Записки за упражненията по Числени методи на линейната алгебра 2019/2020 година

Тихомир Иванов

Съдържание

Глава 1

Въведение в числените методи на линейната алгебра

1.1 Какво представляват числените методи?

Най-общо казано, числените методи са техники, чрез които математически задачи се представят във вид, в който могат да бъдат решени с помощта на аритметични операции. Въпреки че има много видове числени методи, те имат обща характеристика – изискват голям брой аритметични пресмятания. Ето защо тяхното прилагане става посредством имплементирането им в компютърни програми.

Обикновено числените методи включват **апроксимация** (т.е. приближение) на оригиналната математическа задача. Ето защо можем да ги разглеждаме като техники за **приближеното решаване** на дадена математическа задача посредством аритметични операции.

Преди да преминем към разглеждането на конкретни числени методи, нека разгледаме въпроса защо изобщо е необходимо тяхното използване. **Математиката е езикът, който се използва от науката и технологиите, за да може да се опише заобикалящият ни свят.** Всички дялове на науката се интересуват от това как се случват определени процеси. Физиците задават въпроси като “Как се движат обектите?”, “Как една енергия се трансформира в друга?”, “Какво представляват черните дупки?”. Химиците задават въпроси като “Как си взаимодействат веществата?”, биолозите – “Как се делят клетките?”, “Как функционира човешкото тяло?”. Икономистите задават въпроси за това как работи пазарът, как се движат паричните потоци и т.н. Ние искаме да отговорим на такива въпроси, за да разберем всички тези процеси и как можем да ги контролираме. Това ни позволява да правим съответните инженерни решения, като например да построим определени устройства, апарати, да получим нови химични вещества и материали и т.н.

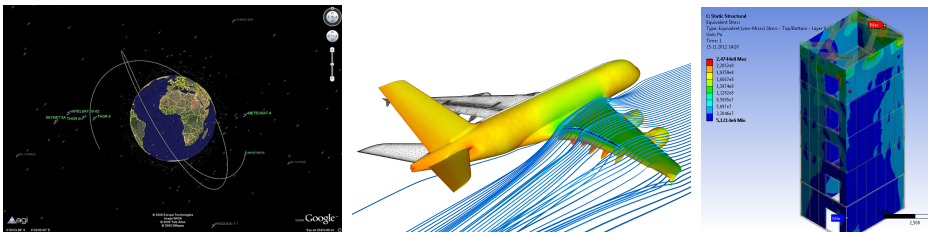
Има два начина да отговорим на такива въпроси – да изградим количествени и качествени теории. Качествени означава да опишем процесите с думи (на естествен език), например “Обектите се движат, защото на тях действат сили” и т.н. Такива отговори обаче често не са достатъчни, за да може да се опише как се случват такива процеси. Ето защо в тези случаи, науката гради количествена теория – описание на процесите на езика на математиката.

За да изучим даден реален процес или да решим дадена практическа за-

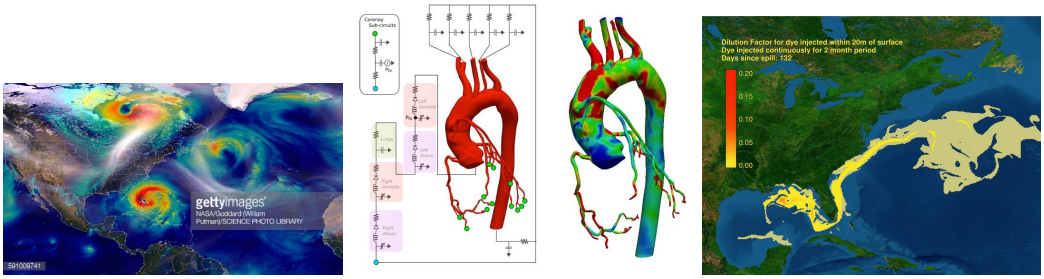
дача, ние трябва да определим кои са основните характеристики, които ги описват – това са някакви величини, които дават информация за съответния процес (скорост, сила, концентрация, температура, стойност, печалба, износ и др.). Величините се измерват в дадени мерни единици, т.е. им се съпоставят някакви числени стойности. Изучавайки даден процес, ние искаме да изучим зависимостите между величините, които го описват, като за целта създаваме и изследваме математически модел на процеса.

