KARADENİZ TEKNİK ÜNİVERSİTESİ
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

RAPOR

Korelasyon Tabanlı DFT

Ad-Soyad	Uğurcan YILMAZ
Numara	383242
Ders Sorumlusu	Doç.Dr. Sedat GÖRMÜŞ

İÇİNDEKİLER

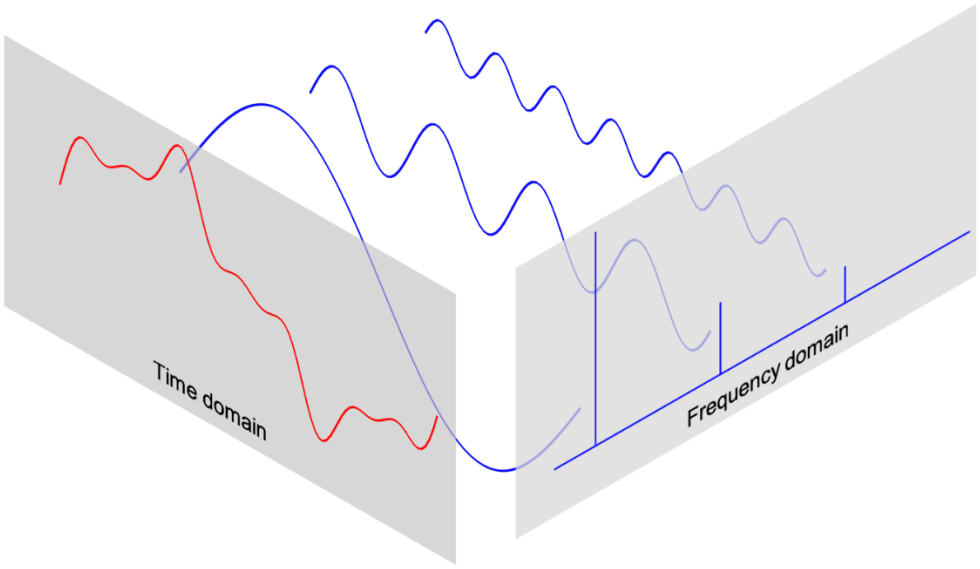
Giriş

Sinüs dalgası, birçok farklı alanda kullanılan temel bir sinyal türüdür. Ses dalgaları, radyo dalgaları ve hatta günlük yaşamda karşılaştığımız birçok dalga, temelde bir sinüs dalgasıdır ya da sinüs dalgalarının süperpozisyonu olarak görülebilir. Bu sebeple, bir sinüs dalgası oluşturmak ve bu dalgayı analiz etmek, sinyal işleme ve iletişim sistemleri açısından büyük önem taşır.

Örnekleme, sürekli bir sinyali belirli zaman aralıklarında ölçmek ve bu ölçümleri bir dizi sayı olarak kaydetme işlemidir. Bu işlem, sürekli zamanlı bir sinyali dijital bir sinyale dönüştürür, böylece sinyal üzerinde dijital sinyal işleme teknikleri uygulanabilir. *Ancak bu projede oluşturacağım sinüs dalgası da yine dijital bir dalga olacaktır. Yani oluşturduğum, dijital bir dalga olan, sinüs dalgasından örnekler alarak, basit bir ayrık sinüs dalgası üreteceğim. En başta direkt olarak ayrık sinüs dalgası da üretilebilirdi ancak orijinal sinüs dalgasının nasıl görüneceğini gözlemlemek için bu şekilde bir yol izledim.* Burada, örnekleme oranı, sinyalin bir saniye içinde kaç kez örneklenmesi gerektiğini belirler.

Örnekleme teorisine göre, bir sinyalin doğru bir şekilde yeniden oluşturulabilmesi için örnekleme frekansının, sinyalin en yüksek frekansının iki katı olması gerekmektedir. Bu örnekleme teorisine "Nyquist-Shannon Örnekleme Teorisi" [1] adı verilir. *Ancak yine bu örnekleme teorisini bu projede kullanmayacağım çünkü örnek sayısını 128 olarak belirlememiz istenmiş ve bu sebeple saniye başına alınacak örnek sayısını 128 olarak belirledim. Yani sinüs dalgası oluşturulurken her saniyede 128 örnek ekrana basılacak ve ayrık sinüs dalgası otomatikman orijinal sinüs dalgasının ilk saniyesindeki örnekleri içerecek.*

DFT (Discrete Fourier Transform – Ayrık Fourier Dönüşümü), bir sinyalin frekans spektrumunu hesaplamak için kullanılan bir matematiksel tekniktir. DFT, zaman serisi verilerini frekans bileşenlerine dönüştürür, böylece bir sinyalin hangi frekanslarda ne kadar enerjiye sahip olduğunu belirlememizi sağlar.



