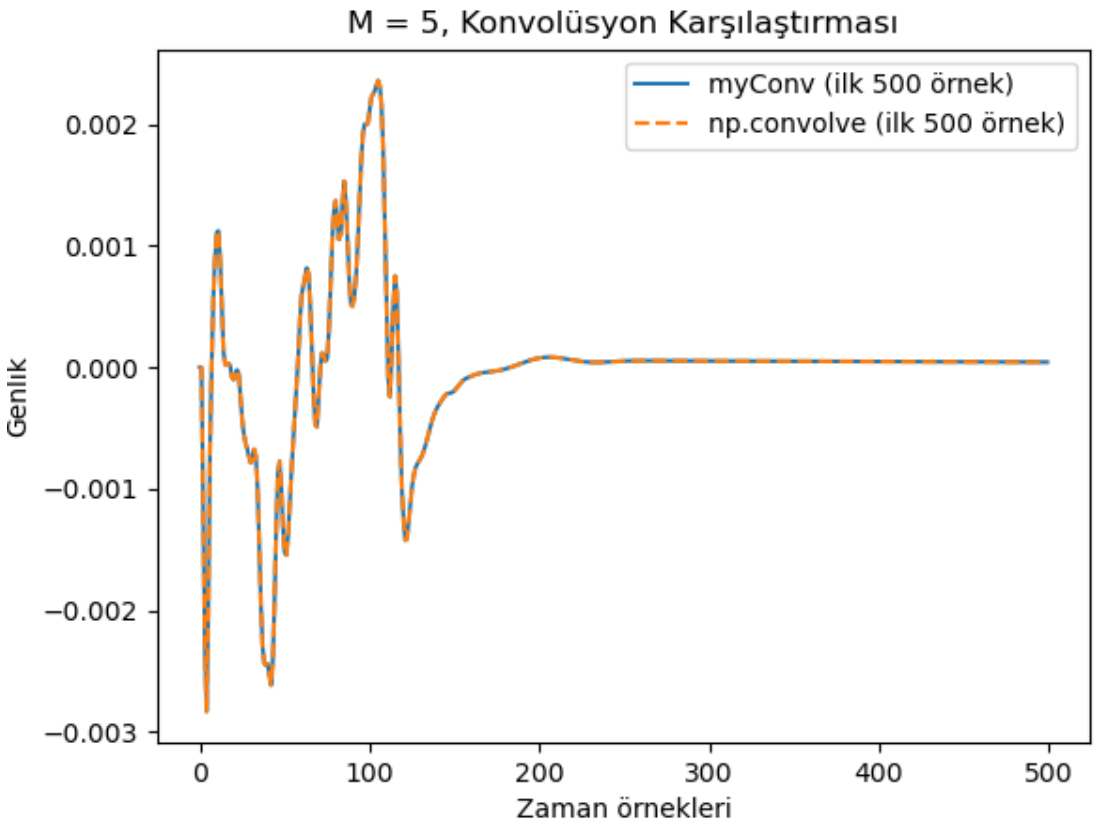
M = 3


M = 3, Konvolüsyon Karşılaştırması
myConv (ilk 500 örnek)
np.convolve (ilk 500 örnek)
Genlik
Zaman örnekleri
0.002
0.001
0.000
−0.001
−0.002
−0.003
0
100
200
300
400
500

M = 4


M = 4, Konvolüsyon Karşılaştırması
myConv (ilk 500 örnek)
np.convolve (ilk 500 örnek)
Genlik
Zaman örnekleri
0.002
0.001
0.000
−0.001
−0.002
−0.003
0
100
200
300
400
500

M = 5


M = 5, Konvolüsyon Karşılaştırması
myConv (ilk 500 örnek)
np.convolve (ilk 500 örnek)
Genlik
Zaman örnekleri
0.002
0.001
0.000
−0.001
−0.002
−0.003
0
100
200
300
400
500