801.526 Asterosismoloji

Ders 4 :
Asterosismolojide Veri Analizi II

Konvolüsyon

- f ve g fonksiyonlarının **konvolüsyonu** aşağıdaki şekilde tanımlanır:

$$(f * g)(x) := \int_{u=-\infty}^{\infty} f(u)g(x-u)\,du$$

- Eğer fonksiyonlar süreksizse;

$$(f * g)[n] := \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

- **Örnek 1:** Her yerde integrali alınabilir hangi bir f(x) fonksiyonu ile aşağıdaki şekilde tanımlı bir $a_h(x)$ fonksiyonunun konvolüsyonunu hesaplamaya çalışalım.

$$a_h(x) := \begin{cases} 1/2h & \text{if } -h < x < h \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned}(f * a_h)(x) &= \int_{-\infty}^{\infty} f(u)a_h(x-u)\,du \\ &= \frac{1}{2h}\int_{-h<x-u<h} f(u)\,du \\ &= \frac{1}{2h}\int_{x-h}^{x+h} f(u)\,du,\end{aligned}$$

Konvolüsyon

- Süreksiz fonksiyonlar için konvolüsyon tanımını bir liste, dizi ya da küme şeklinde verilen iki süreksiz fonksiyona uygulayacak olursak;

$$(f * g)[n] := \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$(f * g)[n] = f[0] \cdot g[n] + f[1] \cdot g[n-1] + f[2] \cdot g[n-2] + \cdots + f[N-1] \cdot g[n-(N-1)].$$

- **Örnek 2:** Aşağıda verilen iki fonksiyonun konvolüsyonunu hesaplayalım.

$$f = (f[0], f[1], f[2], f[3]) := (3, 1, 4, 1),$$
$$g = (g[0], g[1], g[2], g[3]) := (5, 9, 2, 6).$$

(f * g)[0] = f(0).g(0) = 3 x 5 = 15

(f * g)[1] = f(0).g(1) + f(1).g(0) = 3 x 9 + 1 x 5 = 32

(f * g)[2] = f(0).g(2) + f(1).g(1) + f(2).g(0) = 3 x 2 + 1 x 9 + 4 x 5 = 35

(f * g)[3] = f(0).g(3) + f(1).g(2) + f(2).g(1) + f(3).g(0) = 3 x 6 + 1 x 2 + 4 x 9 + 1 x 5 = 61

(f * g)[4] = f(1).g(3) + f(2).g(2) + f(3).g(1) = 1 x 6 + 4 x 2 + 1 x 9 = 23

(f * g)[5] = f(2).g(3) + f(3).g(2) = 4 x 6 + 1 x 2= 26

(f * g)[6] = f(3).g(3) = 1 x 6= 6

Konvolüsyon Stratejileri

f * g = [15, 32, 35, 61, 23, 26, 6] (tam kaydırma)

f * g = [32, 35, 61, 23] (f ve g ile aynı uzunlukta)

f * g = [61] (f ve g'nin bütün elemanlarının kullanıldığı)

Python'da Konvolüsyon

- ✔ Aynı soruyu farklı konvolüsyon stratejileri lle Python `numpy` modülü fonksiyonlarından `convolve` ile çözelim (konvolusyon.py).

- ✔ Öncelikle g'nin ters çevrildikten sonra f üzerine baştan sona kadar kaydırıldığı, yani her iki fonksiyonun üstüste geldiği her noktada konvolüsyon işleminin uygulandığı ve sonuç olarak `n+m-1` (örneğimizde 4 + 4 – 1 = 7) uzunluğunda bir dizi döndüren "full" stratejisini görelim.

```
import numpy as np
f = np.array([3,1,4,1])
g = np.array([5,9,2,6])
f_konv_g = np.convolve(f,g, mode="full")
print("f*g (full) = ", f_konv_g)
```

```
In [1]: run konvolusyon.py
f*g (full) =  [15 32 35 61 23 26  6]
```

- ✔ Bu bir önceki slaytta verilen çözüm ile aynıdır. Eğer konvolüsyon işleminin sonuçta, konvolüsyona tabi tutulan her iki fonksiyondan (diziden) uzun olanının uzunluğuna eşit (`max(m,n)`) bir dizi isteniyorsa bu kez "same" stratejisi kullanılır. Konvolüsyon her iki fonksiyonun tamamen örtüştüğü dizi elemanları için uygulanmak isteniyorsa bu kez de "valid" stratejisi kullanılır.

```
f_konv_g = np.convolve(f,g, mode="same")
print("f*g (same) = ", f_konv_g)
f_konv_g = np.convolve(f,g, mode="valid")
print("f*g (valid) = ", f_konv_g)
```

```
In [2]: run konvolusyon.py
f*g (same) =  [32 35 61 23]
f*g (valid) =  [61]
```

Konvolüsyon

- ✔ Konvolüsyon işleminde integralin sınırlarını belirleme konusu önemli bir konudur. Aşağıdaki örneği bu amaçla inceleyelim.

$$f(x) := \begin{cases} 1 & \text{if } 0 < x < 1 \\ 0 & \text{otherwise,} \end{cases} \qquad g(x) := \begin{cases} x & \text{if } 0 < x < 2 \\ 0 & \text{otherwise.} \end{cases}$$

- ✔ Böyle bir durumda integrasyon sınırları belirlenirken her iki fonksiyonunun sınırlarını dikkate almak gerekir. Örneğimizde sınırlar aşağıdaki şekilde belirlenmiştir.

$$(f * g)(x) = \int_{\substack{0<u<1 \\ \text{and} \\ 0<x-u<2}} f(u)g(x-u)\,du = \int_{\min\{1,\max\{0,x-2\}\}}^{\max\{0,\min\{1,x\}\}} (x-u)\,du.$$

- ✔ İntegrasyon sınırlarını ve parçalı fonksiyon olarak tanımlanmış fonksiyonlar için bu sınırlar arasında hangi ifadenin geçerli olduğunu belirlemek üzere daha iyi bir yöntem f(u) ve g(x-u) fonksiyonlarının birer grafiğini çizip aşağıdaki işlemleri uygulamaktır.

1) Katlama (ing. folding): g(u) fonksiyonunun y eksenine göre simetriği alınır (g(-u))

2) Kaydırma (ing. displacement): g(-u) fonksiyonu x ekseninde x kadar kaydırılır (g(x-u))

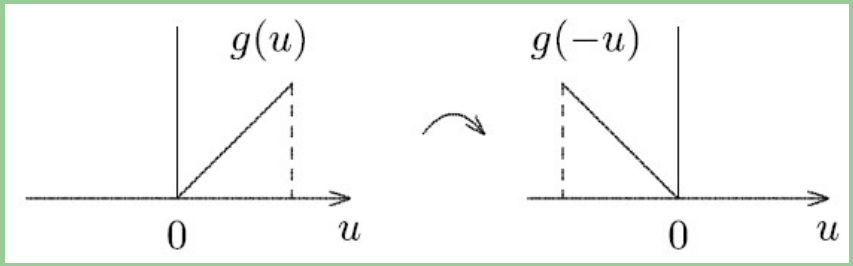
2) Çarpma (ing. multiplication): f(u) ile g(x-u) fonksiyonu çarpılır (f(u).g(x-u))

4) İntegrasyon (ing. integration): Bu işlem bütün x'ler için yapılarak f(u).g(x-u) altında kalan alan hesaplanır.

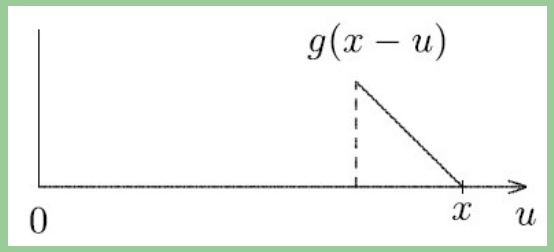
Konvolüsyon

- ✔ Örneğimizdeki fonksiyonlar için bu şemayı adım adım uygulayalım.

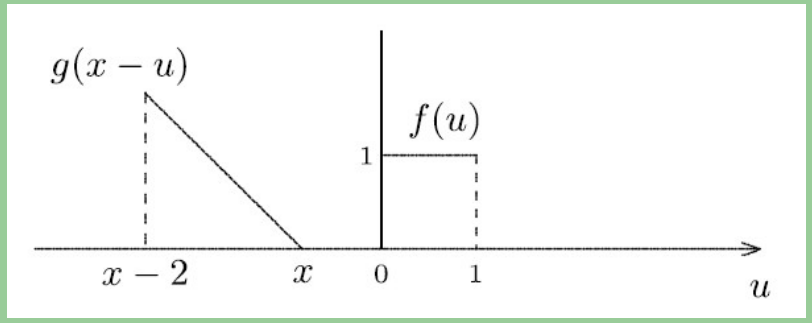
1) Katlama (ing. folding): g(u) fonksiyonunun y eksenine göre simetriği alınır (g(-u))



2) Kaydırma (ing. displacement): g(-u) fonksiyonu x ekseninde x kadar kaydırılır (g(x-u))



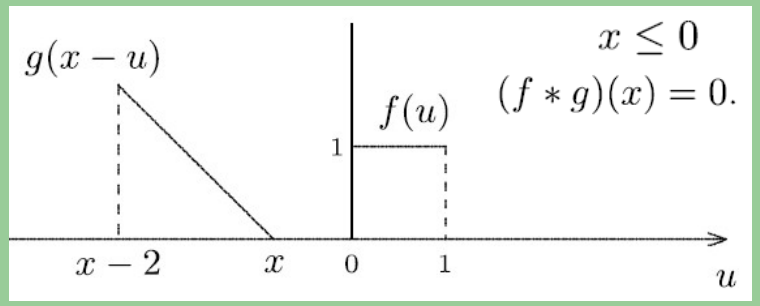
2) Çarpma (ing. multiplication): f(u) ile g(x-u) fonksiyonu çarpılır (f(u).g(x-u))

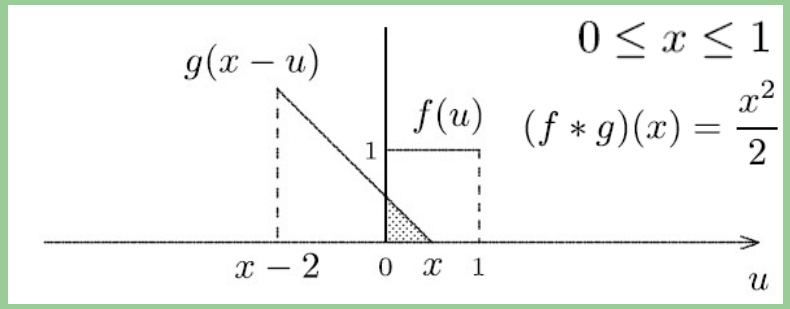


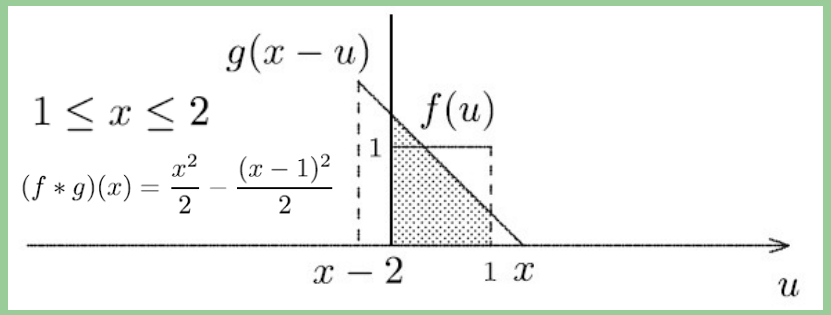
Konvolüsyon

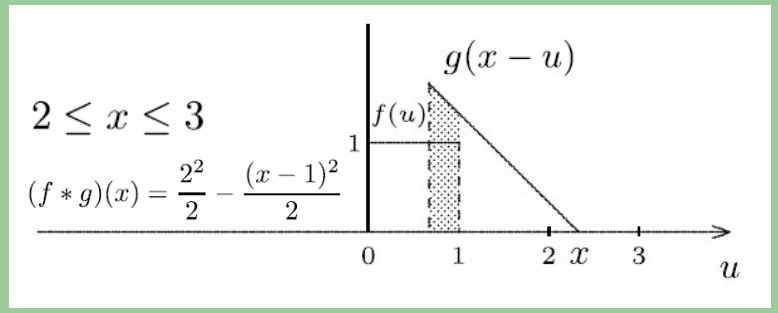
- ✔ Örneğimizdeki fonksiyonlar için bu şemayı adım adım uygulayalım.

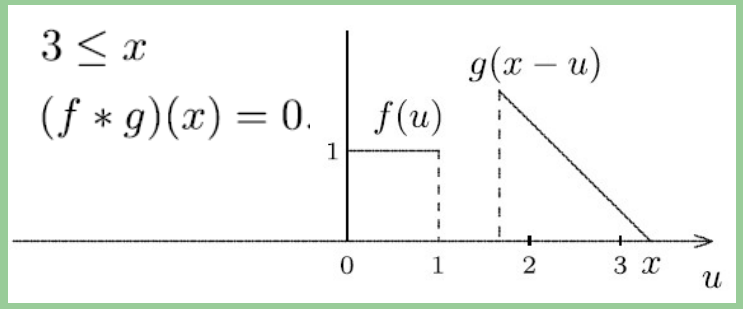
4) İntegrasyon (ing. integration): Bu işlem bütün x'ler için yapılarak f(u).g(x-u) altında kalan alan hesaplanır.





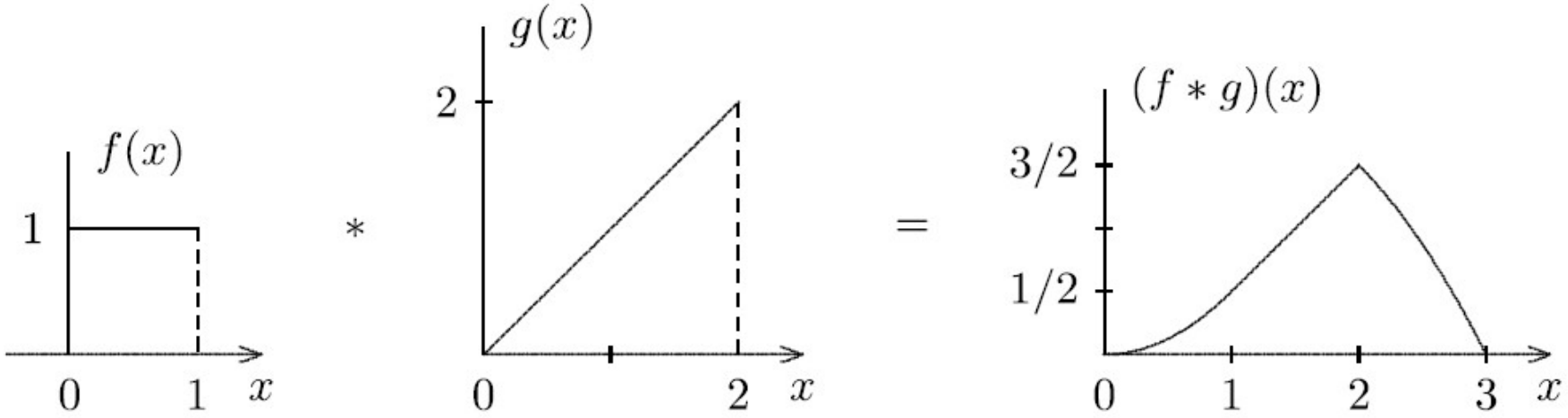






$$(f*g)(x) = \frac{1}{2}\begin{cases} x^2 & \text{if } 0 \le x \le 1 \\ 2x-1 & \text{if } 1 \le x \le 2 \\ -x^2+2x+3 & \text{if } 2 \le x \le 3 \\ 0 & \text{otherwise} \end{cases}$$

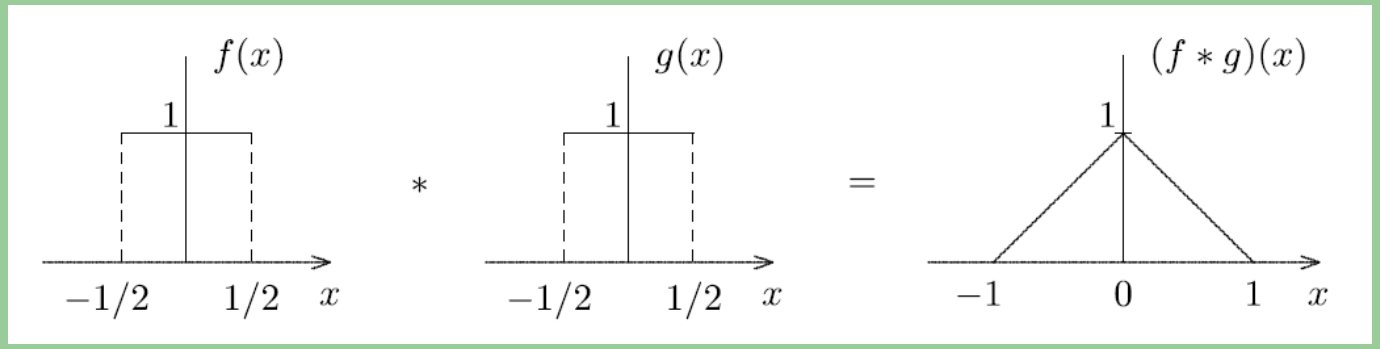
$$(f * g)(x) = \frac{1}{2}\begin{cases} x^2 & \text{if } 0 \le x \le 1 \\ 2x - 1 & \text{if } 1 \le x \le 2 \\ -x^2 + 2x + 3 & \text{if } 2 \le x \le 3 \\ 0 & \text{otherwise} \end{cases}$$



Örnek 2: Aşağıdaki fonksiyonların konvolüsyonunu (f * g) bulunuz.

$$f(x) := g(x) := \begin{cases} 1 & \text{if } -\frac{1}{2} < x < \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

f(u)'yu u = -1/2, u = +1/2 arasında sabit (f(u) = 1), g(x-u)'yu ise u = x -1/2, u = x + ½ arasında bir sabit (g(x-u) = 1) birer kutu fonksiyon olarak düşünebilirsiniz. g(u)'yu katlarken (g(-u)) bir değişim olmadığına dikkat ediniz, bu her zaman bu şekilde gerçekleşmez! x (-∞, +∞) arasında arttıkça g(x-u) x ekseninde kaydırılmış olur ve f(x) fonksiyonu ile g(x-u)'nun ilk kontaktı x = -1 iken sağlanır. Bu sırada g(x-u) fonksiyonunun sağ ucu u = x + 1/2, f(u) fonksiyonunun sol ucu u = -1/2'dedir. [-1, 0] aralığında bu iki fonksiyonun çarpımının altında kalan alan (integrasyonu) lineer olarak artar. x = 0 iken bu iki fonksiyon üstüste biner ve çarpımlarının integrali maksimum değerin alır ve sonra yine lineer olarak azalmaya başlar. İki fonksiyon x = 1 olduğunda son kez birbirine değer ve sonra çarpımları ve dolayısı ile bu çarpımın integrali 0 olur.



$$(f * g)(x) = \begin{cases} 0 & \text{if } x \le -1 \\ 1 + x & \text{if } -1 \le x \le 0 \\ 1 - x & \text{if } 0 \le x \le 1 \\ 0 & \text{if } 1 \le x \end{cases} = \begin{cases} 1 - |x| & \text{if } |x| \le 1 \\ 0 & \text{if } |x| \ge 1, \end{cases}$$

Örnek 3: α > 0 ve β > 0 olmak üzere aşağıdaki fonksiyonların konvolüsyonunu (f * g) bulunuz.

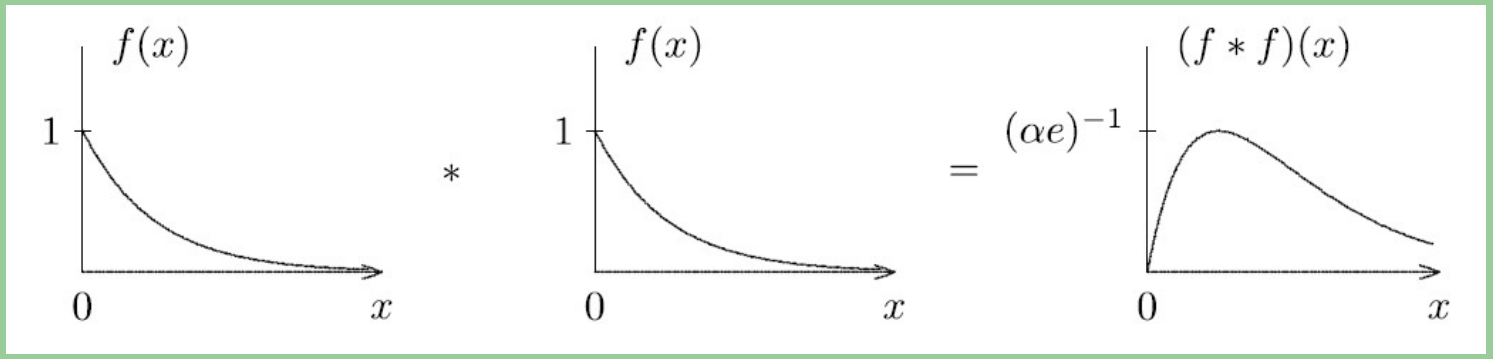
$$f(x) := \begin{cases} e^{-\alpha x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0, \end{cases} \qquad g(x) := \begin{cases} e^{-\beta x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases}$$

Katlama, kaydırma, çarpma ve integral alma adımları sırasıyla uygulanınca aşağıdaki ifadeler elde edilir.

$$(f * g)(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ \displaystyle\int_{u=0}^{x} e^{-\alpha u}\, e^{-\beta(x-u)}\, du & \text{if } x \geq 0 \end{cases}$$

$$= \begin{cases} 0 & \text{if } x \leq 0 \\ x\, e^{-\alpha x} & \text{if } x \geq 0 \text{ and } \beta = \alpha \\ \dfrac{e^{-\beta x} - e^{-\alpha x}}{\alpha - \beta} & \text{if } x \geq 0 \text{ and } \beta \neq \alpha. \end{cases}$$