Най-общо казано, **математически модел** е описание на някакъв реален процес или реална задача на езика на математиката. Често това става чрез функция или уравнение (или система от уравнения), свързващо величините, описващи процеса. Математическият модел обаче може да представлява и друг математически обект.

Целта на математическото моделиране е да се опише даденият процес и по-добре да се разберат механизмите, които го обуславят, както и, евентуално, да се направят компютърни симулации и/или предвиждания за бъдещото му поведение.

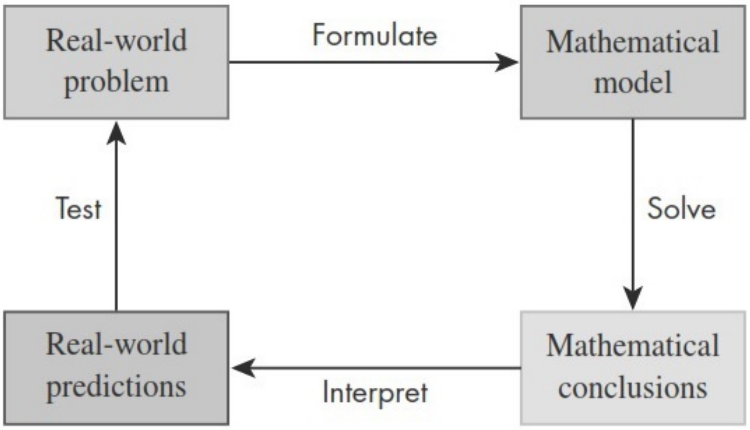


В реално време се изчисляват траекториите на сателитите. Самолетните инженери симулират обтичането на крилата на самолета от въздушния поток и напреженията, възникващи в неговата структура. Строителните инженери симулират поведението на дадена сграда при различни ситуации. От икономическа гледна точка, тези комютърни симулации позволяват пестенето на милиони, тъй като не е необходимо да се правят прототипи, чрез които да се тестват всички възможни поведения на системата.



Математически модели и компютърни симулации се правят още за предвиждания в областта на метеорологията, за описване на различни процеси в медицината (например потока на кръвта), за опазване на околната среда (например за симулация на развитието на даден петролен разлив) и т.н.

Често в литературата се дава следната схема, описваща методологията на математическото моделиране:



Да коментираме накратко етапите, описани в нея.

1. Имайки някаква реална задача, първото, което трябва да направим, е да **формулираме математически модел**, който да я описва. За целта трябва да определим основните величини, които характеризират процеса (от гледна точка на математиката – променливи и параметри), и да съставим математическата задача, която ги свързва (например диференциално уравнение, оптимизационна задача и др.). Важно е да се има предвид, че **всеки математически модел е една абстракция, идеализация на реалния процес**. В него трябва да има баланс – от една страна, моделът трябва достатъчно подробно да описва процеса, така че резултатите от него да бъдат полезни, но, от друга страна, трябва да е достатъчно прост, за да позволява математическо изследване. Всеки модел се базира на някакви допускания (абстракции), които позволяват опростяването на реалната ситуация. При създаването на математически модел използваме физически закони, обуславящи процеса, и математически техники, за да получим уравнения (или други обекти), свързващи променливите. В ситуации, когато не са известни физически закони, които да ни ръководят, може да е необходимо да се съберат данни от експерименти, на базата на които да се състави математическият модел.

2. Имайки предвид, че математическият модел на един процес представлява математическа задача, **вторият етап е да решим тази задача** и да получим математически заключения. **В настоящия курс ние ще разгледаме именно техники, които ще можем да използваме в този етап**. Важно е да се отбележи, че практическите задачи водят твърде често до математически задачи, които не могат да бъдат решени със стандартните аналитични техники. Както знаем, дори просто изглеждащи алгебрични уравнения като полиномиалните уравнения от пета и по-висока степен в общия случай не могат да бъдат решени точно. Същото се отнася за повечето определени интеграли и др. Въпреки това обаче съществуват техники за тяхното **приближено решаване** и именно с такива ще се запознаем в курса по Числени методи.