DFT'nin çıktısı genellikle kompleks sayılardan oluşur, her biri belirli bir frekansta sinyalin genliği ve faz açısını temsil eder. *Bu projede ayrık sinüs dalgasının değerleri üzerinde DFT uygulayıp çıkan gerçek ve sanal değerleri bir spektruma dönüştürdüm. Spektrumu ise ekrana basarak sinüs sinyalinin hangi frekans değerlerinde nasıl bir güce sahip olduğunu gösterdim.*

Sinüs Dalgası Oluşturma İşlemi

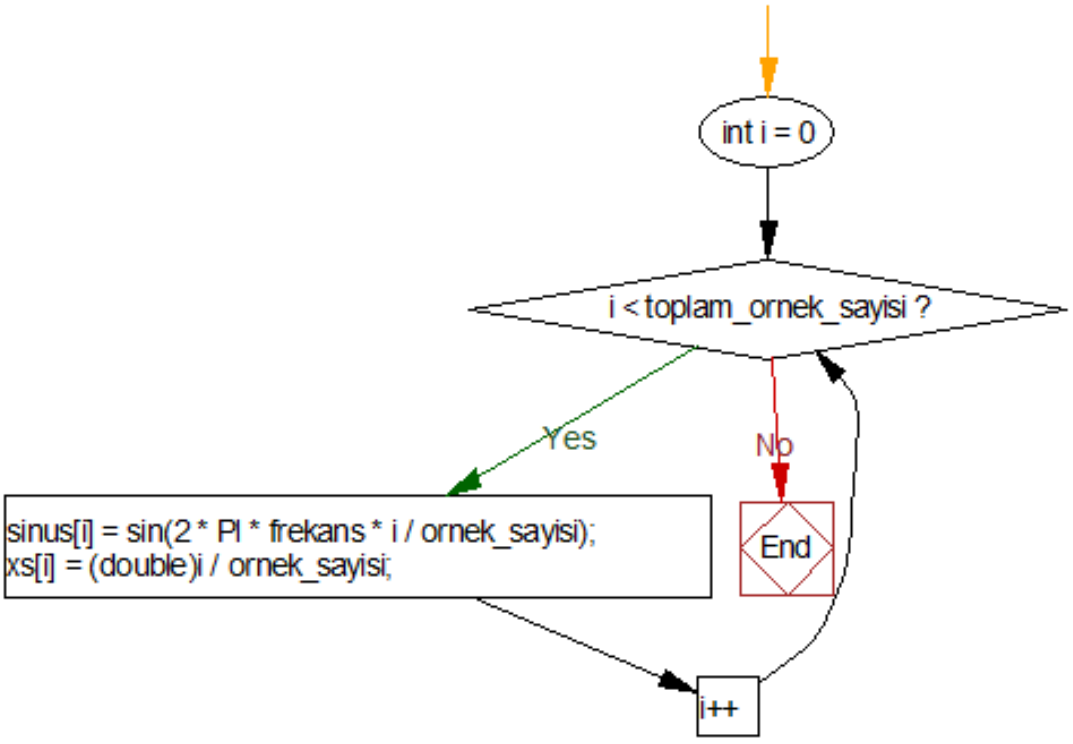
Sinüs dalgası, periyodik bir fonksiyondur ve genellikle aşağıdaki gibi bir formülle ifade edilir:[2]

$$y(t) = A\sin(2\pi f t + \varphi) = A\sin(\omega t + \varphi)$$

- ***A* → *Genlik***, *dalganın tepesinden çukuruna kadar olan düşey uzaklığın yarısı.*
- ***f* → *Frekans***, *her saniye meydana gelen salınımların (döngülerin) sayısı.*
- ***t* → *Zaman (saniye).***
- ***ω* → *Açısal Frekans***, *birim zamandaki radyan sayısı.*
- ***φ* → *Faz***, *salınımın kendi döngüsünde t=0'da olduğu yeri (radyan cinsinden) belirtir.*

Programın sinüs dalgası oluşturma kısmında yukarıdaki formülden, sinüs dalgasının genel formülünden, yararlandım. Faz değerini sıfır olarak kabul ettim. *Buradaki t yani zaman değerini ise dijital bir dalga olacağı için index olarak kabul ettim*. Dalganın zamanının doğru bir şekilde gösterilmesi için ise *her değeri saniyedeki örnek sayısına böldüm.* Bu şekilde dijital bir sinüs dalgası üretip görüntüde gerçek sinüs dalgasına benzer bir hale getirdim.

Sinüs dalgası oluşturmak için yazdığım kodun akış diyagramı aşağıdadır:



Sinüs dalgası oluşturmak için yazdığım kod bloğu aşağıdadır:

```
void sinus_olustur(int toplam_ornek_sayisi, double *sinus, double
frekans)
{
    double xs[toplam_ornek_sayisi];

    for (int i = 0; i < toplam_ornek_sayisi; i++)
    {
        sinus[i] = sin(2 * PI * frekans * i / ornek_sayisi);
        xs[i] = (double)i / ornek_sayisi;
    }

    orijinal_sinus_ciz(xs, sinus, toplam_ornek_sayisi);
}
```

Görüldüğü gibi yukarıda, sinüs dalgasının genel formülünün biraz farklı ve modifiye edilmiş halini yazdım.

