Записки за упражненията по Числени методи на линейната алгебра 2022/2023 година

Тихомир Иванов

Съдържание

Увод

Навлизайки в тематиката "Числени методи на линейната алгебра", естествено е да попитаме с какво се занимава тя и какво е мястото ѝ сред другите дисциплини, които изучаваме; какви са приложенията ѝ на практика; защо е важно да се запознаем с методите, за които става въпрос, и кога би ни се наложило да ги използваме и т.н. Ние обаче ще отложим за кратко тази дискусия, за да можем първо сами да "пипнем" и да "усетим" част от основните задачи, с които курсът ще се занимава и тогава ще можем да дадем по-разбираем и по-пълноценен отговор включително и на формулираните по-горе въпроси.

Все пак, както името на курса показва, ясно е, че основните теми, с които предстои да се занимаваме, са свързани с:

- Линейна алгебра – основната задача на линейната алгебра, както тя е възникнала, е решаването на линейни алгебрични системи. Това е и основното, с което ще се занимаваме в настоящия курс, като ще разгледаме различни методи за решаването на линейни системи и ще коментираме какви са предимствата и недостатъците на всеки един от тях и кога се налага да ги използваме на практика. В края на курса ще отделим известно внимание и на въпроса за намирането на собствени стойности на матрица.

  Да припомним, че, ако имаме една примерна линейна алгебрична система:

  $$\begin{aligned} x + y + z &= 1, \\ x + 2y + 2z &= 2, \\ x + 2y + 3z &= 3, \end{aligned}$$

  естественото ѝ представяне в нотацията на линейната алгебра е векторно-матричната форма:

  $$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

  Следователно основното, с което ще се занимаваме, е да решаваме системи от вида

  $$A\mathbf{x} = \mathbf{b}.$$

- Числени методи – това са методи, които позволяват да решаваме математически задачи (най-често приближено) само с помощта на аритметични операции (събиране, изваждане, умножение, деление, коренуване). Можем да кажем тогава и че това са методи, които позволяват да решаваме математически задачи с помощта на компютър, тъй като именно

компютърът е добър в това да извършва много на брой пресмятания (а числените методи обикновено изискват огромен брой пресмятания, когато се използват на практика). Причината да имаме необходимостта от използването на такива методи е, че твърде често математическите задачи, които срещаме, не могат да бъдат решени с помощта на познатите ни аналитични техники (така например повечето интеграли не могат да бъдат пресметнати точно, повечето уравнения не могат да бъдат решени точно и т.н.). Тъй като, както ще видим, линейните алгебрични системи, възникващи на практика, могат да имат огромен брой уравнения (дори милиони, милиарди или трилиони), става ясна връзката между линейната алгебра и числените методи – трябва да видим как можем да използваме компютъра, за да решаваме линейни системи, след като е очевидно, че това в общия случай няма как да стане на ръка. Основна цел за нас, значи, ще бъде и да се научим да имплементираме програмно даден метод или алгоритъм, като ще използваме системата за компютърна алгебра Wolfram Mathematica.

Глава 1

Директни методи

1.1 Метод на Гаус

Класическият метод за решаването на линейни алгебрични системи, познат ни още от първи курс, е методът на Гаус. Ето защо естествено начало на настоящия курс е да си поставим за цел да реализираме именно този метод.

1.1.1 Някои предварителни сведения за Mathematica

Първо, ще припомним някои основни неща, свързани със синтаксиса и програмирането със системата Mathematica, които ще ни бъдат необходими за реализирането на метода на Гаус. Естествено, първо е необходимо да си припомним как се работи с матрици, тъй като те са основен обект, който ще трябва да използваме в курса.

Дефинираме матрица по следния начин:

```
In[1]:= A = {{1, 1, 1}, {1, 2, 2}, {1, 2, 3}}
```

Всеки от вложените списъци отговаря на един ред от матрицата, т.е. в променливата A сме запазили следната матрица:

```
In[2]:= MatrixForm[A]
Out[3]//MatrixForm=
```

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

Да обърнем внимание, че функцията *MatrixForm* се използва само за визуализиране на дадена матрица, но не можем да работим с матрица, която е в *MatrixForm*, т.е. не можем например да я умножим с число и пр.

Ако искаме да достъпим даден елемент от матрицата, например третия елемент на втория ред, това става по следния начин:

```
In[4]:= A[[2, 3]]

Out[4]= 2
```

Освен да изведем стойността му, разбира се, можем и например да запишем друго число в съответния елемент:

```
In[5]:= A[[2, 3]] = 5
```

Сега вече в паметта се пази следната матрица:

Out[6]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 5 \\ 1 & 2 & 3 \end{pmatrix}$$

Дотук припомнихме как можем да дефинираме и работим с конкретна матрица в Mathematica. За нашите цели обаче много често ще се налага да работим с произволни матрици или ние сами да пресмятаме матрицата по определен начин. Първото, което трябва да можем да направим тогава, е да заделим място в паметта, където да се пази резултатът от нашите изчисления. Можем да направим това по следния начин, например за да инициализираме една матрица 10×10:

```
In[8]:= B = Table[0, {10}, {10}];
        MatrixForm[B]
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

След като вече сме заделили място в паметта, можем да го използваме, за да правим определени изчисления, преобразувания и т.н., които да записваме на това място.

Пример 1. Нека например създадем една диагонална матрица, която има единици по главния диагонал.

```
B = Table[0, {10}, {10}];
For[i = 1, i ≤ Length[B], i++,
 B[[i, i]] = 1
]
```

Така, в B се пази следната матрица:

Out[11]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

За да я създадем, последователно:

- заделихме място в паметта с подходяща размерност;
- обходихме редовете на матрицата (т.е. направихме цикъл, който последователно минава през всеки от редовете) и в i-тия ред записахме 1 на главния диагонал.

Интересно

Думата *matrix* (матрица) идва от латинската дума *mater*, означаваща *майка*. Когато се прибави наставката $-ix$, думата означава *утроба*. Както утробата на майката пази в себе си едно бебе, така и матрицата съхранява в себе си дадени елементи/данни.

Интересно: Джеймс Силвестър

"*Mathematics is the music of reason*"

Джеймс Джоузеф Силвестър (1814-1897) играе водеща роля в американската математика през втората половина на XIX век. Негова заслуга е въвеждането на редица термини в математиката като матрица, граф, дискриминанта.
Една от страстите на Силвестър била поезията. Затова много от математическите му статии съдържат илюстративни цитати от класическата поезия.

Пример 2. Нека дадем още един пример – да създадем долно-триъгълна матрица 20×20 с единици по главния диагонал и под него. За целта трябва:

- да заделим подходящо място в паметта (както вече казахме, това обикновено е естествената ни първа стъпка – иначе няма да има къде да записваме резултата, който искаме да получим);
- да обходим редовете на матрицата, като в i-тия ред запишем единици в първите i елемента – след като искаме да извършим действието "записване на единица" неколкократно (по една единица във всеки от първите i елемента от съответния ред), значи ще трябва да използваме вложен цикъл, който да обхожда съответните елементи в ред i.

Тогава примерна реализация би изглеждала така:

