

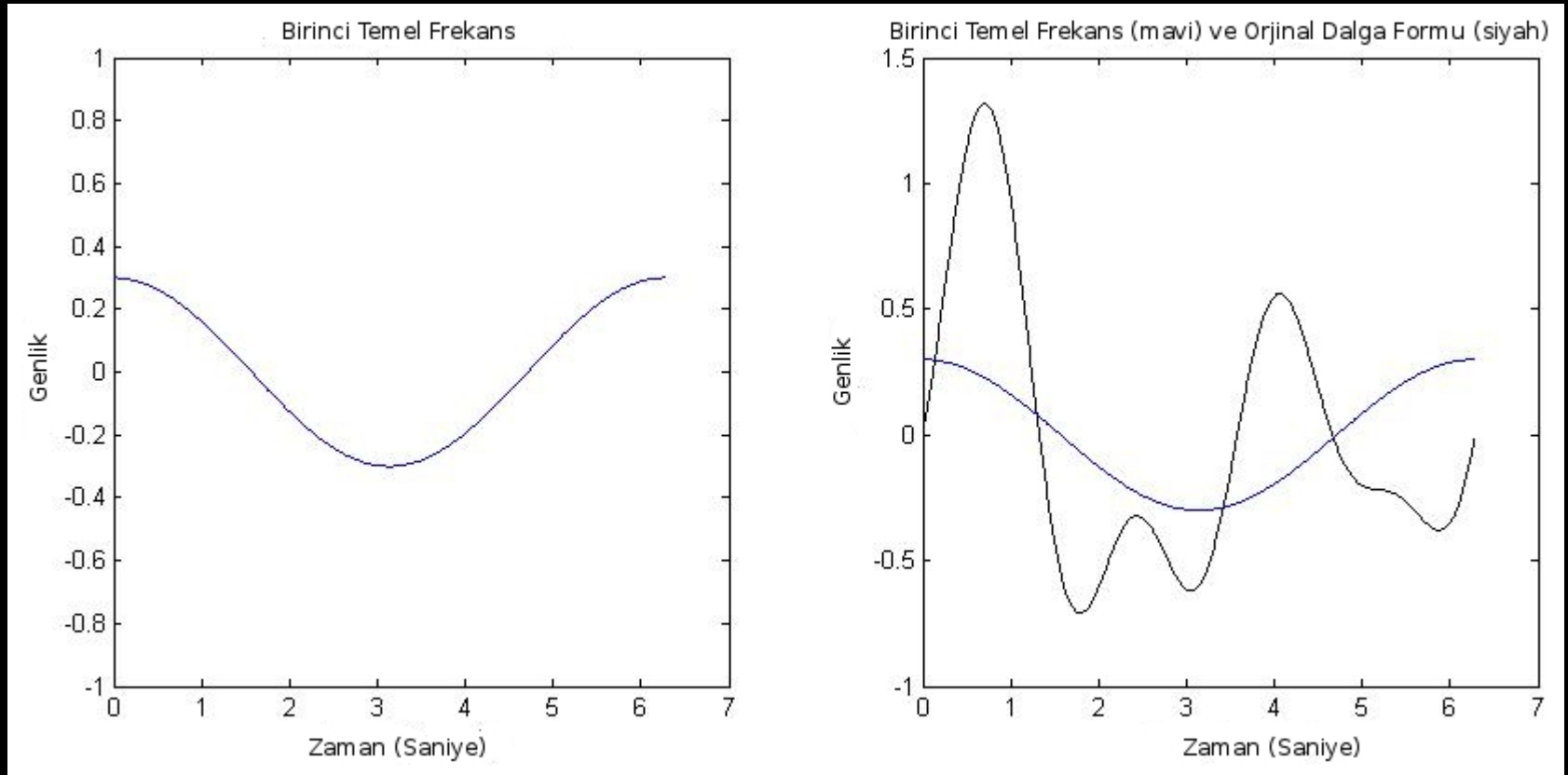
A416

Astronomide Sayısal Çözümleme - II

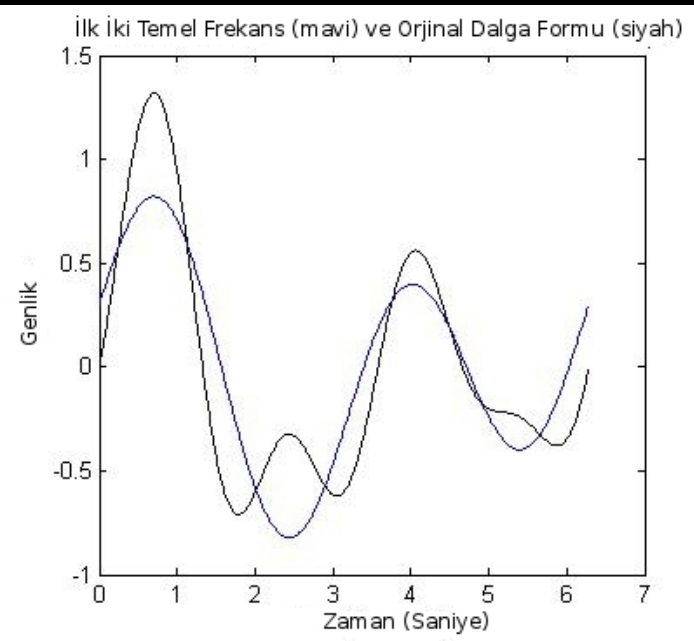
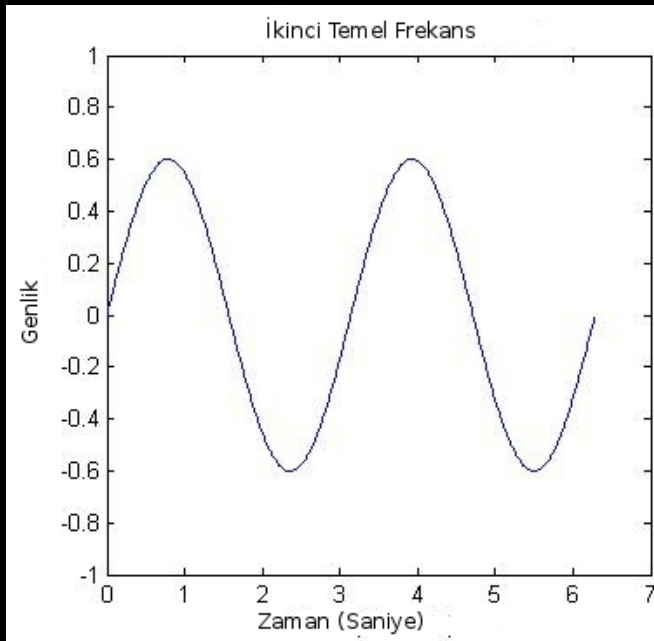
8. Fourier Dönüşümleri

Fourier Dönüşümü

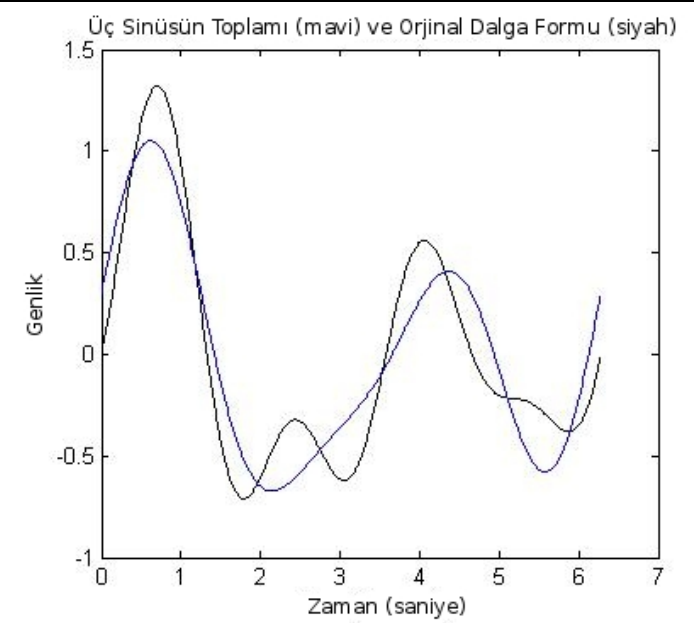
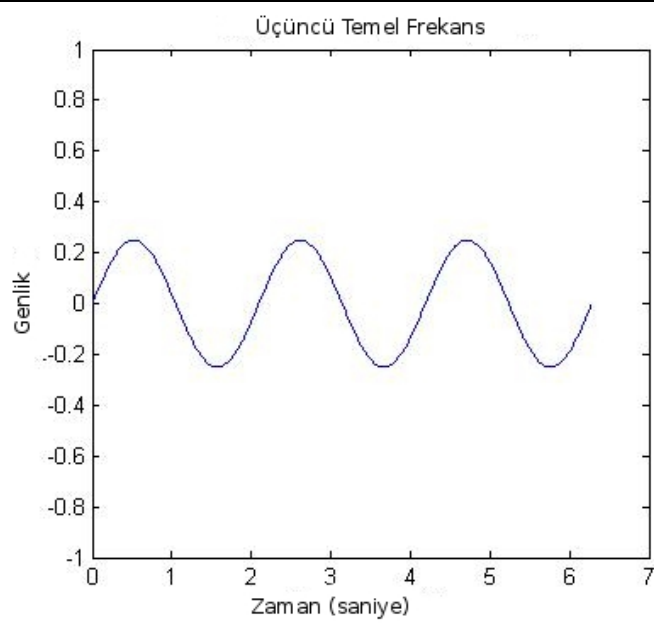
Fourier Dönüşümü bir dalga formunu sinüs ve kosinüs fonksiyonlarının bir kombinasyonu olarak ifade etmek üzere yapılan matematiksel işleme verilen isimdir. Neredeyse her şey (zamana bağlı bir fonksiyon ya da sinyal, elektromanyetik dalgalar, ses dalgaları, hisse senetlerinin fiyat değişimi gibi) bir dalga formu şeklinde tanımlanabilir. Fourier Dönüşümü bu formlarla işlem ve değerlendirme yapmak üzere kullandığımız oldukça güçlü bir araçtır. **“Evrende gözlediğiniz tüm dalga formları farklı frekans ve genliklere sahip sinüs fonksiyonlarının toplamından ibarettir!”**



