به نام خدا

دانشکده مهندسی برق و کامپیوتر

تمرین کامپیوتری سری اول

تجزیه و تحلیل سیستم ها

تاریخ تحویل: ۹۷/۰۸/۲۷

دانشگاه تهران

پردیس دانشکده های فنی

سیگنال ها و سیستم ها – سیستم های LTI – سری فوریه

توابع پایتون برای نمایش سیگنال ها (آموزش)

در حالت عمومی، سیگنال ها با یک بردار ردیفی یا ستونی نمایش داده می‌شوند. اندیس تمامی بردار ها در پایتون از صفر شروع می‌شود. در پایتون از آرایه[1]های کتابخانه نام‌پای[2]برای ایجاد بردار ها استفاده می‌کنیم. `x[0]` اولین عضو آرایه x است. در مواردی که برای نمایش سیگنال ها به اندیس های منفی نیاز دارد، می‌توانیم از یک آرایه به عنوان اندیس زمانی استفاده کنیم. برای مثال، برای نمایش سیگنال

$$x[n] = \begin{cases} 2n, & -3 \le n \le 3 \\ 0, & otherwise \end{cases}$$

می‌توانیم نمونه های غیر صفر را از ضرب ۲ در یک آرایه که شامل اعداد ۳- تا ۳ است، بدست آوریم.

```
import numpy as np

n = np.array([n for n in range(-3, 4)])
x = 2 * n
```

حال فرض کنید می خواهیم سیگنال x را در بازه $[-10, 10]$ رسم کنیم. پس لازم است به آرایه x از هر دو طرف به تعداد ۷ صفر اضافه کنیم.

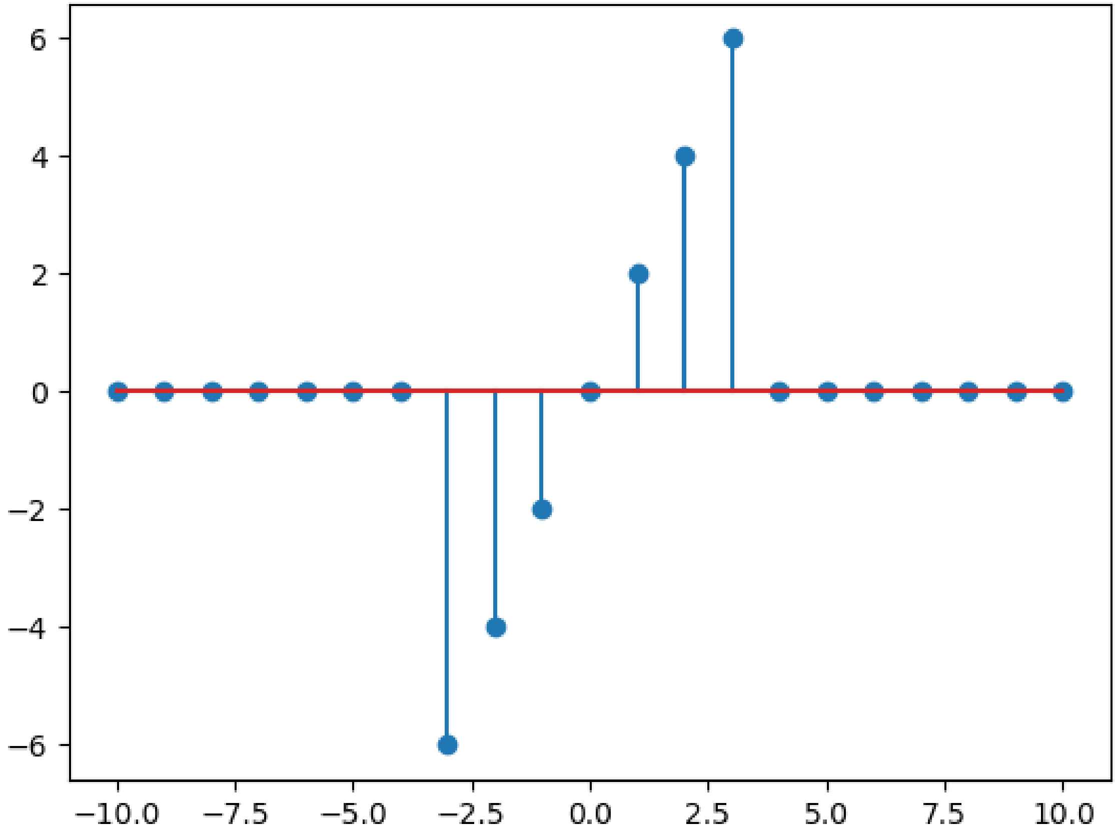
```
import numpy as np

n = np.array([n for n in range(-3, 4)])
x = 2 * n
n = np.array([n for n in range(-10, 11)])
x = np.concatenate([np.zeros(7), x, np.zeros(7)])
```

اکنون با دستور `stem` کتابخانه `matplotlib` می‌توانیم سیگنال x را رسم کنیم و با دستور `show` آن را نمایش دهیم.

