دانشگاه تهران

پردیس دانشکده‌های فنی

سیگنال‌ها و سیستم‌ها

تمرین کامپیوتری شماره‌ی ۱

مهلت تحویل: چهارشنبه ۲۲ اسفند ۱۳۹۷، ساعت ۲۳:۵۵

استاد: **امیرمسعود ربیعی**

طراح: **هدی برخورداریور**، **علی رنجبر**

۱ مقدمه

در این تمرین کامپیوتری قصد داریم با نرم‌افزار متلب[۱] و متمتیکا[۲] آشنا شویم.

۱.۱ متلب

متلب، یک محیط نرم‌افزاری برای انجام محاسبات عددی و یک زبان برنامه‌نویسی نسل چهارم است که از ترکیب دو واژهٔ MATrix (ماتریس) و LABoratory (آزمایشگاه) ایجاد شده‌است. این نام حاکی از رویکرد ماتریس محور برنامه است، که در آن حتی اعداد منفرد هم به عنوان ماتریس در نظر گرفته می‌شوند.

۲.۱ متمتیکا

متمتیکا، یک نرم‌افزار جبری بسیار رایج، که توسط شرکت ولفرم ریسرچ پدید آورده شده است و اکثر توابع نرم‌افزاری موردنیاز در ریاضی و علوم طبیعی را در اختیار استفاده‌کنندگان آن قرار می‌دهد.

۳.۱ مقایسه‌ی متلب و متمتیکا

- جهت‌گیری متلب بیشتر برای کار با داده هاست (که در این بسیار خوب عمل می‌کند) اما با اینکه امکان محاسبات نمادین در متلب وجود دارد، این امکان در متمتیکا بسیار آسان‌تر و کارآمدتر است.
- متلب یک محیط برنامه‌نویسی در حوزه‌ی مهندسی است و چون محاسبات آن با استفاده از تقریب و تخمین‌های ریاضیست بنابراین در کارهای ریاضی کاربردی که اصل کار همان ساختن تقریب هاست ممکن است زیاد مناسب نباشد.
  متمتیکا یک نرم‌افزار ریاضی است که هم در ریاضیات وهم در مهندسی کاربرد دارد. محاسبات نمادین و محض مثل حدگیری و مسایل جبر را به راحتی انجام داده و تمام مراحل حل را به کاربر می‌تواند نشان دهد.
- مصورسازی و رسم نمودار در هر دو نرم افزار به خوبی انجام می‌شود.
- ساختن رابط کاربری برای نرم‌افزار در متمتیکا بسیار آسان‌تر از متلب است.
- مهمترین انتقادات از متلب به خاطر متن بازنبودن و گران بودن آن است که امکان اجرای کدهای نوشته‌شده در متلب را در هر محیطی محدود می‌کند. متمتیکا به نسبت ارزان‌تر است و اجرای کدهای به محیط محدود نمی‌شود.

[۱] MATLAB
[۲] Mathematica

۴.۱ سیگنال‌ها در متلب

سیگنال‌های پیوسته‌ـ زمان (به اختصار پیوسته) متناظر با هر نقطه‌ای از محور زمان یک مقداری دارند در حالی که سیگنال‌های گسسته‌ـ زمان (به اختصار گسسته) فقط در مقادیر صحیح از محور زمانی مقدار دارند. $x[n]$ یک سیگنال گسسته را نشان می‌دهد که n فقط می‌تواند مقادیر صحیح اختیار کند.

همان‌طور که می‌دانید ذخیره تمام مقادیر یک سیگنال پیوسته در طول یک بازه‌ی زمانی ناممکن است. پس چگونه سیگنال‌های پیوسته را پردازش کنیم؟ در آینده خواهید آموخت که چگونه یک سیگنال پیوسته را با نمونه‌برداری به سیگنال گسسته تبدیل می‌کنیم. (به کمک دستور syms می‌توان به شکل پیوسته کار کرد، که به هیچ وجه توصیه نمی‌شود و در صورت استفاده نمره‌ای تعلق نخواهد گرفت.)

۵.۱ سیگنال‌ها در متمتیکا

سیگنال‌های پیوسته‌ـ زمان، به صورت نمادین تعریف شده و توابع خاص مورد استفاده دارد. سیگنال‌های گسسته‌ـ زمان، می‌توانند با ساختمان لیست (معادل ماتریس در متلب) تعریف شوند و با توابع مخصوص پردازش گسسته مورد استفاده قرار گیرند. می‌توان به این توابع سیگنال‌های پیوسته‌ـ زمان نیز به عنوان ورودی داد و خروجی گسسته دریافت کرد.