Yukarıdaki “sinüs_olustur()” fonksiyonunu main fonksiyonundan aşağıdaki şekilde çağırıyorum:

```
    int toplam_ornek_sayisi = zaman * ornek_sayisi;
    double *sinus = malloc(toplam_ornek_sayisi * sizeof(double));
    sinus_olustur(toplam_ornek_sayisi, sinus, frekans);
```

Buradaki “toplam_ornek_sayisi” değişkeni, 128 olarak belirlenmiş olan “ornek_sayisi” değişkeninin programa ikinci argüman olarak verilen “zaman” değişkeniyle çarpılması sonucu oluşur. Bunun sebebi ise komut satırından frekansı ve zamanı girilen sinüs dalgasını doğru bir şekilde çizebilmektir.

“sinüs_olustur” fonksiyonunun sonunda çağrılmış olan “orijinal_sinus_ciz” fonksiyonu, pbPlots kütüphanesini kullanarak oluşturulan sinüs dalgasını resim dosyası (orijinal_sinus_dalgasi.png) olarak kaydeder.

Aşağıda frekansı 1 Hz olan ve 4 saniye süren bir sinüs dalgası oluşturulmuştur:

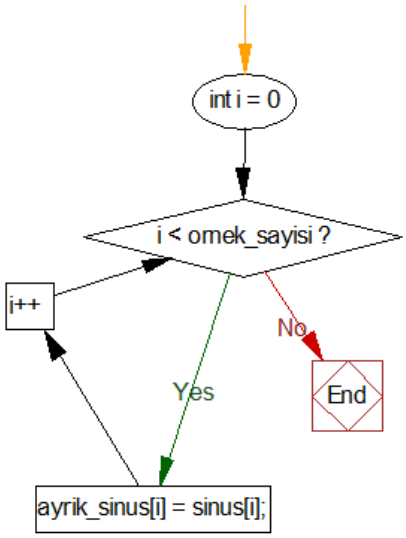
Sinüs Dalgasından Örnek Alma İşlemi

Önceki kısımda oluşturulan sinüs dalgasının ilk 128 örneğini alıp yeni bir ayrık sinüs dalgası oluşturdum. Ayrık sinüs dalgası oluşturma işlemini aşağıdaki formül ile yaptım.

$$A_i = X_i \, , i = 0, \dots , N - 1$$

- $\boldsymbol{A \rightarrow Ayrık\ sinüs\ dalgası}$
- $\boldsymbol{X \rightarrow Orijinal\ sinüs\ dalgası}$
- $\boldsymbol{i \rightarrow indeks}$
- $\boldsymbol{N \rightarrow Örnek\ sayısı}$

Ayrık sinüs dalgası oluşturmak için yazdığım kodun akış diyagramı aşağıdaki şekildedir:



Yukarıdaki akış diyagramına denk gelen kod ise aşağıdaki gibidir:

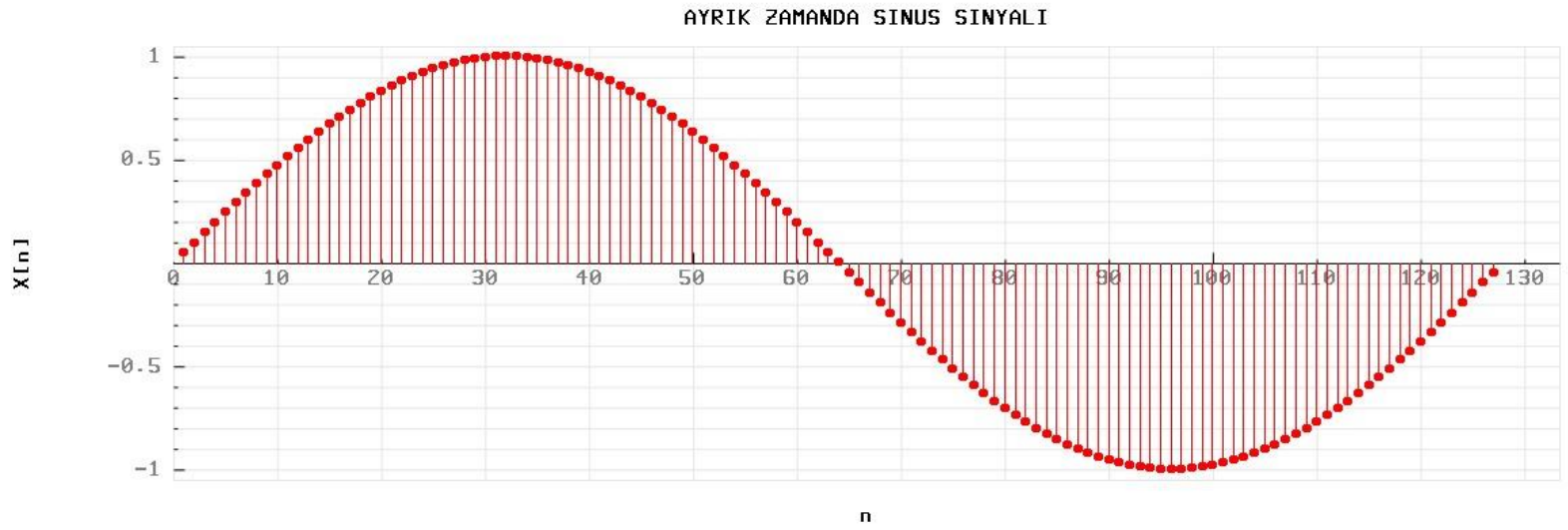
```
void ornek_al(double *sinus, double *ayrik_sinus)
{
    for (int i = 0; i < ornek_sayisi; i++)
    {
        ayrik_sinus[i] = sinus[i];
        printf("Ornek %d: %f\n", i, ayrik_sinus[i]);
    }

    ayrik_sinus_ciz(ornek_sayisi, ayrik_sinus);

    printf("------------------------------\n");
}
```