```
In[12]:= B = Table[0, {20}, {20}];
         For[i = 1, i ≤ Length[B], i++, (*Обхождаме редовете*)
          For[j = 1, j ≤ i, j++, (*В ред i обхождаме първите i елемента,
           B[[i, j]] = 1
          ]
         ]
```

Така получихме матрицата

Out[18]//MatrixForm=

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

Както става видно от горните примери, обикновено, за да можем да направим нещо с дадена матрица, се налага да я обходим по подходящ начин.

Пример 3. Да дадем още един пример, свързан с обхождането на матрица. Нека изведем последователно всеки от редовете на матрицата от предходния пример. Очевидно отново трябва да обходим редовете и да извеждаме всеки от тях:

In[19]:=
```
For[i = 1, i ≤ Length[B], i++,
 Print[B[[i]]]
]
```

Получаваме следния изход от горния код:

```
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0}
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

Пример 4. Нека сега сложим още един "слой" върху горния пример. Да изведем първо първия ред на матрицата, после – първите два, след това – първите три и т.н.

След като искаме да повтаряме действието "извеждане на първите няколко реда", трябва да направим един цикъл за $k = \overline{1,n}$ и на k-тата стъпка от този цикъл да извеждаме първите k реда от матрицата, аналогично на горния пример (т.е. с цикъл, който обхожда първите k реда на матрицата):

```
In[20]:= For[k = 1, k ≤ Length[B], k++,
          For[i = 1, i ≤ k, i++,
           Print[B[[i]]]
          ]
         ]
```

Нека, преди да продължим по-нататък, да разгледаме един последен пример, свързан с прибавянето на един ред от матрицата към друг – операцията, която, както знаем, е в основата на метода на Гаус.

Пример 5. Нека прибавим 3-тия ред на матрицата B, умножен с 10, към 5-ия ред. За да съберем тези две неща, трябва да напишем

```
In[21]:= B[[3]] * 10 + B[[5]]
```

С това обаче все още не сме изпълнили задачата си, тъй като този код събира двата реда, но резултата не се запазва никъде. Винаги, когато извършваме някакви операции, отнасящи се до резултата, който искаме да получим, трябва да отразяваме това в паметта на компютъра. В случая трябва да променим петия ред на матрицата, с която работим:

```
In[22]:= B[[5]] = B[[3]] * 10 + B[[5]]
```

Сега вече в паметта на компютъра имаме следната матрица:

Out[23]//MatrixForm=

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 11 & 11 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

1.1.2 Имплементация на метода на Гаус

И така, вече сме готови да реализираме метода на Гаус. Както знаем, целта при него е да сведем матрицата на системата до горно-триъгълна форма, след което лесно можем да решим системата. Да го припомним с два примера, като за момента ще работим само с матрицата на системата, без да коментираме дясната страна.

$$\begin{bmatrix} \boxed{1} & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 \\ 0 & \boxed{1} & 1 \\ 0 & 1 & 2 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & \boxed{1} \end{bmatrix}$$

$$\begin{bmatrix} \boxed{1} & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 4 \\ 1 & 6 & 8 & 13 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \boxed{1} & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 5 & 7 & 12 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & \boxed{1} & 2 \\ 0 & 0 & 2 & 7 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & \boxed{3} \end{bmatrix}$$

Да припомним, че огражденият във всяка стъпка елемент се нарича **главен или водещ елемент.**

И така, за матрицата 3×3 методът на Гаус има 2 стъпки (съответстващи на двете стрелки; на първата стъпка занулява елементите в първия стълб, а на втората – елементите във втория стълб), а за матрицата 4×4 – 3 стъпки (съответстващи на трите стрелки). Разсъждавайки така, ясно е, че за матрица $n \times n$ методът трябва да направи $n - 1$ стъпки.

На всяка стъпка вземаме реда, в който е водещият елемент (на първата стъпка – първия ред, на втората – втория; изобщо, на k-тата стъпка – k-тия ред) и го прибавяме към всички редове по-надолу (т.е. прибавяме k-тия ред към всички редове от $k + 1$-вия до последния). Тогава общата структура на кода за реализирането на този метод би трябвало да има следния вид:

```
A = {{1, 1, 1, 1}, {1, 2, 2, 2}, {1, 2, 3, 4}, {1, 6, 8, 13}};
n = Length[A];
For[k = 1, k ≤ n - 1, k++,
 (*Прибавяме k-тия ред, умножен с
  подходящо число, към всички следващи
 *)
]
```

За да прибавим k-тия ред "към всички следващи", значи, на k-тата стъпка трябва да обходим редовете от $k + 1$ до последния (т.е. да напишем цикъл за $i = \overline{k+1, n}$) и да прибавим k-тия ред, умножен по $l = -a_{i,k}/a_{k,k}$, към i-тия ред:

```
A = {{1, 1, 1, 1}, {1, 2, 2, 2}, {1, 2, 3, 4}, {1, 6, 8, 13}};
n = Length[A];
For[k = 1, k ≤ n - 1, k++,
 For[i = k + 1, i ≤ n, i++,
  l = -A[[i, k]]/A[[k, k]];
  A[[i]] = A[[k]] * l + A[[i]]
 ]
]
```

Действително, лесно можем да се убедим, че изборът ни на числото l, с което умножаваме k-тия ред, за да го прибавим към i-тия, е правилен, ако

разгледаме k-тия и i-тия ред:

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \boxed{a_{kk}} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a_{ik} & \cdots \end{bmatrix}.$$

Разбира се, ако искаме да решим дадена линейна алгебрична система, то освен матрицата A, би трябвало да имаме и вектор с десните страни.