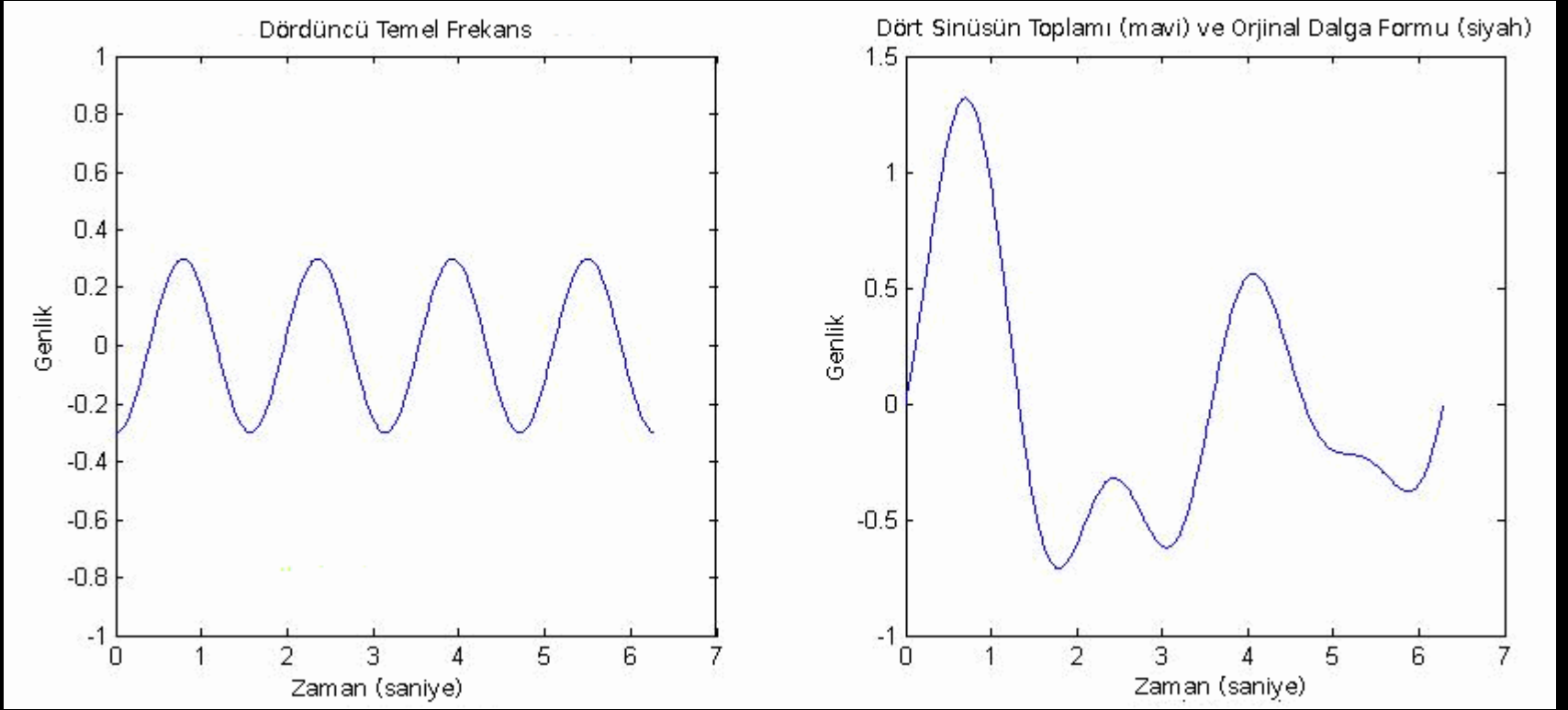
Şekil 1. Orjinal Dalga Formunu (sağda siyah) elde etmek üzere 1 tane sinüs fonksiyonu (solda mavi) başlayalım. Bu sinüs fonksiyonunun periyodu $T = 6.8 \cdot 2\pi$, genliği $A = 0.3$ olsun.



Şekil 2. Birincinin üzerine ekleyeceğimiz 2. sinüs fonksiyonunun (solda, mavi) periyodu ve genliğı birincinin 2 katı olsun.



Şekil 3. İlk iki sinüsün üzerine ekleyeceğimiz 3. sinüs fonksiyonunun (solda, mavi) periyodu birincinin 3 katı olsun.



Şekil 4. Son olarak 4. temel frekansı da eklediğimizde orjinal dalga formumuzu elde etmiş oluruz.

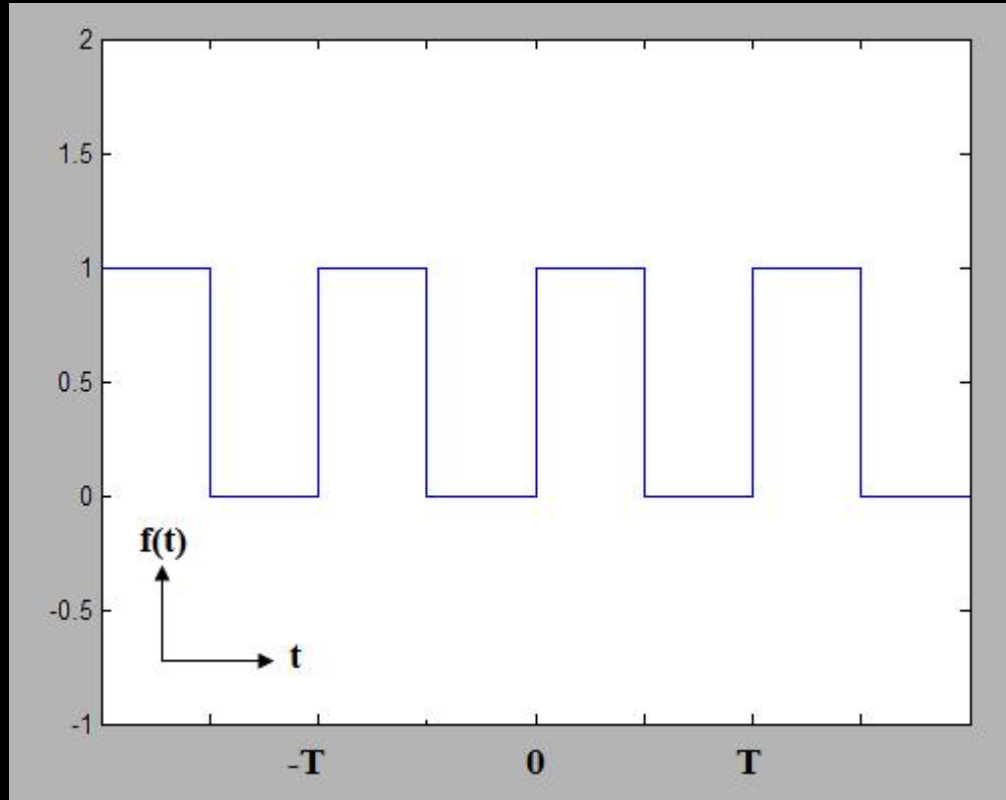
Fourier Dönüşümleri dalga formlarını, birleştirildiklerinde onu oluşturan sinüs fonksiyonlarına ayırmamızı sağlayan kuvvetli bir matematiksel yöntem sağlar. Bu şekilde özellikle periyodik bazı fiziksel olayların yapısını bileşenlerine ayırmamızı sağlayarak anlamamıza yardımcı olur. Fourier dönüşümlerini anlayabilmek için öncelikle Fourier Serileri'ni anlamamız gerekir.

Fourier Serileri

Fourier Serileri **periyodik** bir fonksiyonu birleştirildiklerinde o fonksiyonu oluşturacak sinüs fonksiyonlarına ayırmak için kullanılır. Periyodik fonksiyonlar için Fourier dönüşümleri olarak düşünülebilirler. Bu nedenle öncelikle periyodik fonksiyonları tanımlamalıyız. Bir periyodik fonksiyon

$$f(t + T) = f(t)$$

şeklinde tanımlanır. Bu şu anlama gelir: fonksiyonu T (periyod) kadar süre sonra gözlerseniz o anda gözlediğinizdeki değeri ile aynı değeri alır. Örnek olarak Şekil 5'e bakalım.



Şekil 5. Periyodik Kare Dalga Formu

Fourier Serileri

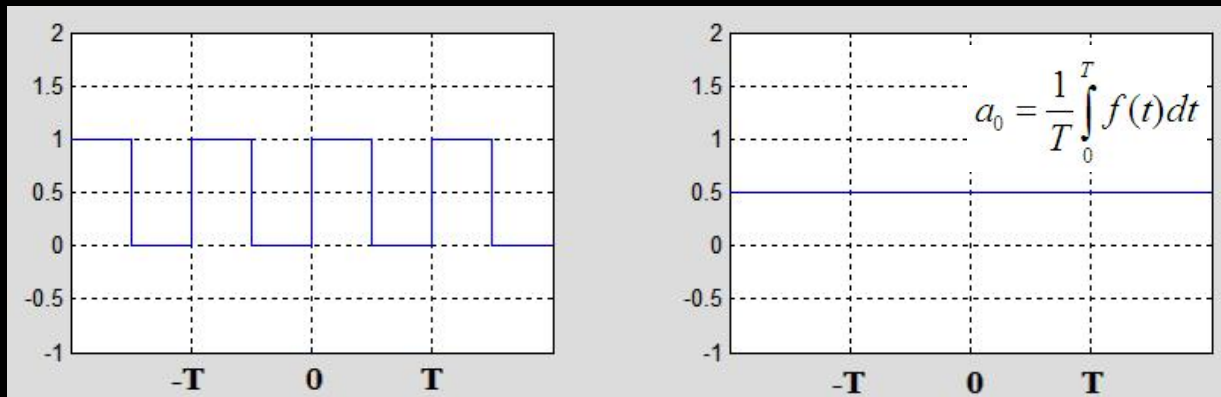
T dönemine sahip bir Fourier Serisi her biri $1/T$ 'nin tam katı frekanslı sonsuz sayıda sinüs ve kosinüs fonksiyonlarının toplamıdır. Ayrıca x eksenindeki kaymayı ifade eden bir de sabit terim (a_0) içerir. a_m , b_n sabitleri Fourier Serisi katsayılarıdır ve her bir sinüsün (ya da kosinüsün) toplama hangi ağırlıkla katılacağını gösterir.

$$g(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos\left(\frac{2\pi mt}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi nt}{T}\right)$$
$$= \sum_{m=0}^{\infty} a_m \cos\left(\frac{2\pi mt}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi nt}{T}\right)$$

Keyfi seçilmiş bir $f(t)$ fonksiyonu için temel frekansın tam katı frekansa sahip basit sinüs fonksiyonlarının üstüste bindirilmesi ile $f(t)$ fonksiyonuna acaba ne kadar yakınsanabilir? Yani yukarıdaki $g(t)$ fonksiyonu orijinal $f(t)$ fonksiyonuna ne kadar yakındır?

$f(t)$, sürekli ve düzgün (her noktasında türevli) bir fonksiyon ise bu sorunun cevabı tam olarak yakınsayabileceğimiz şeklinde verilebilir. Yani sürekli ve düzgün fonksiyonlar Fourier Serileri ile tam olarak temsil edilebilirler.

Öncelikle ilk terimle başlayalım $g(t) = a_0$. $f(t)$ 'yi tek bir sabitle temsil etmek istesek bu sabit ne olurdu? En uygun sabit terim $f(t)$ 'nin $0 - T$ arasındaki ortalama değeri olacaktır.