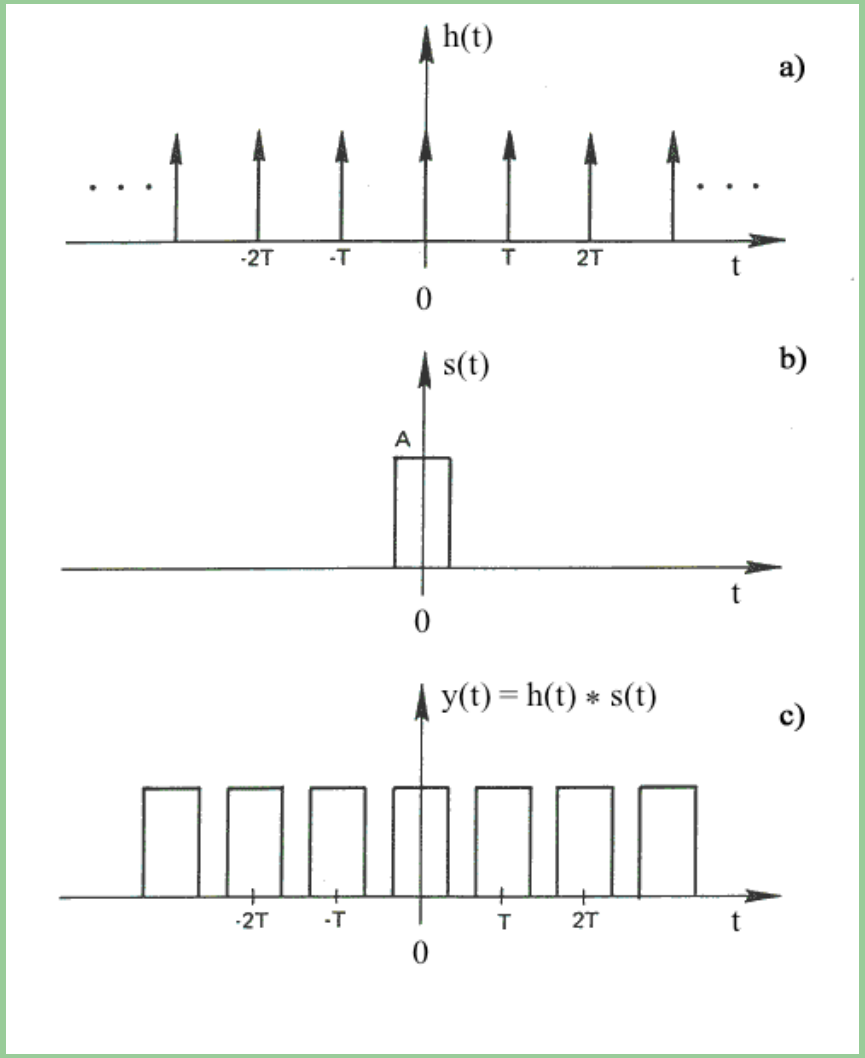
α = β için çözüm



Örnek 4:Herhangi bir fonksiyonun delta fonksiyonu ile konvolüsyonu

$$y(t) = s(t) * \sum_{k=-\infty}^{+\infty} \delta(t - k\Delta t).$$

$$\sum_{k=-\infty}^{+\infty} s(t)\ \delta(t - k\Delta t) = \sum_{k=-\infty}^{+\infty} s(t - k\Delta t).$$



Konvolüsyon Teoremi

f(x) ve g(x) fonksiyonlarının Fourier dönüşümleri aşağıdaki şekilde tanımlanmak üzere,

$$G(\nu) = \mathcal{F}\{g\} = \int_{\mathbb{R}^n} g(x) e^{-2\pi i x \cdot \nu}\, \mathrm{d}x,$$

$$F(\nu) = \mathcal{F}\{f\} = \int_{\mathbb{R}^n} f(x) e^{-2\pi i x \cdot \nu}\, \mathrm{d}x$$

h(z) fonksiyonu f(x) ile g(x) fonksiyonlarının konvolüsyonu, H(v) onun Fourier dönüşümü olsun.

$$h(z) = \int_{\mathbb{R}^n} f(x) g(z - x)\, \mathrm{d}x.$$

$$\begin{aligned} H(\nu) = \mathcal{F}\{h\} &= \int_{\mathbb{R}^n} h(z) e^{-2\pi i z \cdot \nu}\, dz \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(x) g(z - x)\, dx\, e^{-2\pi i z \cdot \nu}\, dz. \end{aligned}$$

Bu durumda;

$$H(\nu) = \int_{\mathbb{R}^n} f(x) \left(\int_{\mathbb{R}^n} g(z - x) e^{-2\pi i z \cdot \nu}\, dz \right) dx.$$

y = z – x, dy = dz değişken değişimi uygulanınca;

$$\begin{aligned} H(\nu) &= \int_{\mathbb{R}^n} f(x) \left(\int_{\mathbb{R}^n} g(y) e^{-2\pi i (y+x) \cdot \nu}\, dy \right) dx \\ &= \int_{\mathbb{R}^n} f(x) e^{-2\pi i x \cdot \nu} \left(\int_{\mathbb{R}^n} g(y) e^{-2\pi i y \cdot \nu}\, dy \right) dx \\ &= \int_{\mathbb{R}^n} f(x) e^{-2\pi i x \cdot \nu}\, dx \int_{\mathbb{R}^n} g(y) e^{-2\pi i y \cdot \nu}\, dy. \end{aligned}$$

Sonuç olarak $H(\nu) = F(\nu) \cdot G(\nu)$

Yani, iki fonksiyonun konvolüsyonlarının Fourier dönüşümü, her bir fonksiyonun Fourier dönüşümlerinin çarpımıdır.

Aynı şekilde iki fonksiyonun çarpımlarının Fourier dönüşümü de fonksiyonların Fourier dönüşümlerinin konvolüsyonuna eşittir!

$$f(x)g(x) \iff F(u) * G(u)$$

Korelasyon

Korelasyon da konvolüsyona çok benzer tanımlanır ve katlama dışında (u → -u) aynı işlem gibi görünür., zira kaydırma, çarpma ve integrasyon aynı şekilde uygulanır. Hatta f(-u) = f(u) şartını sağlayan (çift) reel fonksiyonlar için korelasyon ve konvolüsyon aynı şeydir. f(u) kompleks bir fonksiyon olduğunda onun eşleniği ($\overline{f(u)}$) integrasyon işlemine girer.

$$(f \star g)(x) := \int_{u=-\infty}^{\infty} \overline{f(u)} g(u+x)\, du$$

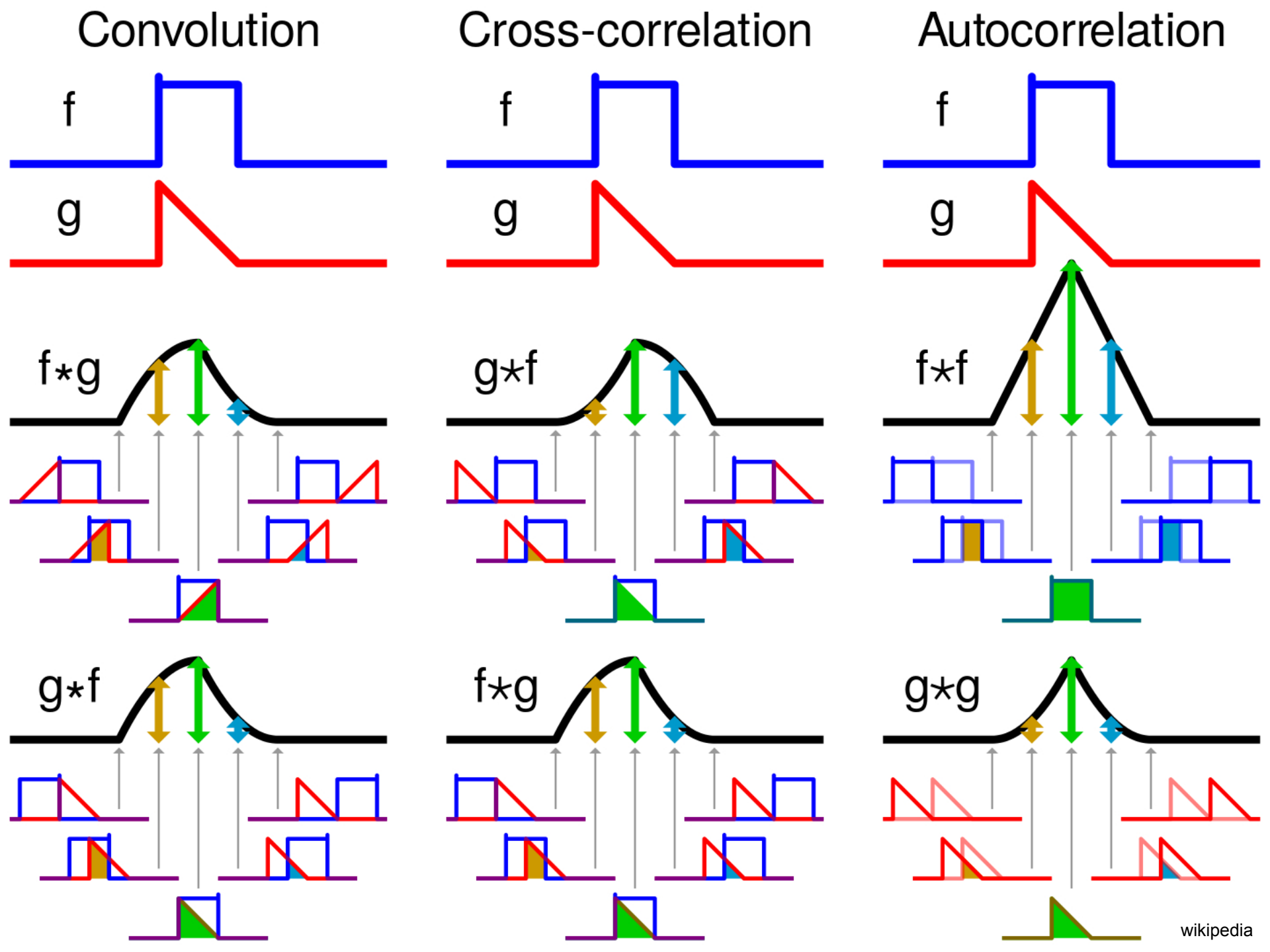
Fonksiyonlar süreksizse tanım benzerdir.

$$(f \star g)[n] := \sum_{m=-\infty}^{\infty} \overline{f[m]} g[m+n]$$

Korelasyon ifadesinde f ve g fonksiyonların birbirinden farklı fonksiyonlar ise işleme çapraz korelasyon (ing. cross correlation), işlem aynı fonksiyonlar arasında ise bu kez otokorelasyon (ing. autocorrelation) adını alır.

Çapraz korelasyonlar iki fonksiyon arasındaki benzerliği belirlemek için kullanılır. Çapraz korelasyonla bir fonksiyonun x (ya da zaman) ekseninde ne kadar kaydırıldığında diğerine "benziyor" olduğu belirlenir. Sinyal işlemede bir sinyallin diğerine göre ne kadar geciktiği (ing. time lag) ya da spektroskopide bir tayfın referans bir tayfa göre dalgaboyu ekseninde ne kadar kaydığı (ing. Doppler shift) gibi benzerlikler çapraz korelasyon fonksiyonları ile araştırılır.

$$f(t) \star g(t) = f(-t) * g(t) = f(t) * g^{*}(-t)$$


Convolution
Cross-correlation
Autocorrelation
f
g
f
g
f
g
f∗g
g∗f
f⋆f
g∗f
f∗g
g⋆g
wikipedia

Korelasyon Teoremi

F(v) ve G(v) sırasıyla f(t) ve g(t) fonksiyonlarının Fourier dönüşümleri olmak üzere;

$$
\begin{aligned}
f \star g &= \int_{-\infty}^{\infty} \overline{f}(\tau)\, g(t+\tau)\, d\tau \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} \overline{F}(v)\, e^{2\pi i v \tau}\, dv \int_{-\infty}^{\infty} G(v')\, e^{-2\pi i v' (t+\tau)}\, dv' \right] d\tau \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \overline{F}(v)\, G(v')\, e^{-2\pi i \tau (v'-v)}\, e^{-2\pi i v' t}\, d\tau\, dv\, dv' \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \overline{F}(v)\, G(v')\, e^{-2\pi i v' t} \left[\int_{-\infty}^{\infty} e^{-2\pi i \tau (v'-v)}\, d\tau \right] dv\, dv' \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \overline{F}(v)\, G(v')\, e^{-2\pi i v' t}\, \delta(v'-v)\, dv'\, dv \\
&= \int_{-\infty}^{\infty} \overline{F}(v)\, G(v)\, e^{-2\pi i v t}\, dv \\
&= \mathcal{F}\left[\overline{F}(v)\, G(v) \right],
\end{aligned}
$$

Yani f ve g fonksiyonlarının çapraz korelasyonu bu fonksiyonların Fourier dönüşümlerinin çarpımının Fourier dönüşümüne eşittir.

Wiener-Khinchin Teoremi

Otokorelasyon fonksiyonu yandaki şekilde tanımlanır.

$$C(t) \equiv \int_{-\infty}^{\infty} \overline{E}(\tau)\, E(t+\tau)\, d\tau.$$

Aynı fonksiyonun (E(t)) Fourier dönüşümü ise;

$$E(\tau) = \int_{-\infty}^{\infty} E_\nu\, e^{-2\pi i \nu \tau}\, d\nu,$$

Dönüşümün kompleks eşleniği:

$$\overline{E}(\tau) = \int_{-\infty}^{\infty} \overline{E}_\nu\, e^{2\pi i \nu \tau}\, d\nu.$$

olmak üzere;

$$\begin{aligned}
C(t) &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} \overline{E}_\nu\, e^{2\pi i \nu \tau}\, d\nu \right] \left[\int_{-\infty}^{\infty} E_{\nu'}\, e^{-2\pi i \nu' (t+\tau)}\, d\nu' \right] d\tau \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \overline{E}_\nu\, E_{\nu'}\, e^{-2\pi i \tau (\nu' - \nu)}\, e^{-2\pi i \nu' t}\, d\tau\, d\nu\, d\nu' \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \overline{E}_\nu\, E_{\nu'}\, \delta(\nu' - \nu)\, e^{-2\pi i \nu' t}\, d\nu\, d\nu' \\
&= \int_{-\infty}^{\infty} \overline{E}_\nu\, E_\nu\, e^{-2\pi i \nu t}\, d\nu \\
&= \int_{-\infty}^{\infty} |E_\nu|^2\, e^{-2\pi i \nu t}\, d\nu \\
&= \mathcal{F}_\nu \left[|E_\nu|^2 \right](t),
\end{aligned}$$

Görüldüğü üzere bir fonksiyon için tanımlanan otokorelasyon fonksiyonu ile onun Fourier dönüşümü aynı enerjiye ($E^2(\tau) \rightarrow (E^2(\nu))$ sahiptir. Bu özellik **Parseval teoremi** ile ifade olunur.

$$\int_{-\infty}^{\infty} |x(t)|^2\, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2\, d\omega = \int_{-\infty}^{\infty} |X(2\pi f)|^2\, df$$

Python'da Korelasyon

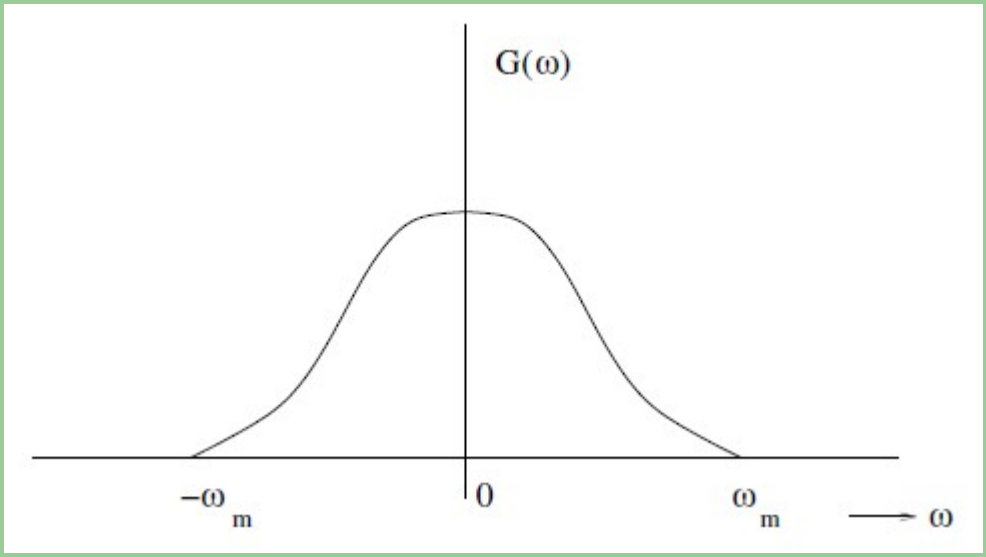
- ✔ Python'da korelasyon konvolüsyona çok benzer şekilde yapılır. Doğası gereği bu işlemin farkı ikinci fonksiyonun "katlanmamasıdır" (y-ekseninde simetriği alınmaz (korelasyon.py).
- ✔ Tıpkı konvolüsyonda olduğu gibi "full" (varsayılan), "same" ve "valid" stratejileriyle çapraz ve otokorelasyon yapılabilir.

```
import numpy as np
f = np.array([3,1,4,1])
g = np.array([5,9,2,6])
f_corr_g = np.correlate(f,g, mode="full")
print("f corr g (full) = ", f_corr_g)
f_corr_g = np.correlate(f,g, mode="same")
print("f corr g (same) = ", f_corr_g)
f_corr_g = np.correlate(f,g, mode="valid")
print("f corr g (valid) = ", f_corr_g)
f_corr_f = np.correlate(f,f, mode="full")
print("f corr f (full autocorrelation) = ", f_corr_f)
g_corr_g = np.correlate(g,g, mode="full")
print("g corr g (full autocorrelation) = ", g_corr_g)
```

```
In [13]: run korelasyon.py
f =  [3 1 4 1]
g =  [5 9 2 6]
f corr g (full) =  [18 12 53 38 43 29  5]
f corr g (same) =  [12 53 38 43]
f corr g (valid) =  [38]
f corr f (full autocorrelation) =  [ 3 13 11 27 11 13  3]
g corr g (full autocorrelation) =  [ 30  64  75 146  75  64  30]
```

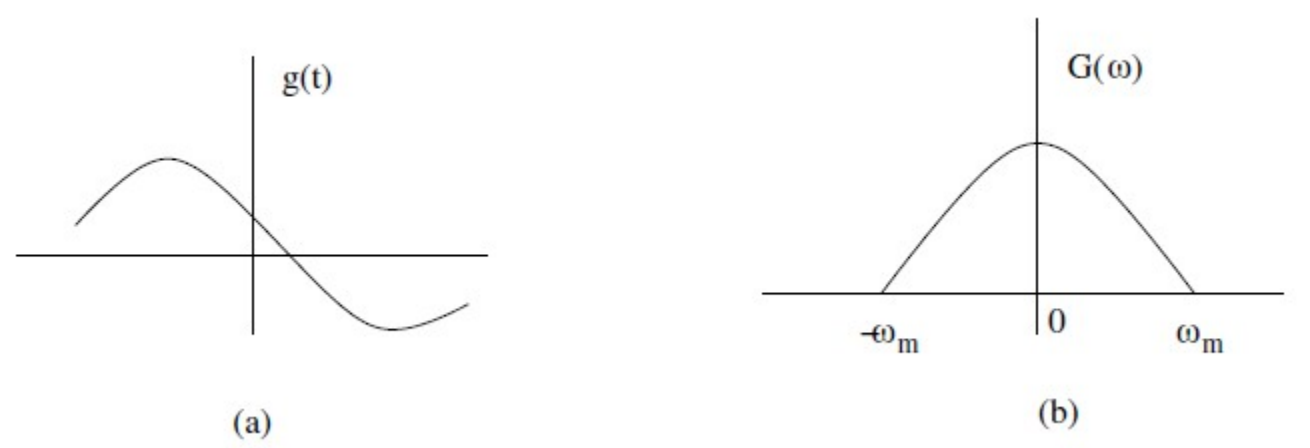
Örneklendirme (ing. Sampling) Teoremi

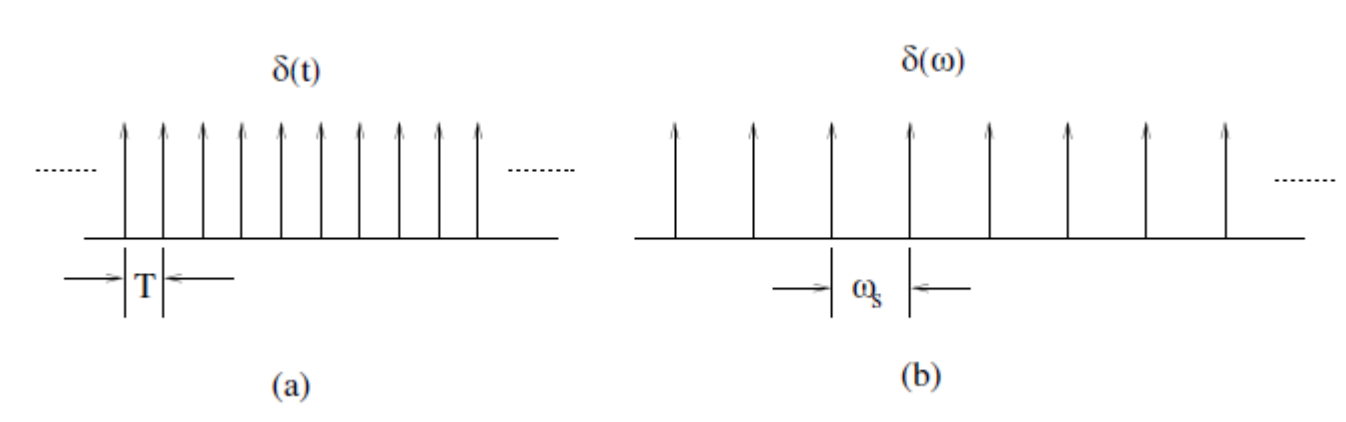
“Sınırlı (ing. band-limited) bir sinyallin gözlemler sonucu tekrar aynen oluşturulabilmesi (ing. reconstruction) için içerdiği maksimum frekansın en az iki katı frekansta örneklenmesi gerekir.



Yukarıda gördüğünüz sinyalin frekansı f_m olsun. Bu sinyalin gözlemlerle aynen elde edilebilmesi (gözlem noktaları kullanılarak sinyallin oluşturulabilmesi)için $f_s \geq 2\ f_m$ ‘lik bir frekansta gözlem noktası alınması gerekir. İhtiyaç duyulan bu minimum frekansa ($2\ f_m$) **Nyquist frekansı** adı verilir.

Orjinal sinyalimiz $g_s(t)$, onun Fourier dönüşümü (açısal frekans birimlerinde) $G_s(\omega)$ ($\omega_m = 2\pi f_m$) olsun. $g_s(t)$ sinyalini $\delta_T(t)$ fonksiyonuyla örneklendiriyor (T dönemi kadar aralıklarla gözlüyor) olalım.





$\delta_T(t)$ fonksiyonu örneklendirme fonksiyonu (ing. sampling function), $\delta_T(\omega)$ ise frekans örneklendirme fonksiyonu (ing. frequency sampling function) adını alır.