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

n = np.array([n for n in range(-3, 4)])
x = 2 * n
n = np.array([n for n in range(-10, 11)])
x = np.concatenate([np.zeros(7), x, np.zeros(7)])
plt.stem(n, x)
plt.show()
```

[1] List
[2] NumPy

نتیجه:

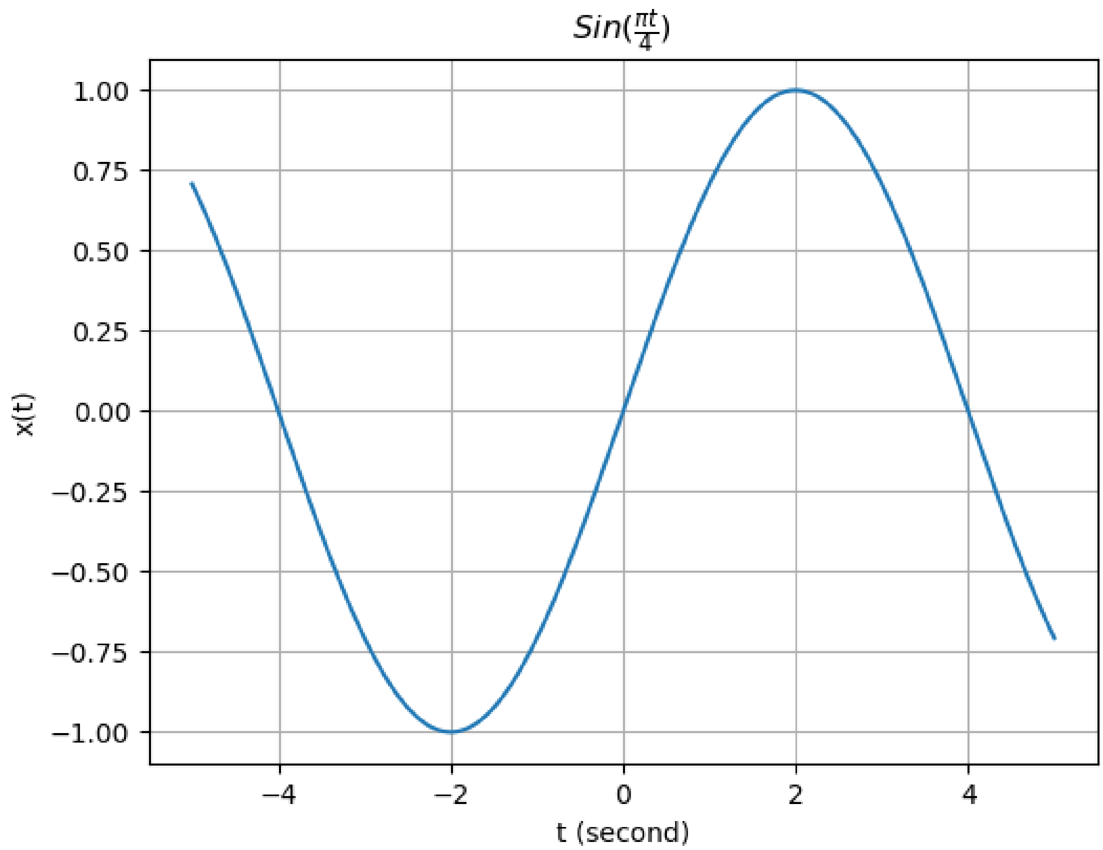


شکل ۱

سیگنال های سینوسی و نمایی مختلط، اهمیت زیادی در مطالعه سیستم های خطی دارند. برای رسم سیگنال های پیوسته-زمان، به صورت تقریبی از تابع `plot` استفاده می‌کنیم. دستور `plot` برخلاف `stem` نقاط مجاور را به هم وصل می‌کند. پس اگر فاصله نقاط آرایه زمانی را به صورت مناسب تعیین کنیم، می‌توانیم نمایش خوبی از سیگنال های پیوسته-زمان داشته باشیم. برای مثال سیگنال $x(t) = \sin(\pi t/4)$ را در فاصله $-5 \leq t \leq 5$ رسم می‌کنیم. برای ایجاد آرایه اندیس زمانی می‌توانیم از تابع `linspace` استفاده کنیم. ورودی های این تابع به ترتیب نقطه شروع، نقطه پایان و تعداد نقاط از نقطه شروع تا نقطه پایان است. با استفاده از توابع `title`، `xlabel` و `ylabel` می‌توانیم برچسب های مختلفی به نمودار بزنیم. تابع `grid` شبکه پشت نمودار را رسم می‌کند.

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

t = np.linspace(-5, 5, 1000)
x = np.sin(np.pi * t / 4)

plt.plot(t, x)
plt.title(r'$Sin(\frac{\pi t}{4})$')
plt.xlabel('t (second)')
plt.ylabel('x(t)')
plt.grid()
plt.show()
```

نتیجه:



شکل ۲

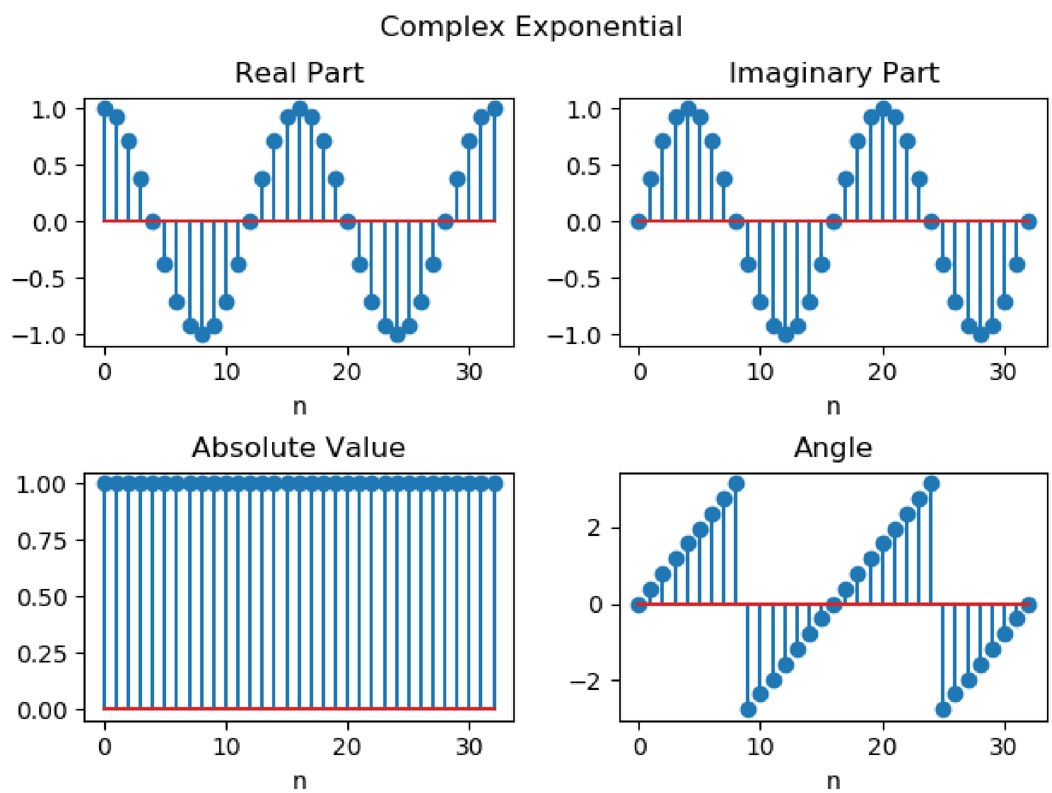
با تغییر عدد ۱۰۰۰، به اعداد کوچکتر تغییرات خروجی را مشاهده کنید.