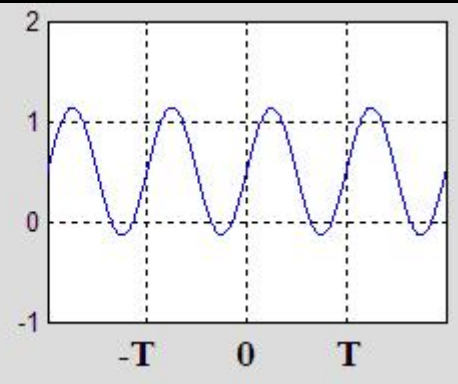
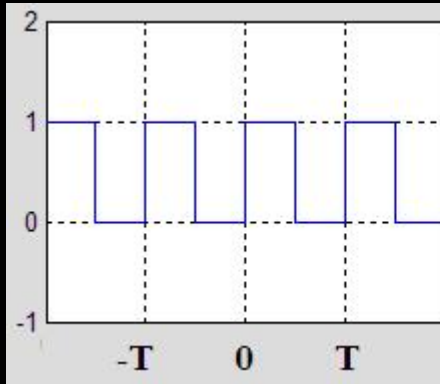
Fourier Serileri

Şimdi bir terim daha alalım. Diyelim ki sinüslü terimi alıyoruz. b_1 katsayısının $f(t)$ 'yi en iyi yakınsayacak $g(t)$ fonksiyonunu vermesi için optimal değeri ne olmalıdır? Bunun için $f(t)$ ile sinüs fonksiyonunun en iyi korelasyonu verdiği katsayıyı bulmalıyız. İki fonksiyon arasındaki en iyi korelasyonu gösteren ifade aşağıda ortada bulunan denklemle verilir. Bu integral alınırsa b_1 'in optimal değeri $2 / \pi$ olarak elde edilir.

$$g(t) = a_0 + b_1 \sin\left(\frac{2\pi t}{T}\right)$$

$$b_1 = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi t}{T}\right) dt$$

$$b_1 = \frac{2}{\pi}$$



İlk iki terim birleştirildiğinde yukarıdaki (sağda) $g(t)$ fonksiyonu elde edilir. Şimdiden orjinal dalga formuyla benzerliği görebiliyoruz. Tek tek ilerlemek yerine a_m ve b_n terimlerinin optimum değerlerini birlikte vermek istersek;

$$a_0 = \frac{1}{T} \int_0^T f(t) dt$$

$$a_m = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi m t}{T}\right) dt$$

$$b_n = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi n t}{T}\right) dt$$

verilen kare dalga için

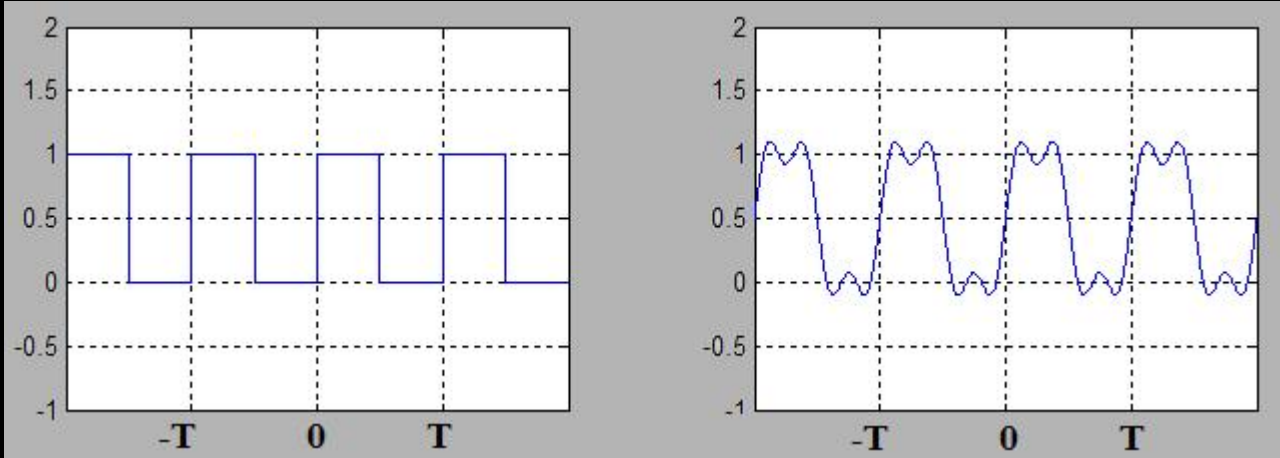
$$a_0 = \frac{1}{2}$$

$$a_m = 0, \quad m = 1, 2, \dots$$

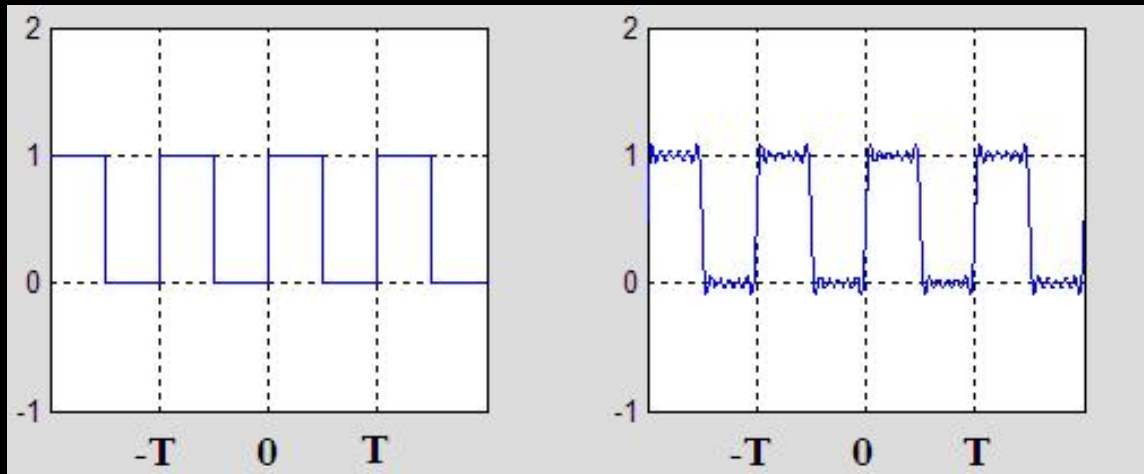
$$b_n = \begin{cases} \frac{2}{\pi n}, & n \text{ tek} \\ 0, & n \text{ çift} \end{cases}$$

Fourier Serileri

n çift olduğundaki terimlerin neden 0 olduğunu kolayca görebiliriz. $a_0 + g(t)$ fonksiyonunda a_0 sadece bir kaym miktarıdır. $g(t) = f(t) - a_0$ fonksiyonuna bakacak olursak bunun tek bir fonksiyon olduğunu kolaylıkla görebiliriz. Kosinüs çift bir fonksiyon olduğundan onun tüm katsayıları $a_1, a_2, \dots, a_m = 0$ olacaktır. b_3 'ü hesaplayıp eklediğimiz vakit aşağıdaki yaklaşımı elde ederiz.



Giderek orjinal dalga formuna yaklaşıyoruz. 7 terim alınırsa ($a_0, b_1, b_3, b_5, b_7, b_9, b_{11}$) aşağıdaki yaklaşım elde edilir.



Fourier Serileri – Kompleks Katsayılar

Fourier Serileri oluşturulurken iyi bilinen Euler eşitliğinden faydalanılarak reel katsayılar yerine kompleks katsayılar da kullanılabilir. Aşağıdaki seri formu dikkate alınırsa yapılması gereken c_n katsayılarını elde etmek olacaktır.

$$\cos t = \frac{e^{it} + e^{-it}}{2}$$
$$\sin t = \frac{e^{it} - e^{-it}}{2i}$$

ve

$$e^{it} = \cos t + i \sin t$$
$$i = \sqrt{-1}$$



$$g(t) = \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n t}{T}}$$

c_n katsayılarının optimum değeri

$$c_n = \frac{1}{T} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt$$

c_n katsayılarının kare dalga için değerleri aşağıdaki şekilde bulunur.

$$c_0 = \frac{1}{2}$$
$$c_n = \frac{1}{i\pi n}, \quad n = \pm 1, \pm 3, \pm 5, \dots$$
$$c_n = 0, \quad n = \pm 2, \pm 4, \pm 6, \dots$$

c_n katsayıları ile c_{-n} birbirinin kompleks eşleniği ise yani $c_n^* = c_{-n} \rightarrow g(t)$ fonksiyonu reel bir fonksiyon olur.

İspat

Fourier Serilerinin kompleks formunda (yanda) katsayıların nasıl bulunduğunun ispatına geçelim. Eğer yandaki seri gerçekten açılımı olduğu $f(t)$ fonksiyonuna yakınsıyorsa aşağıdaki eşitlik doğrudur.

$$c_n = \frac{1}{T} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt$$

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n t}{T}}$$

Şimdi eşitliğin her iki tarafını $e^{-i2\pi n t/T}$ ile çarpıp $[0, T]$ aralığında integralini alalım.

$$\begin{aligned} f(t) e^{-i \frac{2\pi n t}{T}} &= \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n t}{T}} e^{-i \frac{2\pi n t}{T}} \\ &= \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi (n-m) t}{T}} \end{aligned}$$



$$\begin{aligned} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt &= \int_0^T \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi (n-m) t}{T}} dt \\ &= \sum_{n=-\infty}^{\infty} \int_0^T c_n e^{i \frac{2\pi (n-m) t}{T}} dt \\ &= \sum_{n=-\infty}^{\infty} c_n \int_0^T e^{i \frac{2\pi (n-m) t}{T}} dt \end{aligned}$$



$$\int_0^T e^{i \frac{2\pi (n-m) t}{T}} dt = \begin{cases} 0, & n \neq m \\ T, & n = m \end{cases}$$

Sonuç olarak bu eşitliğin 0'dan farklı olduğu tek nokta $n = m$ noktasıdır. Bu eşitliğin sağ tarafı ile yukarıda ortada bulunan eşitliğin sol tarafı birbirine eşittir.

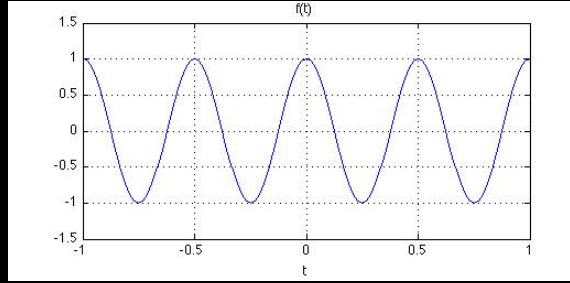
$$\int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt = c_n T$$



$$c_n = \frac{1}{T} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt$$

Kosinüs (cos) Fonksiyonunun Fourier Serisi

Aşağıda $\cos(4\pi t)$ fonksiyonunun grafiği verilmiştir.