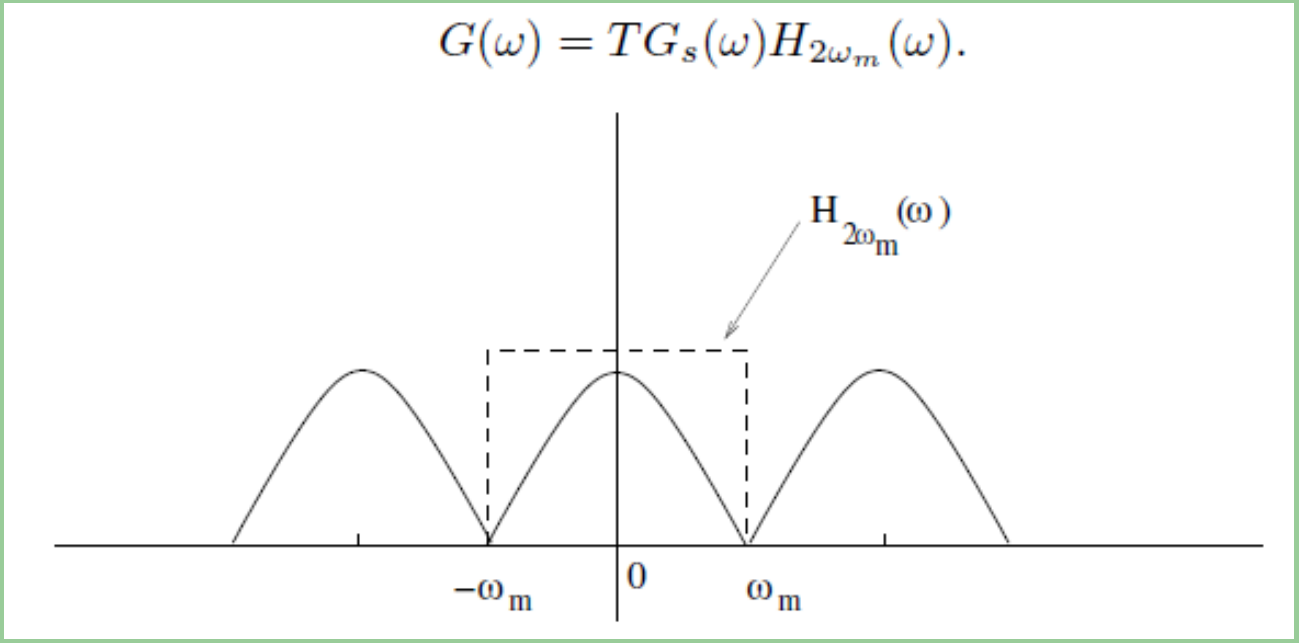
Bu durumda sinyalin Fourier dönüşümü aşağıdaki şekilde elde edilir.

$$
\begin{aligned}
\mathcal{F}(g_s(t)) &= \mathcal{F}\left[g(t)\delta_T(t)\right] \\
&= \mathcal{F}\left[g(t)\sum_{n=-\infty}^{+\infty}\delta(t-nT)\right] \\
&= \frac{1}{2\pi}\left[G(\omega) * \omega_0 \sum_{n=-\infty}^{+\infty}\delta(\omega-n\omega_0)\right] \\
G_s(\omega) &= \frac{1}{T}\sum_{n=-\infty}^{+\infty} G(\omega) * \delta(\omega-n\omega_0) \\
G_s(\omega) &= \mathcal{F}\left[g(t)+2g(t)\cos(\omega_0 t)+2g(t)\cos(2\omega_0 t)+\cdots\right] \\
G_s(\omega) &= \frac{1}{T}\sum_{n=-\infty}^{+\infty} G(\omega-n\omega_0)
\end{aligned}
$$

Gözlem frekansı $\omega_s = 2\,\omega_m$ ise ($T = 1 / 2\,f_m$) bu sinyal tekrar oluşturulabilir.

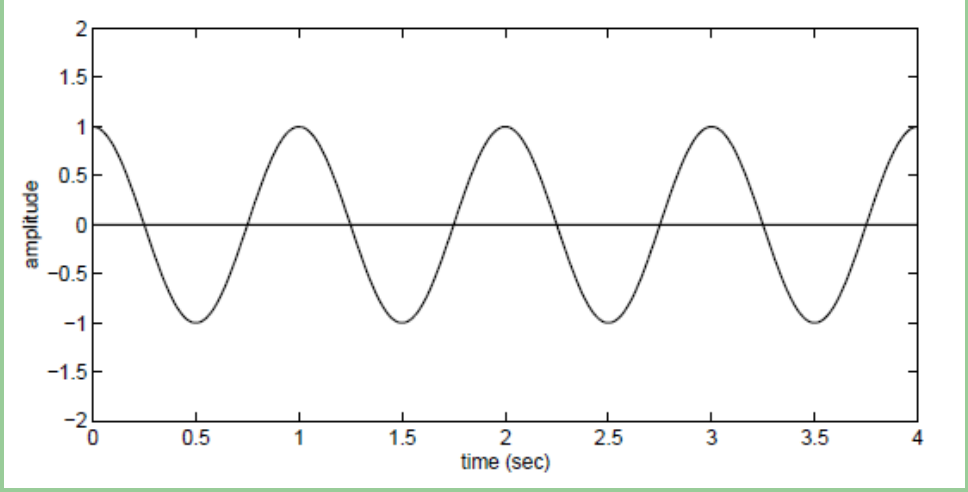
$$G_s(\omega) = \frac{1}{T}\sum_{n=-\infty}^{+\infty} G(\omega-n\omega_m)$$

Gözlem penceresinin ($H_{2\omega}$) (ing. spectral window) en az $2\omega_m$ büyüklüğünde olması gerekir ki sinyal yeniden oluşturulabilsin. Gözlem penceresinin zaman tanım kümesindeki karşılığı **pencere fonksiyonu** (ing. window function) adını alır.

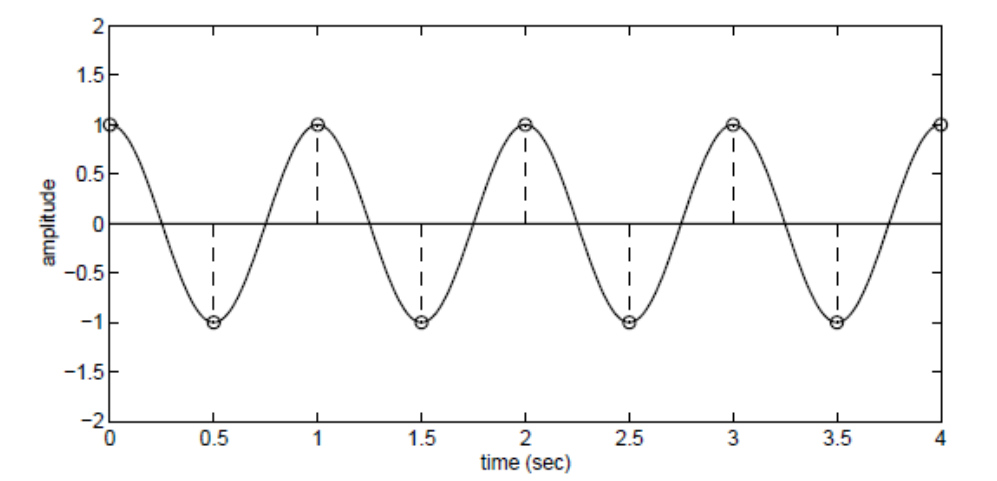


Sinyal içerdiği maksimum frekanstan daha düşük bir frekansta gözlenirse tekrar oluşturulamaz. Bu duruma **frekans örtüşmesi (ing. aliasing)** adı verilir.

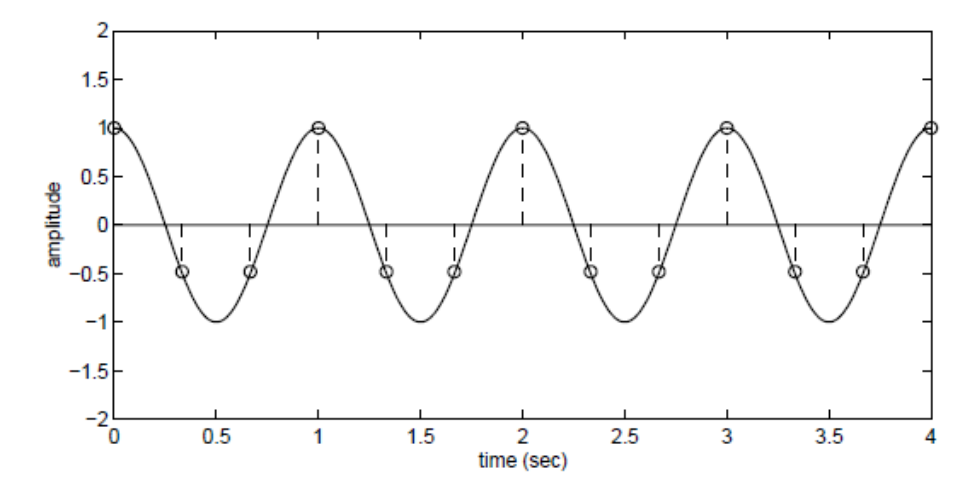
Örnek 1. 1 Hz frekansa sahip aşağıdaki sinyali ele alalım.



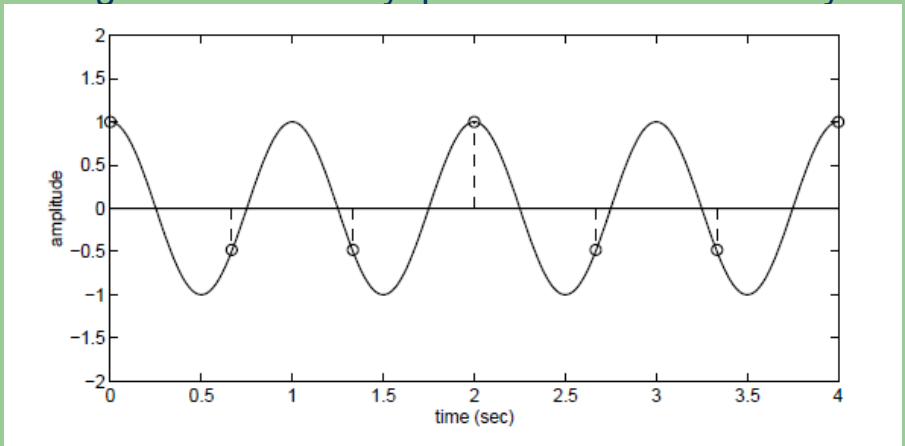
Sinyali 2 Hz frekansla örneklersek tekrar oluşturmamız mümkün olur.

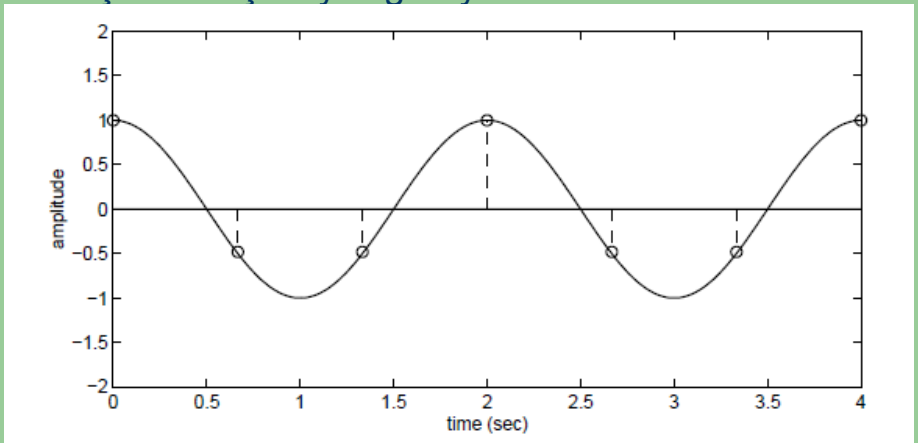


Sinyali 3 Hz frekansla örneklersek orjinal sinyali oluşturmamız daha da kolay olur. (ing. oversampling)

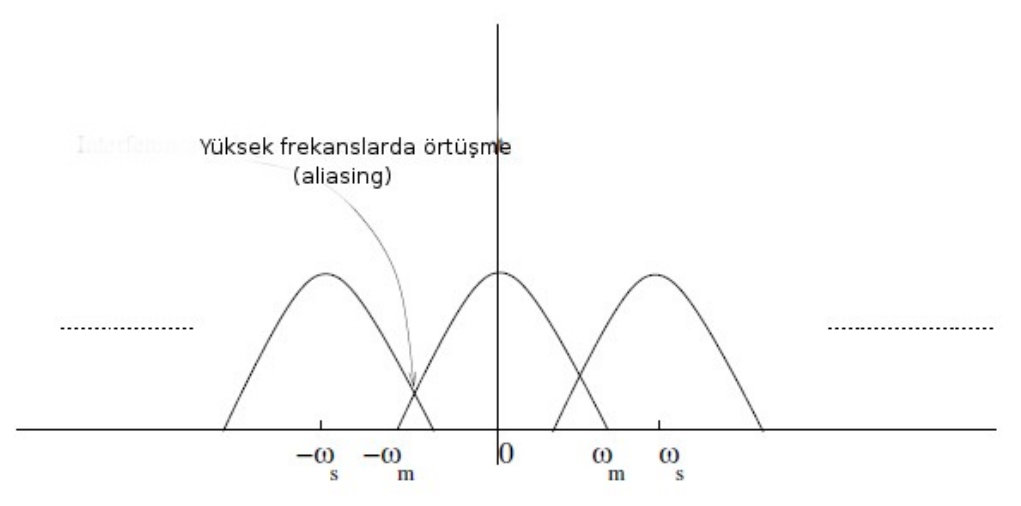


Ancak sinyali 1.5 Hz frekansla örneklersek tekrar oluşturmamız mümkün olmaz (ing. undersampling). Sadece bilgi kaybetmemiş aynı zamanda bu sinyal hakkında yanlış bir izlenim de edinmiş oluruz. Alt sağda bu frekansla yapılan örneklendirmeden yeniden oluşturulmuş sinyali görüyorsunuz.

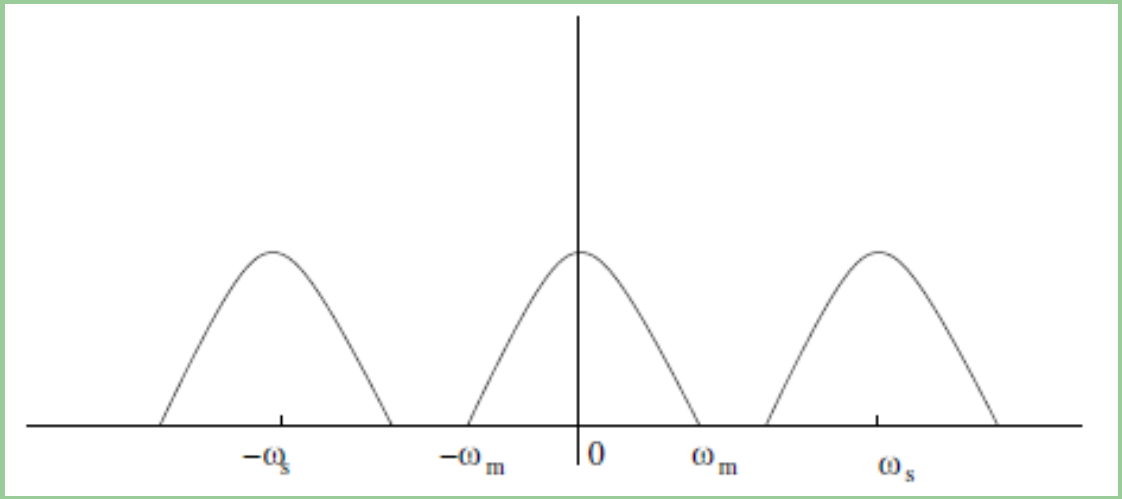




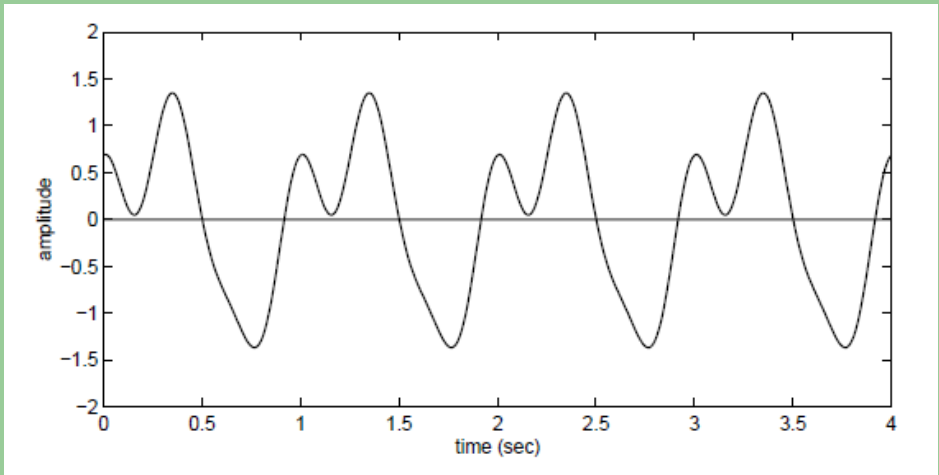
Yüksek frekans içeriğinin gözlenememesi durumuna neden olan frekans örtüşmesi olgusuyla astronomide, özel olarak da yüksek frekanslı değişimlerin gözlenebildiği asterosismolojide sıklıkla karşıaşılır.



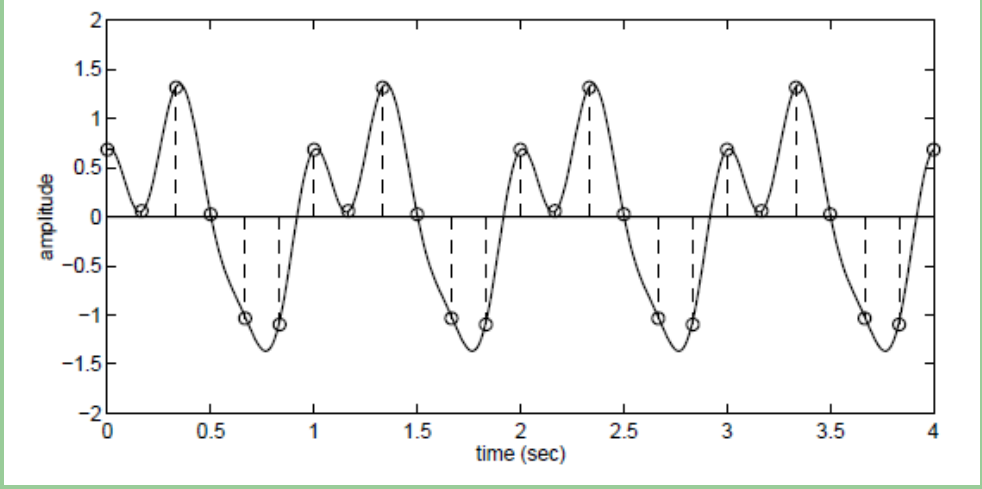
Gözlemlerin sinyalin içerdiği maksimum frekanstan daha yüksek bir frekansla gerçekleşmesi durumunda Fourier tanım kümesinde frekans piklerinin birbirinden iyice ayrıldığı ve bu nedenle sinyalin gözlenen frekanstan tekrar oluşturulabileceği görülür. Nyquist frekansının üstünde gözlem yapılan duruma **ing. oversampling** adı verilir.



Örnek 2. Çoğul frekansların varlığı durumunda verinin orjinal sinyalin içerdiği en yüksek frekansın iki katı frekansta alınması gerekir.



Bu örnekteki en yüksek frekans 3 Hz olup, 6 Hz frekansla örneklersek tekrar oluşturabiliriz.



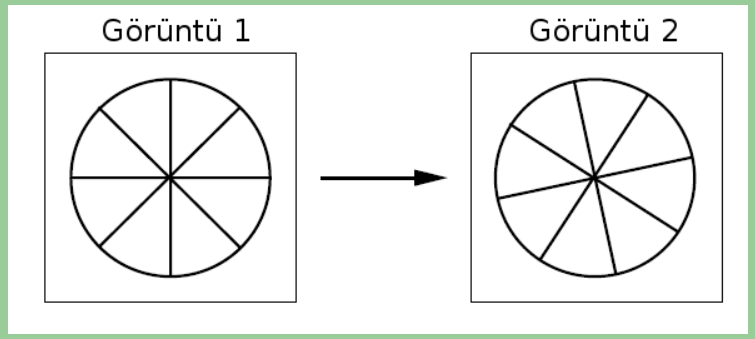
Örnek 3. Aliasing olgusunun iyi bir örneği 24 görüntü / saniye (ing. frame per second) üzerinden oluşturulan videolarda yaşanır. Jantında 8 doğru parçası bulunan bir tekerlek düşünelim. Bu tekerlek saat yönünde saniyede 3 kez dönüyor olsun (3 Hz → 180 rpm). Bu tekerlek 24 görüntü / saniye ile görüntülenirse;

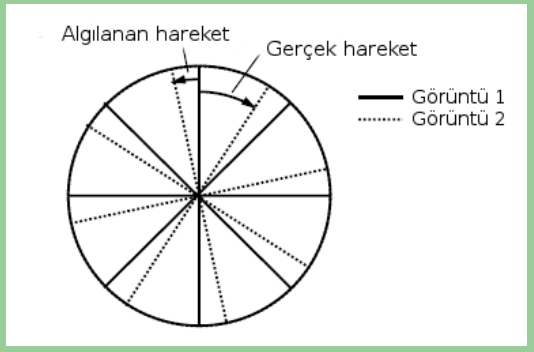
$$\frac{\text{3 dönüş / saniye x 8 doğru / dönüş}}{\text{24 görüntü / saniye}} = \text{1 doğru / görüntü}$$

görüntü başına 1 doğru yer değiştirme gerçekleşmiş olur. Yani bir doğru iki görüntü arasında (360 / 8)° yer değiştirmiş ve diğerinin yerini almış olur. Dolayısı ile izleyici tekeri hiç dönmüyormuş gibi görür.

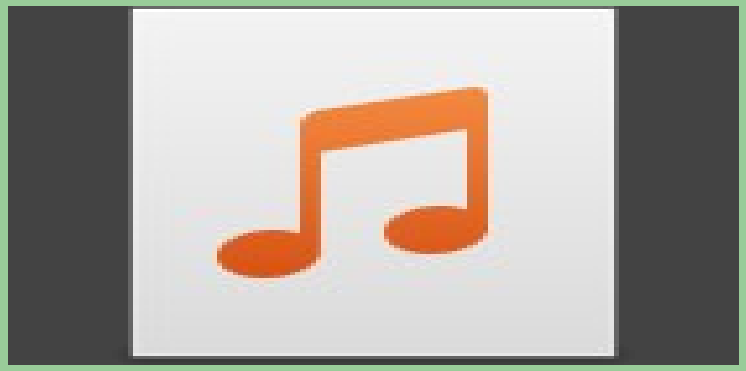
Diyelim ki tekerlik yine saat yönünde saniyede 2.5 kez dönüyor olsun (2.5 dönüş / saniye). Bu durumda 2.5 x 8 / 24 = 0.83'lük bir yer değiştirme gerçekleşir. Yani bir doğru iki görüntü arasında, iki doğru arasındaki uzaklığın saat yönünde %83'ü kadar yer değiştirmiş olsun.

Beynimiz doğru parçalarını birbirinin aynı olarak değerlendirdiği için birbirinden ayıramaz ve iki görüntü arasındaki değişime alternatif bir açıklama da getirebilir. Her bir doğru parçası saat yönünde %83 değil de ona ters yönde %17 yer değiştiriyormuş gibi de görünebilir ve görünen o ki beynimiz bu alternatif açıklamayı tercih etmekte, bu nedenle tekerlek saat yönünün tersine dönüyor gibi görünmektedir (altta).





Dolayısı ile tekerlek bize ters dönüyormuş gibi görünür. Bunun nedeni "frekans örtüşmesi" (ing. aliasing) etkisidir. Tekerleğin dönüşünü doğru hız ve yönde görebimemiz için görüntü alma hızımız tekerleğin dönme hızının en az iki katı olmalıdır.



Cool Wagon-wheel Effect (Storboscopic Effect)
https://www.youtube.com/channel/UCLWPPO_2CtJSK9Z0Ez-BWaA

Süreksiz Fourier Dönüşümü
(ing. Discrete Fourier Transform, DFT)

Fourier dönüşümlerinin sürekli fonksiyonlara nasıl uygulandığını gördük ancak gerçekte sürekli fonksiyonlar yerine süreksiz (ing. discontinous) ve belirli bir gözlem penceresi içinde yer alan (ing. band-limited) veriyle karşılaşılır. Bu durumda süreksiz Fourier dönüşümlerine başvurulur.