در مثال بعدی برای سیگنال گسسته-زمان $x[n] = \exp(j(\pi/8)n)$ در بازه $0 \leq n \leq 32$ قسمت حقیقی، قسمت موهومی، اندازه و فاز را رسم می‌کنیم. در پایتون `1j` برای تعریف اعداد مختلط استفاده می‌شود.

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

n = np.linspace(0, 32, 33)
x = np.exp(1j * (np.pi/8) * n)
fig, axs = plt.subplots(2, 2)
fig.suptitle('Complex Exponential')


axs[0, 0].stem(np.real(x))
axs[0, 0].set_title('Real Part')
axs[0, 0].set_xlabel('n')

axs[0, 1].stem(np.imag(x))
axs[0, 1].set_title('Imaginary Part')
axs[0, 1].set_xlabel('n')

axs[1, 0].stem(np.abs(x))
axs[1, 0].set_title('Absolute Value')
axs[1, 0].set_xlabel('n')
```

```
axs[1, 1].stem(np.angle(x))
axs[1, 1].set_title('Angle')
axs[1, 1].set_xlabel('n')

plt.tight_layout()
fig.subplots_adjust(top=0.88)

plt.show()
```

نتیجه:



شکل ۳

خروجی تابع `angle` به رادیان است. برای تبدیل آن به درجه، باید خروجی را به $180/\pi$ ضرب کنیم.

در پایتون می‌توان بر روی سیگنال ها، در صورتی که تعداد المان آنها در آرایه یکسان باشد، اعمال چهارگانه (جمع، تفریق، ضرب و تقسیم) انجام داد. همچنین می‌توان سیگنال ها را در یک عدد ضرب کرد و یا آن ها را به توان عددی رساند. (البته برای چهار عمل اصلی باید فرض کنید که هر دو سیگنال در ناحیه زمانی یکسانی تعریف شده اند.)

```
import numpy as np

x1 = np.sin((np.pi/4) * np.linspace(0, 15, 16))
x2 = np.sin((np.pi/4) * np.linspace(0, 15, 16))

y1 = x1 + x2
y2 = x1 - x2
y3 = x1 * x2
y4 = x1 / x2
y5 = 2 * x1
y6 = x1 ** 2 # x1 to the power of 2
```

مزایای استفاده از توابع در برنامه نویسی را می‌دانید. در پایتون به صورت زیر می‌توانیم یک تابع تعریف کنیم. scope ها در پایتون با indentation مشخص می‌شوند و باید آن را رعایت کنید.

```python
def foo(x):
    y = 2 * x
    z = (5/9) * (x - 32)
    return (y, z)

(y, z) = foo(-40)
print((y, z))

(y, z) = foo(212)
print((y, z))
```

نتیجه:

```
(-80, -40.0)
(424, 100.0)
```

سیگنال های سینوسی گسسته-زمان

تمرین ۱: سیگنال زیر را در نظر بگیرید:

$$x_1[n] = \cos^2\left(\frac{\pi n}{4}\right)$$
$$x_2[n] = \sin\left(\frac{\pi n}{4}\right)\cos\left(\frac{\pi n}{8}\right)$$
$$x_3[n] = \cos\left(\frac{2\pi n}{N}\right) + 3\sin\left(\frac{5\pi n}{2N}\right)$$

فرض کنید در سیگنال $x_3[n]$، $N = 6$ است. تعیین کنید هر یک از سیگنال ها متناوب هستند یا خیر. سیگنال $x_1[n]$ و $x_2[n]$ را در بازه‌ی $0 \leq n \leq 31$ و سیگنال $x_3[n]$ اگر متناوب بود، نمودار آن را در دو تناوب با شروع از $n = 0$ و اگر متناوب نبود، در بازه $0 \leq n \leq 4N$ رسم کنید و توضیح دهید که چرا متناوب نیست. (از `stem` استفاده کنید و حتما نمودار های خود را به خوبی برچسب بزنید.)

تناوب اصلی هر سیگنال چقدر است؟ برای هر سیگنال بدون استفاده از پایتون چگونه می‌توانید تناوب اصلی آنها را بدست آورید؟

تمرین ۲: سیگنال هایی را که در تمرین ۱ رسم کردید، در نظر بگیرید. آیا جمع دو سیگنال متناوب لزوماً متناوب است؟ ضرب دو سیگنال چطور؟ دلایل خود را به وضوح توضیح دهید.

ویژگی سیستم های گسسته-زمان

سیستم های گسسته-زمان را معمولا با یکسری از ویژگی هایشان می‌شناسند. ویژگی هایی مانند خطی بودن، تغییر ناپذیری با زمان، پایداری، علّی بودن و مکوس پذیری. طرز نمایش دارا بودن یا نبودن یک ویژگی توسط یک سیستم بسیار اهمیت دارد. در این بخش به کمک پایتون و با استفاده از مثال نقض، دارا نبودن یکسری از ویژگی ها را برای سیستم های داده شده، نشان می‌دهید.

تمرین ۳: در این تمرین یک ویژگی از هر سیستم و همچنین ورودی (ورودی ها) که می‌توانید به کمک آن نشان دهید سیستم داده شده آن ویژگی را ندارد، مشخص شده‌اند. برای سیگنال ورودی و خروجی از `numpy.array` استفاده کنید. سپس برای هر سیگنال نمودار آن را رسم کنید و به وضوح بیان کنید چگونه از روی نمودار ها می‌توان تحلیل کرد که سیستم داده شده آن ویژگی را ندارد.

الف) سیستم $y[n] = \sin\big((\pi/2)x[n]\big)$ خطی نیست. از سیگنال های $x_1[n] = \delta[n]$ و $x_2[n] = 2\delta[n]$ استفاده کنید و (با رسم نمودار سیگنال ها) نشان دهید که سیستم داده شده، چگونه خطی بودن را نقض می‌کند.

ب) سیستم $y[n] = x[n] + x[n+1]$ علّی نیست. با استفاده از سیگنال $x[n] = u[n]$ آن را نشان دهید. (راهنمایی: با استفاده از `numpy.array` دو بردار تعریف کنید و سیگنال ورودی را در بازه $-5 \leq n \leq 9$ و سیگنال خروجی را در بازه $-6 \leq n \leq 9$ رسم کنید.)

ج) نشان دهید سیستم $y[n] = \log(x[n])$ پایدار نیست.

جابجایی (شیفت) زمانی سیگنال های پیوسته-زمان

در این قسمت اثرات انتقال و تغییر مقیاس زمانی سیگنال های پیوسته-زمان را بررسی می‌کنیم. سیگنال زیر را در نظر بگیرید:

$$f(t) = t\big(u(t) - u(t-2)\big)$$

که در آن $u(t)$ سیگنال پله واحد است:

$$u(t) = \begin{cases} 1, t \geq 0, \\ 0, t < 0. \end{cases}$$

در کد زیر ابتدا تابع $u(t)$ را تعریف و سپس به کمک آن سیگنال $f(t)$ را رسم می‌کنیم.

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

def heaviside(t):
    """ HEAVISIDE Unit Step function
            f = Heaviside(t1 returns a vector f the same size as
            the input vector, where each element of f is 1 if the
            corresponding element of t is greater than or equal to
            zero."""
    return (t >= 0) * 1

t = np.arange(-2, 3, 0.1)
f = t * (heaviside(t) - heaviside(t - 2))
plt.plot(t, f)
plt.show()
```