۲ آشنایی با متمتیکا

در این قسمت با متمتیکا و برخی دستورهای آن جهت محاسبه‌ی انتگرال و رسم توابع آشنا می‌شوید.

۱.۲ رسم توابع و برخی ویژگی‌ها

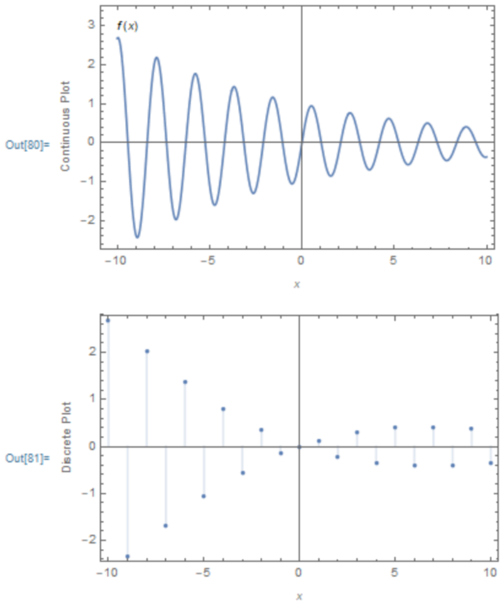
در متمتیکا می‌توان به راحتی توابع را به صورت پیوسته تعریف کرد، سپس به صورت پیوسته و گسسته رسم کرد. به قطعه کد زیر و خروجی آن دقت کنید.

```
f[x_] = Sin[3x] Exp[-0.1x];
plot1 = Plot[f[x], {x, -10, 10},
  PlotLabels -> Placed[Automatic, Above], Frame -> True,
  FrameLabel -> {x, "Continuous Plot"}]
plot2 = DiscretePlot[f[x], {x, -10, 10}, Frame -> True,
  FrameLabel -> {x, "Discrete Plot"}]
```

۲.۲ انجام دهید!

سیگنال‌های زیر را در بازه‌ی زمانی $-5 \le t \le 5$ رسم کنید. (توزیع ضربه را باید خودتان تعریف کنید، دامنه‌ی آن را برای راحتی واحد در نظر بگیرید)

1. $x_1(t) = sinc(t)$



شکل ۱: خروجی قطعه کد فوق

2. $x_2(t) = u(t+1) - u(t-1)$

3. $x_3(t) = \begin{cases} 0, & t < -1, \\ 1, & -1 < t < 0 \\ e^{-\frac{t}{2}}, & 0 < t. \end{cases}$

4. $x_4(t) = \delta(t-3) + 2\delta(t+1)$

5. $x_5(t) = StandardGaussianFunction$

۳.۲ انجام دهید!

دو تابع انرژی و توان را طوری بنوسید که یک سیگنال را به عنوان ورودی بگیرد و انرژی یا توان آن را گزارش کند. انرژی و توان سیگنال‌های فوق را در قالب Table گزارش کنید. مقادیر بدست آمده را با تحلیل دستی خود مقایسه کنید.

۴.۲ انجام دهید!

مقدار DC توابع قسمت ۲.۲ را نیز محاسبه کرده و در قالب Table ارائه دهید. مقادیر بدست آمده را با تحلیل دستی خود مقایسه کنید.

۵.۲ انجام دهید!

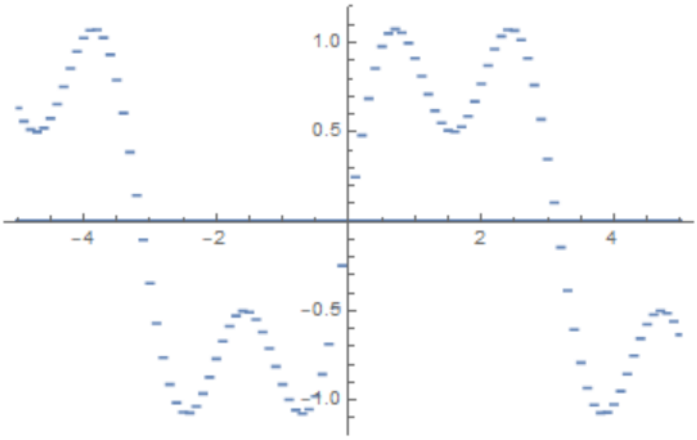
می‌خواهیم برای پردازش‌های بعدی از سیگنال زیر نمونه‌برداری کنیم. برای این که ابتدا سیگنال را تعریف کرده سپس آن را در بازه‌ی $-5 \leq t \leq 5$ رسم کنید.