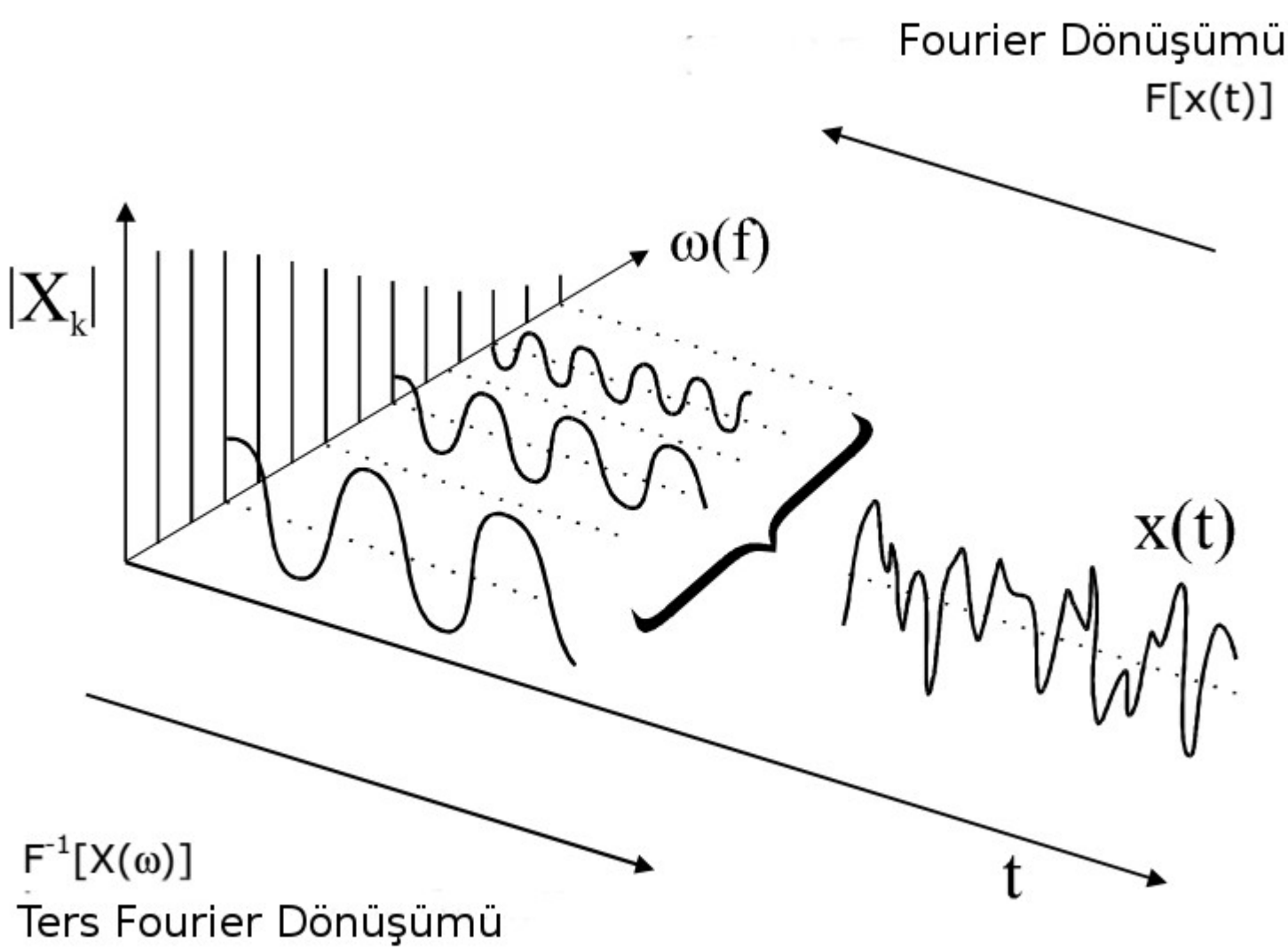
$$\int_{-\infty}^{\infty} f(t)\, e^{-2\pi i \nu t}\, dt. \longrightarrow F_n \equiv \sum_{k=0}^{N-1} f_k\, e^{-2\pi i n k/N}$$

Bu durumda ters süreksiz Fourier dönüşümü (ing. Inverse Discrete Fourier Transform)

$$f_k = \mathcal{F}_n^{-1}\left[\{F_n\}_{n=0}^{N-1}\right](k) \longrightarrow f_k = \frac{1}{N}\sum_{n=0}^{N-1} F_n\, e^{2\pi i k n/N}$$

Yukarıdaki dönüşümün bir sonucu olarak periyodik bir fonksiyonun Fourier dönüşümünde sadece periyodun karşılık geldiği frekansta bir pik yerine, bir pik de o frekansın negatif değerinde oluşur.

$n = 0 – \omega_0$, $1 \le n \le N/2 – 1 \rightarrow 0 < \omega < \omega_c$, $N/2 + 1 \le n \le N - 1 \rightarrow -\omega_c < \omega < 0 \rightarrow N = N/2 \rightarrow \omega = -/+ \omega_c$


Fourier Dönüşümü
F[x(t)]
ω(f)
|X_k|
x(t)
t
F^-1[X(ω)]
Ters Fourier Dönüşümü

Hızlı Fourier Dönüşümü
(ing. Fast Fourier Transform)

Süreksiz Fourier dönüşümünde f_k'nın $e^{-2\pi ikn / N}$ terimi ile N kez çarpımı (ki bu çarpımların hepsi kompleks uzayda yapılır) ve elde edilen değerlerin N kez toplamı gerekir (işlemin karmaşıklığı N^2 ile orantılıdır). Bu nedenle "Yavaş" Fourier Dönüşümü olarak da adlandırılır. Hızlı Fourier Dönüşümleri Fourier dönüşümlerinin simetri gibi bazı özelliklerinden faydalanarak bu karmaşıklığı $N \log_2 N$ düzeyine indirir.

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-i\,2\pi\,k\,n\,/\,N} \\
&= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\,2\pi\,k\,(2m)\,/\,N} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\,2\pi\,k\,(2m+1)\,/\,N} \\
&= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\,2\pi\,k\,m\,/\,(N/2)} + e^{-i\,2\pi\,k\,/\,N} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\,2\pi\,k\,m\,/\,(N/2)}
\end{aligned}
$$

Cooley – Tukey algoritmasında olduğu gibi dönüşüm tek ve çift bileşenlerine ayrıldıktan sonra her bir bileşenin de içinde simetrik olduğu görülür. Bu simetri sinyalin yapısına da bağlı olarak daha fazla basitleştirme de getirir ve DFT matrisinin pek çok elemanı 0 olur. Bu elemanlarla çarpma yapmak anlamılı olmayacağı için hiç işlem yapılmaz. Bu noktada FFT'nin yapısı gereği N'in 2'nin kuvvetleri olması durumunda çalışacağını, aksi takdirde ya atma (ing. truncation) ya da 0'la doldurma (zero padding) yapılması gerektiğini vurgulamak gerekir.

“Yavaş” Fourier Dönüşümü
(ing. “Slow” Fourier Transform)

```
import numpy as np
def DFT_slow(x):
    ""x 1D dizisinin sureksiz Fourier donusumu"""
    x = np.asarray(x, dtype=float)
    N = x.shape[0]
    n = np.arange(N)
    k = n.reshape((N, 1))
    M = np.exp(-2j * np.pi * k * n / N)
    return np.dot(M, x)
```

dft_slow.py

Hızlı Fourier Dönüşümü
(ing. Fast Fourier Transform)

```
def FFT(x):
    "1D Cooley-Tukey FFT"""
    x = np.asarray(x, dtype=float)
    N = x.shape[0]

    if N % 2 > 0:
        raise ValueError("x 2nin kuvveti olmali”)
    elif N <= 32:  # N yeterince kucukse yavas algoritma
        return DFT_slow(x)
    else:
        X_cift = FFT(x[::2])
        X_tek = FFT(x[1::2])
        carpan = np.exp(-2j * np.pi * np.arange(N) / N)
        return np.concatenate([X_cift + carpan[:N / 2] * X_tek,\
                               X_cift + carpan[N / 2:] * X_tek])
```

fft.py

Spektral Yoğunluk (ing. Spectral Density)

Tüm fiziksel sinyallerin sürekli ve süreksiz Fourier dönüşümü yoluyla içerdiği frekans bileşenlerine ayrıştırılabildiğini gördük. Bir sinyalin spektral güç yoğunluğu (ing. power spectral density (PSD)) sinyalin her bir frekansta sahip olduğu gücü gösterir. Bir sinyalin spektral enerji yoğunluğu ise enerjisinin frekansa bağlı olarak nasıl değiştiğini gösterir. Bir sinyalin enerjisi,

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt. \quad \xrightarrow{\text{Parseval teoremi}} \quad \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |\hat{x}(f)|^2 \, df,$$

şeklinde tanımlanır. Daha önce görüdüğümüz Parseval teoremi bunu frekans kümesine dönüştürmemize de olanak sağlar. Burada x(f) aşağıda tanımlandığı şekliyle x(t) zaman serisinin sürekli Fourier dönüşümüdür. Süreksiz versiyonu da integralin yerini toplamın almasıyla birlikte benzerdir.

$$\hat{x}(f) = \int_{-\infty}^{\infty} e^{-2\pi i f t} x(t) \, dt,$$

$|x(f)|^2$, sinyalin f frekansında sahip olduğu enerjiyi, yani frekans başına enerjiyi tarif ettiği için bir yoğunluk fonksiyonu olarak düşünülebilir. Dolayısı ile spektral enerji yoğunluğu kavramı aşağıdaki ifadelerle sürekli ve süreksiz durumlar için tanımlanabilir.

$$S_{xx}(f) = |\hat{x}(f)|^2 \qquad S_{xx}(f) = (\Delta t)^2 \left| \sum_{n=-\infty}^{\infty} x_n e^{-2\pi i f n} \right|^2 = (\Delta t)^2 \hat{x}_d(f) \hat{x}_d^*(f),$$

Spektral Güç Yoğunluğu (ing. Power Spectral Density)

Enerji yoğunluğu ifadesi atımlar şeklinde gerçekleşen enerjinin tek bir frekans etrafına yoğunlaştığı değişimler için kullanışlıdır. Zamanda sürekli sinyaller için daha çok **spektral güç yoğunluğu** kullanılır. Buradaki güç kavramı gerçekten fiziksel güç olabildiği gibi tanım kümesi farklılık gösterdiğinde fiziksel anlamda bir güçten söz edilmemekle birlikte yine de güç spektrumu kavramı kullanılır. x(t) sinyalinin zaman üzerinden ortalama gücü

$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |x(t)|^2 \, dt.$$

şeklinde tanımlanır. [0, T] aralığında süreksiz Fourier dönüşümü ise aşağıdaki gibi hesaplanır.

$$\hat{x}_T(\omega) = \frac{1}{\sqrt{T}} \int_0^T x(t) e^{-i\omega t} \, dt.$$

Spektral güç dağılımı ise E; beklenen değeri göstermek üzere aşağıdaki şekilde verilir. Beklenen değer tekdüze bir dağılım için dağılımın ortalamasından başka bir şey değildir. Görüldüğü gibi bir kompleks fonkisyon için kendisiyle kompleks eşleniğinin [0,T] aralığında integre edlip çarpımının sinyal uzunluğuna bölümünden ibarettir.

$$S_{xx}(\omega) = \lim_{T\to\infty} \mathbf{E}\left[|\hat{x}_T(\omega)|^2\right].$$

$$\mathbf{E}\left[|\hat{x}_T(\omega)|^2\right] = \mathbf{E}\left[\frac{1}{T}\int_0^T x^*(t) e^{i\omega t}\, dt \int_0^T x(t') e^{-i\omega t'}\, dt'\right] = \frac{1}{T}\int_0^T \int_0^T \mathbf{E}\left[x^*(t)x(t')\right] e^{i\omega(t-t')}\, dt\, dt'.$$

Periyodogram

Sonuç olarak klasik periyodogram (ya da güç spektrumu (ing. power spectrum)) fonksiyonla süreksiz Fourier dönüşümünün toplamda aynı güce sahip olması için gücünün nokta sayısına (N) normalize edilmesiyle elde edilir ve x(t) zaman serisine karşılık gelen X(v) frekanslarından her birinin ne kadar güce sahip olduğunu ifade eder.

$$P_N(\nu) = \frac{1}{N}\left|F_N(\nu)\right|^2 = \frac{1}{N}\left|\sum_{i=1}^{N} x(t_i)\exp(2\pi \mathrm{i}\,\nu t_i)\right|^2$$

$$= \frac{1}{N}\left\{\left(\sum_{i=1}^{N} x(t_i)\sin(2\pi\nu t_i)\right)^2 + \left(\sum_{i=1}^{N} x(t_i)\cos(2\pi\nu t_i)\right)^2\right\}$$

x(t) sinyali $x(t_i) = A\cos(2\pi\nu_1 t_i)$ şeklinde bir sinüsoidalse periyodogramın ν_1 frekansında değeri

$$P_N(\nu_1) = \frac{1}{N}\left\{\sum_{i=1}^{N} A\cos(2\pi\nu_1 t_i)\sin(2\pi\nu_1 t_i)\right\}^2 + \frac{1}{N}\left\{\sum_{i=1}^{N} A\cos^2(2\pi\nu_1 t_i)\right\}^2$$

N yeterince büyükse (çok sayıda veri noktası varsa, $N \to \infty$)

$$\sum_{i=1}^{N}\cos(2\pi\nu_1 t_i)\sin(2\pi\nu_1 t_i) \approx 0, \qquad \sum_{i=1}^{N}\cos^2(2\pi\nu_1 t_i) \approx N/2$$

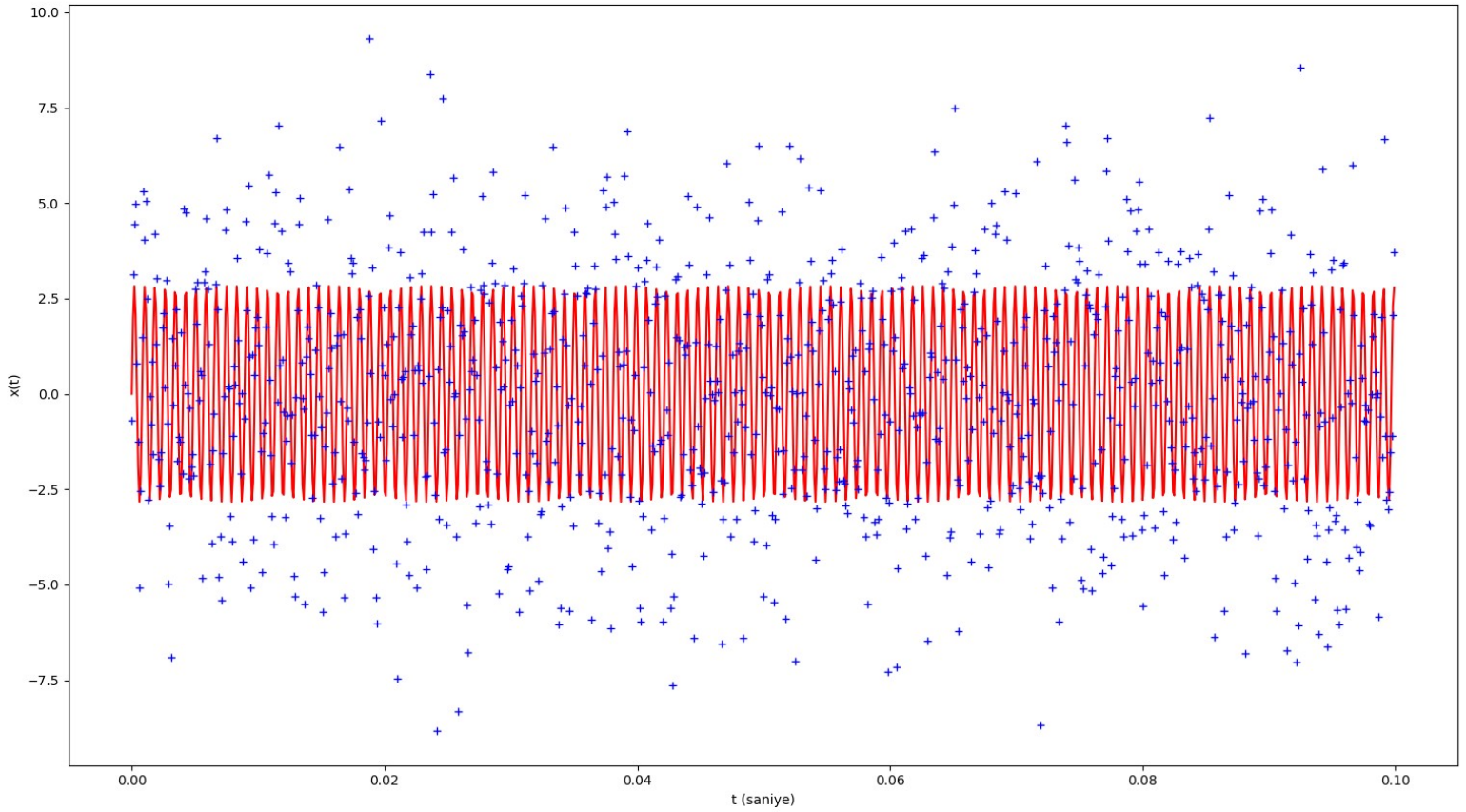
ν_1 frekansında güç spektrumunun değeri bu nedenle $A^2\,N\,/\,4$ 'e yaklaşır.

scipy.signal.periodogram

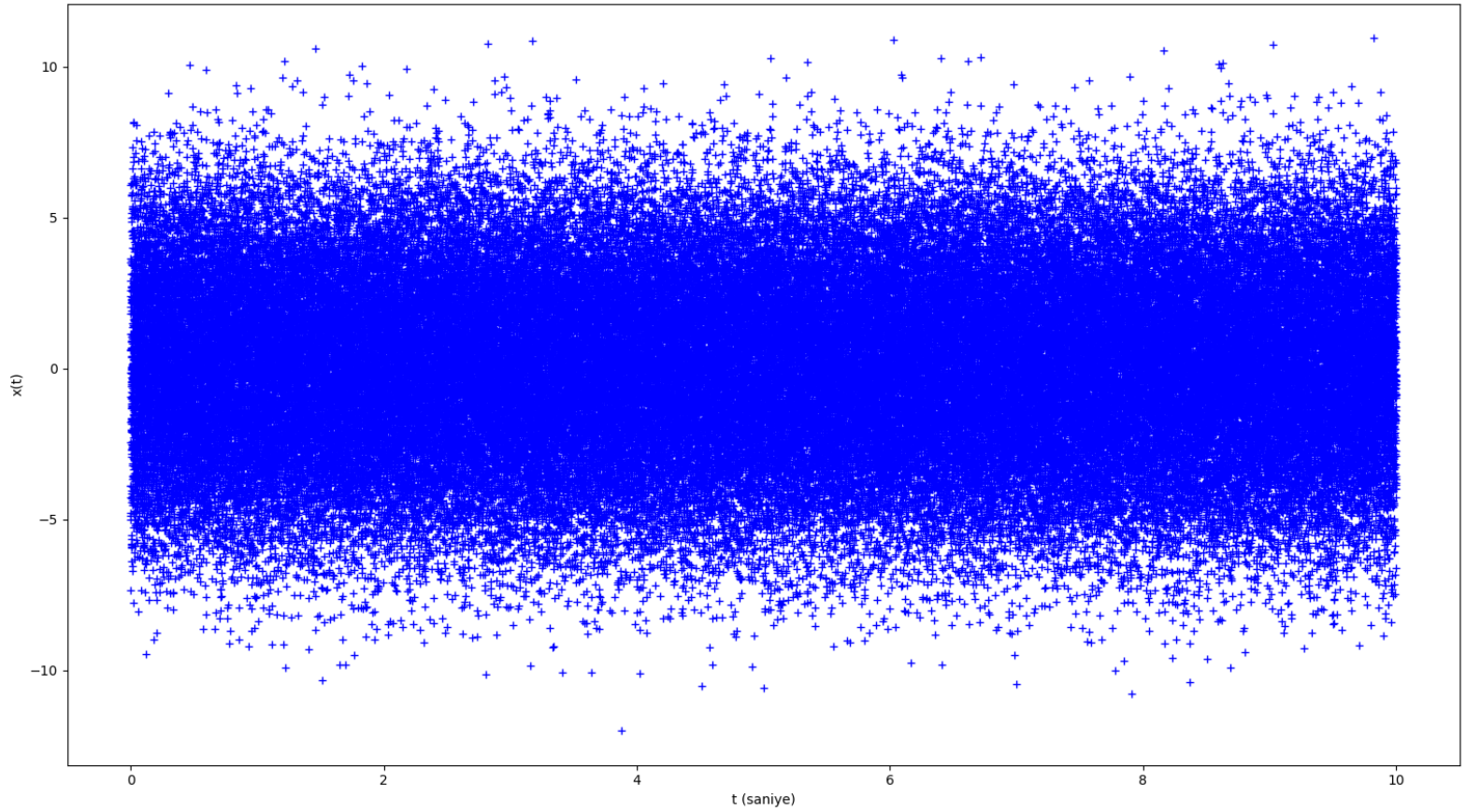
`scipy.signal.periodogram` verilen bir x(t) zaman serisi için spektral güç yoğunluğunu bir periyodogramla verir. Zaman serisi veri x 'in örnekleme frekansı `fs`, gözlem penceresi (spectral window) `window` parametreleriyle sağlanır. İstendiği takdirde hızlı Fourier dönüşümünün uzunluğu (`nfft`), kompleks sinyaller yerine gerçek sinyaller söz konusu olduğunda tek taraflı güç spektrumu (`return_onesided`) istenirse yoğunluk (v^2 / Hz, karekökü doğrudan genliği verecek şekilde ölçeklendirilmiş) istenirse de v biriminde (`scaling`) bir periyodogram alınır. Örnek olarak önce sentetik bir veri üretip sonra onu analiz edelim.

```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
# Sentetik bir sinyal yaratip uzerine gurultu ekleyelim.
fs = 1e4 # ornekleme frekansi (veri alinma sikligi)
N = 1e5 # nokta sayisi
A = 2*np.sqrt(2) # degisim genligi
v = 1234.0 # degisim frekansi
gurultu = 0.001 * fs / 2 # sentetik gurultu (beyaz)
t = np.arange(N) / fs # zaman
x = A*np.sin(2*np.pi*v*t) # x sentetik sinyali
plt.plot(t,x,'r-')
# sinyale gurultu ekleyelim
x        +=        np.random.normal(scale=np.sqrt(gurultu),
size=t.shape)
# sinyali cizdirelim
plt.plot(t,x,'b+')
plt.xlabel("t (saniye)")
plt.ylabel("x(t)")
plt.show()
```

periyoodogram.py



periyoodgram.py koduyla ürettiğimiz sinüsoidal sinyal (kırmızı) ve üzerine gürültü bindirilmiş hali (mavi +). Analiz ederek içerisindeki frekansı belirlemek istediğimiz veri bu mavi veri noktalarıdır. Kolay gösterim için veri noktası sayısı N = 1000 ile sınırlandırılmıştır.