تمرین ۴: با توجه به سیگنال $f(t)$ نمودار سیگنال های پیوسته-زمان زیر را رسم کنید. در هر مورد بیان کنید سیگنال $f(t)$ چه تغییری کرده است؟ (مثلا "۲ واحد به سمت بالا/پایین/راست/چپ جابجا شده" یا "با ضریب ۲ فشرده/گسترده شده" یا "نسبت به محور افقی/عمودی منعکس شده")

$$g_1(t) = f(-t)$$
$$g_2(t) = f(t+1)$$
$$g_3(t) = f(t-3)$$
$$g_4(t) = f(-t+1)$$
$$g_5(t) = f(-2t+1)$$

انرژی و توان سیگنال های پیوسته-زمان

برای یک سیگنال پیوسته-زمان مانند $x(t)$ انرژی سیگنال در بازه $-a \leq t \leq a$ به صورت زیر تعریف می‌شود:

$$E_a = \int_{-a}^{a} |x(t)|^2 \, dt$$

که در آن $|x|^2 = xx^*$ و x^* مزدوج مختلط x است. پس برای یک سیگنال متناوب با تناوب اصلی T، $E_{T/2}$ انرژی سیگنال در یک تناوب است. انرژی کل سیگنال نیز به صورت زیر تعریف می‌شود: (اگر حد زیر موجود باشد.)

$$E_\infty = \lim_{a\to\infty} E_a$$

در حالی که بسیاری از سیگنال های طبیعی انرژی محدودی دارند، بسیاری از سیگنال های پیوسته-زمان در این درس اینگونه نیستند. برای مثال هر سیگتال متناوب انرژی نامحدودی دارد. برای این دسته از سیگنال ها توان متوسط را اندازه گیری می‌کنند که به صورت انرژی سیگنال در یک بازه تقسیم بر طول بازه، تعریف می‌شود. به این ترتیب توان متوسط (زمانی) سیگنال در بازه $-a \leq t \leq a$ به صورت زیر تعریف می‌شود:

$$P_a = \frac{E_a}{2a}, \;\; a > 0$$

و توان متوسط کل سیگنال به صورت زیر تعریف می‌شود: (اگر حد زیر موجود باشد.)

$$P_\infty = \lim_{a\to\infty} P_a$$

در این تمرین ارتباط P_∞ و $E_{T/2}$ برای سیگنال های متناوب را بررسی می‌کنید.

تمرین ۵: سیگنال های زیر را در نظر بگیرید:

$$x_1(t) = \cos(\pi t/5)$$
$$x_2(t) = e^{j2\pi t/3} + e^{j\pi t}$$
$$x_3(t) = 2r(t) - r(t-1) + u(t+1)$$
$$x_4(t) = 4\delta(t)$$

الف) برای هر یک از سیگنال های بالا E_a را بدست آورید (با محاسبات دستی) و آن را در بازه $0 \leq a \leq 30$ رسم کنید. با افزایش a انرژی سیگنال ها چگونه تغییر می‌کند؟ با توجه به تغییرات انرژی چه انتظاری برای E_∞ دارید؟

ب) انرژی و توان سیگنال های بالا را به دست آورید و با نتایج حاصل از تحلیل دستی مقایسه کنید. تعیین کنید هر کدام از سیگنال ها انرژی هستند یا توان و یا هیچ کدام. آیا دلتا سیگنال انرژی است؟

ج) برای هر یک از سیگنال های بالا P_a را بدست آورید (با محاسبات دستی) و آن را در بازه $0.1 \leq a \leq 60$ رسم کنید. (به خاطر داشته باشید که P_a برای $a = 0$ تعریف نشده است.) با افزایش a توان سیگنال ها چگونه تغییر می‌کند؟ برای هر سیگنال P_∞ را با $E_{T/2}/T$ مقایسه کنید و به وضوح بیان کنید این نتایج را چگونه می‌توان از تعریف P_∞ و $E_{T/2}$ برداشت کرد؟

کانولوشن گسسته-زمان

همان‌طور که می‌دانید کانولوشن دو سیگنال گسسته-زمان $h[n]$ و $x[n]$ به صورت زیر تعریف می‌شود:

$$y[n] = \sum_{m=-\infty}^{+\infty} x[m]h[n-m]$$

تصویری از تعریف بالا را می‌توان به این صورت شرح داد: ابتدا دنباله $h[m]$ نسبت به محور عمودی منعکس می‌شود و n نمونه به سمت چپ یا راست (با توجه به علامت n) جابجا می‌شود. سپس دنباله $h[n-m]$ در دنباله $x[m]$ ضرب می‌شود و حاصل جمع دنباله حاصل را بدست می‌آوریم. این تصویر از ویژگی خطی بودن و تغییر ناپذیری با زمان سیستم های گسسته-زمان بدست می‌آید. در این قسمت استفاده از تابع `numpy.convolve` را یاد می‌گیرید.