Main fonksiyonundan aşağıdaki şekilde bu fonksiyonu çağırıyorum:

```
    double *ayrik_sinus = malloc(ornek_sayisi * sizeof(double));
    ornek_al(sinus, ayrik_sinus);
```

Fonksiyon içerisinden çağrılan “ayrik_sinus_ciz()” fonksiyonu ile oluşturulan ayrık sinüs dalgasını “ayrik_sinus_dalgasi.png” olarak dizine kaydediliyor.

Aşağıda, ilk bölümde oluşturduğumuz frekansı 1 Hz olan ve 4 saniye süren sinüs dalgasının ilk 128 örneği alınarak oluşturulmuş ayrık sinüs dalgasını görebilirsiniz:



Ayrık Sinüs Dalgasına DFT Uygulama İşlemi

DFT (Discrete Fourier Transform – Ayrık Fourier Dönüşümü), bir sinyalin frekans spektrumunu elde etmek için kullanılan bir tekniktir. DFT, zaman domaininde olan bir sinyali frekans domainine çevirir. Bu, sinyal içerisinde hangi frekansların mevcut olduğunu ve bu frekansların ne kadar güçlü olduğunu belirlememizi sağlar.

DFT'nin formülü aşağıdaki gibidir:

$$X_k = \sum_{n=0}^{N-1} A_n \times e^{-i2\pi kn/N} \ , k = 0, \dots ,127$$

DFT formülündeki $e^{-i.x}$ kısmı Euler Formülü kullanılarak aşağıdaki şekilde genişletilebilir:

$$e^{i.x} = \cos x + i \times \sin x$$

Bu dönüşümü kullanarak DFT genel formülünü, C dilinde kolay bir şekilde hesaplayabilmek için, aşağıdaki şekilde dönüştürebiliriz:

$$X_k = \sum_{n=0}^{N-1} A_n \times [\cos(2\pi kn/N) - i \times \sin(2\pi kn/N)]$$

- $\boldsymbol{N}$ → *örnek sayısı*
- $\boldsymbol{n}$ → *şimdiki örnek*
- $\boldsymbol{k}$ → *şimdiki frekans*
- $\boldsymbol{A_n}$ → *sinüs dalgasının n'inci örneği*
- $\boldsymbol{X_k}$ → *Hem genlik hem de faz bilgisini içeren DFT*

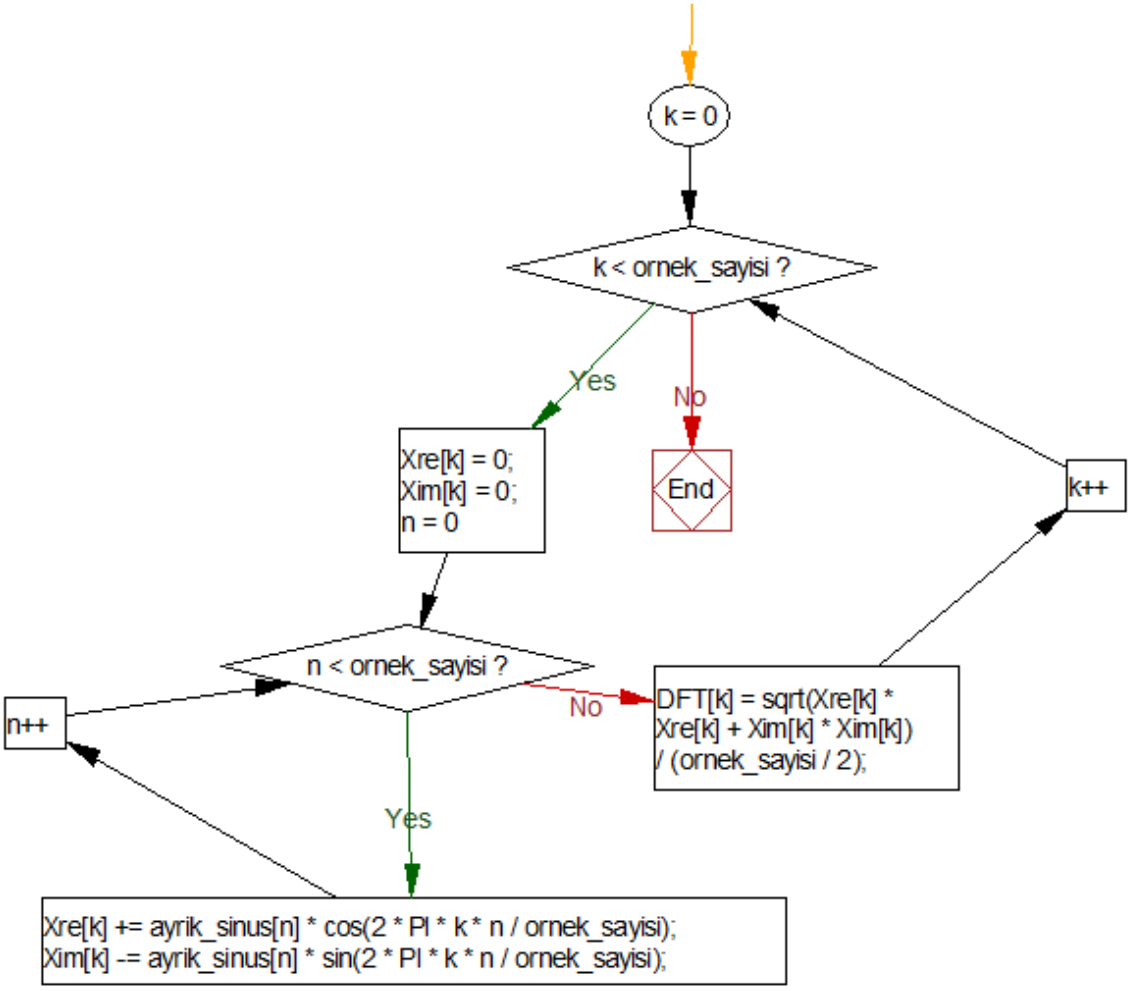
Yukarıdaki formülü kullanarak istediğimiz sayıda noktalı bir Fourier dönüşümü yapabiliriz.