Öncelikle temel frekans ve dönemi bulalım. Fonksiyon $t = 0$ ile $t = 1$ arasında kendini 2 kez tekrar ettiğinde dönemi $T = 0.5$ 'tir. Kompleks katsayılar başvurarak Fourier Serisi açılımını elde etmeye çalışalım.

$$c_n = \frac{1}{T} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt$$
$$= \frac{1}{0.5} \int_0^{0.5} \cos(4\pi t) e^{-i \frac{2\pi n t}{0.5}} dt$$

Kosinüs fonksiyonunun yerine Euler eşitliğini kullanarak üstel formunu koyalım.

$$\cos(t) = \frac{e^{it} + e^{-it}}{2}$$

$$c_n = 2 \int_0^{0.5} \left(\frac{e^{i4\pi t} + e^{-i4\pi t}}{2} \right) e^{-i4\pi n t} dt$$
$$= \int_0^{0.5} e^{i4\pi(1-n)t} dt + \int_0^{0.5} e^{-i4\pi(n+1)t} dt$$

$n \neq 1$
için

$$\int_0^{0.5} e^{i4\pi(1-n)t} dt = \frac{e^{i4\pi(1-n)t}}{i4\pi(1-n)} \Big|_0^{0.5}$$
$$= \frac{1}{i4\pi(1-n)} [e^{i2\pi(1-n)} - 1]$$
$$= 0, \quad n \neq 1$$

$n = 1$
için

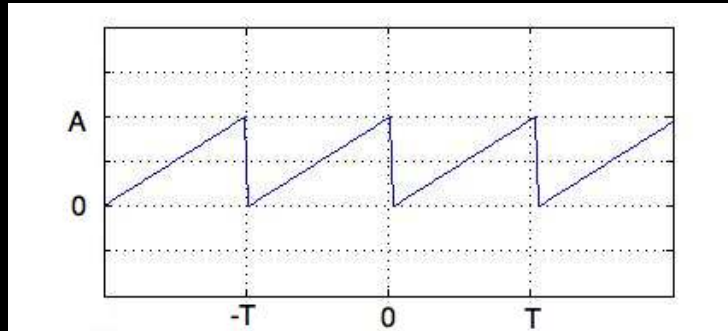
$$c_1 = \int_0^{0.5} e^{i4\pi(1-1)t} dt \stackrel{(n=1)}{=} \int_0^{0.5} e^{i4\pi(1-1)t} dt$$
$$= \int_0^{0.5} 1 dt$$
$$= 0.5$$

Sonuç olarak;

$$g(t) = \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n t}{T}}$$
$$= c_1 e^{i \frac{2\pi t}{T}} + c_{-1} e^{-i \frac{2\pi t}{T}}$$
$$= 0.5 (e^{i4\pi t} + e^{-i4\pi t})$$
$$= \cos(4\pi t)$$
$$= f(t)$$

Testere Fonksiyonunun Fourier Serisi

Aşağıda $f(t) = A t / T$ olarak ifade edilebilecek olan testere fonksiyonunun grafiği verilmiştir.



c_n katsayılarının elde edilmesi aşağıdaki integralin kısmi integrasyon yöntemiyle hesaplanmasıyla mümkündür.

$$c_n = \frac{1}{T} \int_0^T f(t) e^{-i \frac{2\pi n t}{T}} dt \quad e^{-i 2\pi n} = 1$$

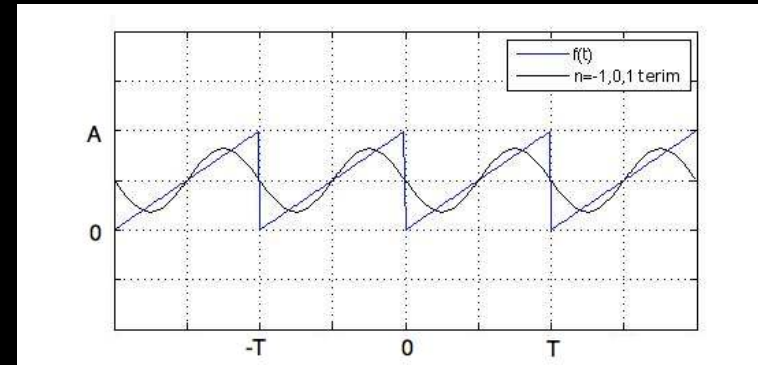
$$= \frac{A}{T^2} \int_0^T t e^{-i \frac{2\pi n t}{T}} dt$$

$$= \frac{A}{T^2} \left[\left\{ \frac{tT}{-i2\pi n} e^{-i \frac{2\pi n t}{T}} + \frac{T^2}{(2\pi n)^2} e^{-i \frac{2\pi n t}{T}} \right\} \right]_0^T$$

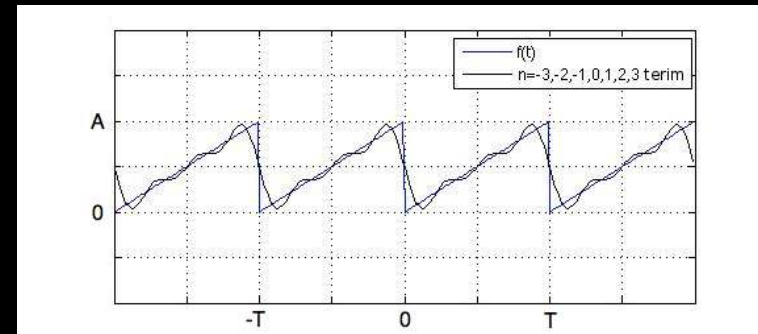
$$= \frac{A}{T^2} \left[\frac{T^2}{-i2\pi n} \right] = \frac{iA}{2\pi n}, \quad n \neq 0$$

$$c_0 = \frac{A}{2}$$

→
 $n = -1, 0, 1$ için



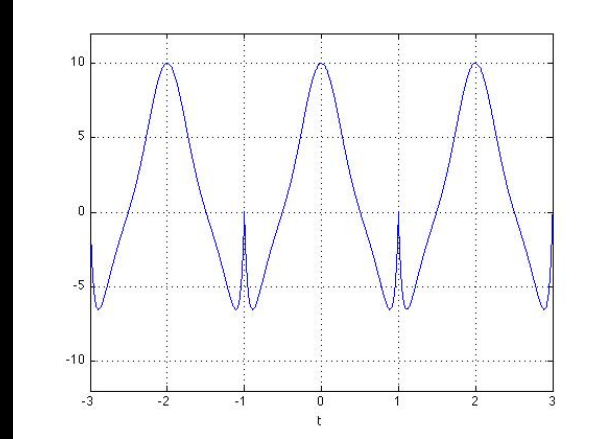
→
 $n = -3, -2, -1, 0, 1, 2, 3$ için



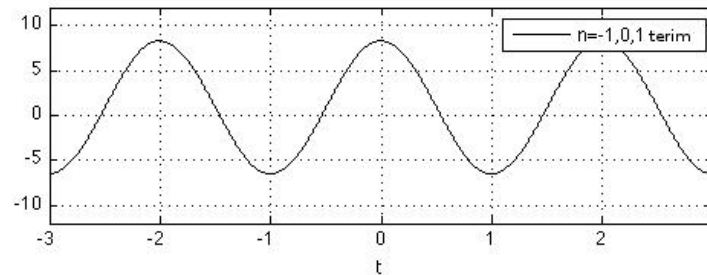
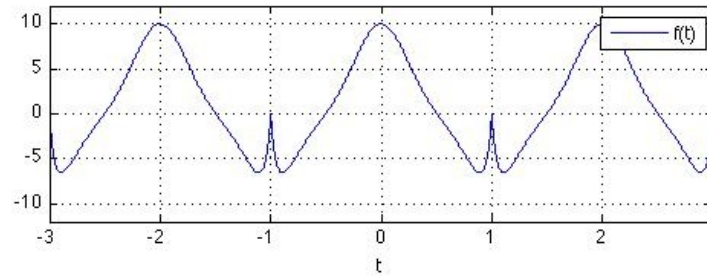
Komplike Bir Fonksiyonunun Fourier Serisi

Aşağıda komplike bir fonksiyonun matematiksel ifadesi (solda) ve grafiği verilmiştir (sağda).

$$f(t) = \frac{(t-1)(t+1)(2t)^4}{\cos(1.4t)} + 10e^{-9t^2}, \quad -1 \leq t < 1$$

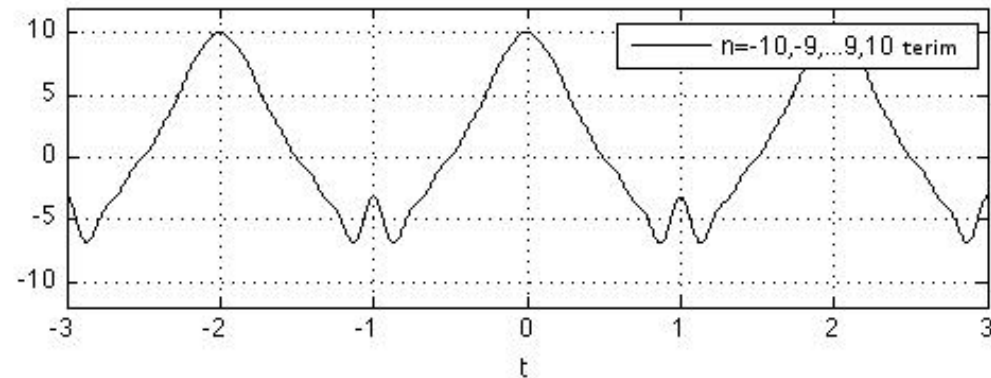
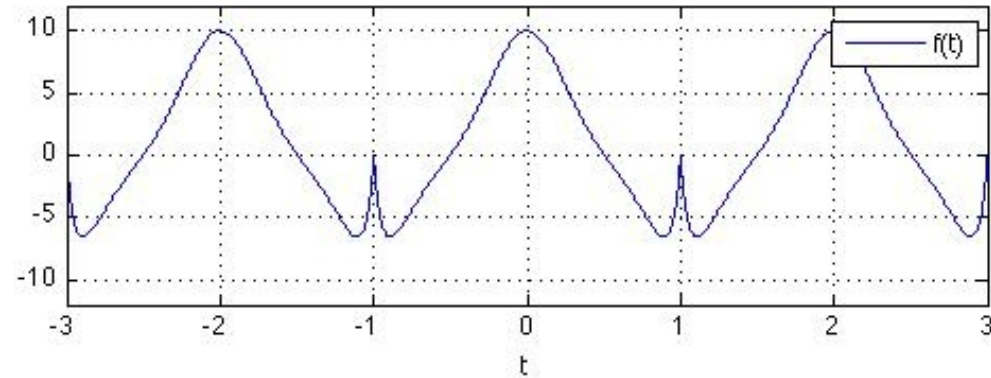