تمرین ۶: در این تمرین سیگنالی گسسته-زمان و پاسخ ضربه یک سیستم LTI را تعریف می‌کنید. سپس می‌توانید با استفاده از `numpy.convolve` خروجی سیستم را حساب کنید.

الف) از آنجا که تابع `numpy.convolve` اندیس زمانی سیگنال های ورودی را در نظر نمی‌گیرد، برای رسیدن به جواب مطلوب نیاز به یکسری ملاحظات دارید. برای سیگنال های $h[n] = 2\delta[n+1] - 2\delta[n-1]$ و $x[n] = \delta[n] + \delta[n-2]$ بردار h و x را بسازید. بردار y را به صورت $y[n] = x[n] * h[n]$ تعریف کنید و اندیس زمانی مناسب برای y را در بردار ny بسازید. نمودار y را با استفاده از تابع stem رسم کنید.

ب) دو سیگنال $h[n]$ و $x[n]$ با طول محدود را در نظر بگیرید که با بردار های h و x تعریف شده‌اند. اندیس زمانی این دو سیگنال به صورت `nh=numpy.arange(a, b)` و `nx=numpy.arange(c, d)` تعریف شده‌اند. بردار y را با کانوالو کردن دو بردار x و h بدست آورید. حال باید اندیس زمانی مناسب برای بردار y را بسازید (ny). برای این کار کانولوشن دو سیگنال $h[n] = \delta[n-a] + \delta[n-b]$ و $x[n] = \delta[n-c] + \delta[n-d]$ را به صورت دستی حساب کنید. با توجه به جواب می‌توانید ny را برای هر مقدار a، b، c و d بدست آورید. (راهنمایی: اگر $a = 0$، $b = N-1$، $c = 0$ و $d = M-1$ باشد، طول بردار y باید $M+N-1$ شود.)

ج) سیگنال گسسته $x[n]$ و پاسخ ضربه واحد $h[n]$ را در نظر بگیرید که به صورت زیر تعریف شده‌اند:

$$x[n] = \left(\frac{1}{2}\right)^{n-2} u[n-2]$$
$$h[n] = u[n+2]$$

اگر بخواهید خروجی این سیستم یعنی $y[n] = x[n] * h[n]$ را با کمک دستور `numpy.convolve` بدست آورید، باید ملاحظاتی برای طول بینهایت دو سیگنال $x[n]$ و $h[n]$ در نظر بگیرید. مقادیر $x[n]$ در بازه‌ی $0 \le n \le 24$ را در بردار x و مقادیر $h[n]$ در بازه‌ی $0 \le n \le 14$ را در بردار h ذخیره کنید. حالا در بردار y مقدار `numpy.convolve(x, h)` را ذخیره کنید. این در حالی است که شما فقط قسمتی از دو سیگنال $x[n]$ و $y[n]$ را در نظر گرفته‌اید پس فقط بخشی از سیگنال خروجی دارای مقادیر صحیح است. تعیین کنید به عنوان خروجی سیستم، چه بخشی از مقادیر دنباله خروجی با ارزش و چه بخشی بی ارزش است. مقادیر a، b، c و d را برای `nx=numpy.arange(a, b)` و `nh=numpy.arange(c, d)` تعیین کنید. سپس با استفاده از نتیجه قسمت ب اندیس زمانی مناسب (ny) برای سیگنال خروجی را بدست آورید. با استفاده از stem سیگنال خروجی را رسم کنید و بخش با ارزش آن را مشخص کنید. از برچسب های مناسب برای نمایش اندیس زمانی سیگنال خروجی استفاده کنید.

تمرین ۷: در این تمرین می‌خواهیم از روش کانولوشن بلوکی استفاده کنیم. این روش در پیاده سازی زمان برخط (real-time) فیلتر های دیجیتال برای پردازش صوت/تصویر استفاده می‌شود. در این روش سیگنال ورودی (که سیگنالی با طول بینهایت/نا معلوم است) را به بلوک های کوچکتر تقسیم می‌کنیم. حال می‌توانیم هر کدام از این بلوک ها را به صورت مستقل پردازش کنیم البته با کمی تأخیر. خطی بود کانولوشن این تضمین را می‌دهید که برهم‌نهی (superposition) خروجی های حاصل از پردازش از بلوک ها، با کانولوشن کل سیگنال با پاسخ ضربه یکسان است. وجود سخت افزار با کارایی مناسب و الگوریتم هایی برای محاسبه کانولوشن سیگنال هایی با طول محدود، بر اهمیت روش کانولوشن بلوکی می‌افزاید. در این تمرین هر کدام از کانولوشن های کوچک را با دستور `numpy.convolve` حساب می‌کنید.