3. След като сме решили (в някакъв смисъл) математическата задача, **следва да интерпретираме резултатите от гледна точка на реалния**

процес.

4. Да обърнем внимание, че резултатите за реалния процес, които получихме, са следствие на математическия модел, а не на самия процес. От друга страна, казахме, че математическият модел е една абстракция на реалния процес, т.е. може и да не го описва достатъчно добре. Затова е необходимо да направим **проверка дали тези резултати съответстват на реалността**. Ако това е така, можем да считаме, че моделът ни е удачен. В противен случай се връщаме в началото и трябва да модифицираме модела така, че той да отразява действителността по-добре. С други думи, математическото моделиране е един **итеративен процес**.

Да формулираме няколко причини за изучаването на числени методи:

- Числените методи са много мощни средства за решаването на реални задачи. С тяхна помощ е възможно решаването на големи системи уравнения, справянето с нелинейности и сложни геометрии, които са присъщи за задачите от практиката и към които често е невъзможно да се подходи аналитично.

- Често в практиката се налага използването на готови софтуерни продукти, чието действие се базира на дадени числени методи. Интелигентното използване на тези продукти изисква познаването на основната теория, обуславяща съответните числени методи.

- Невинаги готовите софтуерни продукти са достатъчни за решаването на дадена практическа задача. В тези случаи познаването на основната теория в областта на числените методи ни позволява проектирането и направата на собствени програми.

1.2 Линейни алгебрични системи и задачи, при които възникват

Основна тема на настоящия курс ще бъде разглеждането на различни методи за решаването на линейни алгебрични системи – централната задача, около която се е развила линейната алгебра.

Почти невъзможно е да се занимаваме с числен анализ на даден реален процес, без да възникне необходимостта от решаването на линейни алгебрични системи.

Важно

Линейни алгебрични системи възникват:

- директно при разглеждането на стационарното състояние на многокомпонентни физически системи, които се описват с линейни зависимости между основните величини;
- индиректно, като стъпка в решаването на по-сложни задачи. Почесто необходимостта от решаването на линейни алгебрични системи възниква именно по този начин, например:
  - при числената решаване на диференциални уравнения – огромна част от математическите модели на реални процеси представляват диференциални уравнения, които не могат да бъдат решени по друг начин освен числено;
  - при решаването на нелинейни задачи – всъщност повечето процеси не се описват с линейни задачи и следователно не водят директно до решаването на линейни системи, а до нелинейни. Подходът за решаването на последните обаче обикновено е свързан с линеаризация (т.е. приближаване на нелинейната задача с линейна, например вж. метода на Нютон за решаване на нелинейни алгебрични уравнения);
- при редица математически задачи – метод на най-малките квадрати, интерполация, оптимизационни задачи и др.

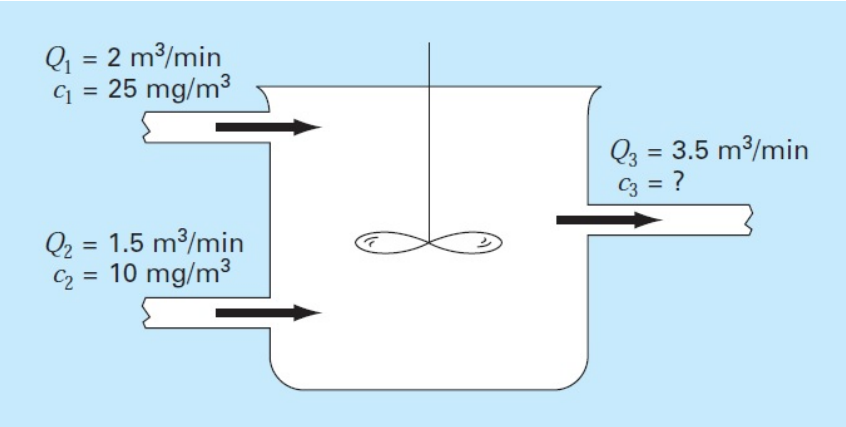
Нека разгледаме няколко примера във връзка с последния коментар.