Verilen fonksiyonun integrali analitik olarak alınamayacağından Fourier Serisi dönüşümünde başka bir yola başvurmalıyız. İntegralin nümerik olarak alınamadığı böyle durumlarda Fourier serisi katsayılarını bulabilmek için nümerik bir çözüme başvurmak gerekir ([katsayi_hesabi.py](#)).



n = -1, 0, 1 için çözüm

Komplike Bir Fonksiyonunun Fourier Serisi



$N = -10, -9, \dots, 9, 10$ için çözüm

Dönüşümün Başarısının Ölçütü

Ortalama Hata (Mean Squared Error -MSE-)

$x = [1 \ 2 \ 1] = [x_1 \ x_2 \ x_3]$ ve $y = [0 \ -1 \ 3] = [y_1 \ y_2 \ y_3]$ şeklinde tanımlanmış iki vektörümüz olsun. x ile y 'nin birbirine ne kadar yakın olduklarını bulmak için aralarındaki uzaklığa bakarız.

$$\begin{aligned} \text{uzaklık}(x,y) &= \| \mathbf{x} - \mathbf{y} \| \\ &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2} \end{aligned}$$

İki fonksiyonun birbirine yakın olup olmadıkları da aynı şekilde belirlenir. Aralarındaki farkların karelerinin toplamının (integralinin) karekökü bize bu iki fonksiyonun uzaklıklarını verir.

$$\| f - g \| = \sqrt{\int_0^T |f(t) - g(t)|^2 dt}$$

Bu uzaklık aynı zamanda Ortalama Hata (Mean Squared Error) olarak da adlandırılır ve fonksiyonların yakınsamasını nicel hale getirmek üzere kullanılan önemli bir metriktir. $f(t)$ fonksiyonuna karşılık gelen Fourier serisi $g(t)$ aşağıdaki gibi tanımlanmış olsun:

$$g(t) = \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n t}{T}}$$

Belirli bir aralık için
sonlu Fourier serisi

$$g_N(t) = \sum_{n=-N}^N c_n e^{i \frac{2\pi n t}{T}}$$

MSE

$$\sqrt{\int_0^T \left| f(t) - \sum_{n=-N}^N c_n e^{i \frac{2\pi n t}{T}} \right|^2 dt}$$

Fourier Dönüşümü

Fourier Serileri ile herhangi bir periyodik fonksiyonu sinüs ve kosinüs fonksiyonlarının (ya da Euler dönüşümü yoluyla üstel bir fonksiyonun) sonsuz toplamı olarak ifade edebileceğimizi gördük. Fourier Dönüşümü bu konseptin periyodik olmayan fonksiyonlara da uyarlanarak genişletilmiş halidir. Herhangi bir $g(t)$ fonksiyonunun Fourier Dönüşümü aşağıdaki ifade ile verilir.

$$\mathcal{F}\{g(t)\} = G(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift} dt$$

Dönüşüm sonucunda frekansın (f) fonksiyonu olan ve $g(t)$ 'nin f frekansındaki gücünü veren bir $G(f)$ fonksiyonu elde edilir. $G(f)$ fonksiyonu bu nedenle $g(t)$ 'nin güç spektrumu olarak isimlendirilir. Ters Fourier Dönüşümü ile $G(f)$ 'ten $g(t)$ 'ye dönüşüm sağlanır.

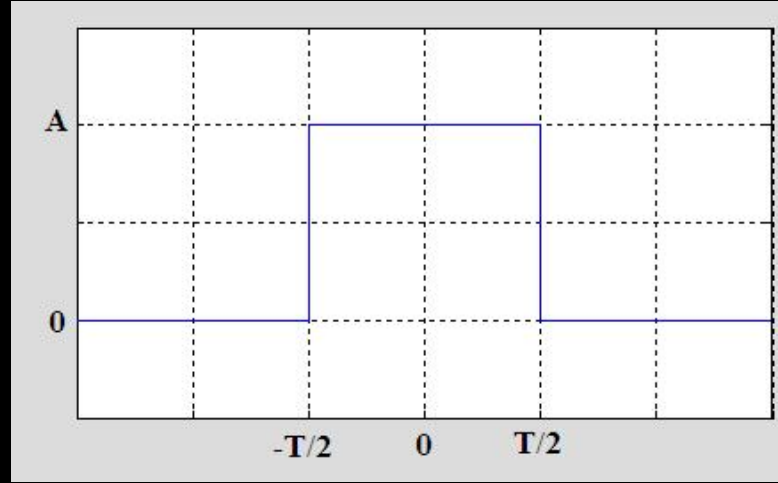
$$\mathcal{F}^{-1}\{G(f)\} = \int_{-\infty}^{\infty} G(f)e^{2\pi ift} df = g(t)$$

Sonuç olarak $g(t)$ 'den $G(f)$ 'e (zaman tanım kümesinden frekans tanım kümesine) dönüşüm Fourier Dönüşümü ile, $G(f)$ 'ten $g(t)$ 'ye (frekans tanım kümesinden zaman tanım kümesine) dönüşüm ise Ters Fourier Dönüşümü ile sağlanır. Bu iki fonksiyona bir Fourier Çifti adı verilir.

$$g \xLeftrightarrow{\mathcal{F}} G$$

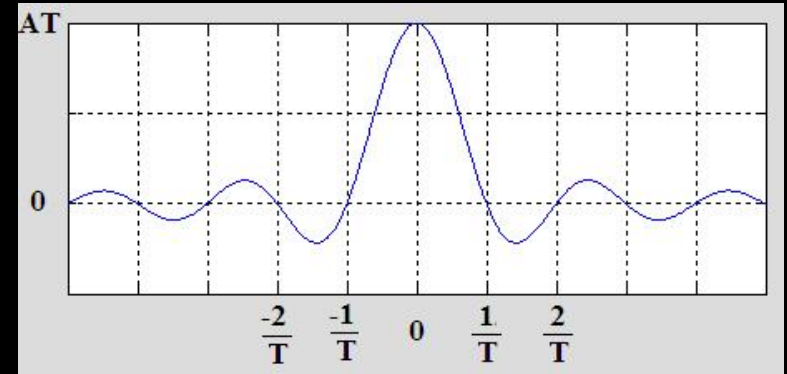
Kare Dalga Fonksiyonunun Fourier Dönüşümü

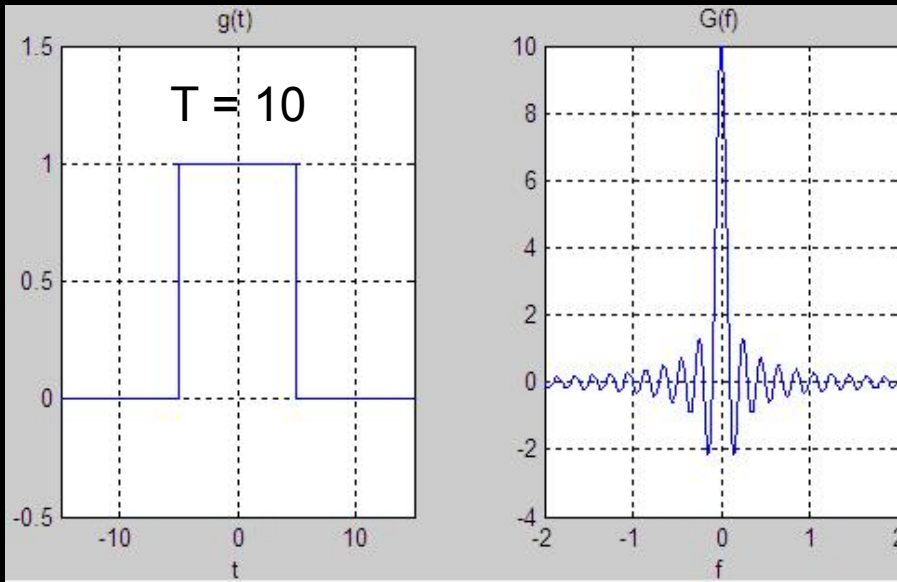
Kare dalga, kare atım, ya da kutu fonksiyonu olarak bilinen fonksiyon $t = -T/2$ ile $T/2$ arasında A genlik değerini, onun dışında ($|t| > T/2$) 0 değerini alan parçalı bir fonksiyondur.



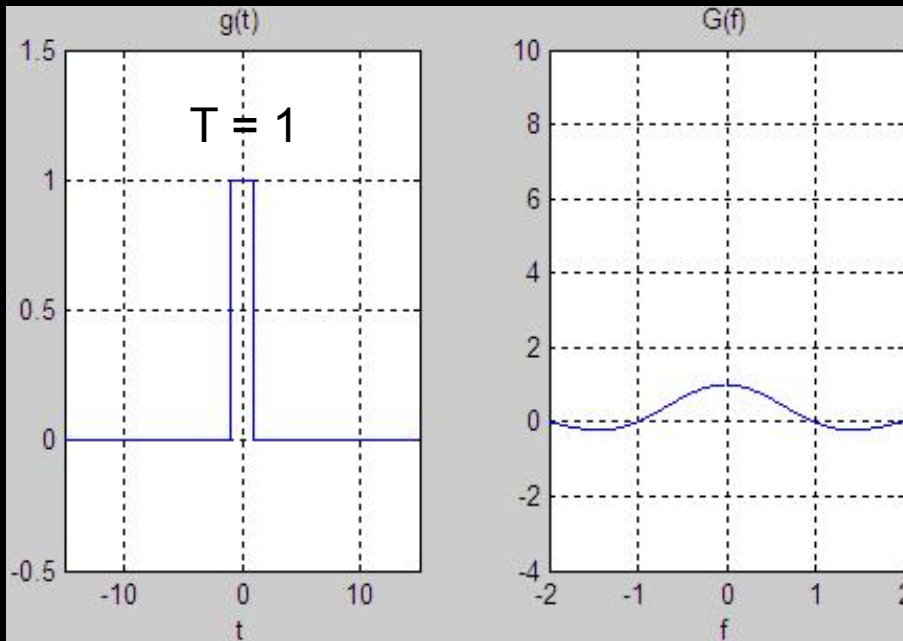
Fourier Dönüşümü tanımını kullanarak bu fonksiyonun Fourier Dönüşümü aşağıdaki şekilde elde edilir.

$$\begin{aligned}\mathcal{F}\{g(t)\} &= G(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift} dt \quad \text{sinc}(t) = \frac{\sin(\pi t)}{\pi t} \\ &= \int_{-T/2}^{T/2} Ae^{-2\pi ift} dt = \frac{A}{-2\pi if} \left[e^{-2\pi ift} \right]_{-T/2}^{T/2} \\ &= \frac{A}{-2\pi if} \left[e^{-\pi ifT} - e^{\pi ifT} \right] = \frac{AT}{\pi fT} \left[\frac{e^{\pi ifT} - e^{-\pi ifT}}{2i} \right] \\ &= \frac{AT}{\pi fT} \sin(\pi fT) = AT[\text{sinc}(fT)]\end{aligned}$$





1. Geniş kare dalga daha dar ve pek çok piki olan bir spektrum üretir.



2. Dar kare dalga daha geniş ve az sayıda piki olan bir spektrum üretir.

Sonuç: Hızlı değişen fonksiyonların Fourier dönüşümleri daha çok yüksek frekans içerirken (2), daha yavaş değişen fonksiyonları (1) ise yüksek frekanslarda daha az enerjiye sahip pikler üretirler. Yavaş değişen fonksiyonların frekans içerikleri düşük frekanslara konsantre olduğundan bu frekanslardaki enerjileri daha yüksektir.