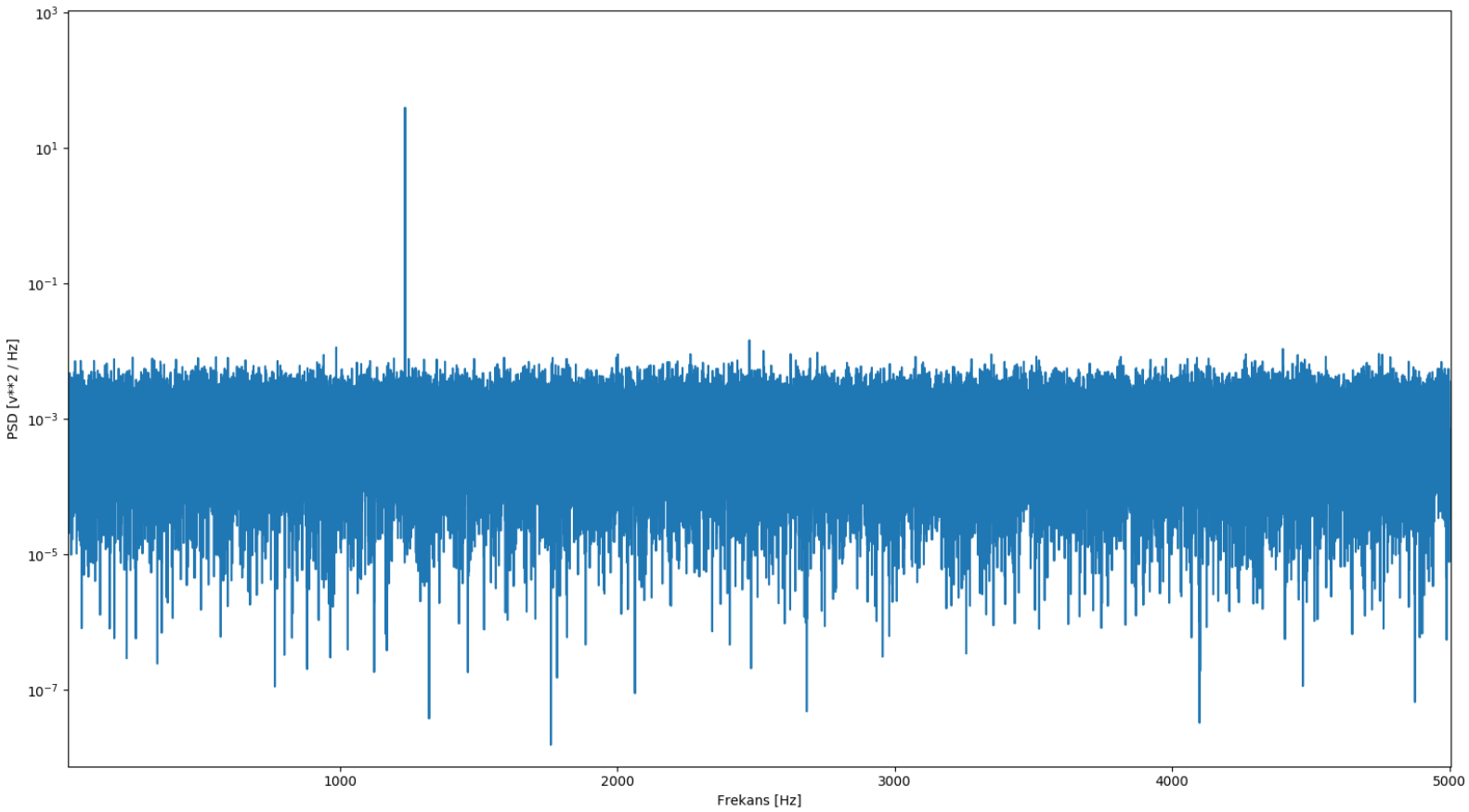
periyoodgram.py koduyla ürettiğimiz ve analiz etmek istediğimiz sentetik veri. fs = 10000 Hz ile örneklenmiş N = 100000 veri noktası içermektedir.

scipy.signal.periodogram

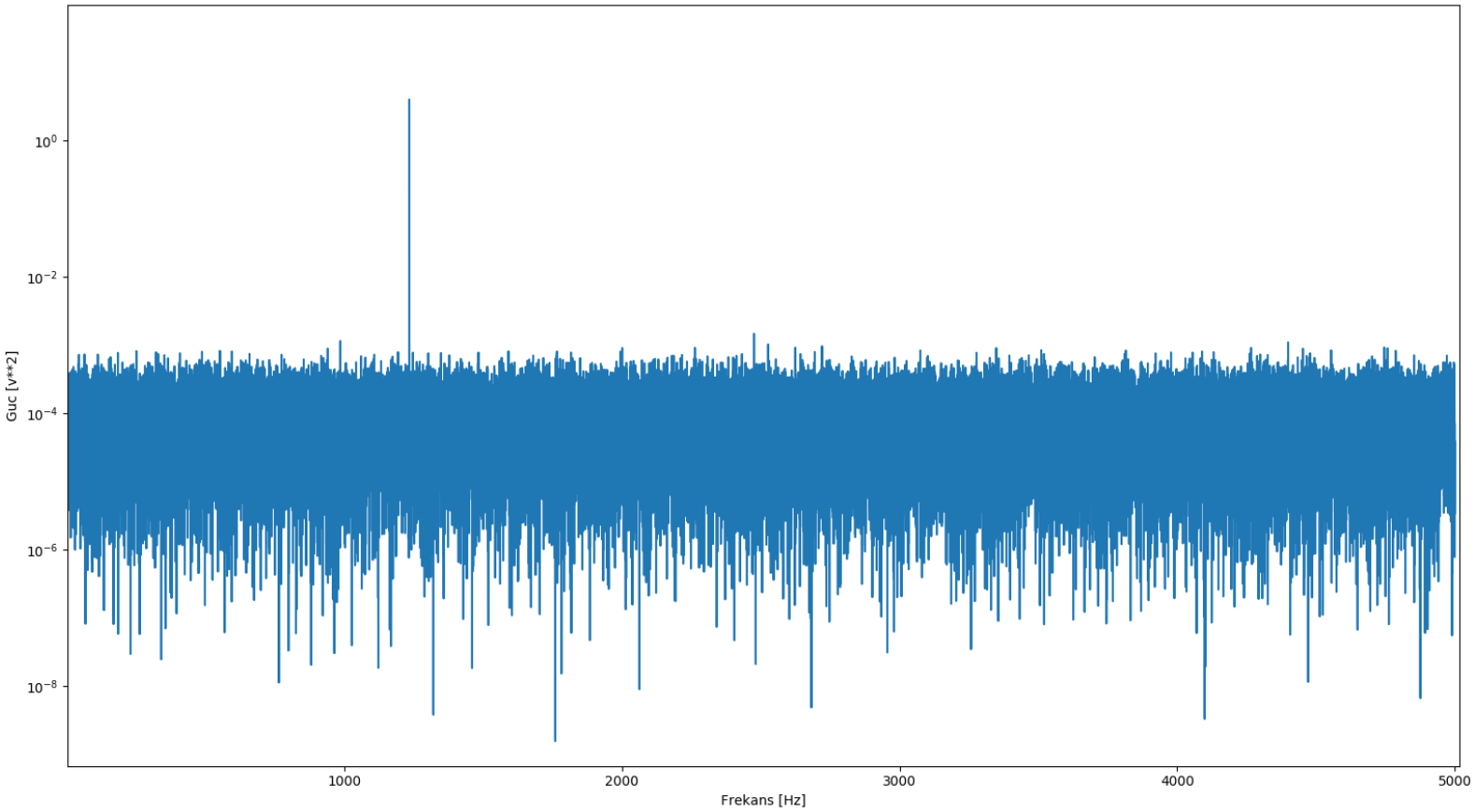
`scipy.signal.periodogram` örneklemenin yapıldığı frekansları ve bu frekanslardaki spektral güç yoğunluğunu (periyodogramı) verir. Aşağıdaki kod parçasıyla bu parametreleri alıp çizdirelim. Yalnız numpy.random.normal fonksiyonuyla ortalaması 0, standart sapması 1 olan bir normal dağılımdan rastgele çektiğimiz noktalarla kendi yarattığmız gürültünün frekans uzayındaki karşılığını göstermek üzere y-eksenini logaritmik çizdirelim (plt.semilogy). İsterseniz plt.plot ile lineer çizdirmeyi deneyebilirsiniz. Ayrıca gürültünün seviyesini de hesaplayıp, ekrana getirelim.

```
# frekanslari ve her bir frekanstaki spektral guc yogunlugunu
# periodogram fonksiyonunda alalim
frekans, guc_yogunlugu = signal.periodogram(x, fs)
plt.semilogy(frekans, guc_yogunlugu)
plt.xlabel('Frekans [Hz]')
plt.ylabel('Spektral guc yogunlugu [v**2/Hz]')
plt.show()
# Gurultu seviyesi
gurultu_f = np.mean(guc_yogunlugu[256:])
print("Beyaz gurultu seviyesi: ", gurultu_f)
# Lineer spektrumu cizdirelim
plt.semilogy(frekans, np.sqrt(guc_yogunlugu))
plt.xlabel('Frekans [Hz]')
plt.ylabel('Lineer spektrum [v]')
plt.show()
```

periyoodogram.py



periyoodgram.py koduyla 1234 Hz. frekanslı bir sinüs üzerinden kendi ürettiğimiz ve analiz etmek istediğimiz sentetik verinin spektral güç yoğunluğu diyagramı (periyodogramı). Y-ekseninde spektral güç yoğunluğu bulunmaktadır. Gördüğünüz gibi 1234 Hz. piki periyodogramda açıkça görünmektedir. Gürültünün gücü ortalama 0.0018'dir. Ortalama gürültü için kodda `guc_yogunlugu` dizisinin frekans pikinin bulunduğu frekanstan sonraki indeksleri almak üzere [256:] dilimlemesi yapılmıştır.



İstendiği takdirde spektral güç yoğunluğu yerine **güç spektrumu** (ing. power spectrum) adı verilen, genliğin karesiyle ölçekli spektral güç `scaling` parametresi "`spectrum`" değerine atanarak çizdirilebilir. Bu durumda y-ekseninde genliğin karesiyle ölçeklendirilmiş güç parametresi bulunur. Eksenin yine logaritmik olduğuna dikkat ediniz.

Genlik Spektrumu (ing. Amplitude Spectrum)

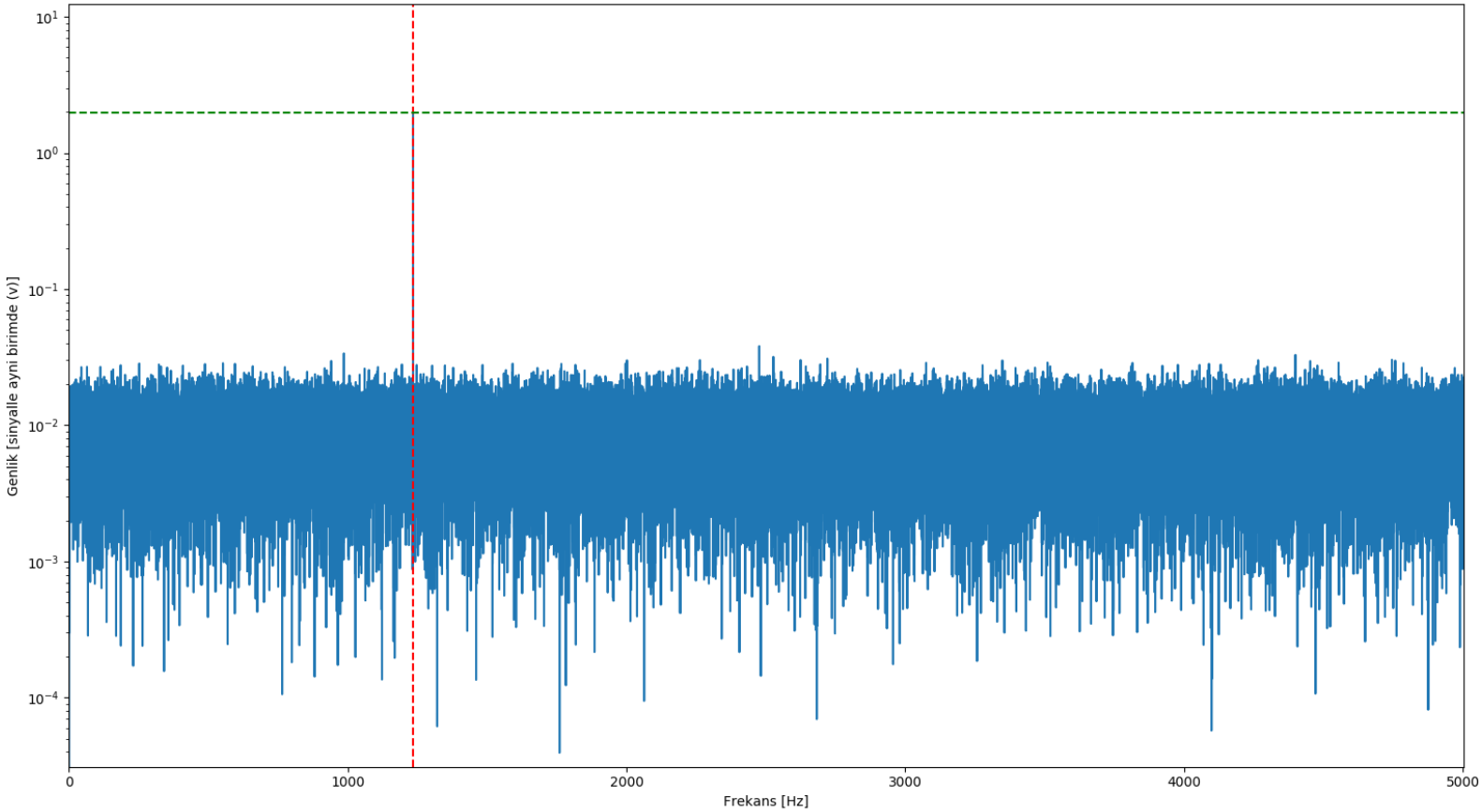
$\nu \neq \nu_1$ frekanslarında hem $x(t_i)$ sin $(2\pi\nu_1 t_i)$ terimleri, hem de $x(t_i)$ cos $(2\pi\nu_1 t_i)$ terimlerinde negatif ve pozitif terimler oluşur ve toplamda bu terimler birbirlerini büyük ölçüde sadeleştirdiği için toplam küçük olur. Yani verinin içindeki ν_1 frekansı dışındaki test frekanslarında P_N değeri küçük olur.

$P_N(\nu)$ maksiumum değerini verinin içindeki frekansta alır. 100 civarında nokta sayısından itibaren $P_N \approx A^2 N / 4$ gayet iyi bir yaklaşım haline gelir. Bu nedenle genliğin değerini direkt periyodogramdan görmek amacıyla frekansa karşılık

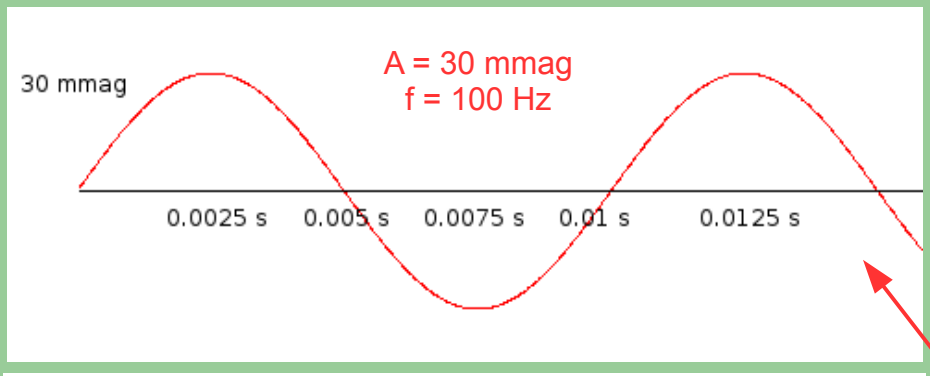
$$A(\nu) = \sqrt{\frac{4P_N(\nu)}{N}}$$

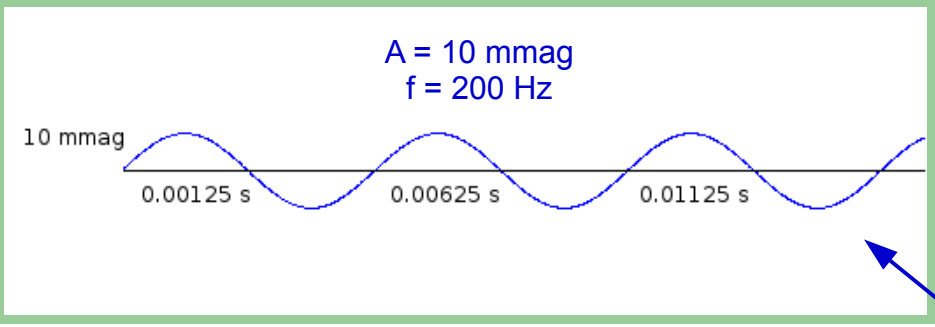
grafiği çizilir. Böylece her bir frekanstaki değişimin genliği direkt grafik üzerinden okunmuş olur. Bu şekilde oluşturulan periyodograma genlik periyodogramı ya da genlik spektrumu (ing. amplitude spectrum) adı verilir.

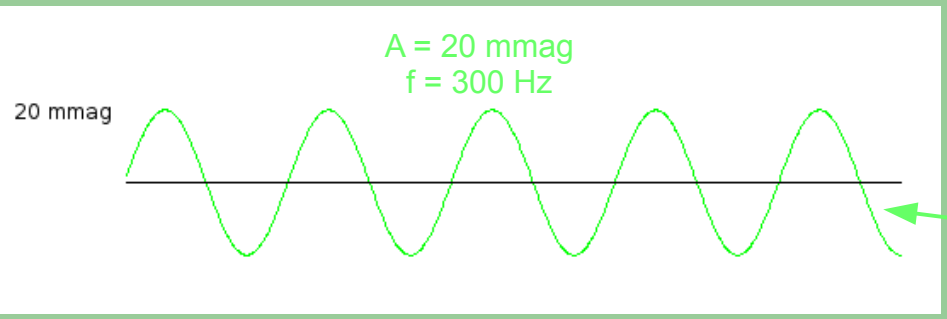
Sonuç olarak amacımız her bir değişim modunun genliğini belirlemek ve bu bilgiyi yorumlamaktır. Bu noktada dikkat çekilmesi gereken bir husus bilgisayar kodları ya da paket programlar aracılığıyla kullanılan periyodogramlarda elde edilen güç spektrumlarının ne şekilde ölçeklendiğidir. Örneğin sciy.signal.periodogram ekseni $P_N(v)^2 / N$ ile ölçeklendirir. Dolayısı ile **yarı-genlik** için verilen dizinin karekökünü almak yeterli olur.

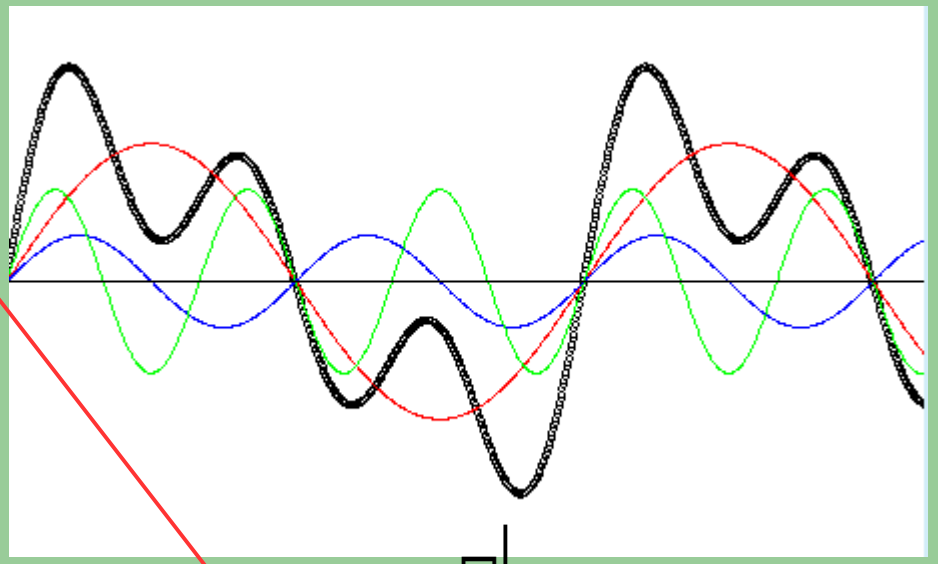


Son olarak kodumuzun maksimum güce sahip piki hangi frekansta bulduğunu ve bu pikte sinyalin genliğini numpy.max() ve numpy.argmax() fonksiyonlarını kullanarak elde edelim. Maksimum gücün olduğu frekanstaki genlik, güç spektrumunun o frekans için değerinin karekköküyle elde edilir. Pikin maksimumu 1234.00 Hz'te bullunmuştur ve genligi 2.0161'dır. Orjinall sinyalimizin yarı-genliği $2\sqrt{2} \sim 2.83$ idi ancak biz onun üzerine gürültü de eklemiştik. Gürültü barındıran verinin içindeki sinyalin genliğini belirlerken hata olması normaldir. Bu hatanın düzeyi RMS = $\sqrt{\Sigma(PSD*df)} = 0.0278$ ile verilir. Burada df: tayfsal çözünürlük olup, iki nokta arasındaki frekans biriminden uzaklıktır. Sinyal üzerindeki gürültüyü rastgele ürettiğimizden bu değerler sizin için farklı olabilir.

