مقدمه و توضیح الگوریتم

در این مینی پروژه قصد داریم با توجه به مفاهیمی که از فصل اول یاد گرفته‌ایم به حل یک دستگاه معادله بپردازیم. به طور کلی اگر بخواهیم مفاهیمی که یاد گرفته‌ایم را با یک مثال حل کنیم تا هم مطالب دوره شود و هم به گام‌هایی که بایستی در الگوریتم طی کنیم، توجه کنیم، می‌توانیم از مثال زیر استفاده کنیم:

$$\begin{cases} x_3 - 2x_4 = -3 \\ x_1 - 7x_2 + 6x_4 = 5 \\ -x_1 + 7x_2 - 4x_3 + 2x_4 = 7 \end{cases}$$

مثال فوق مثال ۱۲ فصل ۱.۲ کتاب درسی می‌باشد. اگر **Augmented Matrix** دستگاه فوق را در نظر بگیریم داریم:

$$\begin{pmatrix} 0 & 0 & 1 & -2 & -3 \\ 1 & -7 & 0 & 6 & 5 \\ -1 & 7 & -4 & 2 & 7 \end{pmatrix}$$

حال در ۴ گام می‌توانیم ماتریس فوق را به صورت **Echelon form** تبدیل کنیم (**forward phase**):

۱. تشخیص **pivot column**: از سمت چپ به دنبال ستونی می‌گردیم که حداقل یک درایه غیر صفر داشته باشد. این ستون یک **pivot column** می‌باشد که حاوی یک **leading entry** است.
۲. یک درایه غیر صفر از **pivot column** انتخاب می‌شود و در صورت نیاز سطر حاوی این درایه **interchange** می‌شود و با بالاترین سطر مورد نظر جا به جا می‌گردد. حال این درایه در **pivot position** قرار دارد.
۳. به کمک **row replacement** تمامی درایه‌های پایین **pivot position** برابر با صفر می‌گردد.
۴. از ستون (و ستون‌های قبل‌تر) و سطری که **pivot position** در آن قرار دارد چشم‌پوشی می‌گردد و مراحل فوق روی زیر ماتریس حاصل تکرار می‌گردند.

حال گام‌های مورد نیاز را مدام در مثال فوق تکرار می‌کنیم تا ماتریس ما به صورت **Echelon form** حاصل گردد:

(۱) ستون اول بدون شک **pivot column** است، زیرا که ۲ درایه ۱ و ۱- دارد که هردوی آن‌ها غیر صفر هستند:

$$\begin{pmatrix} 0 & 0 & 1 & -2 & -3 \\ 1 & -7 & 0 & 6 & 5 \\ -1 & 7 & -4 & 2 & 7 \end{pmatrix}$$

(۲) سطری که دارای درایه ۱ می‌باشد انتخاب و با سطر فوق **interchange** می‌شود تا درایه غیر صفر در بالاترین سطر باشد:

$$\begin{pmatrix} 0 & 0 & 1 & -2 & -3 \\ 1 & -7 & 0 & 6 & 5 \\ -1 & 7 & -4 & 2 & 7 \end{pmatrix} \sim \begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ -1 & 7 & -4 & 2 & 7 \end{pmatrix}$$

(۳) حال سطر آخر که در **pivot column** درایه غیر صفر دارد را صفر می‌کنیم. برای این کار به راحتی می‌توانیم سطر اول را با سطر آخر جمع کنیم (به طور کلی سطر مبدا ضربدر یک عدد می‌شود و با سطر مقصد جمع می‌گردد که در این مثال عددی که ضرب می‌شود برابر با ۱ است):

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ -1 & 7 & -4 & 2 & 7 \end{pmatrix} \sim \begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & -4 & 8 & 12 \end{pmatrix}$$

(۴) حال سطر و ستونی که حاوی **pivot position** می‌باشد؛ یعنی سطر اول و ستون اول را چشم پوشی می‌کنیم و مراحل را روی زیرماتریس حاصل تکرار می‌کنیم:

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & -4 & 8 & 12 \end{pmatrix}$$

(۱) در زیرماتریس بدست آمده به دنبال **pivot column** می‌گردیم. ستون دوم هیچ درایه غیر صفری ندارد ولی ستون سوم دارای دو درایه ۱ و ۴- می‌باشد. پس ستون سوم نیز **pivot column** و دارای یک **leading entry** است:

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & -4 & 8 & 12 \end{pmatrix}$$

(۲) درایه ۱ از سطر دوم و ستون سوم انتخاب می‌گردد. با توجه به این که این درایه در بالاترین سطر زیرماتریس است، نیازی به **interchange** نیست و این درایه در **pivot position** قرار دارد:

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & -4 & 8 & 12 \end{pmatrix}$$

(۳) حال در این **pivot column** انتخابی سطر آخر ۴- و غیر صفر می‌باشد. لذا با عمل **row replacement** سطر دوم را در ۴ ضرب و با سطر آخر جمع می‌کنیم تا در **pivot column** بدست آمده از زیرماتریس فرضی، تنها یک درایه غیر صفر موجود باشد.

(۴) حال اگر از سطر حاوی **pivot position** و ستون حاوی آن و (ستون قبل آن) چشم‌پوشی کنیم، تنها سطر آخر باقی می‌ماند که بیانگر پایان مساله است و انتظار داریم **Echelon form** حاصل شده باشد:

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

پس از آن که به کمک ۴ گام معرفی شده **echelon form** بدست آوردیم، به سراغ گام ۵ ام (**backward phase**) می‌رویم:

۵. از پایین سمت راست ترین **pivot position** شروع می‌کنیم. اگر ستون‌های بالای آن صفر نباشند، با **row replacement** آن ها را صفر می‌کنیم. سپس اگر درایه‌ای که در **pivot position** است غیر ۱ باشد، کل سطر را بر درایه تقسیم می‌کنیم تا درایه ۱ شود و سراغ **pivot position** سطرهای بالاتر می‌رویم.

در مثال فوق بر حسب اتفاق دو **pivot position** حاصل گردید که اولی در سطر اول و دومی در سطر دوم است که هردوی آن‌ها ۱ و درایه‌های بالای **pivot position** نیز برابر با صفر می‌باشند. لذا ماتریسی که در فوق بدست آورده‌ایم ماتریس به فرم **reduced echelon form** می‌باشد و نیازی به گام ۵ ندارد.

$$\begin{pmatrix} 1 & -7 & 0 & 6 & 5 \\ 0 & 0 & 1 & -2 & -3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

قوانین

۱. مینی پروژه به صورت انفرادی می‌باشد. تقلب‌ها توسط سامانه کوئرا چک می‌شود اما تصحیح توسط تیم تدریسیاری صورت می‌گیرد.
۲. در این پروژه تنها مجاز بر استفاده کتابخانه **NumPy** و توابع معرفی شده در ویدیو ۸ می‌باشید و استفاده از توابع دیگر و یا کتابخانه‌های دیگر تحت هیچ عنوان قابل قبول نمی‌باشد و هدف، یادگیری شما از عملیات سطری معرفی شده در درس برای حل سوال می‌باشد.
۳. با توجه به مجازی بودن درس و عدم امکان تحویل حضوری، انتظار می‌رود کد پیاده‌سازی شده از توابع مختلف تشکیل شده باشد که فهم مطلب را آسان‌تر کند. جود مستند برای توضیح کد به شدت استقبال می‌گردد.