DFT dizisini oluşturduktan sonra spektrum oluşturmak için her bir karmaşık DFT değerinin büyüklüğünü aşağıdaki formül ile hesaplamamız gerekir. Değerleri 2 ile çarpmamızın sebebi ise zaman alanındaki ayrık sinüs dalgamızın genliğini doğru bir şekilde DFT spektrumunda gösterebilmektir. Yani normalize ediyoruz. [3]

$$mag = \frac{2 \times |X_k|}{N} = \frac{2 \times \sqrt{Re(X_k)^2 + Im(X_k)^2}}{N}$$

- $\boldsymbol{mag}$ → *magnitude, büyüklük*
- $\boldsymbol{X_k}$ → Hem genlik hem de faz bilgisini içeren DFT
- $\boldsymbol{N}$ → *örnek sayısı*
- $\boldsymbol{Re(X_k)}$ → *DFT'nin gerçek kısmı*
- $\boldsymbol{Im(X_k)}$ → *DFT'nin sanal kısmı*

DFT hesaplamak için yazdığım fonksiyonun akış diyagramı aşağıdadır:



Yukarıdaki akış diyagramına denk gelen fonksiyon ise aşağıdadır:

```
void DFT_hesapla(double *ayrik_sinus, double *DFT)
{
    int k, n;
    double Xre[ornek_sayisi], Xim[ornek_sayisi];

    for (k = 0; k < ornek_sayisi; k++)
    {
        Xre[k] = 0;
        Xim[k] = 0;
        for (n = 0; n < ornek_sayisi; n++)
        {
            Xre[k] += ayrik_sinus[n] * cos(2 * PI * k * n
                                           / ornek_sayisi);
```

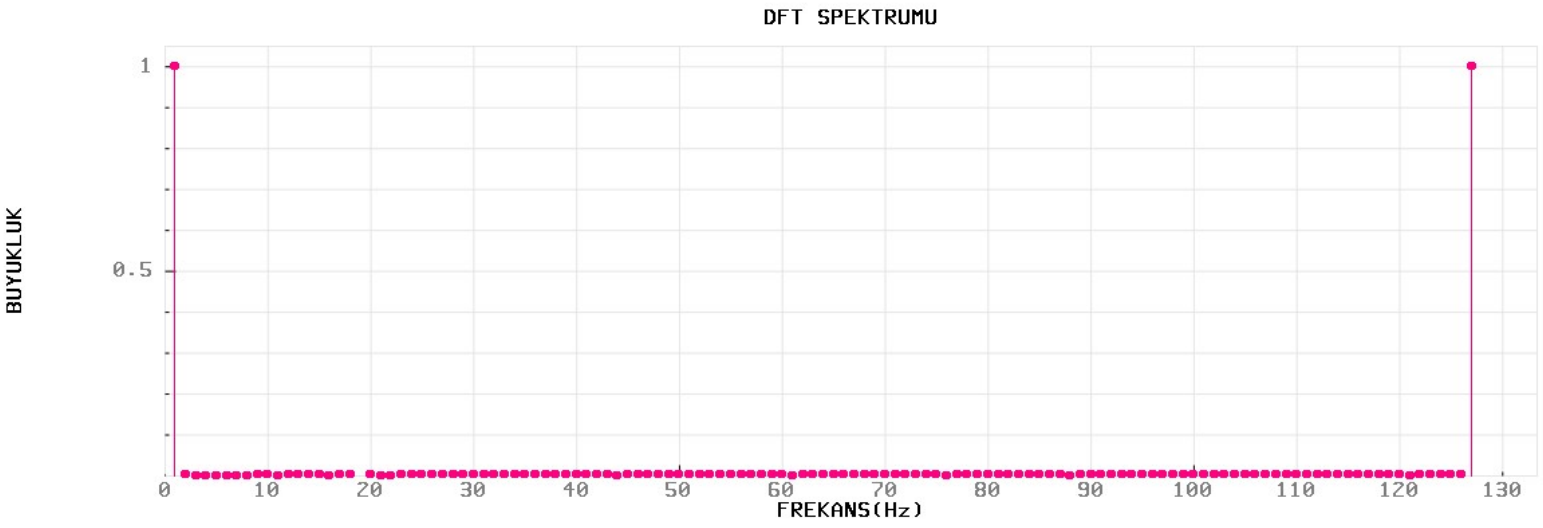
```
            Xim[k] -= ayrik_sinus[n] * sin(2 * PI * k * n
                                            / ornek_sayisi);
        }
        DFT[k] = sqrt(Xre[k] * Xre[k] + Xim[k] * Xim[k])
                    / (ornek_sayisi / 2);
        printf("DFT[%d] = %.2f %.2fi --> %.2f\n",
                    k, Xre[k], Xim[k], DFT[k]);
    }

    DFT_ciz(ornek_sayisi, DFT);

    printf("-----------------------------\n");
}
```