1.2.1 Моделиране на стационарно състояние на дадена физическа система

Най-простият случай, който обаче възниква достатъчно често в практиката, е даден процес да се описва с линейна зависимост. Тогава, ако поискаме дадено условие за участващите величини да е изпълнено, е естествено това да доведе до линейно уравнение. Нека разгледаме няколко примера.

Пример 1. Анализ на стационарното състояние на система реактори

Нека разгледаме следния реактор:

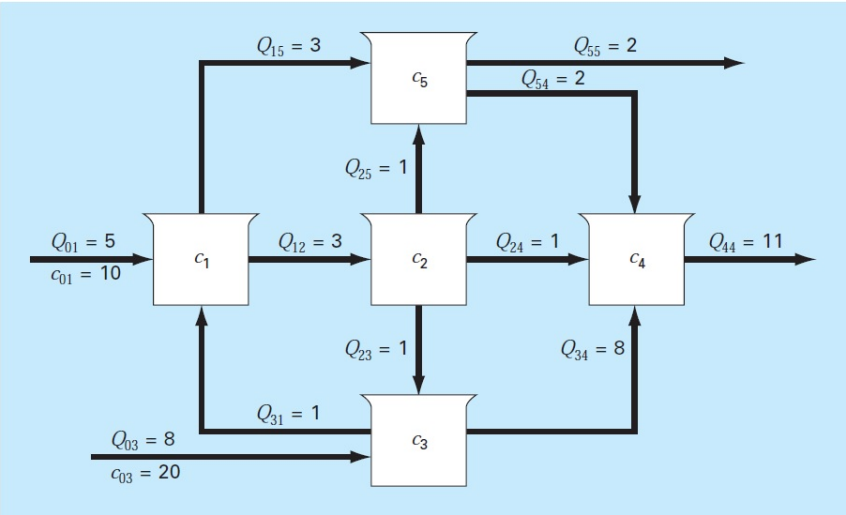


За да бъде той в стационарно състояние, е необходимо входът в реактора точно да се компенсира от изхода. С други думи, трябва да е изпълнено

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

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

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, $i = 1, \ldots, 5$)

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

Замествайки 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

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

Интересно

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

Интересно: Джеймс Силвестър

"*Mathematics is the music of reason*"

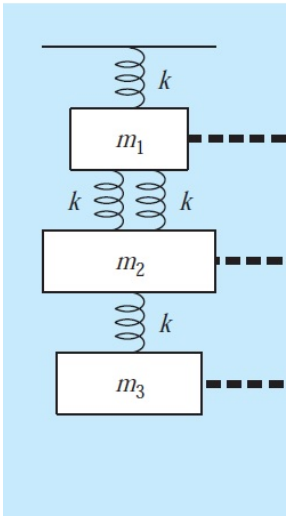
Джеймс Джоузеф Силвестър (1814-1897) играе водеща роля в американската математика през втората половина на XIX век. Негова заслуга е въвеждането на редица термини в математиката като матрица, граф, дискриминанта.

Една от страстите на Силвестър била поезията. Затова много от математическите му статии съдържат илюстративни цитати от класическата поезия.

Пример 2. Стационарно състояние на система от пружини и маси.

Системи от пружини и маси играят важна роля като идеализация в редица приложения в механиката, инженерното дело и развлекателната индустрия (например в компютърната анимация).

Нека разгледаме следната примерна система:



Нека телата имат маси съответно $m_1 = 2kg$, $m_2 = 3kg$, $m_3 = 2.5kg$, а еластичните сили в пружините се описват със закона на Хук $F = -kx$, където $k = 10kg/s^2$, а x е дължината на пружината. Нека в покой означим разстоянията на трите тела до основата, на която са закачени с x_1, x_2, x_3.