Пример 6. Да разгледаме системата

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= 1, \\ x_1 + 2x_2 + 2x_3 + 2x_4 &= 2, \\ x_1 + 2x_2 + 3x_3 + 4x_4 &= 3, \\ x_1 + 6x_2 + 8x_3 + 13x_4 &= 4, \end{aligned}$$

на която съответства матрицата, разгледана по-горе, и вектор на десните страни $\mathbf{b} = (1, 2, 3, 4)^T$. Сега вече трябва да работим с разширената матрица на системата:

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 4 & 3 \\ 1 & 6 & 8 & 13 & 4 \end{array}\right] \to \left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 2 \\ 0 & 5 & 7 & 12 & 3 \end{array}\right] \to \left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 7 & -2 \end{array}\right] \to \left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 3 & -4 \end{array}\right].$$

Това, което направихме дотук, е известно като **прав ход на метода на Гаус**. С други думи, правият ход на метода започва от оргиналната задача и я свежда до система с горно-триъгълна матрица, която вече лесно може да се реши (самото решаване е т.нар. обратен ход, на който ще се спрем след малко).

Както е видно от горния пример, трябва с вектора на десните страни да извършваме същите операции, както с редовете на матрицата A. Тогава окончателно получаваме следния код за правия ход на метода на Гаус:

Прав ход на метода на Гаус:

```
A = {{1, 1, 1, 1}, {1, 2, 2, 2}, {1, 2, 3, 4}, {1, 6, 8, 13}};
b = {1, 2, 3, 4}
n = Length[A];
For[k = 1, k ≤ n - 1, k++,
 For[i = k + 1, i ≤ n, i++,
  l = -A[[i, k]]/A[[k, k]];
  A[[i]] = A[[k]] * l + A[[i]];
  b[[i]] = b[[k]] * l + b[[i]]
 ]
]
```

Остава да реализираме и обратния ход, т.е. да решим системата, след като сме я привели в горно-триъгълен вид. Нека първо направим един пълен пример, в който да приложим метода на Гаус, за да решим една проста система:

Пример 7. Да се реши системата

$$\begin{array}{rcrcrcr} 4x_1 & - & 2x_2 & + & x_3 & = & 11 \\ -2x_1 & + & 4x_2 & - & 2x_3 & = & -16 \\ x_1 & - & 2x_2 & + & 4x_3 & = & 17 \end{array}$$

Решение. Разширената матрица на системата има вида

$$[A|\mathbf{b}] = \left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right].$$

- Прав ход на алгоритъма:

$$\left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right] \to \left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & -3 & -3/2 & -10.5 \\ 0 & -3/2 & 15/4 & 14.25 \end{array}\right] \to \left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -3/2 & -10.5 \\ 0 & 0 & 3 & 9 \end{array}\right].$$

  На първата стъпка прибавихме първия ред, съответно умножен с $1/2$ и -$1/4$, към втория и третия ред. На втората стъпка прибавихме втория ред, умножен с -$1/2$ към третия.

- Обратен ход на алгоритъма: След като сме свели системата до такава с горно-триъгълна матрица, можем ведната да намерим последното неизвестно от последното уравнение, което има вида

$$3x_3 = 9 \Rightarrow x_3 = 3.$$

Заместваме го в предпоследното уравнение, което добива вида

$$3x_2 - \frac{3}{2} \times 3 = -10.5$$

и от него намираме предпоследното неизвестно:

$$x_2 = \frac{-10.5 + 1.5 \times 3}{3} = -2$$

. Накрая заместваме в първото уравнение и получаваме

$$x_1 = \frac{11 - 1 \times 3 + 2 \times (-2)}{4} = 1.$$

□

И така, за да можем да имплементираме обратния ход на метода, трябва да напишем общия вид на формулата, по която пресмятаме неизвестните x_i, започвайки от последното към първото. За целта нека разгледаме едно произволно i-то уравнение след края на правия ход на метода. С други думи, разглеждаме системата, след като матрицата ѝ е приведена в горно-триъгълен вид:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,i-1} & a_{1i} & a_{1,i+1} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2,i-1} & a_{2i} & a_{2,i+1} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a_{ii} & a_{i,i+1} & \cdots & a_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}.$$

Следователно i-тото уравнение има вида

$$a_{ii}x_i + a_{i,i+1}x_{i+1} + \cdots + a_{in}x_n = b_i,$$

откъдето лесно можем да получим

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}},\ i = \overline{n, 1}.$$

Сега вече можем да реализираме обратния ход (приемаме, че сме стигнали до триъгълна матрица A и съответния ѝ вектор $\mathbf{b}$, т.е. сме изпълнили кода за правия ход, описан по-горе). Както вече стана ясно, е необходимо да направим две неща:

- Да заделим място в паметта, където да пазим резултата, т.е. ще създадем подходящ списък с Table;
- Да обходим този списък и да запишем стойностите на неизвестните, като започваме от последното към първото.

Получаваме следния примерен код:

Обратен ход на метода на Гаус
(след като е изпълнен кода за правия ход по-горе):

```
x = Table[0, n];
For[i = n, i ≥ 1, i--,
 x[[i]] = (b[[i]] - Sum[A[[i, j]] x[[j]], {j, i + 1, n}])/A[[i, i]]
]
x
```

1.1.3 Недостатъци на метода на Гаус

Методът на Гаус има един много сериозен недостатък. Нека разгледаме следната матрица.

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

На първата стъпка алгоритъмът ни казва от втория ред да извадим първия, умножен по a_{21}/a_{11}, но $a_{11} = 0$, което означава, че алгоритъмът не може да продължи работа. Изобщо казано, на k-тата стъпка имаме операцията деление на a_{kk}. Следователно **методът на Гаус е реализуем, само когато всички водещи елементи са различни от 0.**

По-сериозният проблем обаче се вижда от следния пример. Да разгледаме системата, чиято разширена матрица е

$$[A|\mathbf{b}] = \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 1 & 0 \end{array}\right].$$

След като прибавим първия ред, умножен по -10^{20} към втория, получаваме

$$[A^{(1)}|\mathbf{b}^{(1)}] = \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 0 & 1-10^{20} & -10^{20} \end{array}\right].$$

Тази система, разбира се, е еквивалентна на оригиналната. Ние обаче в общия случай извършваме пресмятанията с помощта на компютър. Малко по-късно в курса ще дадем повече информация за особеностите на компютърната аритметика, но нека тук споменем някои основни неща. Когато работим с числа с двойна точност например (както по подразбиране прави и Mathematica), ние имаме около 15 значещи цифри (т.е. можем да записваме числа от вида $\pm m \times 10^e$), където m е число между 0 и 1 с не повече от 15 цифри след десетичната запетая. За да запишем числото $1 - 10^{20}$ обаче ще са ни необходими 20 значещи цифри. С други думи, в паметта това число ще се закръгли към най-близкото число от посочения по-горе вид, т.е. -10^{20}. Тогава, вместо матрицата $[A^{(1)}|\mathbf{b}^{(1)}]$, в паметта ще се запази матрицата

$$float([A^{(1)}|\mathbf{b}^{(1)}]) = \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 0 & -10^{20} & -10^{20} \end{array}\right].$$

Единствената грешка от закръгляване е в елемента a_{22}, като тази грешка изглежда пренебрежимо малка. Да видим обаче до какво води тази изключително малка грешка. Лесно се вижда, че решението на системата с разширена матрица $float([A^{(1)}|\mathbf{b}^{(1)}])$ е $[0,1]^T$. Оригиналната система обаче има решение, което е приблизително равно на $[-1,1]^T$! Виждаме, че съвсем малка грешка от закръгляване при метода на Гаус може да доведе до съвършено различен резултат.