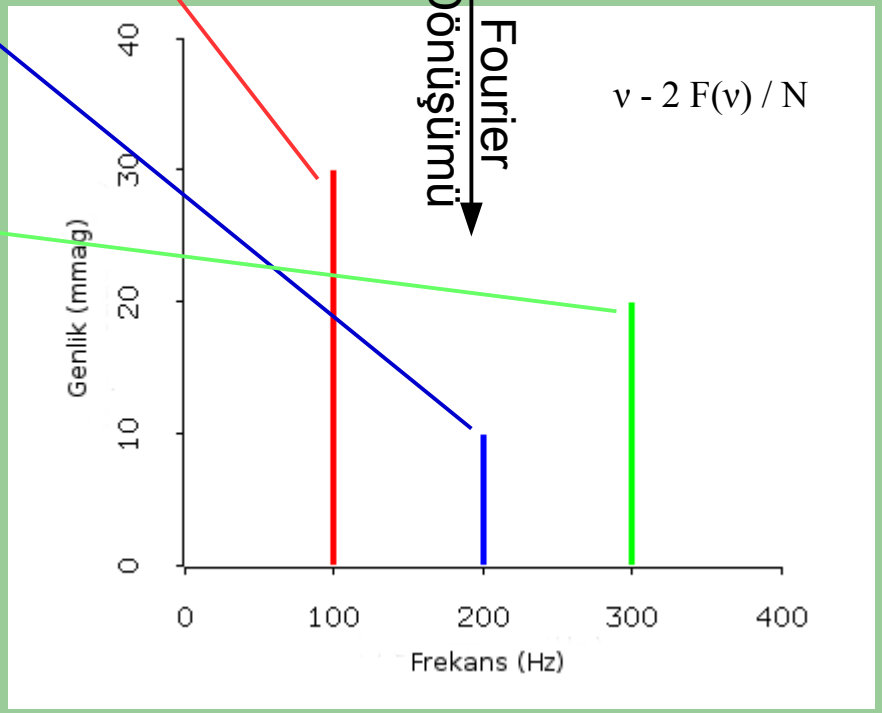




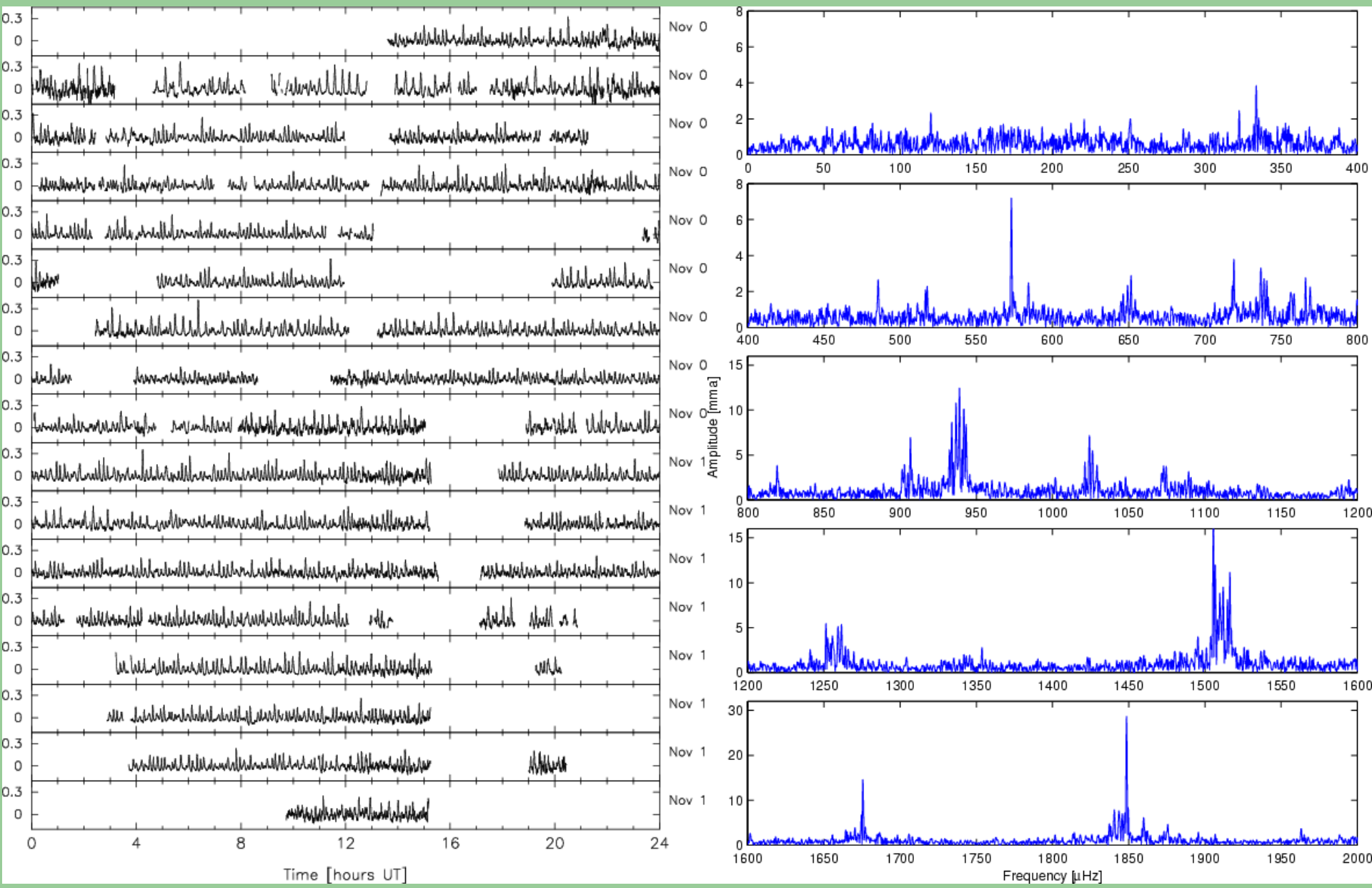




Fourier Dönüşümü



Yeteri sıklıkta ve hatası çok küçük ölçümler almamız durumunda sağdaki genlik spektrumundan yukarıdaki üç sinyali ayrı ayrı elde edebilmemiz gerekir. Ancak bunun ideal durum olduğunu belirtmemiz gerekir.



ZZ Ceti yıldızı HL Tau 76'nın 16 günlük ışık eğrisi (solda) ve genlik spektrumu (sağda) (Dolez vd. 2006). Görüldüğü üzere asterosismolojide gerçek durum idealdan uzaktır.

Lomb-Scargle Periyodogramı

Fourier analizinden farklı olarak veriye en küçük kareler yöntemiyle sinüs fonksiyonları uyumlamaya (fit etmeye) dayanır. Fourier analizi sadece eş aralıkla (Δt) alınmış veriye uygulanabilirken en küçük kareler yöntemiyle spektral analiz eş aralıkla alınmamış veri setine de uygulanabilir. Temelinde

$$\phi(t) = A\sin\omega t + B\cos\omega t.$$

şeklindeki bir sinüsoidal fonksiyonu veriye uyumlamak vardır. Bunun için τ parametresi (referans epoch), t_i zamanı için aşağıdaki şekilde tanımlanır.

$$\sum_{i=1}^{N}\cos[2\pi\nu(t_i-\tau)]\sin[2\pi\nu(t_i-\tau)] = 0,$$

$$\tan(4\pi\nu\tau) = \frac{\sum_{i=1}^{N}\sin(4\pi\nu t_i)}{\sum_{i=1}^{N}\cos(4\pi\nu t_i)}.$$

Bu durumda i. gözlemsel veri $x_i = x(t_i)$ olmak üzere ω frekansı için Lomb-Scargle periyodogramı

$$P_{\rm LS}(\nu) = \frac{1}{2}\frac{\left\{\sum_{i=1}^{N}x(t_i)\cos[2\pi\nu(t_i-\tau)]\right\}^2}{\sum_{i=1}^{N}\cos^2[2\pi\nu(t_i-\tau)]} + \frac{\left\{\sum_{i=1}^{N}x(t_i)\sin[2\pi\nu(t_i-\tau)]\right\}^2}{\sum_{i=1}^{N}\sin^2[2\pi\nu(t_i-\tau)]}.$$

şeklinde hesaplanır. Bu haliyle periyodogram ortalaması 0'da olan bir veri seti için geçerlidir. Bu problemi aşmak için bir yol periyodogram hesaplanmadan önce verinin ortalamasını çıkarmaktır. Diğer bir yol ise aşağıdaki şekilde bir sinüsoidal fiti en küçük kareler yöntemiyle yapıp periyodogramı o şekilde oluşturmaktır. Bu şekilde oluşturulan periyodograma genelleştirilmiş Lomb-Scargle periyodogramı adı verilir.

Lomb-Scargle Periyodogramı

Aynı şekilde N (nokta sayısı) yeterince büyük olduğunda ν_1 frekansındaki güç değeri yine A^2 N / 4 'e yakınsar. Lomb-Scargle periyodogramına dayalı genlik spektrumunda genlik

$$A_{\mathrm{LS}}(\nu) = \sqrt{\frac{4P_{\mathrm{LS}}(\nu)}{N}}$$

Lomb-Scargle periyodogramının önemli bir avantajı eş aralıklı ve / veya sinüsoidal olmayan veriye de uygulanabilmesidir. Bu yöntem de temelinde en küçük kareler yöntemiyle sinüs uyumlamaları (fiti) ile aynı sonucu verdiği için parametrik olmayan yöntemlerin (en küçük kareler yöntemiyle iteratif sinüs fiti) etkinliğini göstermek bakımından önemlidir.

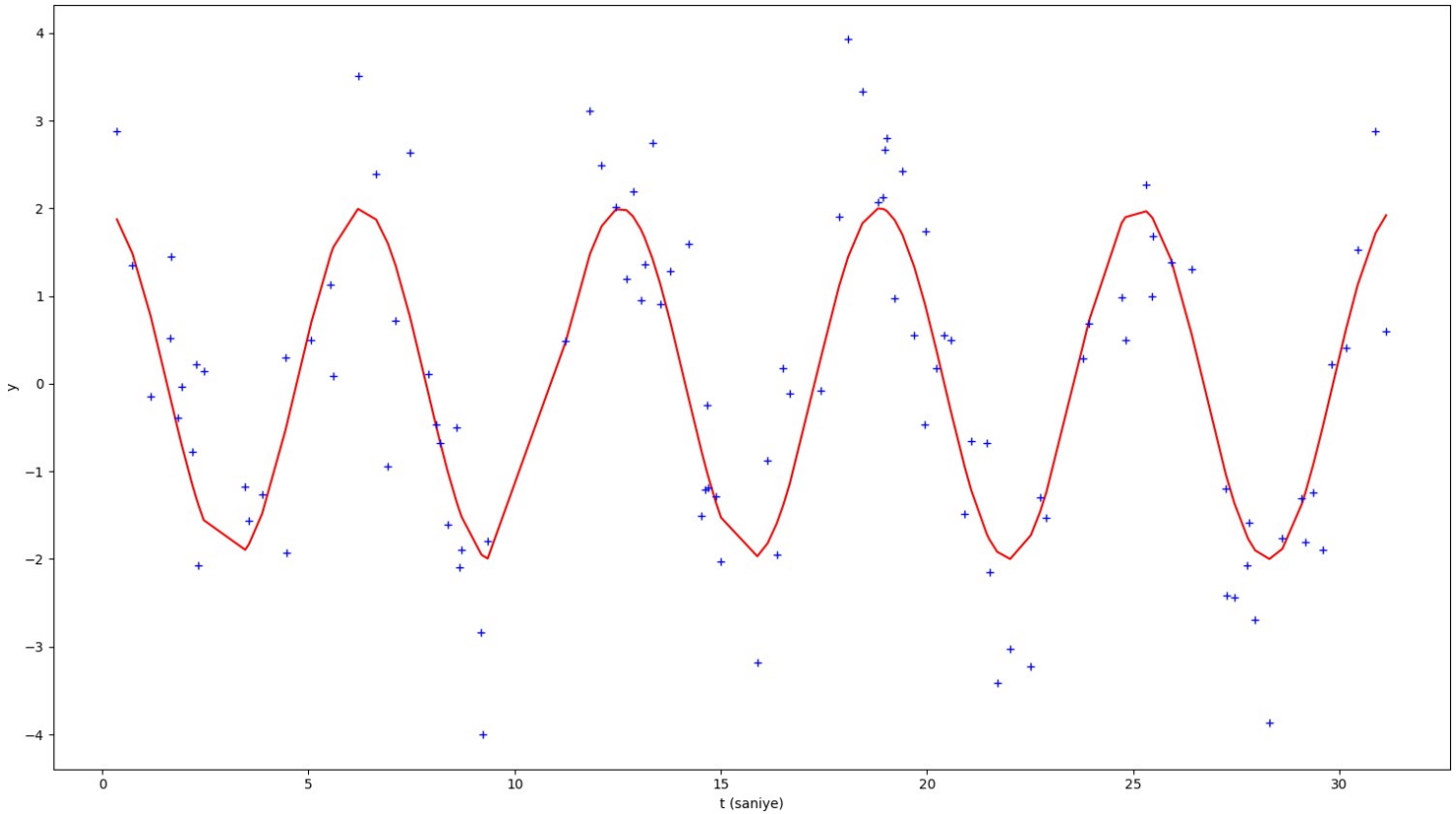
Aarhus grubu bu nedenle halen iteratif sinüs fiti yöntemini parametrik yöntemlere (Fourier analizine dayanan yöntemler) tercih etmekte, böylece hesaplama yükü, frekans örtüşmeleri (aliasing) ve frekans girişimleri gibi istenmeyen sonuçlardan kaçınmaktadır. Bu analiz yöntemlerinin yerini giderek Bayesian istatistiğe dayalı yöntemler (peak bagging) almaktadır.

scipy.signal.lombscargle

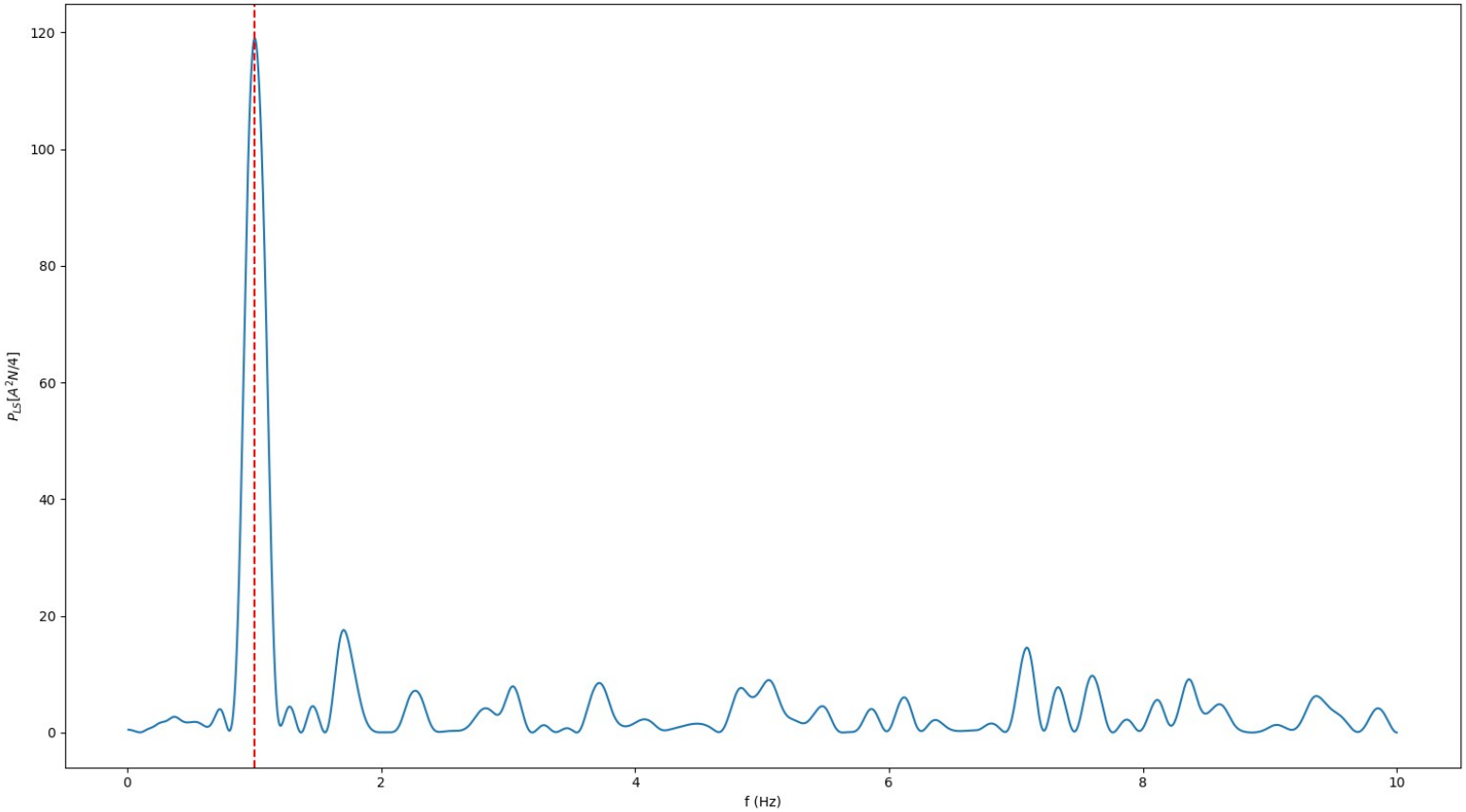
`scipy.signal.lombscargle` Lomb [(1976) tarafından eşit aralıklarla alınmamış (eş frekansla örneklenmemiş) zayıf periyodik sinyallerin analizi için üretilen ve Scargle (1982) tarafından daha da geliştirilen Lomb-Scargle periyodogramlarının üretilmesine olanak sağlayan bir fonksiyondur. Normalizasyon yapılmazsa (`normalize = False`), yeterince büyük N için A genliğine sahip bir harmonik sinyal söz konusu olduğunda (A^2) N / 4 değerini alır. Normalizasyon (`normalize = True`), sabit referans modelden (0 noktasına) farklara göre yapılır. Yine öncelikle bir sentetik veri seti üretelim.

```
A = 2. # genlik
w = 1. # acisal frekans
phi = 0.5 * np.pi # aci
N1 = 1000
N2 = 100000
# Veriyi farkli zaman araliklarinda alinmis bir veriye
# donusturmek uzere karsilastirma amaciyla kullanacagimiz
parametre
atla = 0.9
# ortalamasi 0, standart sapmasi 1 olan bir normal dagilimdan
# 1000 nokta secelim
r = np.random.rand(N1)
# 0.01 ile 10pi arasinda 1000 noktadan olusan bir x eksenimzi
olsun
x = np.linspace(0.01, 10*np.pi, N1)
# dagilimdan secilen sayi 0.9'dan buyukse alalim
# onun karsilik geldigi indeksteki x'i alalim yoksa almayalim
x = x[r >= atla]
# Sinyalimizi hazırlayalim
y = A * np.sin(w*x+phi)
plt.plot(x,y,'r-')
# Biraz sentetik gurultu ekleyelim
y += np.random.normal(size=len(x))
plt.plot(x,y,'b+')
plt.xlabel("t (saniye)")
plt.ylabel("y")
plt.show()
```

lombscargle.py

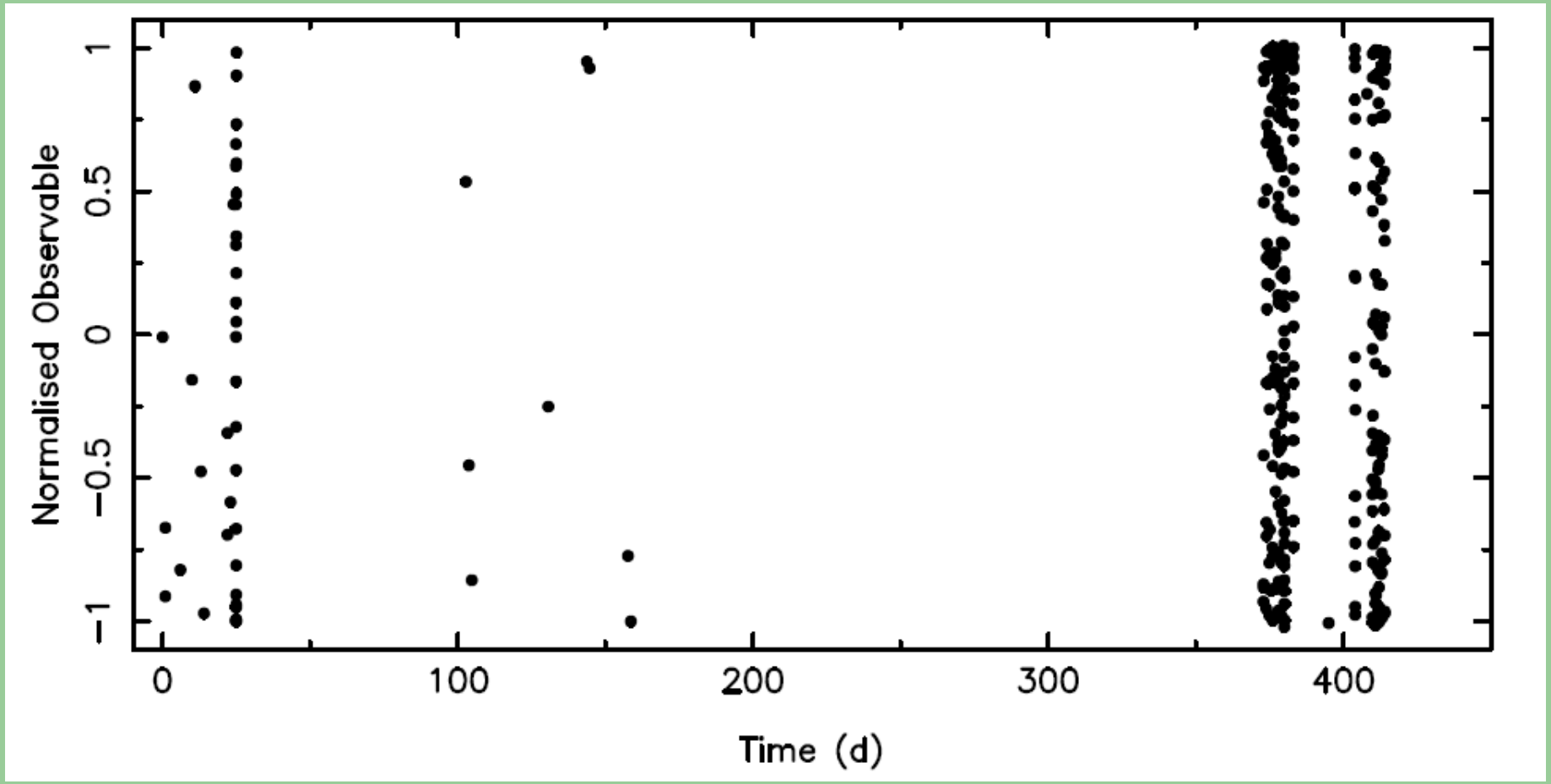


lombscargle.py koduyla ürettiğimiz sinüsoidal sinyal (kırmızı) ve üzerine gürültü bindirilmiş hali (mavi +). Analiz ederek içerisindeki frekansı belirlemek istediğimiz veri, bu mavi veri noktalarıdır.

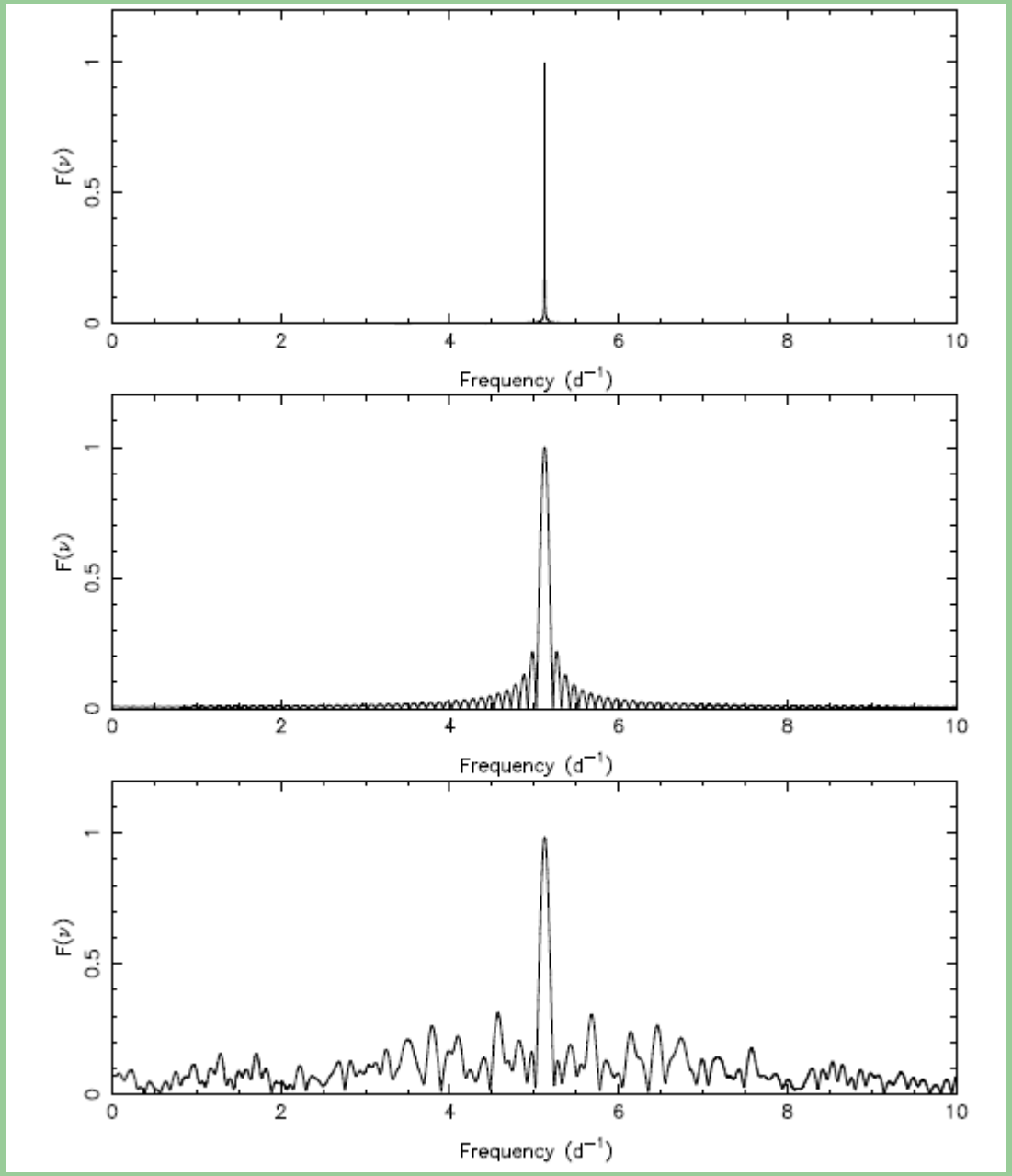


Son olarak kodumuzun maksimum güce sahip piki hangi frekansta bulduğunu ve bu pikte sinyalin genliğini numpy.max() ve numpy.argmax() fonksiyonlarını kullanarak elde edelim. Pikin maksimumu 1.0043 Hz'te bullunmuştur ve genligi 2.7270'tirr. Orjinall sinyalimizin yarı-genliği 2.0 idi ancak biz onun üzerine gürültü de eklemiştik. Gürültü barındıran verinin içindeki sinyalin genliğini belirlerken hata olması normaldir. Lomb-Scargle periyodogramı (scipy.signal.lombscargle) tarafından üretilen 4 A^2 / N ile ölçekli olduğundan genliğe geçmek için periyodogramın maksimum değerinin karekökü 4'e bölünmelidir.