$$x_6(t) = sin(t) + 0.5sin(3t)$$

نمونه‌برداری به این صورت است که تابع ضربه‌ای را از چپ به راست حرکت داده و در سیگنال مورد نظر ضرب می‌کنیم.

$$x_s(t) = x(t)\delta_{T_s}(t) = \sum_n x(nT_s)\delta(t - nT_s)$$

برای این کار، از تابع ضربه‌ای که برای رسم قسمت ۲.۲ از آن استفاده کردید کمک بگیرید. همچنین کافی است این ضرب را در بازه‌ی $-5 \leq t \leq 5$ و با فاصله‌های 0.1 انجام داده، نتایج را در یک Table ذخیره کنید، سپس شکل حاصل را رسم کنید. شکل حاصل باید مشابه شکل زیر باشد.



شکل ۲: سیگنال پس از نمونه‌برداری

۳ جابجایی زمانی سیگنال های پیوسته‌ـزمان

۱.۳ آموزش

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

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

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.

○ کد متلب

```
u = @(t) double(t >= 0);

t = -2 : 0.1 : 3;
f = t .* (u(t) - u(t - 2));
plot(t, f)
```

○ کد پایتون

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

def u(t):
    """ HEAVISIDE Unit Step function
            f = u(t) 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 * (u(t) - u(t - 2))
plt.plot(t, f)
plt.show()
```

۲.۳ انجام دهید!

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")

$$f(t+1)$$

$$f(-t+1)$$

$$f(-2t+1)$$

۴ کانولوشن گسسته_زمان

کانولوشن دو سیگنال گسسته $x[n]$ و $h[n]$ 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 زمان سیستم های گسسته_زمان بدست می‌آید. در این قسمت استفاده از تابع conv (در پایتون numpy.convolve) را یاد می‌گیرید.

۱.۴ آموزش conv

اگر فرض کنیم سیگنال $x[n]$ فقط در بازه‌ای به طول N_x و سیگنال $h[n]$ فقط در بازه‌ای به طول N_h مقدار غیر صفر داشته باشند، آنگاه سیگنال $y[n]$ فقط در بازه‌ای بطول $N_x + N_h - 1$ غیر صفر خواهد بود. بدین معنی که اگر x برداری N_x بعدی شامل مقادیر سیگنال $x[n]$ و h برداری N_h بعدی شامل مقادیر سیگنال $h[n]$ باشد، دستور

```
y = conv(h, x);
```

به تعداد $N_x + N_h - 1$ نمونه از $y[n]$ را در بردار y برمی‌گرداند.

اگر دقت کرده باشید، این دستور هیچ اطلاعی در مورد اندیس زمانی نمونه‌های سیگنال $y[n]$ (که در بردار y ذخیره شده است) برنمی‌گرداند که مورد انتظار نیز هست. چون هیچ ورودی از اندیس بردارهای x و h نمی‌گیرد. در این حالت باید خودتان اندیس‌های مناسبی بسازید. در ادامه با مثالی ساده نحوه‌ی ساخت این اندیس‌ها را یاد می‌گیرید.

سیگنال زیر با طول محدود را در نظر بگیرید:

$$x[n] = \begin{cases} 1, & 0 \leq n \leq 5, \\ 0, & \text{otherwise.} \end{cases}$$

ابتدا حاصل عبارت $y[n] = x[n] * x[n]$ را با تحلیل دستی حساب کنید.

به کمک کد زیر می‌توانید کانولوشن را حساب کرده و آن را رسم کنید. دقت کنید که باید تابع convIndices را پیاده سازی کنید.

