



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 واژه MATrix (ماتریس) و LABoratory (آزمایشگاه) ایجاد شده است. 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 و هم 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 نرم‌افزار به خوبی انجام می‌شود.
- ساختن رابط کاربری برای نرم‌افزار در متمتیکا بسیار آسان‌تر از متلب است.
- مهمترین انتقادات از متلب به خاطر متن باز نبودن 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 پیوسته-زمان (به اختصار پیوسته) متناظر با هر نقطه‌ای از محور زمان یک مقداری دارند در حالی که سیگنال‌های گسسته-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 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 سیستم‌های گسسته-زمان بدست می‌آید. در این قسمت استفاده از تابع `conv` (در پایتون `convolve.numpy`) 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

```
1 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 اطلاعاتی 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.

```
1 clear; clc
2 nx = 0 : 5;
3 x = ones(size(nx));
4 ny = convIndices(nx, nx); % you need to implement the convIndices function.
5 % In this example it will output a row vector with elements [0 .. 10]
6 y = conv(x, x);
7 stem(ny, y, 'lineWidth', 2)
8 % Graph labels
9 title('plot of signal $y[n]=x[n]*x[n]$', 'interpreter', 'latex', 'fontSize', 16)
10 xlabel('$n$', 'interpreter', 'latex')
11 ylabel('$y[n]$', 'interpreter', 'latex')
```