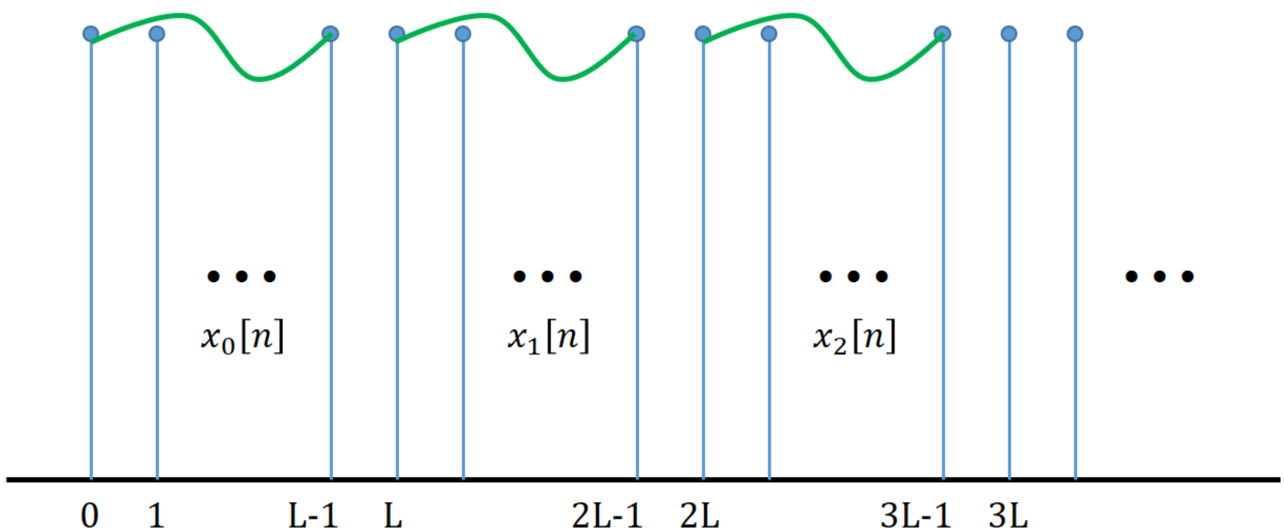
فرض کنید یک فیلتر با پاسخ ضربه $h[n]$ (با طول محدود) دارید که فقط در بازه‌ی $0 \le n \le P-1$ غیر صفر است. همچنین فرض کنید دنباله ورودی یعنی $x[n]$ برای $n < 0$ صفر است و طول آن به طور قابل ملاحظه ای از P بیشتر است. حال می‌توانید به صورت زیر سیگنال $x[n]$ را به بلوک هایی با طول L تقسیم کنید:

$$x[n] = \sum_{r=0}^{\infty} x_r[n - rL]$$

که در آن $L > P$ است و داریم:

$$x_r[n] = \begin{cases} x[n+rL], & 0 \le n \le L-1 \\ 0, & otherwise \end{cases}$$

شکل زیر را ببینید:



شکل ۴ – تجزیه بلوکی سیگنال $x[n]$

الف) برای $h[n] = (0.9)^n(u[n] - u[n-10])$ و $x[n] = \cos(n^2)\sin(2\pi n/5)$ ، $y[n] = h[n] * x[n]$ را در بازه‌ی $0 \le n \le 99$ به صورت مستقیم با استفاده از دستور `numpy.convolve` حساب کنید و نمودار آن را با استفاده از `stem` در بازه داده شده رسم کنید.

ب) با فرض $L = 50$ سیگنال $x[n]$ را به دو بلوک تقسیم کنید که طول هر کدام 50 شود. $y_0[n] = h[n] * x_0[n]$ و $y_1[n] = h[n] * x_1[n]$ را که در آن $x_0[n]$ ۵۰ نمونه اول از سیگنال $x[n]$ و $x_1[n]$ ۵۰ نمونه دوم سیگنال $x[n]$ است، حساب کنید. حال فرم سیگنال خروجی به صورت زیر خواهد بود:

$$y[n] = x[n] * h[n] = y_0[n] + y_1[n-k]$$

در عبارت بالا k مناسب را بدست آورید. (دقت کنید که طول هر کدام از سیگنال های $y_0[n]$ و $y_1[n]$ باید $L + P - 1$ باشد.) وقتی سیگنال $y_0[n]$ و $y_1[n]$ را با هم جمع می‌کنید، ناحیه‌ای وجود دارد که در آن مقادیر غیر صفر از دو سیگنال با هم جمع می‌شوند. به این خاطر به روش کانولوشن بلوکی، "هم‌پوشانی و اضافه کردن" نیز می‌گویند. سیگنال خروجی یعنی $y[n]$ را با اسنفاده از این روش حساب کنید و آن را در بازه $0 \le n \le 99$ با استفاده از stem رسم کنید. آیا به همان نتیجه قسمت الف می‌رسید؟ نتایج را تحلیل کنید.

ج) در این قسمت باید یک تابع در پایتون بنویسید که عمل هم‌پوشانی و اضافه کردن را انجام دهد. ورودی های این تابع پاسخ ضربه(h)، بردار ورودی سیستم (x) و طول هر بلاک (L) هستند. طول بردار x دلخواه و طول هر بلاک (L) یک عدد دلخواه بزرگ تر از طول فیلتر است. حال قسمت ب را دوباره با استفاده از تابع خود انجام دهید.

آموزش (`fft`)

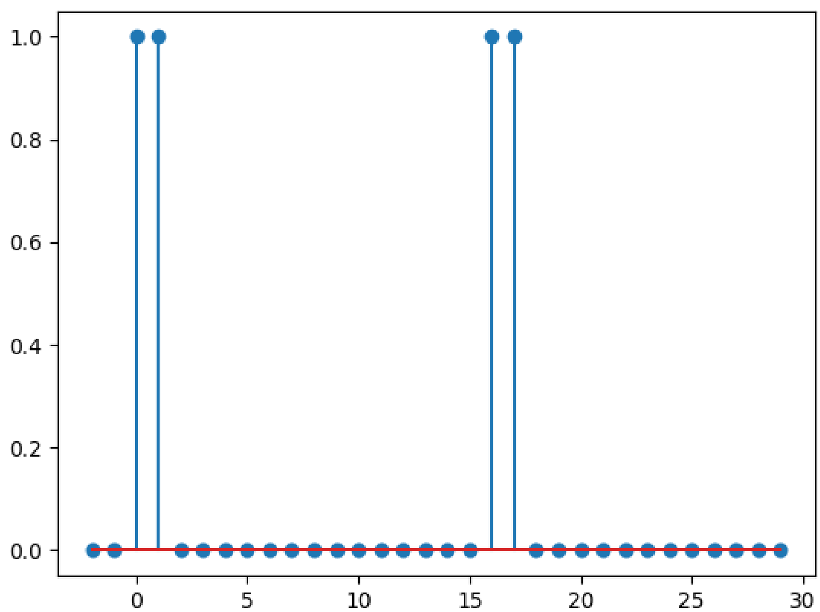
همان‌طور که می‌دانید، معادلات سنتز و تجزیه سری فوریه برای سیگنال های گسسته-زمان (DTFS) برای سیگنال $x[n]$ با تناوب اصلی N از روابط زیر بدست می‌آیند:

$$x[n] = \sum_{k=0}^{N-1} a_k e^{jk(2\pi/N)n} \qquad (1)$$

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk(2\pi/N)n} \quad (2)$$

بخاطر داشته باشید $x[n]$ دارای تناوب N است. پس سری معادله (2) را می‌توانیم روی هر N نمونه متوالی از $x[n]$ حساب کنیم. به همین شکل a_k روی k با تناوب N تکرار می‌شود و سری معادله (1) را می‌توانیم روی هر N نمونه متوالی از a_k حساب کنیم. برای محاسبه دو معادله (1) و (2) در پایتون یک تابع بسیار کارآمد وجود دارد. اگر `x` یک بردار N نقطه‌ای باشد که مقادیر $x[n]$ در بازه‌ی $0 \le n \le N-1$ را شامل شود، DTFS سیگنال $x[n]$ را می‌توان به صورت `a=(1/N)*fft(x)` بدست آورد که در آن `a` مقادیر a_k را در بازه‌ی $0 \le k \le N-1$ دارد. تابع `fft` یک پیاده سازی کارآمد از معادله (2) است که با ضریب N مقیاس شده است. برای مثال سیگنال $x[n]$ با دوره تناوب $N = 16$ را در نظر بگیرید.

$$x[n] = \begin{cases} 1 & n = 0, 1 \\ 0 & otherwise \end{cases}$$

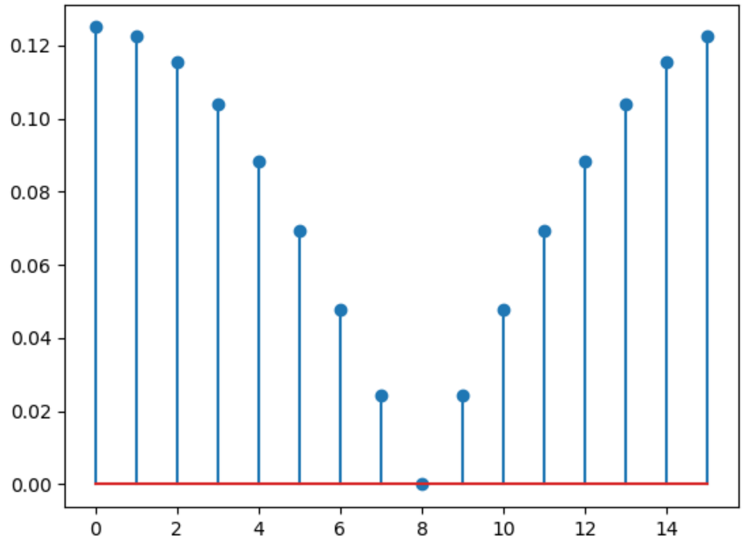


شکل ۵

حال برای بدست آوردن DTFS به این شکل عمل می‌کنیم.

```
import numpy as np

N = 16
x = np.concatenate([np.ones(2), np.zeros(14)])
a = np.fft.fft(x)/N
```

برای این که معمولا ضرایب سری فوریه اعدادی مختلط هستند، برای نمایش از اندازه استفاده می‌شود. برای اندازه می‌توانید از دستور `abs` استفاده کنید. در شکل زیر اندازه ضرایب سری فوریه را برای مثال قبل می‌بینیم.



شکل ۶

برای محاسبه عکس سری فوریه نیز می‌توان از دستور `ifft` استفاده کرد. در اینجا نیز باید عبارت حاصل را در N ضرب کنید.

آموزش(freqz)

سیگنال $e^{j\omega n}$ تابع ویژه برای سیستم های LTI است. برای هر مقدار از ω پاسخ فرکانسی $H(e^{j\omega})$ مقدار ویژه سیستم LTI برای تابع ویژه $e^{j\omega n}$ است؛ یعنی وقتی ورودی سیستم LTI سیگنال $x[n] = e^{j\omega_0 n}$ باشد، خروجی سیستم $y[n] = H(e^{j\omega_0})e^{j\omega_0 n}$ خواهد بود. برای بدست آوردن پاسخ فرکانسی سیستم LTI و علّی که به کمک معادله تفاضلی مشخص می‌شود، می‌توانید از دستور `freqz` استفاده کنید. این دستور در کتابخانه `scipy.signal` قرار دارد. نحوه استفاده از این تابع به شکل زیر است:

```
import scipy.signal as signal

w, h = signal.freqz(b, a, worN=512)
```

a و b ضرایب معادله تفاضلی هستند که به عنوان ورودی به تابع می‌دهیم. خروجی تابع در اصل $H(e^{j\omega})$ است که تابعی پیوسته و متناوب است. بنابراین `worN` مشخص می‌کند، که چه تعداد فرکانس با فاصله مساوی از یکدیگر انتخاب شود. خروجی w مقدار فرکانس‌های انتخاب شده در بازه 0 تا π است. خروجی h نیز مقدار $H(e^{j\omega})$ در فرکانس مربوطه است.

برای به دست‌آوردن ضرایب a و b که دو بردار هستند، معادله تفاضلی زیر را در نظر بگیرید.

$$a_0 y[n] + a_1 y[n-1] + \cdots + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M]$$

بردار a و b به شکل زیر تعریف می‌شوند.

$$a = [a_0. a_1. \dots. a_N] \quad . \quad b = [b_0. b_1. \dots. b_M]$$

سنتز سیگنال‌ها به کمک ضرایب سری فوریه

تمرین ۸: سیگنال‌های زیر را در نظر بگیرید.

$$x_1[n] = \{1. \quad 0 \le n \le 7.$$

$$x_2[n] = \begin{cases} 1. & 0 \le n \le 7. \\ 0. & 8 \le n \le 15. \end{cases}$$

$$x_3[n] = \begin{cases} 1. & 0 \le n \le 7. \\ 0. & 8 \le n \le 31. \end{cases}$$

سیگنال‌های $x_1[n]$، $x_2[n]$ و $x_3[n]$ به ترتیب دارای دوره تناوب $N_1 = 8$، $N_2 = 16$ و $N_3 = 32$ هستند.