```
clear; clc
nx = 0 : 5;
x = ones(size(nx));
ny = convIndices(nx, nx);   % you need to implement the convIndices function.
                            % In this example it will output a row vector with elements [0 .. 10]
y = conv(x, x);
stem(ny, y, 'lineWidth', 2)
% Graph labels
title('plot of signal $y[n]=x[n]*x[n]$', 'interpreter', 'latex', 'fontSize', 16)
xlabel('$n$', 'interpreter', 'latex')
ylabel('$y[n]$', 'interpreter', 'latex')
```

۲.۴ انجام دهید!

در این قسمت تابع convIndices را پیاده‌سازی می‌کنید.

برای بدست آوردن بردار ny دو سیگنال زیر را در نظر بگیرید:

$$h[n] = \delta[n-a] + \delta[n-b],$$
$$x[n] = \delta[n-c] + \delta[n-d].$$

با تحلیل دستی $y[n] = x[n] * x[n]$ را حساب کنید. سپس ny را بر حسب a، b، c و d تعیین کنید. حال می‌توانید تابع convIndices را بنویسید. این تابع اندیس زمانی ورودی‌های کانولوشن را ورودی می‌گیرد و اندیس زمانی مناسبی برای خروجی کانولوشن می‌دهد. در این مثال ورودی های این تابع دو بردار به صورت nh = a : b و nx = c : d هستند.

۳.۴ انجام دهید!

سیگنال ورودی $x[n]$ و پاسخ ضربه ضربه $h[n]$ به صورت زیر تعریف شده‌اند:

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

حال اگر بخواهید $y[n] = h[n] * x[n]$ را با دستور conv حساب کنید، باید ملاحظاتی برای طول بی‌نهایت دو سیگنال $x[n]$ و $h[n]$ بکنید.

مقادیر $x[n]$ در بازه‌ی $0 \leq n \leq 24$ را در بردار x و مقادیر $h[n]$ در بازه‌ی $0 \leq n \leq 14$ را در بردار h ذخیره کنید. حال حاصل کانولوشن این دو سیگنال را در بردار y ذخیره کنید. این در حالی است که شما فقط قسمتی از دو سیگنال $x[n]$ و $h[n]$ را در نظر گرفته‌اید. پس فقط بخشی از سیگنال خروجی دارای مقادیر درست است.

مقادیر a، b، c و d را به نحوی که nh = a : b و nx = c : d باشند، تعیین کنید و از جواب قسمت قبل برای بدست آوردن ny استفاده کنید. با دستور stem سیگنال $y[n]$ را رسم کنید و مشخص کنید چه بخشی از مقادیر آن با ارزش و چه بخشی بی‌ارزش است. (از برچسب‌های مناسب برای نمایش سیگنال خروجی استفاده کنید.)

۴.۴ انجام دهید!

تابع کانولوشن را خودتان پیاده سازی کنید و آن را myConv بنامید. سیستمی با پاسخ ضربه‌ی زیر فرض کنید:

$$h[n] = sinc(2\pi n)(u[n+4] - u[n-5])$$

خروجی این سیستم را یکبار با تابع کانولوشن متلب و یکبار با تابع خوتان برای ورودی زیر حساب کنید و صحت تابع خود را بررسی کنید.

$$x[n] = u[n] - n[n-2]$$

با دستور tic و toc مدت زمان انجام کانولوشن خودتان و کانولوشن متلب را بدست آورید و مقایسه کنید. علت اختلاف را شرح دهید.

۵.۴ انجام دهید (امتیازی)!

در این قسمت می‌خواهیم از روش کانولوشن بلوکی استفاده کنیم. این روش در پیاده سازی بی‌درنگ فیلترهای دیجیتال برای پردازش صوت/تصویر استفاده می‌شود.

در این روش سیگنال ورودی (که سیگنالی با طول بی‌نهایت/نامعلوم است) را به بلوک‌های کوچکتر تقسیم می‌کنیم. حال می‌توانیم هر کدام از این بلوک‌ها را به صورت مستقل پردازش کنیم البته با کمی تأخیر.