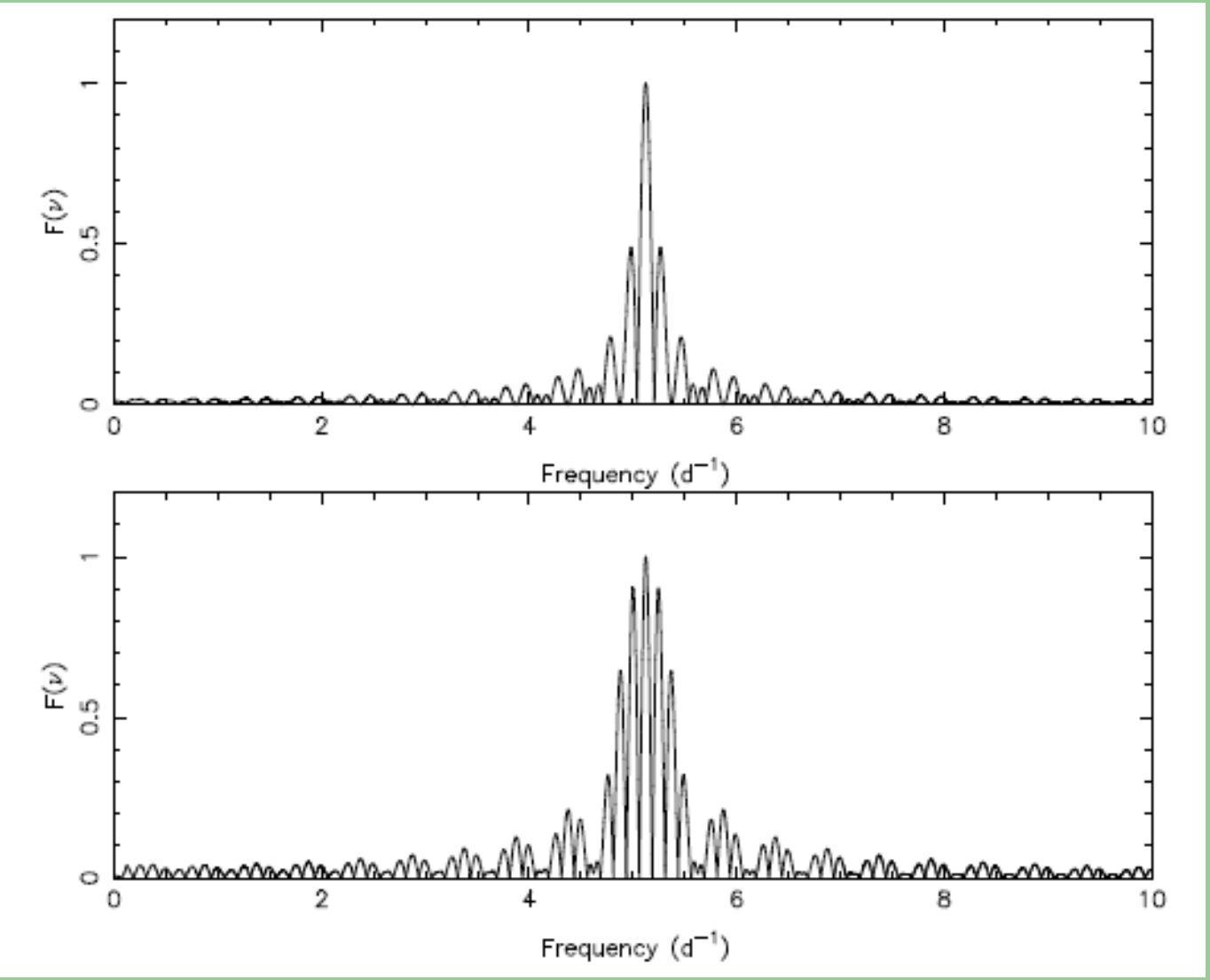
Örnek



Yerden gözlenmiş bir değişen yıldız verisi. Noktalar arası uzaklığın medyan değeri 0.012 gündür (Nyquist frekansı 42 gün^{-1}). Eş zaman aralıklı alınmamış veri söz konusu olduğunda Nyquist frekansı hesabı için noktalar arası uzaklığın medyan değeri ya da ortalaması kullanılır. Verinin standart sapması 0.696, üzerindeki beyaz gürültünün (her frekansta aynı standart sapmaya sahip gürültü) standart sapması 0.01111'dir.

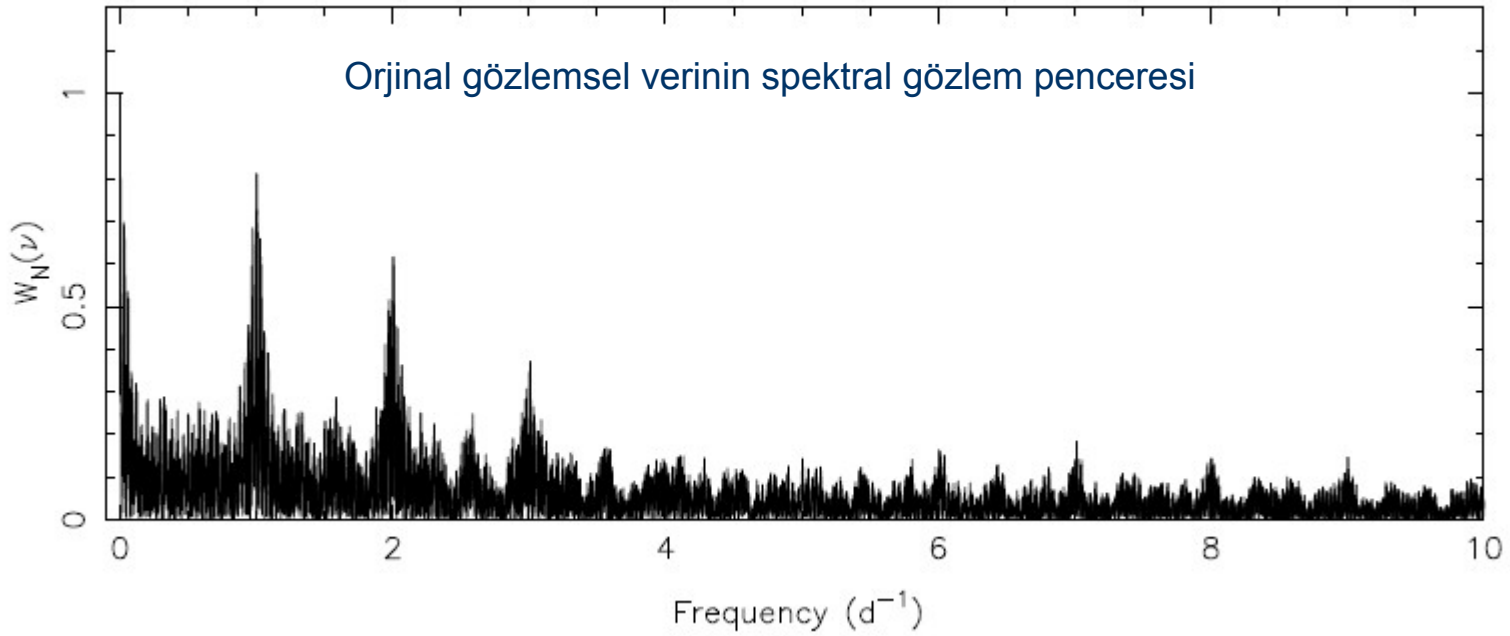


Veri seti hiç gürültü (hata) içermeden alınacak 1000 güne dağılmış 1 milyon nokta (en üstte), 10 güne dağılmış 10000 nokta (ortada) ve aynı zaman aralığında boşluklar da içeren 4472 nokta (altta) içerdiğinde elde edilen Fourier dönüşümleri (FFT).



Aynı sinyal yine hata içermeyecek şekilde 10 günlük bir zaman aralığında 4 ile 6. gün arasında gözlem noktası olmadan (üstte), 2 ile 8. gün arasında gözlem noktası olmadan gözlendiğinde elde edilen Fourier dönüşümleri.

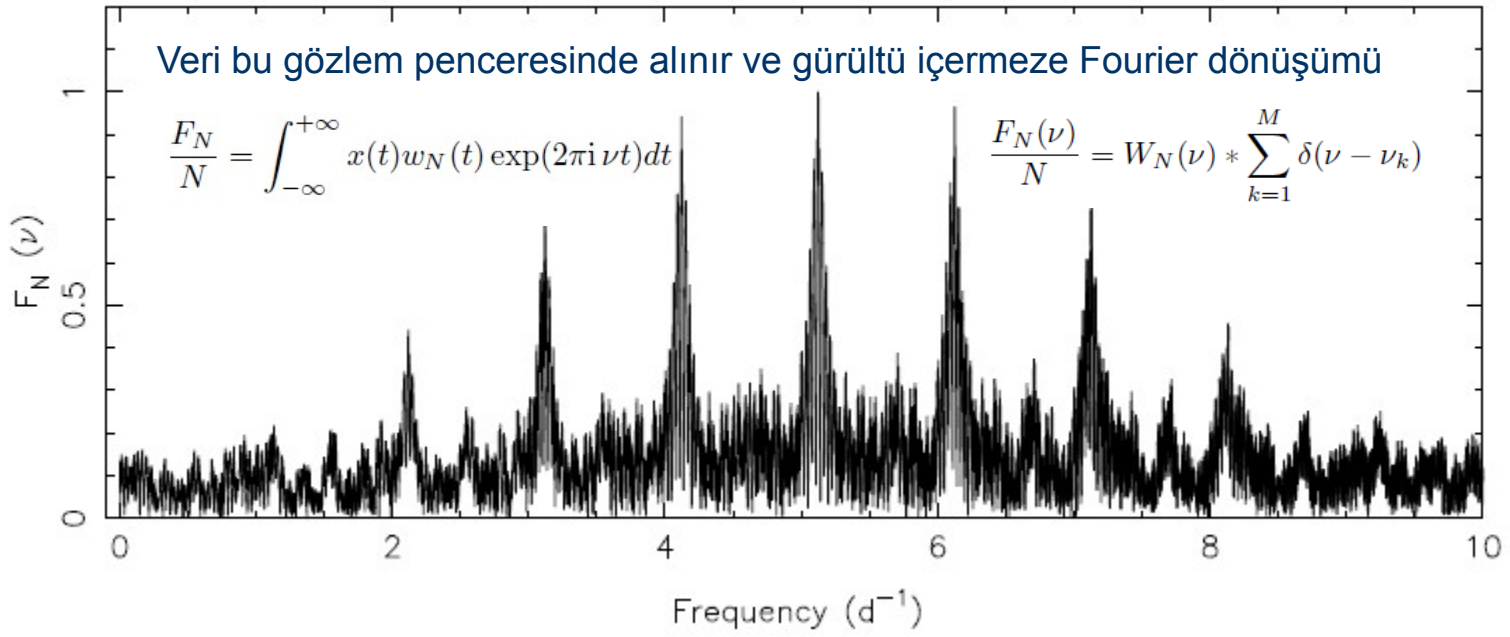
Orjinal gözlemsel verinin spektral gözlem penceresi

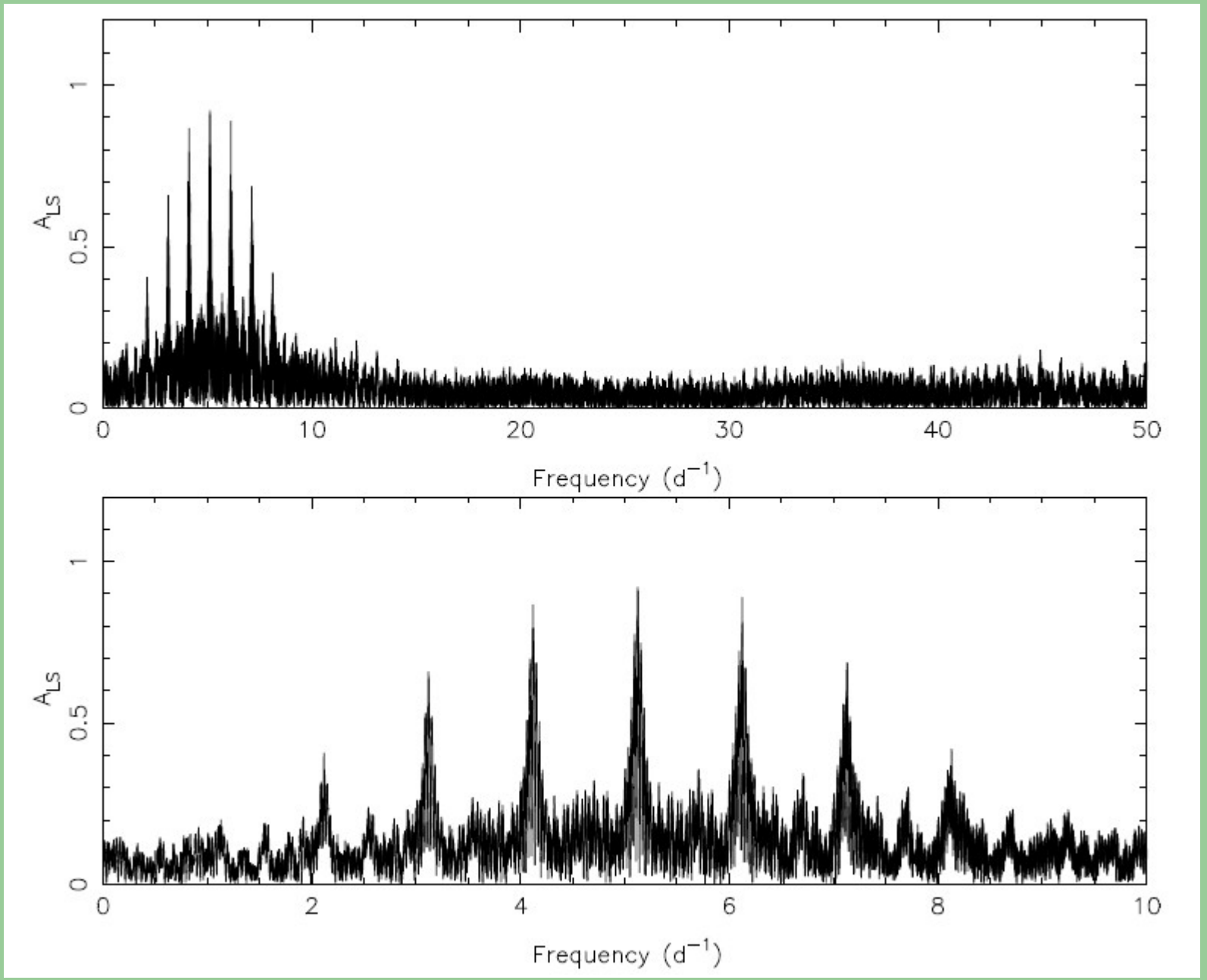


Veri bu gözlem penceresinde alınır ve gürültü içermeze Fourier dönüşümü

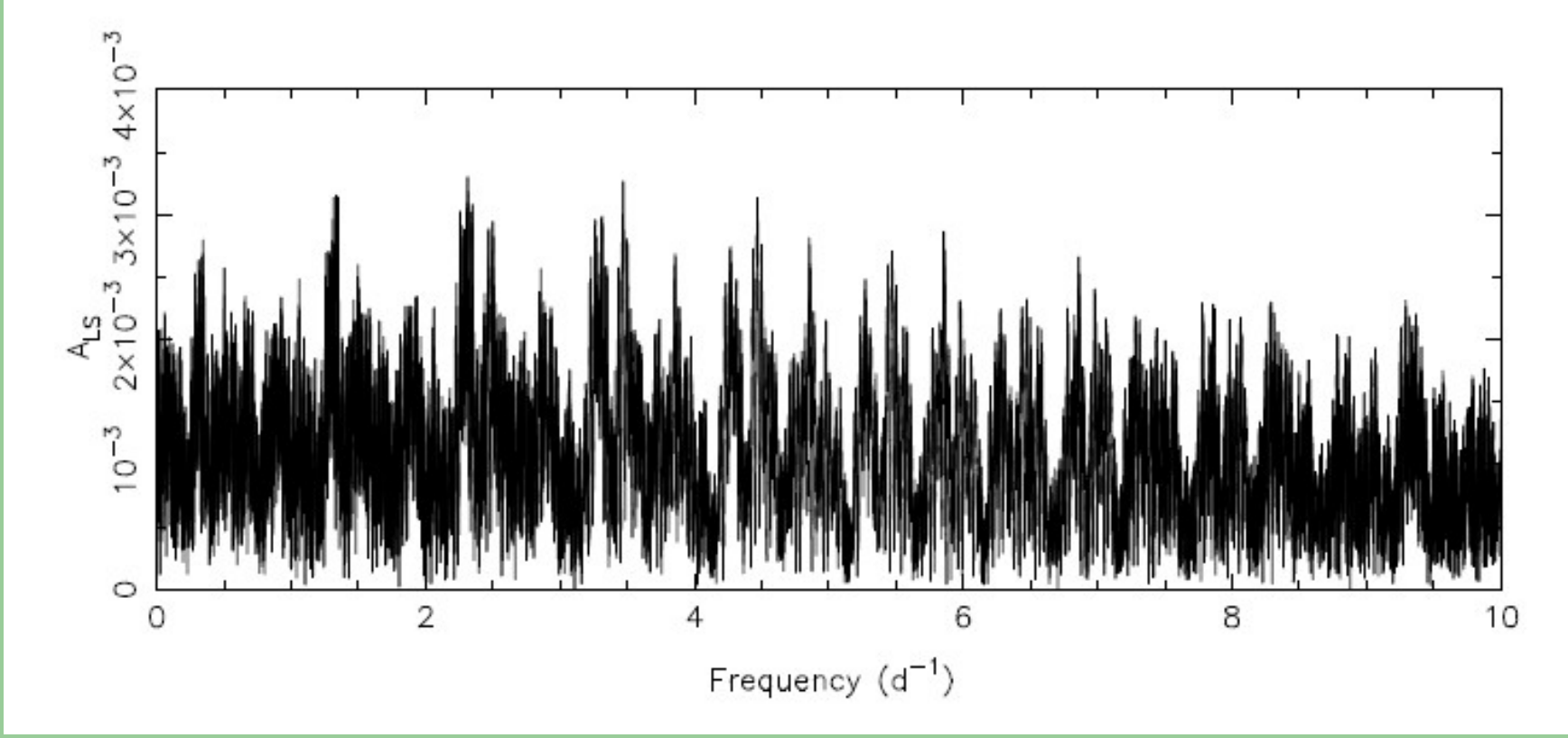
$$\frac{F_N}{N} = \int_{-\infty}^{+\infty} x(t) w_N(t) \exp(2\pi i \nu t) dt$$

$$\frac{F_N(\nu)}{N} = W_N(\nu) * \sum_{k=1}^{M} \delta(\nu - \nu_k)$$





Orjinal verinin Lomb-Scargle periyodogramı. Periyodogramdaki maksimum güç 5.123456789 gün^{-1} frekansındadır. Aşağıda Nyquist frekansına (~10.24 gün^{-1}) kadarki bölüm.

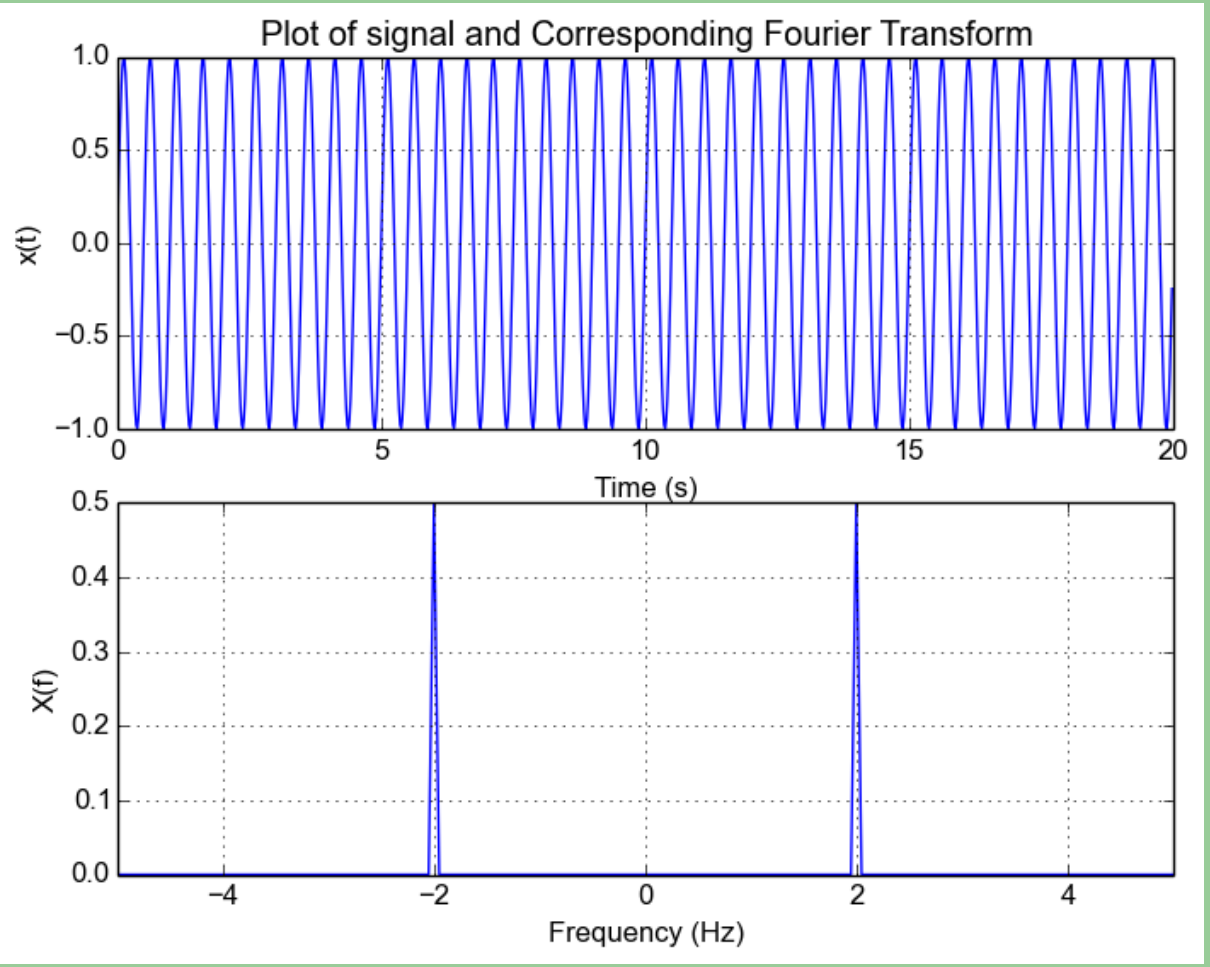


5.123456789 gün^{-1} frekansından arındırılmış verinin artığının (prewhitened) Lomb-Scargle periyodogramı.