Методът на Гаус е неустойчив – малки грешки от закръгляване могат да доведат до много големи грешки в крайния резултат.

Важно е да се подчертае каква е причината за появилата се неустойчивост – тя се крие в много малкия по абсолютна стойност водещ елемент a_{11}. Делейки на него, получихме много голям по абсолютна стойност елемент a_{22}, което доведе до невъзможността да запишем полученото число в система с 16-цифрена мантиса. И така, да обобщим проблемите при метода на Гаус.

Важно

- Ако някой от водещите елементи е 0, алгоритъмът не може да продължи работа, т.е. не е реализуем;
- Ако някой от водещите елементи е малък по абсолютна стойност, това може да доведе до неустойчивост – малки грешки от закръгляване водят до големи грешки в резултата.

Да отбележим, че методът на Гаус е пример за **директен метод**.

Дефиниция 1: Директен метод

Директен метод е метод, при който оригиналната задача се свежда до еквивалентна на нея, която може да се реши непосредствено. Директните методи намират точното решение на задачата (по-точно грешки възникват само заради закръглявания).

Методът на Гаус е директен метод, тъй като той привежда оригиналната линейна алгебрична система до такава с горна триъгълна матрица **само с еквивалентни преобразувания** – умножение на двете страни на дадено уравнение с число и почленно събиране на две уравнения. Получената система може да се реши лесно чрез обратния ход на метода.

Интересно

Въпреки че разглежданият метод за решаване на линейни алгебрични системи се свързва с името на Гаус, той всъщност е бил известен още на древните китайски математици. Нещо повече, в Европа той бива популяризиран от Нютон, който през 1670 отбелязва, че във всички известни му учебници по алгебра липсва урок за решаване на линейни системи и привежда този метод. Така, във времето, когато Гаус живее и работи, този метод е вече стандартен във всички учебници по алгебра и наименованието му на Гаус (което става 50-те години на XX век) е вследствие на историческа грешка. Самият Гаус през 1810 разглежда метода за елиминация при симетрични матрици.

Интересно: сър Исак Нютон

"*If I have seen further than others, it is by standing upon the shoulders of giants.*"

Сър Исак Нютон (1642–1727) е считан за един от гигантите на физиката и математиката. Може би най-известните му достижения са откриването на законите за движение и гравитацията, диференциалното и интегрално смятане, но също така доказва например законите на оптиката и развива методи за решаването на полиномиални уравнения с произволна точност.
Роден е на Рождество Христово и след нещастно детство бива приет в Тринити Колидж, университета Кеймбридж, където изучава математика. По време на годините на чума (1665–1666), когато университетът е затворен, Нютон мисли и записва идеи, които след публикуването си незабавно революционизират науката.
Въпреки огромното влияние и оценка на достиженията на Нютон, включително удостояването с благородническа титла от кралица Анна през 1705, самият той е много по-скромен за резултатите си. Той казва "Изглежда, че бях само момче, играещо си на брега...докато великият океан на истината остава неоткрит пред мен".

1.2 Метод на Гаус с частичен избор на главния елемент

Методът на Гаус с частичен избор на главния елемент има за цел да реши двата проблема, които посочихме в края на предходния параграф. Тъй като причината за тези проблеми е в малка абсолютна стойност на водещия елемент, на k-тата стъпка за водещ елемент избираме най-големия по абсолютна стойност елемент от k-тия стълб на матрицата. За целта на практика разменяме k-тия ред и реда, съдържащ въпросния елемент. Това не променя нищо при решаването на системата, просто пренарежда уравненията.

1.2.1 Някои предварителни примери в системата Mathematica

Преди да пристъпим към разглеждането на метода на Гаус с частичен избор на главния елемент, нека отново започнем с няколко примера за работата в системата Mathematica, идеите от които ще използваме по-късно при реализирането на метода.

Пример 8. Нека имаме даден следния списък: {2,12,25,7,83,-15,7,6,5}. Ще покажем как можем да намерим индекса на най-големия елемент от него. За целта:

- Първо, трябва да си дефинираме променлива, нека я наречем maxIndex, в която да пазим резултата. Инициализираме я с 1, т.е. допускаме, че това

е индексът на най-големия елемент.

- Обхождаме всички следващи елементи и последователно ги сравняваме с най-големия досега намерен елемент (т.е. с индекс maxIndex). Ако текущият е по-голям, maxIndex вече е текущият индекс.

Прилагаме примерна имплементация

```
list = {2, 12, 25, 7, 83, -15, 7, 6, 5};
maxIndex = 1;
For[i = 2, i ≤ Length[list], i++,
 If[list[[i]] > list[[maxIndex]], maxIndex = i]
]
maxIndex
```

Резултатът от изпълнението на горния код е 5, тъй като най-големият елемент в посочения списък е петият.

Пример 9. Нека припомним и как става разменянето на стойностите на две променливи. Нека имаме например следните променливи:

```
a = 5; b = 3;
```

За да разменим техните стойности, трябва да използваме помощна променлива, нека я кръстим *temp*, в която "временно" да запазим стойността на a. След това в a записваме b, а в b записваме старата стойност на a, която е останала запазена в помощната променлива:

```
temp = a;
a = b;
b = temp;
```

В следващия пример ще генерираме една матрица, която ще използваме многократно по време на курса, за да сравняваме бързодействието на различните разглеждани от нас методи.

Пример 10. Нека построим 3-диагонална матрица с размерност $n \times n$, която изглежда по следния начин (n-входен параметър):

$$\begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}.$$

Подхождаме по познатия ни вече начин:

- Създаваме празна таблица с размерност n;
- Обхождаме редовете от втория до предпоследния (защото първия и последния са малко по-различни) и попълваме 1, -2, 1 на съответните позиции;

- Допълваме съответните стойности на първия и последния ред.