За да бъде системата в покой, върху всяко от телата силите, действащи във вертикално направление, трябва да се уравновесяват. За първото тяло имаме една пружина, действаща на тялото нагоре със сила $F_U = kx_1$. Надолу действат силата на тежестта $G = m_1 g$ и две пружини със сили $F_D = k(x_2 - x_1)$. Тогава за първото тяло трябва да е изпълнено

$$kx_1 = m_1 g + 2k(x_2 - x_1).$$

Аналогично, съставяйки уравнения и за другите две тела, получаваме системата

$$\begin{aligned} 3kx_1 - 2kx_2 &= m_1 g, \\ -2kx_1 + 3kx_2 - kx_3 &= m_2 g, \\ -kx_2 + kx_3 &= m_3 g. \end{aligned}$$

Замествайки в системата със съответните стойности на величините и записвайки системата във векторно-матрична форма, получаваме окончателно $A\mathbf{x} = \mathbf{b}$, където

$$A = \begin{bmatrix} 30 & -20 & \\ -20 & 30 & -10 \\ & -10 & 10 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 19.6 \\ 29.4 \\ 24.5 \end{bmatrix}.$$

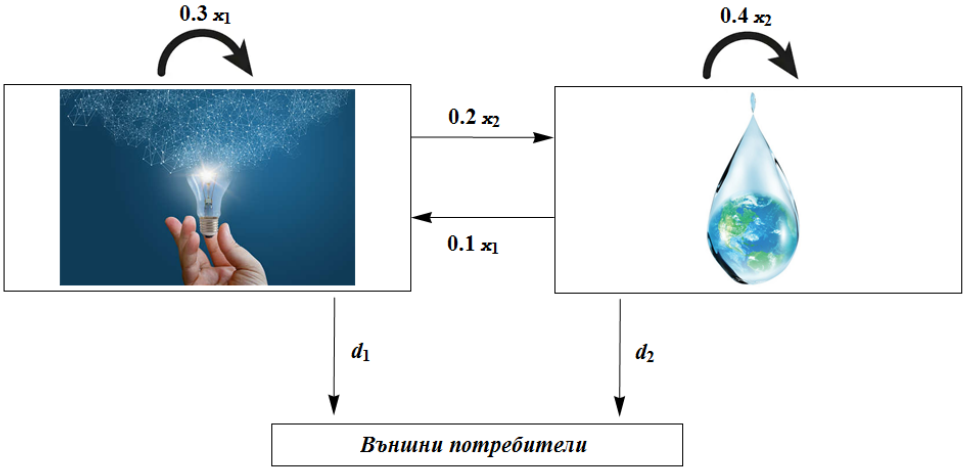
Системата има матрица, която е симетрична и положително определена. **Симетрията в матрицата е естествено отражение на симетрията във физическия процес** – еластичните сили, които действат на първото тяло надолу, действат на второто в обратна посока. Много процеси се характеризират с някаква симетрия, която по аналогичен начин води до симетрия в разглежданите математически задачи, в частност линейни алгебрични системи.

Пример 3. 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).

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

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

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 с много голяма размерност** (т.е. с много на брой уравнения). Затова е важно да се изучат подходящи начини за описването и решаването на такива системи.

1.2.2 Линейни алгебрични системи, възникващи при решаването на други математически задачи

Впрочем, въпреки че има много важни за практиката модели, които представляват линейни алгебрични системи, последните обикновено не възникват директно при моделирането на дадена физическа система, а индиректно, като стъпка при решаването на по-сложна задача.

Пример 4. Числено решаване на диференциални уравнения

Диференциалните уравнения са основният апарат на математическото моделиране. Ние ще се занимаем с този въпрос много по-подробно в курса "Числени методи за диференциални уравнения". Нека тук разгледаме само един пример,

Много физически процеси се описват с диференциални уравнения от втори ред. Да разгледаме следния пример:

$$\begin{aligned} -u''(x) &= \sin x, \quad x \in (0,1), \\ u(0) &= u(1) = 0. \end{aligned} \tag{1.1}$$

Обикновено възникващите в практиката диференциални уравнения са значително по-сложни и не могат да бъдат решени аналитично. Единият основен подход за числено им решаване е следният. Разделяме интервала на подинтервали, въвеждайки мрежата от точки (на разстояние h една от друга):

$$\omega_h := \{x_i = ih, i = \overline{0,n}, n = 1/h\}.$$

Тогава можем да апроксимираме диференциалното уравнение (1.1) в i-тата точка, като използваме следните формули за числено диференциране:

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} + O(h^2),$$
$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2).$$

Нека означим с y_i апроксимацията на решението в точката x_i. Тогава диференциалното уравнение се апроксимира с алгебричното уравнение

$$-\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = \sin x_i.$$

Написвайки по едно уравнение като горното за всяка точка, получаваме система от линейни уравнения, която очевидно има тридиагонална матрица (т.е. в i-тото уравнение участват само y_{i-1}, y_i, y_{i+1}.