Ayrıca fonksiyon “kısa süreli” bir kare dalga ise (2), enerjisinin daha az olacağı düşünülebilir, Fourier Dönüşümü'nde görülen de gerçekten enerjisinin az olduğudur.

Fourier Dönüşümünün Özellikleri

1. Fourier Dönüşümleri lineerdir. Bu özelliğin doğruluğu kolaylıkla gösterilebilir.

$$\mathcal{F}\{c_1g(t) + c_2h(t)\} = c_1G(f) + c_2H(f)$$

$$\begin{aligned}\mathcal{F}\{c_1g(t) + c_2h(t)\} &= \int_{-\infty}^{\infty} c_1g(t)e^{-i2\pi ft} dt + \int_{-\infty}^{\infty} c_2h(t)e^{-i2\pi ft} dt \\ &= c_1 \int_{-\infty}^{\infty} g(t)e^{-i2\pi ft} dt + c_2 \int_{-\infty}^{\infty} h(t)e^{-i2\pi ft} dt \\ &= c_1G(f) + c_2H(f)\end{aligned}$$

2. $g(t)$ fonksiyonu zamanda a reel sayısı kadar kaydırıldığında $g(t-a)$, Fourier dönüşümünün $G(f)$ frekans içeriği ve güç spektrumunun genliği değişmez, değişen sadece evresidir.

$$\begin{aligned}\mathcal{F}\{g(t-a)\} &= \int_{-\infty}^{\infty} g(t-a)e^{-i2\pi ft} dt \\ &= \int_{-\infty}^{\infty} g(u)e^{-i2\pi f(u+a)} du \\ &= e^{-i2\pi fa} \int_{-\infty}^{\infty} g(u)e^{-i2\pi fu} du \\ &= e^{-i2\pi fa} G(f)\end{aligned}$$

3. $g(t)$ fonksiyonu bir c reel sayısı ile ölçeklendirildiğinde $(g(ct))$ Fourier dönüşümü:

$$\mathcal{F}\{g(ct)\} = \frac{G\left(\frac{f}{c}\right)}{|c|}$$

4. $g(t)$ fonksiyonunun türevinin Fourier dönüşümü:

$$\mathcal{F}\left\{\frac{dg(t)}{dt}\right\} = i2\pi f \cdot G(f)$$

Fourier Dönüşümünün Özellikleri

5. $g(t)$ fonksiyonunun $h(t)$ fonksiyonu ile konvolüsyonunun Fourier dönüşümü:

$$g(t) * h(t) = \int_{-\infty}^{\infty} g(\tau)h(t - \tau)d\tau$$

$$\mathcal{F}\{g(t) * h(t)\} = G(f)H(f)$$

6. İki fonksiyonun ($g(t)$ ve $h(t)$) çarpımının (modülasyonunun) Fourier Dönüşümü:

$$\mathcal{F}\{g(t)h(t)\} = G(f) * H(f)$$

7. Parseval Teoremi: $G(f)$, $g(t)$ 'nin Fourier Dönüşümü olmak üzere $g(t)$ ve $G(f)$ 'nin içerdikleri toplam enerji (güç) aynıdır.

$$\int_{-\infty}^{\infty} |g(t)|^2 dt = \int_{-\infty}^{\infty} |G(f)|^2 df$$

8. Düalite Özelliği: $G(f)$, $g(t)$ 'nin Fourier Dönüşümü olmak üzere $G(f)$ 'nin Fourier Dönüşümü:

$$\mathcal{F}\{G(f)\} = g(-t)$$

İntegrasyon Özelliği (Opsiyonel)

- Bu bölümde $X(f)$, $x(t)$ fonksiyonunun Fourier dönüşümü olmak üzere $x(t)$ 'nin integrali olarak tanımlanan $y(t)$ fonksiyonunu Fourier dönüşümünü bulmaya çalışacağız.

$$y(t) = \int_{-\infty}^t x(\tau) d\tau$$

- $g(t)$ fonksiyonunun türevinin Fourier dönüşümünü hatırlayarak başlayalım ve orjinal (sağ üstte) denklemi tekrar yazalım (sağ altta).

$$\mathfrak{F}\left\{\frac{dg(t)}{dt}\right\} = i2\pi f G(f)$$

$$\mathfrak{F}\left\{\frac{dg(t)}{dt}\right\} = i2\pi f \mathfrak{F}\{g(t)\}$$

- $h(t) = g(t)$ 'nin zamana göre türevi olsun. Bu durumda $g(t)$ de $h(t)$ 'nin $-\infty$ ile t arasındaki integrali olacaktır.

$$\mathfrak{F}\{h(t)\} = i2\pi f \mathfrak{F}\left\{\int_{-\infty}^t h(\tau) d\tau\right\}$$

- Bu eşitlik tekrar düzenlendiğinde sağ üstteki eşitlik elde edilir. Ancak bu eşitlik tam olarak doğru değildir. Zira $g(t)$ 'nin yanı sıra c bir sabit olmak üzere tüm $g(t) + c$ fonksiyonlarının da türevi aynıdır. Bu nedenle sağ alttaki koşulu koymak gereklidir.

$$\mathfrak{F}\left\{\int_{-\infty}^t h(\tau) d\tau\right\} = \frac{\mathfrak{F}\{h(t)\}}{i2\pi f}$$

$$\int_{-\infty}^{\infty} (g(\tau) - c) d\tau = 0$$


- Bu koşul konduğunda yandaki eşitlik elde edilir.

$$\begin{aligned}\mathfrak{F}\left\{\int_{-\infty}^t g(\tau) d\tau\right\} &= \frac{\mathfrak{F}\{g(t)\}}{i2\pi f} + c\delta(f) \\ &= \frac{G(f)}{i2\pi f} + c\delta(f)\end{aligned}$$


Dirac δ fonksiyonunun tanımı

$$\delta(t) = \begin{cases} \infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

Bazı Fonksiyonların Fourier Dönüşümleri - I

Fonksiyon	Adı	Frekans (f)	Açısal Frekans ($\omega = 2\pi f$)
$g(t)$		$G(f) = \int_{-\infty}^{\infty} g(t)e^{-i2\pi ft} dt$	$G(\omega) = \int_{-\infty}^{\infty} g(t)e^{-i\omega t} dt$
$A \text{ } (-T/2 < t < T/2)$ $0 \text{ } (t > T/2)$	Kare Dalga	$T \text{sinc}(fT)$	$T \text{sinc}\left(\frac{\omega T}{2\pi}\right)$
$\Lambda(t)$	Üçgen Fonksiyonu	$\text{sinc}^2(f)$	$\text{sinc}^2\left(\frac{\omega}{2\pi}\right)$
$e^{-\pi t^2}$	Gauss Fonksiyonu	$e^{-\pi f^2}$	$e^{-(\omega^2 / 4\pi)}$
C	Sabit Fonksiyon	$C\delta(f)$	$2\pi C\delta(\omega)$
$\delta(t - a)$	Dirac-Delta Fonksiyonu	$e^{-i2\pi fa}$	$e^{-i\omega a}$
$\cos(2\pi At)$	Kosinüs Fonksiyonu	$\frac{1}{2}[\delta(f - A) + \delta(f + A)]$	$\pi[\delta(\omega - 2\pi A) + \delta(\omega + 2\pi A)]$
$\sin(2\pi At)$	Sinüs Fonksiyonu	$\frac{1}{2i}[\delta(f - A) - \delta(f + A)]$	$\frac{\pi}{i}[\delta(\omega - 2\pi A) - \delta(\omega + 2\pi A)]$
$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$	Birim Adım Fonksiyonu	$\frac{1}{2\pi if} + \frac{\delta(f)}{2}$	$\frac{1}{i\omega} + \pi\delta(\omega)$

Bazı Fonksiyonların Fourier Dönüşümleri – II

Fonksiyon	Adı	Frekans (f)	Açısal Frekans ($\omega = 2\pi f$)
$g(t)$		$G(f) = \int_{-\infty}^{\infty} g(t)e^{-i2\pi ft} dt$	$G(\omega) = \int_{-\infty}^{\infty} g(t)e^{-i\omega t} dt$
$\text{sgn}(t) = \begin{cases} 1, & t \geq 0 \\ -1, & t < 0 \end{cases}$	Signum Fonksiyonu	$\frac{1}{\pi if}$	$\frac{2}{i\omega}$
$e^{- a t}u(t)$	Üstel Fonksiyon	$\frac{1}{ a + 2\pi if}$	$\frac{1}{ a + i\omega}$
$e^{ a t}u(-t)$	Üstel Fonksiyon	$\frac{1}{ a - 2\pi if}$	$\frac{1}{ a - i\omega}$
$e^{- at }$	Üstel Fonksiyon	$\frac{2 a }{ a ^2 + (2\pi f)^2}$	$\frac{2 a }{ a ^2 + \omega^2}$
$III(t) = \sum_{n=-\infty}^{\infty} \delta(t - n)$	Shah Fonksiyonu	$III(f)$	$III(\omega)$
$t^n h(t)$	$t^n * h(t)$ Fonksiyonu	$\left(\frac{i}{2\pi}\right)^n \frac{d^n H(f)}{df^n}$	$i^n \frac{d^n H(\omega)}{d\omega^n}$
$ t $	Mutlak Değer Fonksiyonu	$\frac{-1}{2\pi^2 f^2}$	$\frac{-2}{\omega^2}$
$\frac{1}{\sqrt{t}}$	Ters Karekök Fonksiyonu	$\frac{1}{\sqrt{f}}$	$\sqrt{\frac{2\pi}{\omega}}$

scipy.fftpack İle Fourier Analizi

- ✓ Sürekli olan fonksiyon – Fourier transformu çifti (Fourier çifti) süreksiz denkleiryle yer değiştirdiğinde “Süreksiz Fourier Dönüşümü” 'nden söz edilir (DFT: Discrete Fourier Transform). Süreksiz Fourier Dönüşümleri'nin hesaplamak için iyi bir algoritma “Hızlı Fourier Dönüşümleri” 'dir (FFT: Fast Fourier Transform). N uzunluğundaki x[n] dizisinin Hızlı Fourier Dönüşümü (FFT) y[k] serisin ve tersi:

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n]$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi j \frac{kn}{N}} y[k]$$