Прилагаме примерна имплементация:

```
n = 20;
A = Table[0, n, n];
For[i = 2, i ≤ n - 1, i++,
 A[[i, i]] = -2;
 A[[i, i - 1]] = A[[i, i + 1]] = 1
]
A[[1, 1]] = A[[n, n]] = -2;
A[[1, 2]] = A[[n, n - 1]] = 1;
```

1.2.2 Описание на метода на Гаус с частичен избор на главния елемент

Както казахме методът на Гаус с частичен избор на главния елемент е модификация на класическия метод на Гаус, при която преди да направим k-тата стъпка (т.е. да вземем ред k и да го прибавим, умножен с подходящи числа към долните):

- избираме най-големия по абсолютна стойност елемент в k-тия стълб (от ред k надолу) за главен елемент;
- разменяме ред k и реда с новия главен елемент.

Пример 11. И така, нека се върнем на примера от предишния раздел, за който методът на Гаус беше неустойчив:

$$[A|\mathbf{b}] = \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 1 & 0 \end{array}\right].$$

Прилагайки избор на главния елемент, първо ще разменим първия и втория ред (защото най-големият по абсолютна стойност елемент се намира в ред 2). Получаваме

$$[A|\mathbf{b}] = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 10^{-20} & 1 & 1 \end{array}\right] \to \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 1-10^{-20} & 1 \end{array}\right].$$

Дори закръглявайки числата в последната матрица, заради това, че работим с компютър, ще получим системата

$$[A|\mathbf{b}] = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 1 & 1 \end{array}\right],$$

която има решение $(-1, 1)^T$, което съответства на точното решение.

Нека разгледаме още един пример.

Пример 12. Да се направи правият ход на метода на Гаус с частичен избор на главния елемент за матрицата

$$\begin{bmatrix} 2 & 2 & 4 \\ 4 & 8 & 24 \\ 1 & 1 & 5 \end{bmatrix}.$$

Решение. На първата стъпка от алгоритъма избираме най-големия по абсолютна стойност елемент в пъргвия стълб (той е във втория ред) и разменяме първия и втория ред, след което продължаваме по стандартния начин, за да получим нули в първия стълб:

$$\begin{bmatrix} 2 & 2 & 4 \\ \boxed{4} & 8 & 24 \\ 1 & 1 & 5 \end{bmatrix} \to \begin{bmatrix} \boxed{4} & 8 & 24 \\ 2 & 2 & 4 \\ 1 & 1 & 5 \end{bmatrix} \to \begin{bmatrix} \boxed{4} & 8 & 24 \\ 0 & -2 & -8 \\ 0 & -1 & -1 \end{bmatrix}.$$

На втората стъпка избираме най-големия по абсолютна стойност елемент от втория стълб (под ред 2, разбира се) за главен. В случая той се намира на втория ред, т.е. не е необходимо да правим размени на редове. Получаваме

$$\begin{bmatrix} 4 & 8 & 24 \\ 0 & \boxed{\text{-2}} & -8 \\ 0 & -1 & -1 \end{bmatrix} \to \begin{bmatrix} 4 & 8 & 24 \\ 0 & \boxed{\text{-2}} & -8 \\ 0 & 0 & 3 \end{bmatrix}.$$

□

1.2.3 Имплементация на метода

Да разгледаме имплементацията на метода. Единственото, което трябва да променим спрямо класическия метод на Гаус е, че трябва на k-тата стъпка от алгоритъма първо да намерим най-големия по абсолютна стойност елемент в k-тия стълб (подобно на Пример 8) и да разменим ред k с реда, в който е този елемент (аналогично на това, което направихме в Пример 9).

Тогава една примерна имплементация добива следния вид, като с червено сме отбелязали допълнението спрямо кода за класическия метод на Гаус:

```
A = {{1, 1, 1, 1}, {1, 2, 2, 2}, {1, 2, 3, 4}, {1, 6, 8, 13}};
b = {1, 2, 3, 4};
n = Length[A];
For[k = 1, k ≤ n - 1, k++,
 maxIndex = k;
 For[i = k + 1, i ≤ n, i++,
  If[Abs[A[[i, k]]] > Abs[A[[maxIndex, k]]], maxIndex = k]
 ];
 temp = A[[k]];
 A[[k]] = A[[maxIndex]];
 A[[maxIndex]] = temp;
 temp = b[[k]];
 b[[k]] = b[[maxIndex]];
 b[[maxIndex]] = temp;
 For[i = k + 1, i ≤ n, i++,
  l = -A[[i, k]]/A[[k, k]];
  A[[i]] = A[[k]] * l + A[[i]];
  b[[i]] = b[[k]] * l + b[[i]]
 ]
]
x = Table[0, n];
For[i = n, i ≥ 1, i--,
 x[[i]] = (b[[i]] - Sum[A[[i, j]] x[[j]], {j, i + 1, n}])/A[[i, i]]
]
x
```

Могат да се намерят академични примери, за които методът на Гаус с частичен избор на главния елемент е неустойчив. Да разгледаме следния пример. Нека имаме матрицата

$$A = \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

По метода на Гаус с частичен избор на главния елемент (да отбележим, че в случая смени на редове не са необходими) се получават последователно матриците

$$A = \begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & -1 & 1 & & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & 0 & 1 & & 4 \\ 0 & 0 & -1 & 1 & 4 \\ 0 & 0 & -1 & -1 & 4 \end{bmatrix}$$
$$\longrightarrow \cdots \longrightarrow \begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & 0 & 1 & & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{bmatrix}.$$

Ако вземем аналогична на матрицата A, но с по-висока размерност, нека бъде $n \times n$, е ясно, че горната триъгълна матрица, която ще получим по метода на Гаус, ще съдържа елемента 2^{n-1}, което за достатъчно голямо n, аналогично на примера от предишния параграф, ще доведе до съществени грешки в резултата.

Въпреки широката практическа употреба на метода на Гаус с частичен избор на главния елемент обаче, не е известен пример, идващ от практиката, за който методът да е неустойчив. Могат да се направят и някои вероятностни съображения, които показват, че вероятността методът да е неустойчив за произволна зададена матрица е нищожна.

Важно

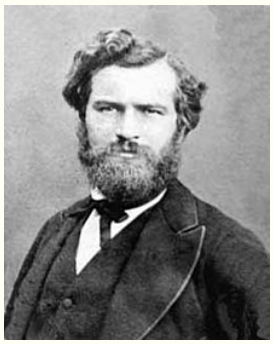
С други думи, **за всички практически цели методът на Гаус с частичен избор на главния елемент може да се разглежда като устойчив**.

Забележка. Тук няма да се спираме на метода на Гаус с (пълен) избор на главния елемент, тъй като неговото програмно реализиране е по-сложно, а подобряването на устойчивостта в сравнение с метода на Гаус с частичен избор на главния елемент е много малко. Затова на практика се използва най-вече методът на Гаус с частичен избор на главния елемент.