Матрицата на тази система изглежда по следния начин:

$$\begin{bmatrix}
\ddots & \ddots & & & & & \\
\ddots & \ddots & \ddots & & & & \\
 & -1/h^2 & 2/h^2 & -1/h^2 & & & \\
 & & -1/h^2 & 2/h^2 & -1/h^2 & & \\
 & & & -1/h^2 & 2/h^2 & -1/h^2 & \\
 & & & & \ddots & \ddots & \ddots \\
 & & & & & \ddots & \ddots
\end{bmatrix}.$$

Единствено първото и последното уравнения (които отговарят на граничните условия) са по-различни, но не нарушават тридиагоналната структура и в момента няма да се спираме на този въпрос.

Да обърнем внимание още, че системата има толкова уравнения, колкото са възлите, с които сме разделили интервала. За да получим висока точност, може да се наложи интервала (или, изобщо, областта, в която се решава задачата и която може да бъде двумерна или тримерна) да се раздели на много подинтервали, т.е. **се налага решаването на система с много висока размерност**.

Пример 5. Метод на най-малките квадрати

Тук само ще отбележим, че методът на най-малките квадрати води също до решаването на една **линейна алгебрична система със симетрична матрица**. Ще се върнем много по-подробно на този въпрос по-късно в курса, когато ще отделим цяло упражнение на метода на най-малките квадрати.

Пример 6. Оптимизационни задачи

Необходимото условие дадена функция на много променливи да има екстремум в дадена точка е частните производни по отношение на всяка от променливите да са равни на 0. Това води до решаването на алгебрични системи.

Могат да се дадат, разбира се, още много примери. Като извод от горните примери е добре да обърнем внимание на следното.

Важно

1. Линейни алгебрични системи възникват при описването на стационарното състояние на някои физически процеси, както и индиректно при решаването на важни за практиката математически задачи;

2. Често линейните системи, които се налага да се решават, имат някаква специална структура. Например матрицата им е симетрична, тридиагонална и др. Ние ще разгледаме както методи, които са приложими за произволни системи, така и методи, които са специализирани в решаването именно на тези важни частни случаи;

3. Често системите, възникващи в практиката са с много голяма размерност. Например, ако при решаване на диференциалното уравнение, дискретизираме интервала $[0, 1]$ с въвеждането на много възли (което е естествено, ако искаме да получим добра точност), ще получим система с много уравнения.

1.3 Грешка. Източници на грешка. Представяне на числата в компютъра.

Както отбелязахме, повечето числени методи включват някаква апроксимация. Ето защо разбирането на идеята за грешка е от много голяма важност за ефективното им използване. Нека първо дадем следните дефиниции:

Дефиниция 1: Абсолютна грешка

Абсолютна грешка наричаме разликата между точната и приближената стойност при дадена апроксимация:

$$\varepsilon_a := exact\ value - approximation.$$

Дефиниция 2: Относителна грешка

Относителна грешка дефинираме по следния начин:

$$\varepsilon_r := \frac{exact\ value - approximation}{exact\ value} = \frac{\varepsilon_a}{exact\ value}.$$

Основните източници на грешка при решаването на една практическа задача са следните:

- Математическият модел – както казахме, математическият модел сам по себе си е една апроксимация на реалността, с други думи самото му съставяне въвежда грешка по отношение на реалния процес.

- Грешка от числения метод – обикновено числените методи се базират на някаква апроксимация, т.е. въвеждат някаква грешка. Тъй като ние на

практика не знаем точното решение на съответната математическа задача, обикновено е невъзможно да намерим каква е грешката при въпросната апроксимация. От друга страна, за да разберем дали даден числен метод е приложим, или не, ние трябва да знаем с каква точност той ще реши съответната задача. Затова се налага да се правят оценки на грешката, например да се намери някаква стойност, която тя със сигурност не надминава, или да се определи нейният порядък.