- ✓ scipy.fftpack modülündeki fft ve ifft kullanılarak her iki dönüşüm de yapılabilir.

```
>>> import numpy as np
>>> from scipy.fftpack import fft, ifft
>>> x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
>>> y = fft(x)
>>> y
array([ 4.50000000+0.j,  2.08155948-1.65109876j,
        -1.83155948+1.60822041j, -1.83155948-1.60822041j,
         2.08155948+1.65109876j])
>>> yinv = ifft(y)
>>> yinv
array([ 1.0+0.j,  2.0+0.j,  1.0+0.j, -1.0+0.j,  1.5+0.j])
```

- ✓ Yukarıda verilen FFT tanımından y[0]'ın x[n] dizisinin toplamını, N'in çift değeri için y[1] ... y[N/2 -1]'in pozitif frekansları y[N/2], ... y[N-1]'in ise negatif frekansları, N'in tek değeri için ise y[1] ... y[(N-1)/2]'nin pozitif frekansları y[(N+1)/2], ... y[N-1]'in ise negatif frekansları verdiği kolayca görülebilir.

```
>>> print np.sum(x)
4.5
```

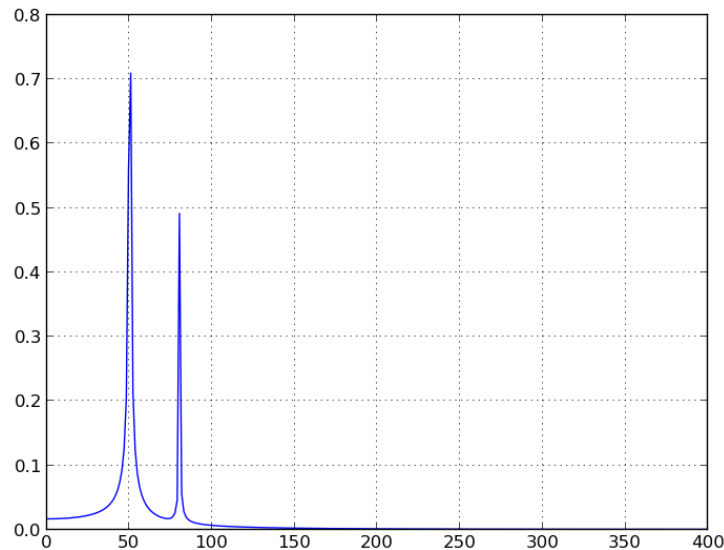
$$y[0] = \sum_{n=0}^{N-1} x[n]$$

Örnek 1



İki farklı frekansa sinüs fonksiyonunun (süreksiz) üstüste bindirilmesi sonucu oluşan sinüs fonksiyonunun Hızlı Fourier Dönüşümü (FFT) ([fft_2sinus.py](#)):

```
>>> from scipy.fftpack import fft
>>> import numpy as np
>>> from matplotlib import pyplot as plt
>>> N = 600 # toplam nokta sayisi
>>> T = 1.0 / 800. #noktalar arasi uzaklik
>>> y = np.linspace(0.0, N*T, N)
>>> yf = fft(y) # y fonksiyonunun hizli fourier donusumu
>>> xf = np.linspace(0.0, 1.0/(2.0*T), N/2) # x'in frekansa donusumu
>>> plt.plot(xf, 2.0/N * np.abs(yf[0:N/2])) # sadece pozitif frekanslari cizdirelim
>>> plt.grid()
>>> plt.show()
```



Örnek 2

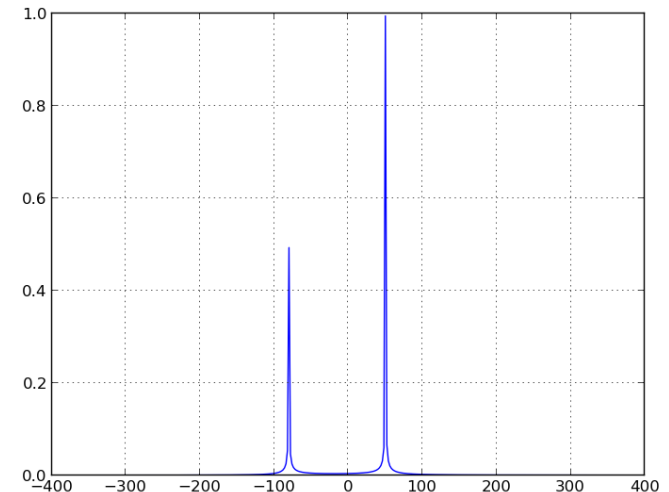


Bu kez iki kompleks üstel fonksiyonunun Hızlı Fourier Dönüşümü (FFT)'ne bakalım (`fft_2exp.py`):

```
>>> from scipy.fftpack import fft, fftfreq, fftshift
>>> import numpy as np
>>> from matplotlib import pyplot as plt
>>> N = 400 # toplam nokta sayisi
>>> T = 1.0 / 800. #noktalar arasi uzaklik
>>> x = np.linspace(0.0, N*T, N)
>>> y = np.exp(50.0 * 1.j * 2.0*np.pi*x) + 0.5*np.exp(-80.0 * 1.j * 2.0*np.pi*x)
>>> yf = fft(y)
>>> xf = fftfreq(N, T) # Frekans tanim kumesinde noktalarin uzakligi = 1 / (T*N)
>>> xf = fftshift(xf) # 0'i guc spektrumunun ortasina aliyoruz.
>>> yplot = fftshift(yf) # xf ile ayni ozellige sahip olmasi icin
>>> import matplotlib.pyplot as plt
>>> plt.plot(xf, 1.0/N * np.abs(yplot))
>>> plt.grid()
>>> plt.show()
```



Burada x 'i frekans tanım kümesinde ifade etmek için bir önceki örnekte olduğu gibi xf 'i hesaplamak yerine `scipy.fftpack` fonksiyonu `fftfreq(n,d)` 'dan (n , nokta sayısını; d noktaların birbirinden uzaklığını (x tanım kümesinde) göstermek üzere) yararlandığımıza dikkat ediniz. Ancak `fftfreq`'in verdiği frekans dizisiinin birinci elemanı 0'ı; ilk yarısı pozitif, ikinci yarısı ise negatif frekanslardan oluşur. Bu nedenle daha anlamlı bir grafik için $f=0$ frekansını güç spektrumunun ortasına kaydıran bir de `fftshift(xf)` fonksiyonu kullanıyoruz.



Örnek 3



`scipy.fftpack` 'in `rfft` ve `irfft` fonksiyonları reel sayılardan oluşan bir `x` dizisinin Hızlı Fourier Dönüşümü'nü hesaplar ve dönüşümün Reel ve İmajiner kısımlarını ayrı ayrı almamıza olanak sağlar. `N`'in tek olması durumunda Fourier Dönüşümü `[y[0],Re(y[1]),Im(y[1]),...,Re(y[N/2])]`, `N`'in çift olması durumunda ise dönüşüm, `[y[0],Re(y[1]),Im(y[1]),...,Re(y[N/2]),Im(y[N/2])]` 'den oluşur.

```
>>> from scipy.fftpack import fft, rfft, irfft
>>> import numpy as np
>>> x = np.array([1.0, 2.0, 1.0, -1.0, 1.5, 1.0])
>>> fft(x)
array([ 5.50+0.j           ,  2.25-0.4330127j , -2.75-1.29903811j,
        1.50+0.j           , -2.75+1.29903811j,  2.25+0.4330127j ])
>>> yr = rfft(x)
>>> yr
array([ 5.5          ,  2.25          , -0.4330127 , -2.75          , -1.29903811,
        1.5          ])
>>> irfft(yr)
array([ 1. ,  2. ,  1. , -1. ,  1.5,  1. ])
>>> x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
>>> fft(x)
array([ 4.50000000+0.j           ,  2.08155948-1.65109876j,
       -1.83155948+1.60822041j, -1.83155948-1.60822041j,
        2.08155948+1.65109876j])
>>> yr = rfft(x)
>>> yr
array([ 4.5          ,  2.08155948, -1.65109876, -1.83155948,  1.60822041])
```

Süreksiz Kosinüs Dönüşümü

Discrete Cosine Transform (DCT)

- ✓ Herhangi bir fonksiyonu kosinüs fonksiyonlarının toplamı şeklinde ($y[k]$) ifade etmek de mümkündür. Bunun için üstüste bindirilen ve frekansları temel frekansın tam katları olan kosinüslerin genlikleri ($x[n]$) aranır. Bu işlemi gerçekleştirmek için literatürde 8 farklı yöntem önerilmiştir ancak bunlardan 3'ü scipy adapte edilmiştir. Genellikle kullanılan 2. tip DCT, ters dönüşüm için kullanılırsa 3. tip DCT'dir. Ayrıca ortogonal normalizasyon kullanma imkanı da bulunmaktadır.

- ✓ **I. tip DCT** (normalizasyon seçeneği yoktur)

$$y[k] = x_0 + (-1)^k x_{N-1} + 2 \sum_{n=1}^{N-2} x[n] \cos\left(\frac{\pi n k}{N-1}\right), \quad 0 \leq k < N.$$

- ✓ **II. tip DCT** Normalizasyon kullanılmaması durumunda:

$$y[k] = 2 \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad 0 \leq k < N.$$

Ortogonal normalizasyon kullanılması durumunda:

$$f = \begin{cases} \sqrt{1/(4N)}, & k \text{ sıfır} \\ \sqrt{1/(2N)}, & k \text{ sıfırdan farklıysa} \end{cases}$$

$y[k]$ için ölçeklendirme çarpanı ve ϕ yandaki şekilde tanımlanmak üzere,

$$\phi_k[n] = 2f \cos\left(\frac{\pi(2n+1)k}{2N}\right)$$

$$\sum_{n=0}^{N-1} \phi_k[n] \phi_l[n] = \delta_{lk}$$

- ✓ **III. tip DCT** Normalizasyon kullanılmaması durumunda:

$$y[k] = x_0 + 2 \sum_{n=1}^{N-1} x[n] \cos\left(\frac{\pi n(2k+1)}{2N}\right) \quad 0 \leq k < N$$

Ortogonal normalizasyon kullanılması durumunda:

$$y[k] = \frac{x_0}{\sqrt{N}} + \frac{2}{\sqrt{N}} \sum_{n=1}^{N-1} x[n] \cos\left(\frac{\pi n(2k+1)}{2N}\right) \quad 0 \leq k < N$$

Ters Dönüşüm (IDCT)



Normalize edilmemiş III. Tip DCT, normalize edilmemiş II. Tip DCT'nin tersidir (2N ölçeklendirme çarpanı olmak kaydıyla). Ortonormal III. Tip DCT ise ortonormal II. Tip DCT'nin tam olarak tersidir. Bunun dışında ters dönüşüm IDCT ile elde edilir.