1.2.4 Метод на Гаус–Жордан

Разликата между метода на Гаус–Жордан и класическия метод на Гаус е, че на k-тата стъпка при метода на Гаус-Жордан k-тият ред се изважда от всички останали редове, а не само от редовете след k-тия. По този начин се получава диагонална матрица и системата може да се реши непосредствено. Методът на Гаус–Жордан също може да бъде приложен с частичен или пълен избор на главния елемент. Имплементацията оставяме за самостоятелна работа.

Интересно: Камий Жордан



Мари Енмон Камий Жордан (1838-1922) е френски математик, известен с основополагащата си работа в теория на групите и приносите си в областта на анализа и алгебрата. Резултати, носещи неговото име, са теоремата на Жордан в топологията (че всяка проста затворена крива в равнината я разделя на две части), Жордановата нормална форма на матрица и др.

1.3 Сложност на метода на Гаус

За да оценим сложността на метода на Гаус, трябва да преброим колко операции се извършват при решаването на дадена система. Първо, нека видим колко са операциите в правия ход на алгоритъма.

Алгоритъмът привежда оригиналната матрица до горна триъгълна за $n-1$ стъпки. Тогава броят операции е

$$N = \sum_{k=1}^{n-1} (\text{брой операции на } k\text{-тата стъпка}).$$

На k-тата стъпка вадим k-тия ред от всички следващи. Тоест операциите на k-тата стъпка ще получим, като съберем операциите при изваждането на k-тия от $k+1$-вия, $k+2$-ия и т.н.

$$N = \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (\text{брой операции при изваждането на } k\text{-тия ред от } i\text{-тия}).$$

Броят операции при изваждането на k-тия ред от i-тия е $2(n-k+1)$. Действително, всеки елемент от k-тия ред първо се умножава по l и после се прибавя към съответния елемент от i-тия ред, т.е. се извършват 2 операции. От друга страна имаме $(n-k+1)$ елемента – от a_{kk} до a_{kn}.

Окончателно получаваме

$$\begin{aligned} N &= \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (2n - 2k + 2) \\ &= \sum_{k=1}^{n-1} (2n - 2k + 2)(n - k) \\ &= \sum_{k=1}^{n-1} (2n^2 - 2nk - 2nk + 2k^2 + 2n - 2k). \end{aligned}$$

Оценявайки сложността на един алгоритъм, ние се интересуваме от членовете от най-висок ред. В случая това са оцветените в червено членове. За пресмятането на горната сума ще използваме известните формули

$$\sum_{i=1}^{n-1} i = \frac{n(n+1)}{2} = \frac{n^2}{2} + O(n),$$
$$\sum_{i=1}^{n-1} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + O(n^2).$$

Тогава

$$N = 2n^3 - 2n\frac{n^2}{2} - 2n\frac{n^2}{2} + 2\frac{n^3}{3} + O(n^2) = \frac{2n^3}{3} + O(n^2).$$

Важно

Окончателно получихме, че методът на Гаус има сложност $\frac{2}{3}n^3 + O(n^2)$. Това означава, че **методът на Гаус не е приложим за системи с много голяма размерност**. Ако искаме да решим система с 10^9 уравнения например, трябва да направим от порядъка на 10^{27} операции, което е твърде много дори за съвременните изчислителни машини. В тези случаи се използват итеративни методи, с които ще се запознаем по-късно в курса. **За системи с по-малка размерност обаче методът на Гаус (с частичен избор на главния елемент) е най-широко използваният метод**, тъй като със сигурност дава решението за практически всяка система (методът е точен, т.е. няма грешка от апроксимация, а само от закръгляванията при работа в компютърна аритметика).

Забележка. При извеждането на сложността разгледахме само правия ход на метода на Гаус. Лесно се вижда обаче, че обратният ход има сложност $O(n^2)$ и следователно не влияе върху оценката ни. Също лесно се вижда, че при метода на Гаус с частичен избор на главния елемент изборът на главен елемент на всяка стъпка и размяната на редовете добавя $O(n^2)$ операции, т.е. неговата сложност е също $2/3n^3 + O(n^2)$.

Интересно

В приведената по-долу таблица са дадени точният брой умножения/деления и събирания/изваждания, които правият ход на метода на Гаус прави за системи с 3, 10, 50, 100 уравнения.

n	Multiplications/Divisions	Additions/Subtractions
3	17	11
10	430	375
50	44,150	42,875
100	343,300	338,250

За да разберем по-добре какво означават резултатите, които получихме по-горе, нека проведем следния експеримент.

Пример 13. Нека сравним времената за изпълнение на метода на Гаус за системи с различна размерност. Ще решаваме задачата

$$\begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

последователно за $n = 50, n = 100, n = 200, n = 400, n = 800$. Матрицата на тази система показахме как може да се генерира в Пример 10. За засичане на времената използваме функцията TimeUsed[ ]. Скицираме как се прави експериментът за n=50:

```
n = 50;
A =. . .(*Тук генерирай матрицата A*)
b = Table[1., n];
start = TimeUsed[];
(*Тук сложи кода за метода на Гаус или го извикай като функция*)
end = TimeUsed[];
end - start
```

Систематизираме резултатите от нашия експеримент в таблицата по-долу, като времената са приведени в секунди:

n	Гаус (с част. избор) $2/3n^3 + O(n^2)$
50	0.14
100	1.078
200	8.422
400	66.29
800	613.594
1600	над 1 час

Струва си да направим няколко коментара, за да интерпретираме правилно данните, които получихме. Основният извод, разбира се, е, че ако увеличим размерността 2 пъти, то времето за изпълнение на метода се увеличава от порядъка на 8 пъти и следователно времето за решаване на системата много бързо нараства значително. Очевидно е, че ако продължим по същия начин, времената за решаване на милиарди уравнения ще станат безнадеждно големи. Трябва да отбележим, че тези времена са постигнати със системата Mathematica на стандартен персонален компютър. Ако използваме език за програмиране като

например C++ или Fortran или използваме високопроизводителни изчислителни системи (клъстери, суперкомпютри и т.н.), времената ще бъдат намалени в пъти. Въпреки това обаче, ако системата има милиарди или трилиони уравнения, това не би могло да помогне по никакъв начин.

Ето защо методът на Гаус (и изобщо като правило директните методи, както ще стане ясно по-нататък) е приложим за системи с малка размерност (което за нас ще остава стотици и хиляди уравнения, но не и милиони). **За системи с малка размерност методът на Гаус е стандартният общоприложим метод**, който обикновено се използва[1].

