سیگنال‌ها و سیستم‌ها

تمرین کامپیوتری شماره‌ی ۱

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

دانشگاه تهران

پردیس دانشکده‌های فنی

استاد: **امیرمسعود ربیعی**

طراح: **هدی برخوردارپور**، **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 نشان می‌دهد.
- 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.

---

[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.)

۲ کانولوشن گسسته‌ـ 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 حاصل‌جمع دنباله حاصل را بدست می‌آوریم. این تصویر از ویژگی خطی بودن و تغییر ناپذیری زمان سیستم های گسسته‌ـ زمان بدست می‌آید. در این قسمت استفاده از تابع conv (در پایتون numpy.convolve) را یاد می‌گیرید.

۱.۲ 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')
```

۲.۲ 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.

۳.۲ انجام دهید!

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

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

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

مقادیر $x[n]$ در بازه‌ی $0 \le n \le 24$ را در بردار x و مقادیر $h[n]$ در بازه‌ی $0 \le n \le 14$ را در بردار h ذخیره کنید. 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.)