Spektral Sızma (ing. Spectral Leakage)

Öncelikle sürekli ve 2Hz frekansına sahip basit bir sinüsoidal fonksiyonu düşünelim → $x(t)=\cos(4\pi t)$.

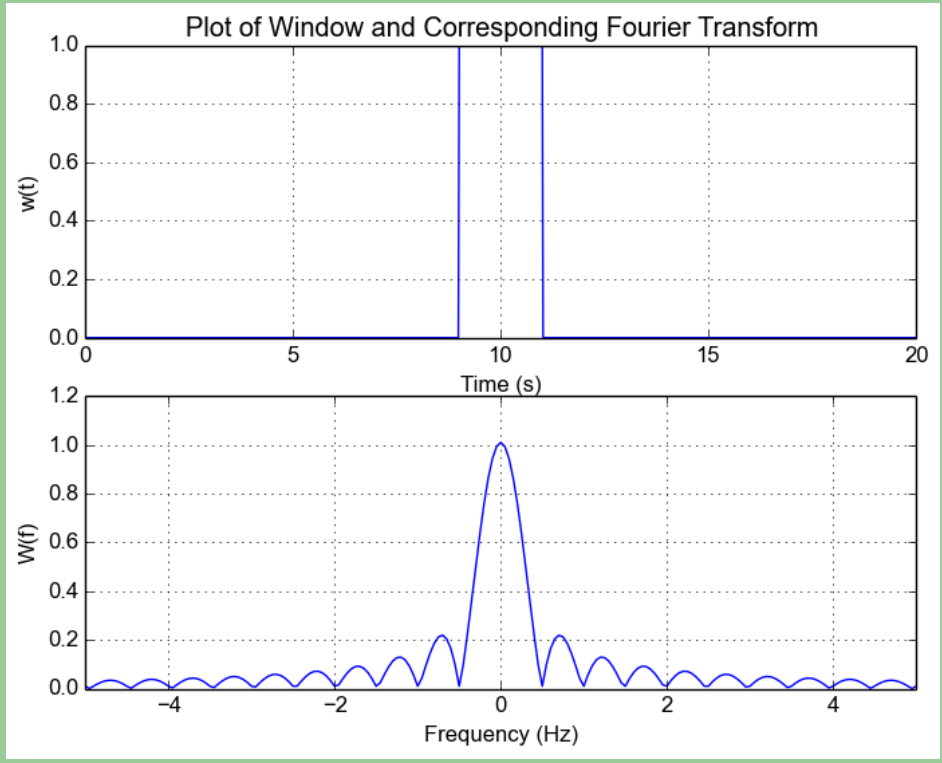
Bu sinyalin sürekli Fourier dönüşümü $X(f)=\delta(f \pm 2)$ verir.

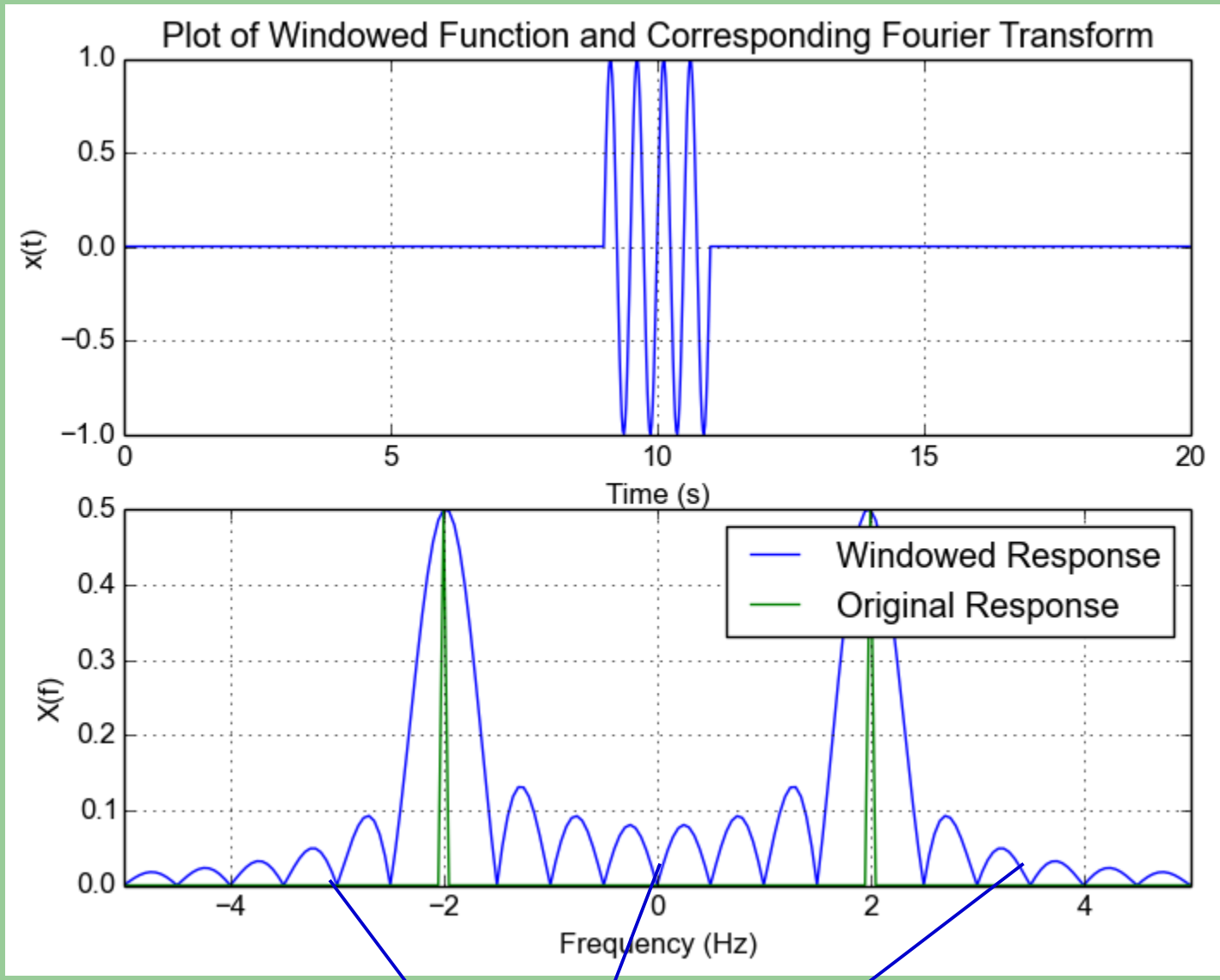


http://saadahmad.ca/fft-spectral-leakage-and-windowing/

Ancak gerçekte biz bu (-∞, ∞) aralığındaki sinyalin sadece bir bölümünü, bir gözlem penceresi (window function) dahilinde gözleriz. $\hat{x}(t) = x(t)w(t)$. Gözlem penceresiyle çerçevelenmiş bu sinyalin Fourier dönüşümü $\hat{X}(f)=X(f)*W(f)$ şeklinde verilir (çarpımın Fourier dönüşümünün konvolüsyon olduğunu hatırlayınız). Burada W(f) spektral pencere (ing. spectral window) adını alır.

Diyelim ki gözlem penceremiz bir dikdörtgen olsun (astronomide genellikle böyledir). Ölçümümüzü $\Delta t = t_2 - t_1$ aralıklarla alıyor olalım. Bu durumda gözlem penceremiz $w(t)=u(t-t_1)-u(t-t_2)$ ve frekans tanım kümesinde karşılık geldiği gözlem penceresi $W(f)=e^{-i2\pi ft1}\cdot\mathrm{sinc}(\Delta t f_2)\cdot\Delta t$ olur. Bu noktada kutu fonksiyonunun Fourier dönüşümünün bir sinc (sinx / x) fonksiyonu olduğunu hatırlayınız.

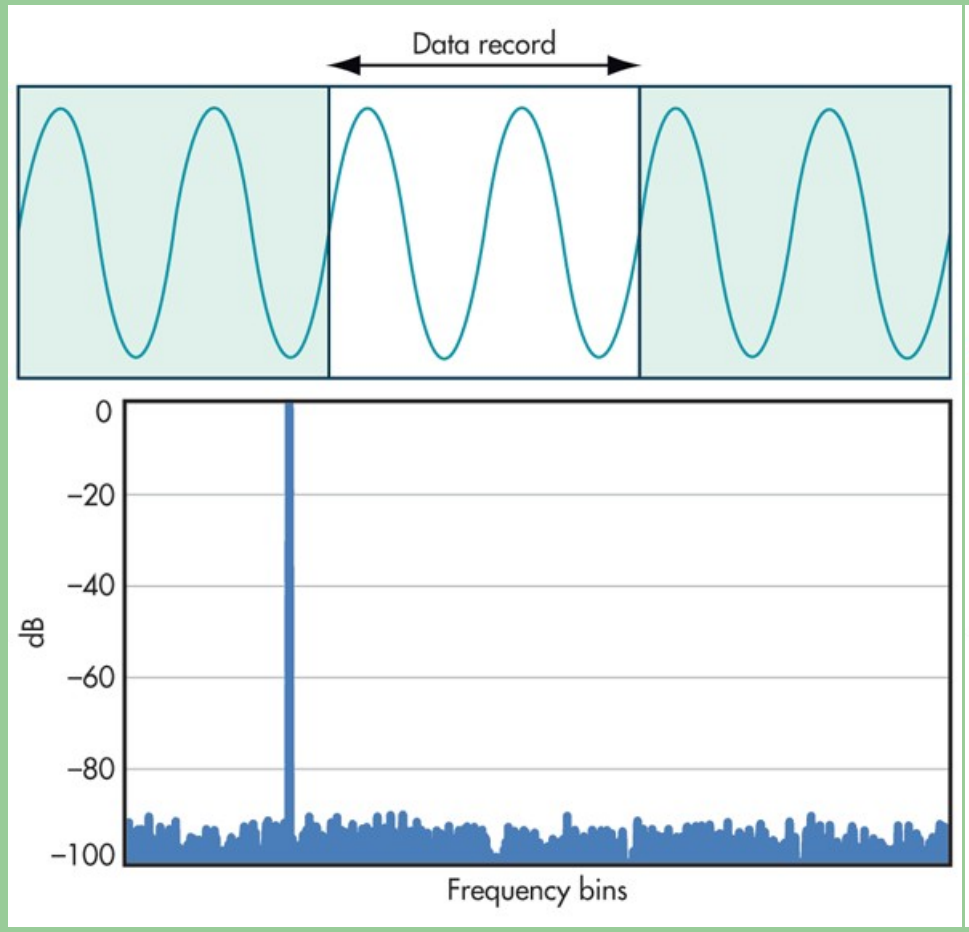




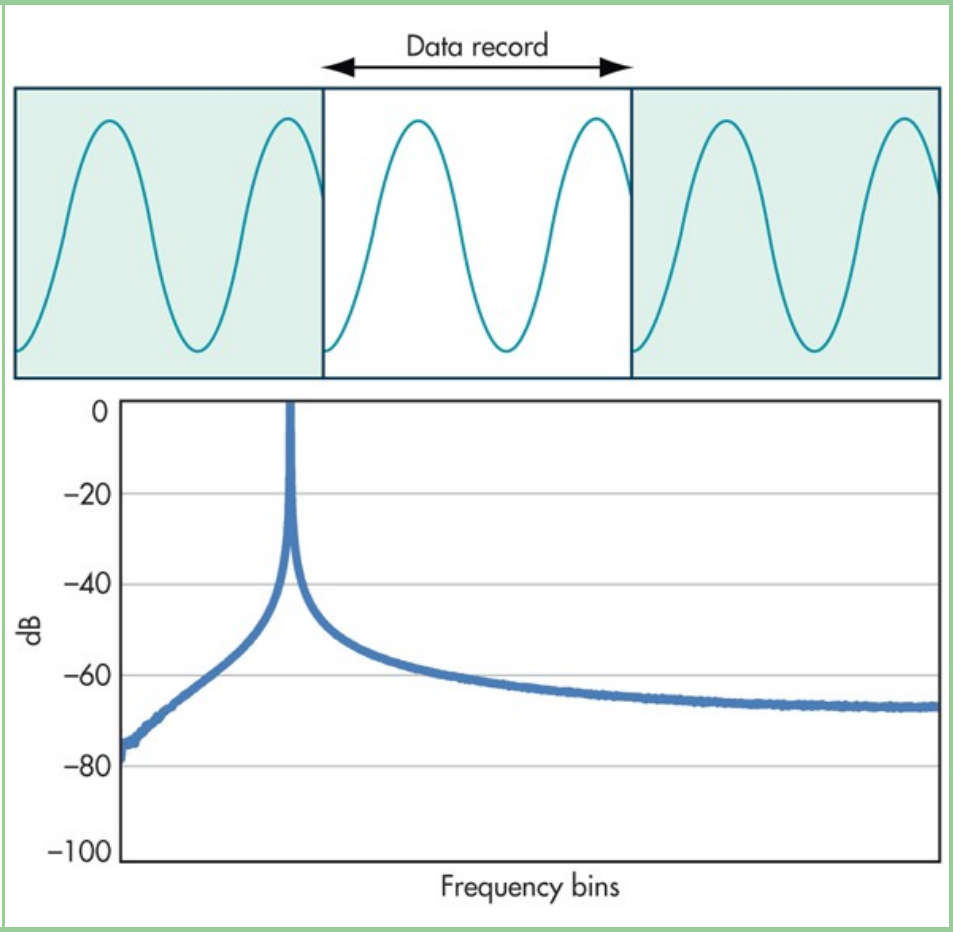
http://saadahmad.ca/fft-spectral-leakage-and-windowing/

Diğer frekanslara “sızan” pikler

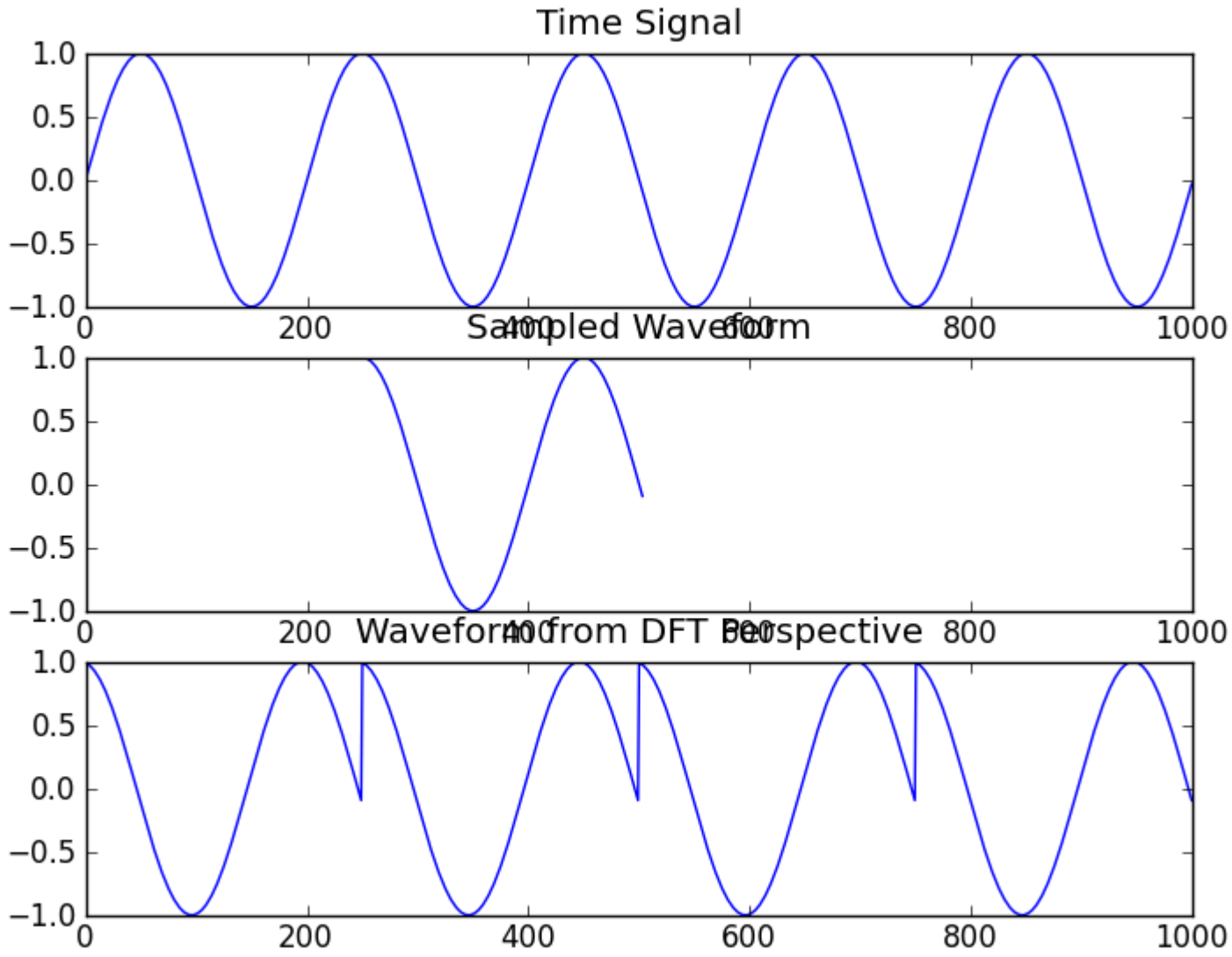
Spektral sızma, gözlenen çevrim sayısının tam olmamasından da kaynaklanır. Özünde bu da bir gözlem penceresi etkisidir.



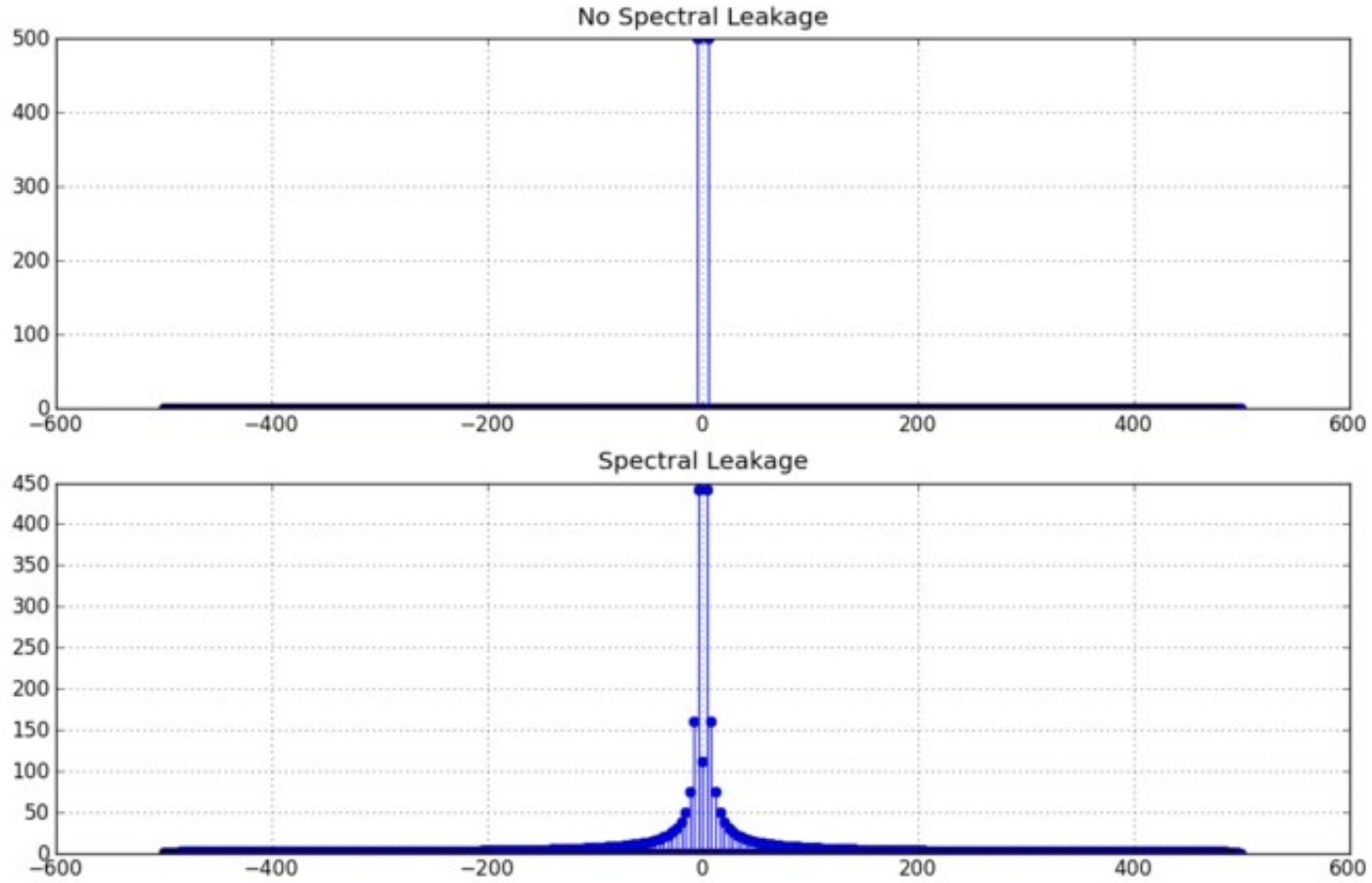
Tam sayıda (2) çevrimin gözlendiği durum. Üstte zaman serisi altta Fourier dönüşümü görülmektedir.



Tam sayıda çevrimin gözlenemediği bu nedenle de sinyalin periyodik görünmediği süreksiz durum. Üstte zaman serisi altta Fourier dönüşümü görülmektedir.



Üstte (-∞, ∞) aralığında sürekli sinyal, ortada gözlem penceresi çerçevesinde gözlenen sürekli bantla sınırlı (ing. Band-limited) sinyal, en altta bu sinyale uygulanan Fourier dönüşümünün ters dönüşümünden elde edilen sinyal.



Bant-limitli olmayan sinyalin Fourier dönüşümü (üstte), tam çevrim sayısında gözlenmemiş, bant-limitli sinyalin Fourier dönüşümü (altta).

Kaynaklar

- Julius Orion Smith III Homepage, “Mathematics of the Discrete Fourier Transform” (DFT)”https://ccrma.stanford.edu/~jos/
- www.fouriertransform.com
- Python örnekleriyle Hızlı Fourier Dönüşümleri: https://www.oreilly.com/library/view/elegant-scipy/9781491922927/ch04.html
- Periyodogram örnekleri: https://www.programcreek.com/python/example/100530/scipy.signal.periodogram
- scipy.signal.periodogram Dokümantasyonu: https://docs.scipy.org/doc/scipy-0.13.0/reference/generated/scipy.signal.periodogram.html
- scipy.signal.lombscargle Dokümantasyonu: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lombscargle.html