Интересно: Карл Фридрих Гаус

"*If others would but reflect on mathematical truths as deeply and as continuously as I have, they would make my discoveries.*"

Карл Фридрих Гаус (1777-1855) е считан от мнозина за най-великия математик в модерната история. Наричан е от своите съвременници "Принцът на математиката". Роден в бедно семейство, като малък Гаус открил грешка в счетоводните изчисления на баща си – едно от ранните събития в живота му, които засвидетелствали математическия му талант. Друга интересна случка разказва за учителя по математика на Гаус в началното училище, който поставил задача на целия клас да намерят сумата на първите 100 естествени числа. Надявал се, че това ще държи учениците ангажирани достатъчно дълго време. Гаус обаче изумил своя учител, като съобразил формула само за няколко минути.
На 22 години в своята докторска дисертация доказва Основната теорема на алгебрата – че всеки алгебричен полином от степен n има точно n комплексни нули. Неговите постижения се простират в практически всеки дял на математиката, както и във физиката и астрономията.

1.4 Case study 1: Моделиране на равновесното състояние на многокомпонентна система

В настоящата секция започваме една важна дискусия, а именно – къде на практика възникват линейни алгебрични системи. Това е необходимо, от една страна, за да започнем да разбираме по-задълбочено какво е приложението на разглежданата от нас математическа теория (и изобщо на математиката като език, който описва реални процеси). От друга страна, разбирайки какви са линейните алгебрични системи, възникващи на практика, ще можем по-добре да

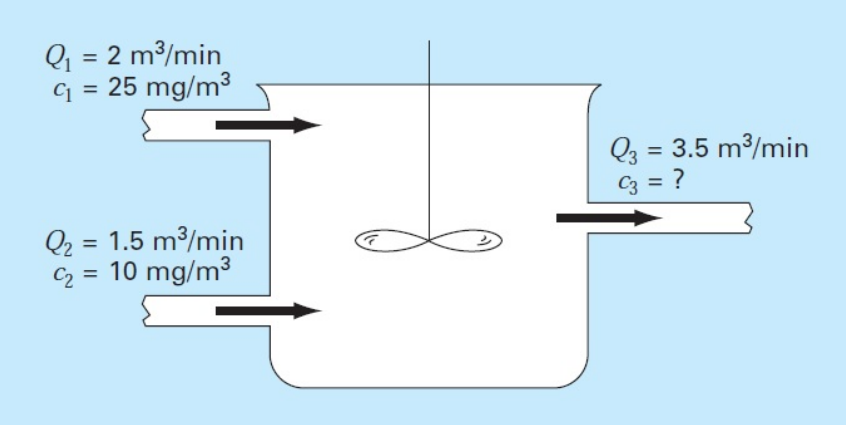
[1] Има ситуации, в които дори и за системи с малка размерност методът на Гаус е твърде бавен – например ако искаме в рамките на един алгоритъм, работещ в реално време, да се реши многократно дадена система, т.е. методът на Гаус да бъде изпълнен много пъти за много ограничено време

разберем какви трябва да са нашите изисквания към методите за решаването им.

Ще започнем с един контекст, в който възникват линейни алгебрични системи, а в аналогични секции по-нататък, ще разглеждаме и други принципни ситуации, които се срещат на практика.

Нека разгледаме следния реактор:

Пример 14. Равновесие на система от реактори.

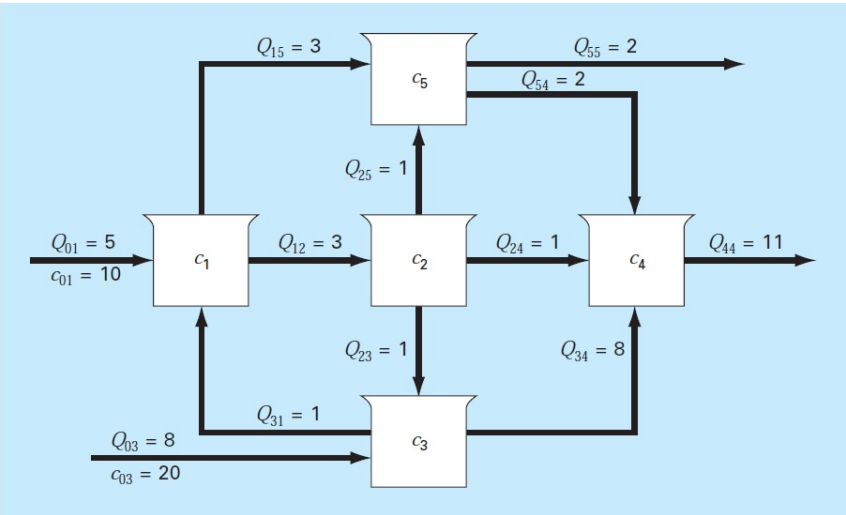


За да бъде той в стационарно състояние, е необходимо входът в реактора точно да се компенсира от изхода. С други думи, трябва да е изпълнено

$$Q_1c_1 + Q_2c_2 = Q_3c_3,$$
$$50 + 15 = 3.5c_3,$$

т.е. стационарната концентрация е решение на едно линейно алгебрично уравнение.

В действителност обаче рядко се срещат физически системи, при които има само една компонента, независима от други такива. Нека сега разгледаме една многокомпонентна система:



Ясно е, че **когато системата е многокомпонентна, ще получим система линейни алгебрични уравнения**, тъй като в този случай за всеки

един реактор в системата ще "отговаря" по едно уравнение Тя има вида (i-тото уравнение отговаря на баланса на масите в i-тия реактор, $i = 1, \ldots, 5$)

$$\begin{aligned}
Q_{01}c_{01} + Q_{31}c_3 &= (Q_{12} + Q_{15})c_1,\\
Q_{12}c_1 &= (Q_{25} + Q_{24} + Q_{23})c_2,\\
Q_{03}c_{03} + Q_{23}c_2 &= (Q_{31} + Q_{34})c_3,\\
Q_{24}c_2 + Q_{34}c_3 + Q_{54}c_5 &= Q_{44}c_4,\\
Q_{15}c_1 + Q_{25}c_2 &= Q_{54}c_5 + Q_{55}c_5.
\end{aligned}$$

Заместваики данните от фигурата, получаваме

$$\begin{aligned}
6c_1 - c_3 &= 50,\\
-3c_1 + 3c_2 &= 0,\\
-c_2 + 9c_3 &= 160,\\
-c_2 - 8c_3 + 11c_4 - 2c_5 &= 0,\\
-3c_1 - c_2 + 4c_5 &= 0.
\end{aligned}$$