الف) بردار های `x1`، `x2` و `x3` نمونه های سیگنال های $x_1[n]$، $x_2[n]$ و $x_3[n]$ را در یک دوره تناوب دارند. این بردار ها بدست آورید و سیگنال های $x_1[n]$، $x_2[n]$ و $x_3[n]$ را در بازه $0 \le n \le 63$ رسم کنید. توجه کنید بدلیل اینکه سیگنال ها متناوب‌اند، باید بردار تعریف شده برای هر سیگنال را به تعدادی تکرار کنید تا کل این بازه تکمیل شود. (برای رسم از برچسب های مناسب استفاده کنید.)

ب) به کمک تابع `fft` که در بخش‌های قبلی توضیح داده شد، بردار ضرایب سری فوریه `a1` و `a2` و `a3` را که به ترتیب برای سیگنال های $x_1[n]$، $x_2[n]$ و $x_3[n]$ هستند، بدست آورید. بخش حقیقی، موهومی و همچنین اندازه هر یک از بردار ضرایب را رسم کنید. مقادیر `a1[0]` و `a2[0]` و `a3[0]` (مقدار DC توابع) را از روی شکل محاسبه کرده و با مقادیر بدست آمده از تابع `fft` مقایسه کنید.

ج) در این قسمت می‌خواهیم سیگنال $x_3[n]$ را به کمک ضرایب سری فوریه آن بازسازی کنیم. اما هر بار تنها بخشی از ضرایب را انتخاب می‌کنیم تا به کمک آن‌ها سیگنال را بازسازی کنیم. ضرایب را به شکل زیر انتخاب می‌کنیم.

$$x_{3_2}[n] = \sum_{k=-2}^{2} a_k e^{jk\left(\frac{2\pi}{32}\right)n}$$

$$x_{3_8}[n] = \sum_{k=-8}^{8} a_k e^{jk\left(\frac{2\pi}{32}\right)n}$$

$$x_{3_{12}}[n] = \sum_{k=-12}^{12} a_k e^{jk\left(\frac{2\pi}{32}\right)n}$$

$$x_{3\_all}[n] = \sum_{k=-15}^{16} a_k e^{jk\left(\frac{2\pi}{32}\right)n}$$

از آن‌جایی که $x_3[n]$ یک سیگنال حقیقی است، پس داریم $a_k = a_{-k}^*$. پس برای محاسبه این مقادیر تمامی مقادیر لازم را در دسترس دارید. همچنین برای صرف نظر از مقادیر موهومی بسیار کوچک تولید شده، مقدار حقیقی این عبارت را در نظر بگیرد. این چهار تابع را در بازه $0 \le n \le 31$ رسم کنید. آیا با افزایش تعداد ضرایب انتخاب شده شکل بهتر می‌شود؟ آیا پدیده `Gibb’s` در اینجا قابل مشاهده است؟

فیلترهای گسسته-زمان بازگشتی مرتبه اول

تمرین ۹: دو سیستم زیر را در نظر بگیرید:

$$system1: y[n] - 0.8y[n-1] = x[n]$$

$$system2: y[n] + 0.8y[n-1] = x[n]$$

که ورودی آن سیگنال $x[n]$ با ضرایب سری فوریه a_k و دوره تناوب $N = 20$ است.

$$a_k = \begin{cases} 3/4 & k = \pm 1 \\ -1/2 & k = \pm 9 \\ 0 & otherwise \end{cases}$$

الف) بردارهای `a1` و `b1` را برای سیستم ۱ - با توجه به تعریف ورودی تابع `freqz` - مشخص کنید. همچنین `a2` و `b2` را برای سیستم ۲ بدست آورید.

ب) پاسخ فرکانسی سیستم ۱ و سیستم ۲ را بدست آورید. اندازه و فاز هر کدام را در بازه 0 تا 2π و برای `worN=1024` رسم کنید. (در حالت عادی تابع `freqz`، در بازه 0 تا π خروجی می‌دهد، برای این که بازه 0 تا 2π خروجی داهد باید گزینه `whole` را `true` قرار دهید.) مشخص کنید کدام یک فیلتر بالاگذر و کدام یک پایین گذر است.

ج) ضرایب سری فوریه را که در قسمت قبل بدست آوردید، در بازه $0 \le n \le 19$ با استفاده از تابع `stem` رسم کنید. از بین دو مؤلفه فرکانسی مشاهده شده، مشخص کنید در هر سیستم هر کدام از این دو مولفه تضعیف یا تقویت می‌شوند.

د) با استفاده از ضرایب سری فوریه و به کمک `ifft`، سیگنال ورودی $x[n]$ را بدست آورید و آن را در بازه $-20 \le n \le 99$ رسم کنید.

ه) به کمک تابع `filter` (که در کتابخانه `scipy.signal` قرار دارد و ورودی `a` و `b` آن همانند `freqz` است.)، خروجی هر سیستم را رسم کنید. کدام خروجی دارای فرکانس‌های بالا و کدام خروجی دارای فرکانس‌های پایین است.

محاسبه سری فوریه گسسته-زمان

تمرین ۱۰(امتیازی): تابع DTFS را مشابه `fft` پیاده‌سازی کنید. به این صورت که یک تناوب سیگنال ورودی را دریافت کند و ضرایب سری فوریه متناظر را به عنوان خروجی بدهد. برای سیگنال‌هایی به طول ۱۰ ، ۱۰۰ ، ۱۰۰۰ و ۱۰۰۰۰ (می‌توانید یک سیگنال تصادفی استفاده کنید.) زمان محاسبه `fft` را با تابع خود مقایسه کنید.

تعداد محاسبات لازم برای DTFS، $O(n^2)$ می‌باشد ولی `fft` از الگوریتمی استفاده می‌کند که محاسبات آن $O(nlogn)$ است.