Bu fonksiyonu, main fonksiyonu içerisinden aşağıdaki şekilde çağırıyorum:

```
    double *DFT = malloc(ornek_sayisi * sizeof(double));
    DFT_hesapla(ayrik_sinus, DFT);
```

“DFT_ciz()” fonksiyonu ile ise oluşturulan DFT spektrumunu dizine “dft_spektrumu.png” ismiyle resim olarak kaydediyorum.

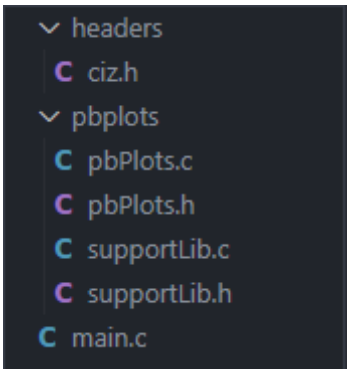
Aşağıdaki şekilde frekansı 1 Hz olan 128 örnekli bir sinüs dalgasının DFT spektrumunun nasıl oluştuğunu görebilirsiniz:



Program Hakkında Bilgiler

- Programdaki grafikleri çizebilmek için pbplots kütüphanesine [4] ihtiyacımız var. Kütüphaneyi kullanmak için herhangi bir şey yüklemeye gerek yok. Sadece programın bulunduğu dizinde kütüphanenin C dosyalarının bulunması yeterlidir. Ben dosya içine ekleyeceğim. Herhangi başka bir şey indirilmesine gerek yoktur.

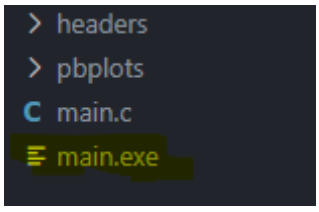
- Dosyalarımız aşağıdaki şekilde görüldüğü gibi olmalıdır:



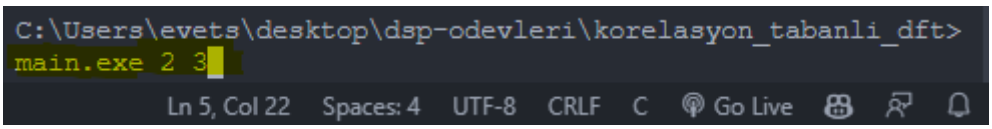
- Programı, herhangi bir C programlama dili derleyicisi kullanarak aşağıdaki şekilde göründüğü gibi derlememiz gerekiyor:

```
C:\Users\evets\desktop\dsp-odevleri\korelasyon_tabanli_dft>
gcc -o main main.c pbplots/pbPlots.c pbplots/supportLib.c
```

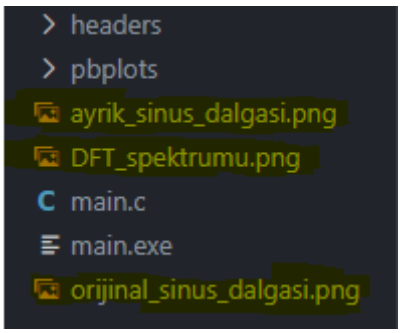
- Programı derledikten sonra aşağıdaki şekilde <main.exe> dosyamız oluşturuluyor:



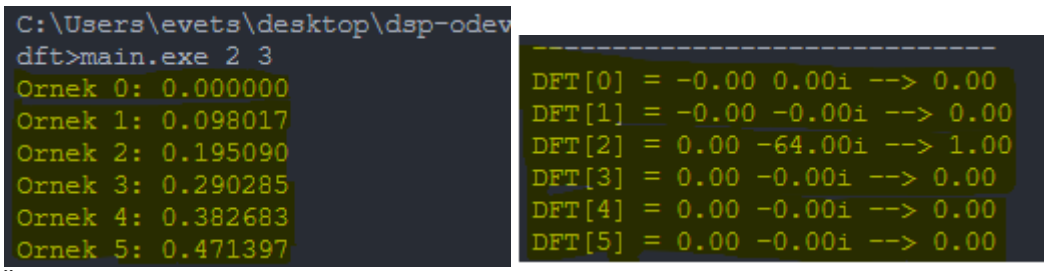
❖ Daha sonra yine terminalden <main.exe> dosyamızı çalıştırmak için aşağıdaki şekilde iki argüman vermemiz gerekiyor. İlk argüman oluşturmak istediğimiz sinüs dalgasının frekansı (Hz), ikinci argüman ise sinüs dalgasının kaç saniye süreceğidir:



❖ Programı çalıştırdıktan sonra dosya dizininde aşağıdaki gibi 3 tane .png dosyası eklenmesi gerekiyor.
"orijinal_sinus_dalgasi.png" oluşturulan orijinal sinüs dalgasının grafiğini, "ayrik_sinus_dalgasi.png" orijinal sinüs dalgasından örnek alınarak oluşturulan ayrık sinüs dalgasının grafiğini ve son olarak "DFT_spektrumu.png" ise ayrık sinüs dalgasının değerlerine DFT uygulandıktan sonra oluşan spektrumun grafiğini bize gösterecektir:

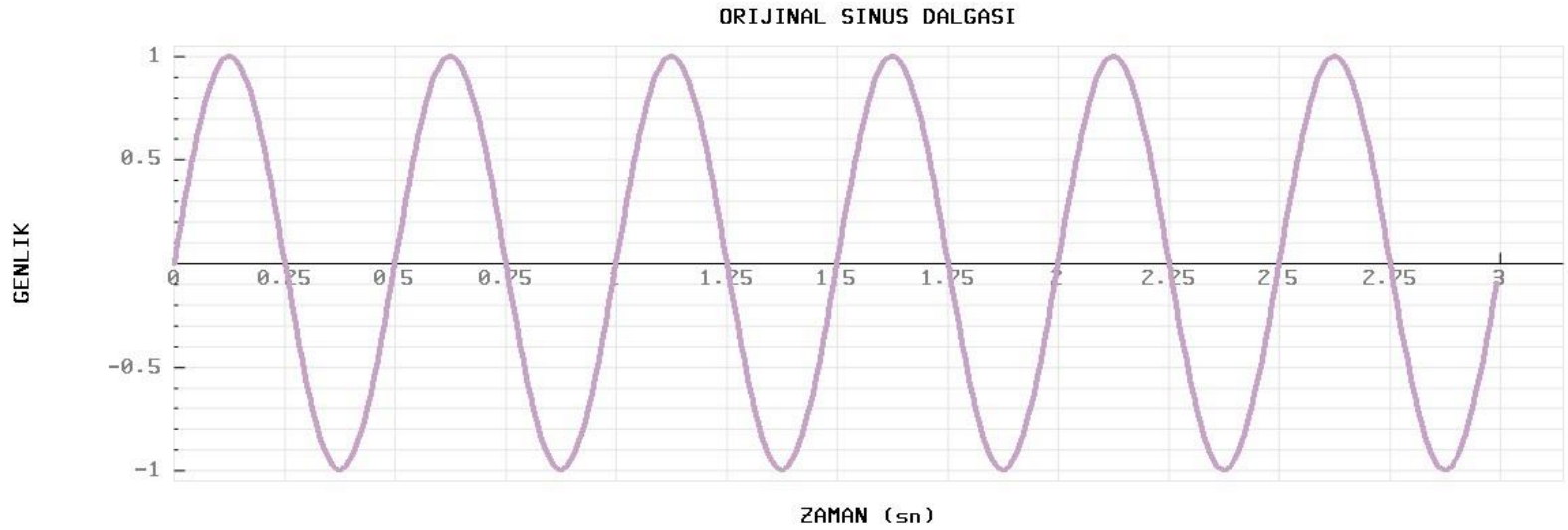


❖ Programın koşulduğu terminalde ise oluşturulan ayrık sinüs dalgasının tüm örneklerinin değerleri ve hesaplanan 128 noktalı DFT'nin tüm değerlerinin gerçek ve sanal kısımları ile büyüklük değerleri aşağıdaki gibi basılacaktır:

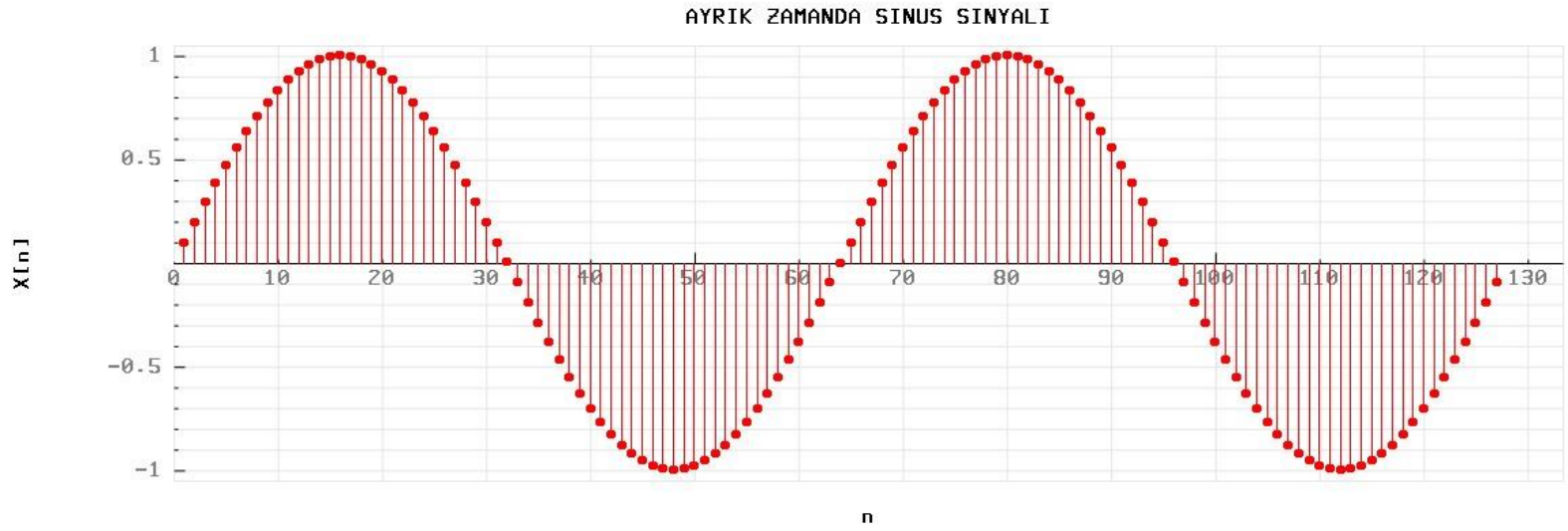


(Örnek olması açısından ilk 6 değerin resimlerini koydum. Terminalde tüm değerler basılacaktır.)

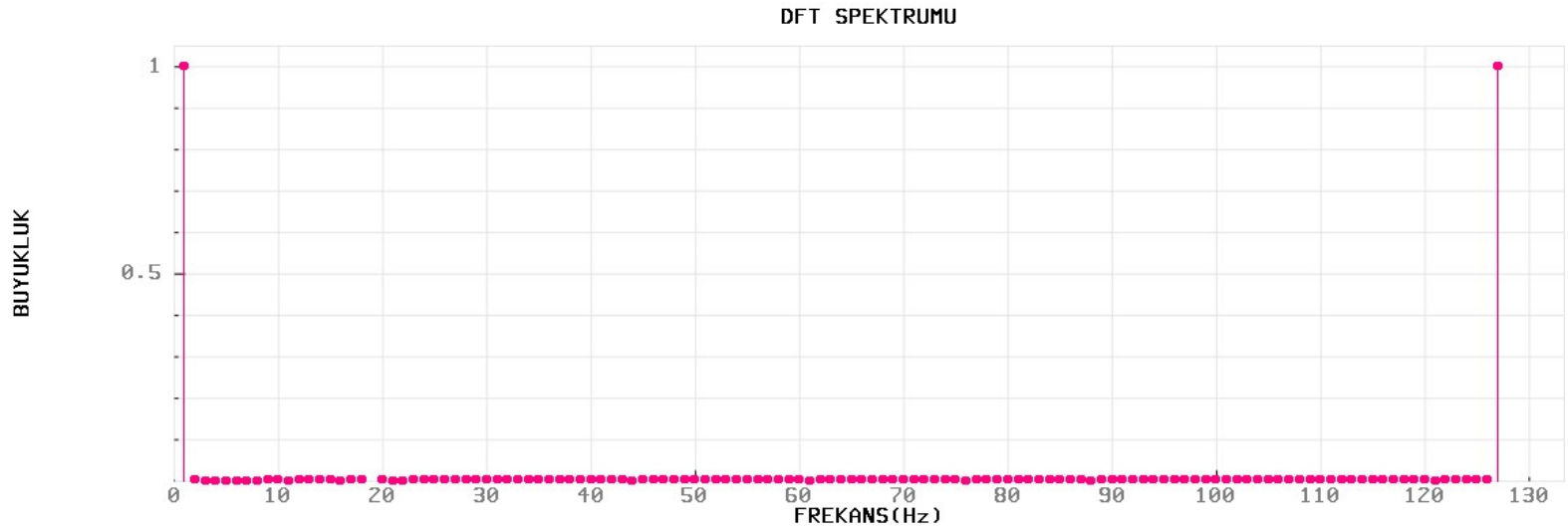
❖ Frekansı 2 Hz olan ve 3 saniye süren orijinal sinüs dalgasinin grafiği:



❖ Ayrık sinüs dalgasının grafiği:



❖ DFT spektrumunun grafiği:



Kaynakça

1. Nyquist–Shannon sampling theorem
2. Sine wave
3. Discrete Fourier Transform (DFT), Python Numerical Methods, chapter 24.2
4. Pbplots library, Martin F. Johansen, Apr 15 2020