Записана във векторно-матрична форма, системата има вида $A\mathbf{c} = \mathbf{b}$, където

$$A = \begin{bmatrix} 6 & 0 & -1 & 0 & 0\\ -3 & 3 & 0 & 0 & 0\\ 0 & -1 & 9 & 0 & 0\\ 0 & -1 & -8 & 11 & -2\\ -3 & -1 & 0 & 0 & 4 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_1\\ c_2\\ c_3\\ c_4\\ c_5 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 50\\ 0\\ 160\\ 0\\ 0 \end{bmatrix}.$$

Можем да приложим написания от нас код за метода на Гаус с избор на главния елемент (който казахме, че е стандартният метод за решаването на система, която няма голяма размерност) за решаването на тази система. Получаваме следните равновесни концентрации:

```
Out[2]= {11.5094, 11.5094, 19.0566, 16.9983, 11.5094}
```

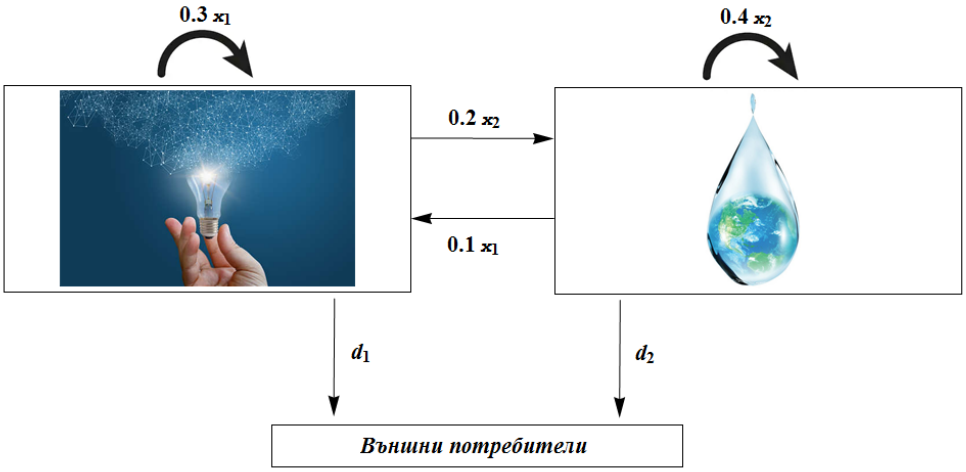
Разгледаният от нас пример, впрочем, илюстрира една принципна ситуация, която по никакъв начин не е ограничена само до конкретния пример, свързан с вход и изход в дадени реактори. Напротив, когато искаме да намерим баланса на една многокомпонентна система, независимо дали тя е инженерна, както в този случай, или има друг характер, е естествено да получим аналогична линейна алгебрична система. Ще разгледаме един известен пример за баланс, идващ от областта на икономиката.

Пример 15. Баланс на икономиката

Интересно

През 1973 г. Василий Леонтиев получава Нобелова награда за икономика. Един от неговите приноси е формулирането на математически модел на икономиката, който описва как си взаимодействат 500 сектора на американската икономика. Последният представлява система от линейни алгебрични уравнения. В следващия пример ние ще разгледаме много опростена версия на модела само за два отрасъла.

Разглеждаме икономика, която се състои от две индустрии – електрическа компания E и водна компания W. Производството на двете компании ще измерваме в долари. Електрическата компания използва електричество и вода, за да произвежда електроенергия, както и водната компания, за да добива вода. Нека за производството на електроенергия на стойност \$1 са необходими електроенергия на стойност \$0.30 и вода на стойност \$0.10. Нека за добива на вода на стойност \$1 са необходими електроенергия на стойност \$0.20 и вода на стойност \$0.40. Нека с x_1 и x_2 означим съответно произведените електроенергия и вода. Схематично казаното дотук е представено на долната графика, като с d_1 и d_2 са означени електроенергията и водата, търсени от крайните потребители на пазара (приемаме, че те са известни числа, установени от анализа на пазара):



Искаме да определим стойностите на x_1 и x_2 така, че пазарът да е в равновесие, т.е. да бъдат произвеждани точно толкова ресурси, колкото е необходимо (без да има излишък и недостиг). Гледайки горната схема, ясно е, че това е абсолютно същата принципна ситуация, като примера ни със системата от реактори.

Вземайки предвид горната схема, общо електроенергията, която трябва да се произведе, е $0.3x_1 + 0.2x_2 + d_1$, т.е., за да бъде пазарът в баланс, трябва да е изпълнено

$$x_1 = 0.3x_1 + 0.2x_2 + d_1.$$

Аналогично за добива на вода трябва да бъде изпълнено

$$x_2 = 0.1x_1 + 0.4x_2 + d_2.$$

Тъй като и в двете уравнения участват и двете неизвестни, трябва да ги решим

едновременно, т.е. да решим линейната алгебрична система

$$\begin{vmatrix} x_1 = 0.3x_1 + 0.2x_2 + d_1 \\ x_2 = 0.1x_1 + 0.4x_2 + d_2 \end{vmatrix}.$$

Разбира се, както стана въпрос по-горе, в реална задача е необходимо да се разгледат много повече отрасли, което води до решаването на система с много повече уравнения. Както споменахме по-горе, оригиналният модел на Леонтиев отразява 500 сектора на американската икономика, които биха довели до 500 уравнения в линейната алгебрична система.

Важно

Както виждаме от горните примери, често линейни алгебрични системи възникват, когато искаме да определим равновесното състояние на дадена реална (икономическа, физическа и пр.) система. Естествено е, разбира се, че когато търсим равновесно състояние, то трябва да бъде изпълнена система от равенства.
В случай, че условията, които трябва да се удовлетворят, са много (например ако има много отрасли в модела на Леонтиев или много реактори в предходния пример), ще се наложи да се решават **системи със стотици или хиляди уравнения** (т.е. с много на брой уравнения). Затова е важно да се изучат подходящи начини за описването и решаването на такива системи и да се научим да използваме възможностите на компютъра за съответните изчисления.

Както подчертахме, съвсем естествено е при разглеждането на такива физически системи да решаваме алгебрични системи със стотици или хиляди уравнения. Едва ли обаче можем да си представим, че в модела на Леонтиев например ще имаме милиони или милиарди уравнения. Такива системи обаче са не по-малко важни, но за тях ще говорим в следващи секции "Case Study".

Важно

За системи със стотици и хиляди уравнения (които за нас ще бъдат **системите с малка размерност**) класическият подход е методът на Гаус с частичен избор на главния елемент, който разгледахме дотук.