خطی بودن کانولوشن این تضمین را می‌دهد که برهم‌نهی [۳] خروجی‌های حاصل از پردازش بلوک‌ها با کانولوشن کل سیگنال با پاسخ ضربه یکسان است. وجود سخت افزار با کارایی مناسب و الگوریتم‌هایی برای محاسبه کانولوشن سیگنال‌هایی با طول

[۳] Superposition

محدود، بر اهمیت روش کانولوشن بلوکی می‌افزاید. در این قسمت هر کدام از کانولوشن‌های کوچک را با دستور conv حساب می‌کنید.

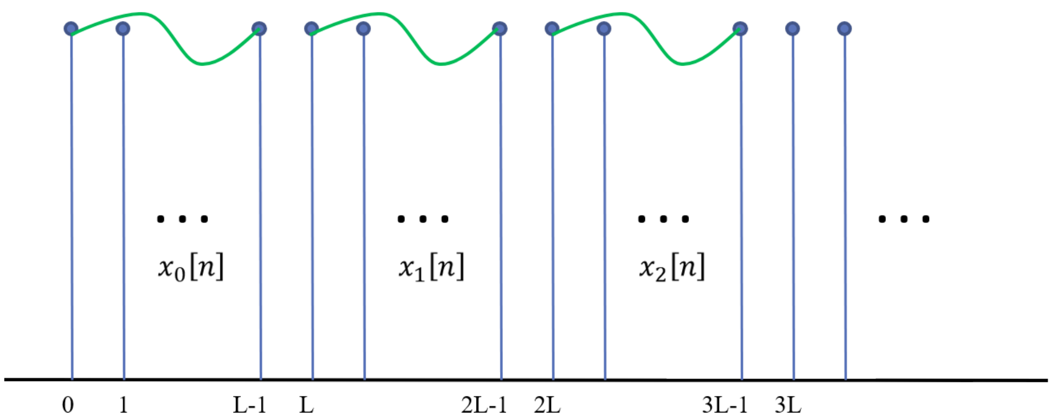
فرض کنید یک سیستم با پاسخ ضربه‌ی $h[n]$ دارید که فقط در بازه‌ی $0 \leq n \leq P-1$ غیر صفر است. همچنین فرض کنید دنباله‌ی ورودی یعنی $x[n]$ برای $n < 0$ صفر است و طول آن به طور قابل ملاحظه‌ای از P بیشتر است.حال می‌توانید به صورت زیر سیگنال $x[n]$ را به بلوک‌هایی با طول L تقسیم کنید:

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

که در آن $L > P$ و داریم:

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

شکل زیر را ببینید:



شکل ۳: تجزیه بلوکی سیگنال $x[n]$

ابتدا برای دو سیگنال زیر $y[n] = h[n] * x[n]$ را با دستور conv در بازه‌ی $0 \leq n \leq 99$ حساب کنید و نمودار آن را با stem در بازه‌ی داده شده رسم کنید.

$$x[n] = cos(n^2)sin(2\pi n/5),$$
$$h[n] = (0.9)^n(u[n] - u[n-10])$$

با فرض $L = 50$ سیگنال $x[n]$ را به دو بلوک تقسیم کنید که طول هر کدام ۵۰ شود. دو سیگنال $y_0[n] = h[n] * x_0[n]$ و $y_1[n] = h[n] * x_1[n]$ را که در آن $x_0[n]$ ۵۰ نمونه اول $x[n]$ و $x_1[n]$ ۵۰ نمونه دوم $x[n]$ 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

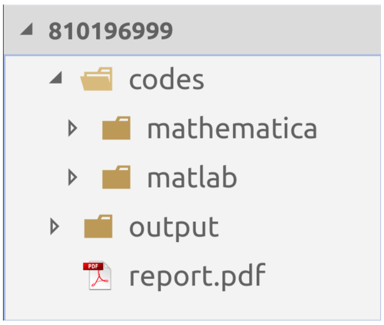
می‌شوند. به این خاطر به روش کانولوشن بلوکی، ”هم‌پوشانی و اضافه کردن“ نیز می‌گویند. سیگنال خروجی یعنی $y[n]$ را با استفاده از این روش حساب کنید و آن را در بازه $0 \leq n \leq 99$ با استفاده از stem رسم کنید. آیا به همان نتیجه قبلی می‌رسید؟ نتایج را تحلیل کنید.

در نهایت یک تابع بنویسید که عمل هم‌پوشانی و اضافه کردن را انجام دهد. ورودی‌های این تابع پاسخ ضربه (h)، بردار ورودی سیستم (x) و طول هر بلاک (L) ات. طول بردار x دلخواه و طول هر بلاک یک عدد دلخواه بزرگتر از طول فیلتر است. حال قسمت قبل را با تابع خود دوباره انجام دهید.

۵ نکات تحویل

فایل‌های خود را با نام A1-SID.zip در صفحه‌ی CECM درس بارگذاری کنید که SID شماره‌ی دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۶۹۹۹ باشد، نام پرونده‌ی شما باید A1-810196999.zip باشد.

- **تحویل حضوری** این تمرین در تاریخ ۲۳ و ۲۴ اسفند ماه خواهد بود.
- اگر اسم توابعی که باید بنویسید، مشخص شده است، حتما آن را رعایت کنید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.
- پیشنهاد می‌شود ساختار فایل‌ها را به صورت زیر رعایت کنید. در پوشه‌ی codes کدها و در output فایل‌های خروجی مانند تصاویر و فایل‌های صوتی (در تمرین‌های بعدی) را قرار دهید.



شکل ۴: ساختار فایل‌ها