پیاده سازی

در ابتدای برنامه بایستی از کاربر سطر و ستون **Coefficient matrix** (ماتریسی که تنها از ضرایب متغیرها بدست آمده است) از کاربر گرفته شود. نمونه‌ای از پیاده‌سازی آن را با توجه به اعداد مثال تعریف شده در فوق می‌توانید در تصویر زیر مشاهده کنید. لازم به ذکر است که ورودی لزومی ندارد حتما به فرمت زیر باشد و هر فرمتی که به نظر شما مناسب‌تر است را انتخاب کنید:

```
Coefficient matrix:
Enter number of rows and columns respectively:
> 3 4
Enter row 1:
> 0 0 1 -2
Enter row 2:
> 1 -7 0 6
Enter row 3:
> -1 7 -4 2
```

پس از گرفتن ورودی‌های **Coefficient matrix**، از کاربر اعداد ثابت سمت راست دستگاه معادله تحت عنوان یک ستون دریافت می‌گردد و با اضافه کردن این ستون به ستون آخر **Coefficient matrix**، انتظار داریم **Augmented matrix** حاصل و توسط برنامه چاپ گردد. نمونه پیاده‌سازی توضیح فوق:

```
Enter constant values:
> -3 5 7
Given matrix:
[[ 0.  0.  1. -2. -3.]
 [ 1. -7.  0.  6.  5.]
 [-1.  7. -4.  2.  7.]]
```

پس از گرفتن ورودی، برنامه باید با کمک الگوریتم معرفی شده در قسمت اول صورت مینی پروژه، به حل دستگاه معادله بپردازد.

با توجه به این که دستگاه می‌تواند بیش از یک جواب داشته باشد (مانند مثال تعریف شده)، انتظار می‌رود که مجموعه جواب به صورت **Parametric Description** توصیف و در خروجی نمایش داده شود. بنابراین برنامه شما بایستی قابلیت تشخیص **base variable** و **free variable** را داشته باشد تا بتواند جواب را به فرم مورد نظر نمایش دهد. نمونه خروجی از برنامه می تواند به فرمت زیر باشد:

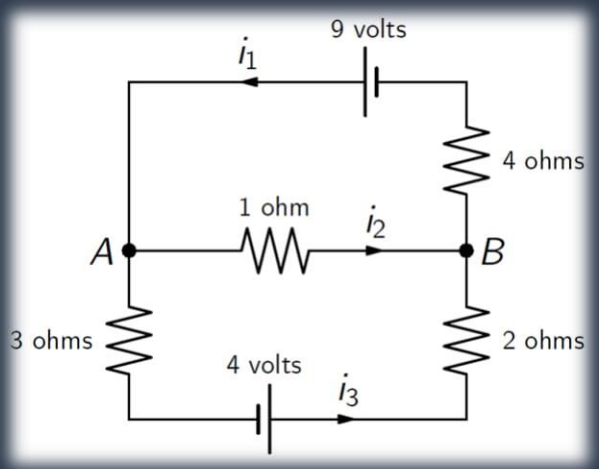
```
x1: 5.0+7.0x2-6.0x4
x2 is free
x3: -3.0+2.0x4
x4 is free
```

کاربردها (برای مطالعه)

حالا ببینیم از کدی که زدیم کجا ها میشه استفاده کرد .

هرچیزی که توی دنیای واقعی بتونیم به فرم دستگاه معادلات در بیاریم می‌تونیم با کدمون حلش کنیم.

ساده‌ترین مثال براش سوالای درس مدار الکتریکیه، مثلا برای مدار زیر ما به این معادله‌ها می‌رسیم:



Top loop: $9 - i_2 - 4i_1 = 0 \quad \Rightarrow -4i_1 - i_2 + 0i_3 = -9$

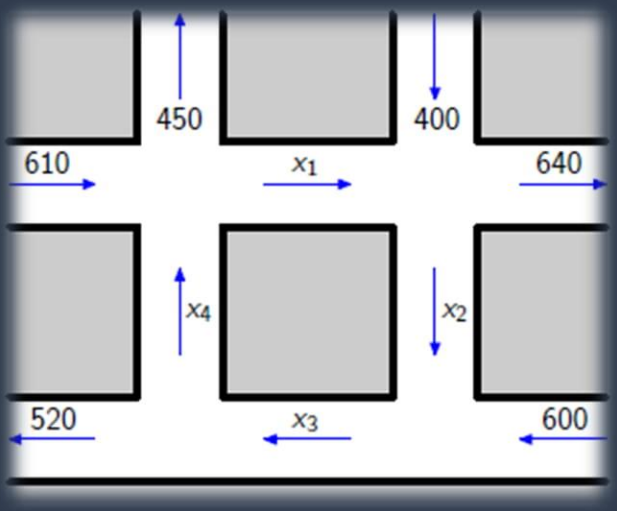
Bottom loop: $4 - 2\,i_3 + i_2 - 3i_3 = 0 \quad \Rightarrow 0\,i_1 + i_2 - 5i_3 = -4$

Big loop: $4 - 2\,i_3 - 4\,i_1 + 9 - 3\,i_3 = 0 \quad \Rightarrow -4i_1 - 0i_2 - 5\,i_3 = -13$

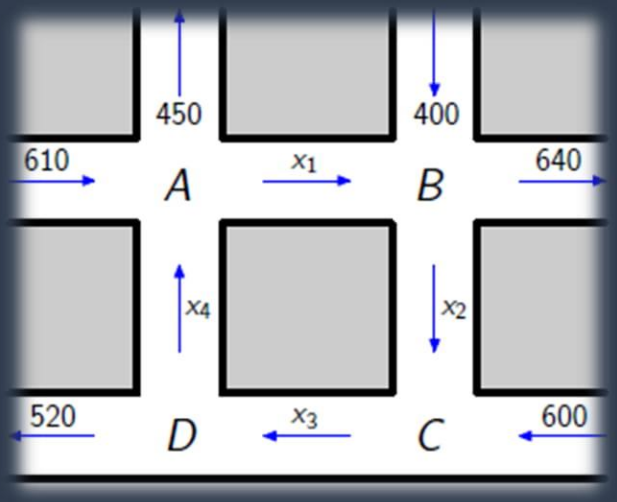
که اگر ماتریس افزونه رو تشکیل بدیم به ماتریس زیر می‌رسیم و می‌تونیم با کدمون مقدار هر کدوم از جریان‌ها رو بدست بیاریم.

$$\begin{pmatrix} -4 & -1 & 0 & -9 \\ -4 & 0 & -5 & -13 \\ 0 & 1 & -5 & -4 \end{pmatrix}$$

یکی دیگه از مثال‌های کاربردی، بدست آوردن میزان جریان ترافیک در مسیرهای درون شهریه . فرض کنید که ما میخوایم میزان جریان ترافیک رو در چهار مسیر x_1, x_2, x_3 و x_4 حساب کنیم.



میدونیم که تعداد ماشین‌های ورودی به هر تقاطع، برابر تعداد ماشین‌های خروجی از اون تقاطع هست. پس برای هر تقاطع یک معادله میشه نوشت .



$$\begin{cases} A \text{ تقاطع} : x_4 + 610 = x_1 + 450 \Rightarrow -x_1 + x_4 = -160 \\ B \text{ تقاطع} : x_1 + 400 = x_2 + 450 \Rightarrow x_1 - x_2 = 240 \\ C \text{ تقاطع} : x_2 + 600 = x_3 \Rightarrow x_2 - x_3 = -600 \\ D \text{ تقاطع} : x_3 = x_4 + 520 \Rightarrow x_3 - x_4 = 520 \end{cases}$$

که میتونیم ماتریس افزونه رو تشکیل بدیم و به ماتریس زیر برسیم و دوباره با استفاده از کدمون، جریان ترافیک رو تو هر کدام از مسیرهای خواسته شده حساب کنیم.

$$\begin{pmatrix} -1 & 0 & 0 & 1 & -160 \\ 1 & -1 & 0 & 0 & 240 \\ 0 & 1 & -1 & 0 & -600 \\ 0 & 0 & 1 & -1 & 520 \end{pmatrix}$$

موفق باشید

تیم تدریس یاری جبر خطی کاربردی

مهر ۹۹