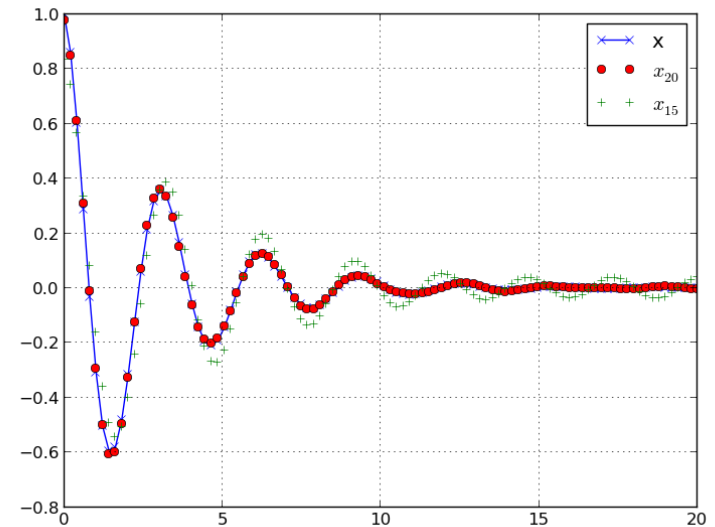
```
>>> from scipy.fftpack import dct, idct
>>> x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
>>> dct(dct(x, type=2, norm='ortho'), type=3, norm='ortho')
array([ 1. ,  2. ,  1. , -1. ,  1.5])
>>> idct(dct(x, type=2), type=2) # 2*N = 10 ile normalize olduguna dikkat ediniz!
array([ 10.,  20.,  10., -10.,  15.])
>>> idct(dct(x, type=2, norm='ortho'), type=2, norm='ortho') # Normalizasyon yok!
array([ 1. ,  2. ,  1. , -1. ,  1.5])
>>> idct(dct(x, type=3), type=3) # 2*N = 10 ile normalize!
array([ 10.,  20.,  10., -10.,  15.])
>>> idct(dct(x, type=3, norm='ortho'), type=3, norm='ortho') # Normalizasyon yok!
array([ 1. ,  2. ,  1. , -1. ,  1.5])
>>> idct(dct(x, type=1), type=1) # 2*(N-1) = 8 ile normalize
array([ 8., 16.,  8., -8., 12.])
```

Örnek



DCT'nin önemli özelliklerinden biri pek çok sinyal için kısa bir süre sonra katsayıların 0'a yaklaşmasıdır (energy compaction property). Bu nedenle sinyali az sayıda katsayı üzerinden yapılandırmak çok büyük bir hataya yol açmaz. Aşağıdaki örnekte x sinyalinin 20 katsayı kullanılarak x_{20} ve 15 katsayı kullanılarak x_{15} dönüştürülmüş formları görülmektedir. Hatanın fazla olmadığı açıktır!

```
>>> from scipy.fftpack import dct, idct
>>> import matplotlib.pyplot as plt
>>> N = 100
>>> t = np.linspace(0,20,N)
>>> x = np.exp(-t/3)*np.cos(2*t)
>>> y = dct(x, norm='ortho')
>>> window = np.zeros(N)
>>> window[:20] = 1
>>> yr = idct(y*window, norm='ortho')
>>> sum(abs(x-yr)**2) / sum(abs(x)**2)
0.0010901402257
>>> plt.plot(t, x, '-bx')
>>> plt.plot(t, yr, 'ro')
>>> window = np.zeros(N)
>>> window[:15] = 1
>>> yr = idct(y*window, norm='ortho')
>>> sum(abs(x-yr)**2) / sum(abs(x)**2)
0.0718818065008
>>> plt.plot(t, yr, 'g+')
>>> plt.legend(['x', '$x_{20}$', '$x_{15}$'])
>>> plt.grid()
>>> plt.show()
```



Süreksiz Sinüs Dönüşümü

Discrete Sine Transform (DST)

- ✓ Herhangi bir fonksiyonu sinüs fonksiyonlarının toplamı şeklinde ($y[k]$) ifade etmek de mümkündür. Bunun için üstüste bindirilen ve frekansları temel frekansın tam katları olan sinüslerin genlikleri ($x[n]$) aranır. Bu işlemi gerçekleştirmek için literatürde yine 8 farklı yöntem önerilmiş ve bunlardan 3'ü scipy tarafından adapte edilmiştir. Dönüşüm, ters dönüşüm ve normalizasyon DCT'de olduğu gibidir.

- ✓ I. tip DST (normalizasyon seçeneği yoktur)

$$y[k] = 2 \sum_{n=0}^{N-1} x[n] \sin\left(\frac{\pi(n+1)(k+1)}{N+1}\right), \quad 0 \leq k < N$$

- ✓ II. tip DST (normalizasyon kullanılması durumunda):

$$y[k] = 2 \sum_{n=0}^{N-1} x[n] \sin\left(\frac{\pi(n+1/2)(k+1)}{N}\right), \quad 0 \leq k < N$$

- ✓ III. tip DST (normalizasyon kullanılmaması durumunda):

$$y[k] = (-1)^k x[N-1] + 2 \sum_{n=0}^{N-2} x[n] \sin\left(\frac{\pi(n+1)(k+1/2)}{N}\right), \quad 0 \leq k < N$$

numpy.fft paketi ile basit bir örnek



numpy.fft paketi de scipy.fftpack fonksiyonlarına çok benzer fonksiyonlarla Fourier Analizi imkanı sağlar. Basit bir örneği inceleyelim ([numpy_fft.py](#)).

```
import matplotlib.pyplot as plt
import numpy as np

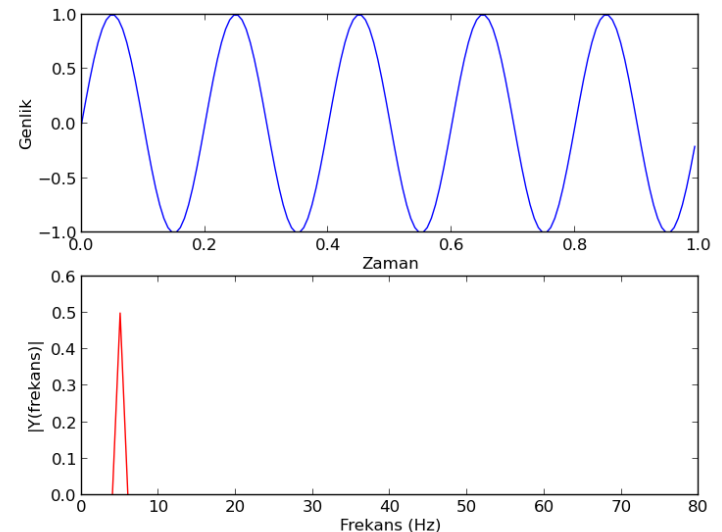
Fs = 150.0; # ne siklikta nokta alindigi (sampling rate)
Ts = 1.0/Fs; # aralik uzunlugu (sampling interval)
t = np.arange(0,1,Ts) # zaman tanim kumesi (x-ekseni)

ff = 5; # Sinaylin frekansi
y = np.sin(2*np.pi*ff*t) # sinyal

n = len(y) # sinyal uzunlugu
k = np.arange(n) # n uzunlugunda tam sayilardan olusan dizi
T = n/Fs # donem
frq = k/T # negatif ve pozitif frekanslar
frq = frq[range(n/2)] # sadece pozitif frekanslar

Y = np.fft.fft(y)/n # FFT ve n ile normalizasyon
Y = Y[range(n/2)]

fig, ax = plt.subplots(2, 1)
ax[0].plot(t,y)
ax[0].set_xlabel('Zaman')
ax[0].set_ylabel('Genlik')
ax[1].plot(frq,abs(Y), 'r') # guc spektrumu
ax[1].set_xlabel('Frekans (Hz)')
ax[1].set_ylabel('|Y(frekans)|')
plt.show()
```

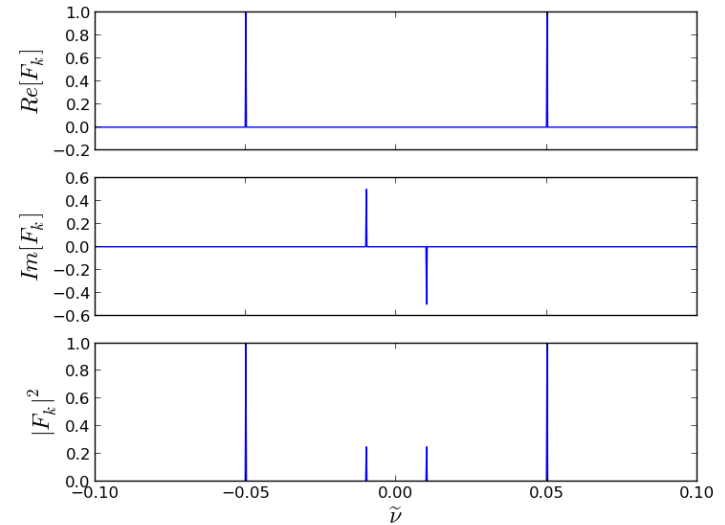


numpy.fft paketi ile basit bir örnek



Basit bir örnek daha inceleyelim (`numpy_fft2.py`).

```
from numpy import fft
import numpy as np
import matplotlib.pyplot as plt
# Veri noktası sayısı
n = 1000
# Metre cinsinden noktalar arası uzaklık
dx = 5.0
# x koordinatları
x = dx*np.arange(0,n)
w1 = 100.0 # dalgaboyu (m)
w2 = 20.0 # dalgaboyu 2 (m)
fx = np.sin(2*np.pi*x/w1) + 2*np.cos(2*np.pi*x/w2) # sinyal
Fk = fft.fft(fx)/n # fourier katsayıları (n'e bölündüğüne dikkat!)
nu = fft.fftfreq(n,dx) # Frekanslar
Fk = fft.fftshift(Fk) # O'ı merkeze kaydırma
nu = fft.fftshift(nu) # dogal frekansa kaydırma
f, ax = plt.subplots(3,1,sharex=True)
ax[0].plot(nu, np.real(Fk)) # kosinuslu (reel) terimler
ax[0].set_ylabel(r'$Re[F_k]$', size = 'x-large')
ax[1].plot(nu, np.imag(Fk)) # sinuslu (imajiner) terimler
ax[1].set_ylabel(r'$Im[F_k]$', size = 'x-large')
ax[2].plot(nu, np.absolute(Fk)**2) # Güç spektrumu
ax[2].set_ylabel(r'$\|F_k\|^2$', size = 'x-large')
ax[2].set_xlabel(r'$\tilde{\nu}$', size = 'x-large')
plt.show()
```



Kaynaklar

- **www.thefouriertransform.com**
- http://en.wikipedia.org/wiki/Window_function
- https://en.wikipedia.org/wiki/Discrete_cosine_transform
- http://en.wikipedia.org/wiki/Discrete_sine_transform
- <http://docs.scipy.org/doc/scipy/reference/tutorial/fftpack.html>