- Грешки от закръгляване – те са свързани с начина, по който числата се представят в компютъра. Ще се спрем по-подробно на този вид грешка в настоящия параграф.

- Грешки от входните данни – математическите модели обикновено зависят от някакви параметри, които се определят чрез провеждането на експерименти, правенето на измервания. Дори и най-съвършената техника позволява измерване с определена точност, т.е. стойностите на измерените величини, с които работим, също носят определена грешка.

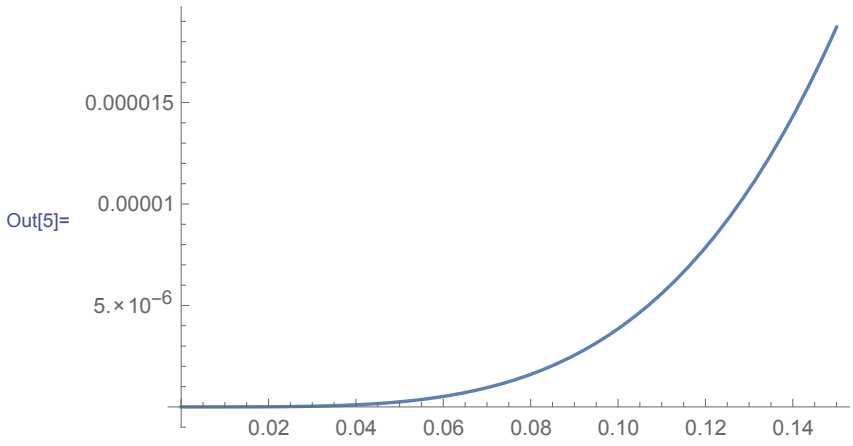
Задача 1. Да се намерят абсолютната и относителната грешка, които се получават, като приближим стойността на e^x с полинома на Тейлър от трета степен $1+x+x^2/2+x^3/6$ за $x = 0.1$. Да се построи графика на относителната грешка за $x \in [0, 0.15]$.

Решение. Ще използваме системата Wolfram Mathematica.

```
In[1]:= absErr[x_] := E^x - (1 + x + x^2 / 2 + x^3 / 6)
        relErr[x_] := absErr[x] / E^x
        absErr[0.1]
        relErr[0.1]
Out[3]= 4.25141 × 10^-6
Out[4]= 3.84683 × 10^-6
```

За графиката получаваме

```
In[5]:= Plot[relErr[x], {x, 0, 0.15}]
```



Грешката расте с нарастването на аргумента, което е естествено, имайки предвид, че експоненциалната функция расте значително по-бързо от полинома. Все пак в разглеждания интервал тя не надминава 0.002%. □

Сега ще се спрем на грешката от закръгляване. Причината за нея, както казахме, е начинът, по който числата се представят в компютъра. По-точно, ще се занимаем с т.нар. числа с плаваща точка (floating-point). При този подход числото се представя чрез дробна част, наречена **мантиса**, и цяло число, което се нарича **експонента** или характеристика по следния начин:

$$m.b^e,$$

където m е мантисата, b е основата на бройната система, в която работим (в компютъра $b = 2$), e – експонентата. Например $156.78 = 0.15678 \times 10^3$ е представянето във вид на число с плаваща точка на числото 156.78 в десетична бройна система. Да обърнем внимание, че обикновено дробната част се нормализира, така че първият знак след десетичната точка да бъде различен от нула.

Предимството на числата с плаваща точка е, че те позволяват представянето както на дроби, така и на много големи числа. От друга страна обаче, се появява т.нар. грешка от закръгляване, тъй като мантисата може да съдържа само краен брой значещи цифри. В компютъра, с t-битова дума могат да се представят най-много 2^t различни реални числа. Очевидно има безброй много числа, които не могат да бъдат представени точно. За тяхното представяне се използва най-близкото число, което може да се представи точно. По този начин въвеждаме грешка от закръгляване. Нещо повече, тъй като има максимално (по абсолютна стойност) число, то при опит да запишем число, което има по-голяма стойност, получаваме т.нар грешка "overflow". Освен това, по аналогична причина, не можем да представяме много малки по абсолютна стойност числа (т.е. близки до нулата). Опитът за записването на такова число води до грешка "underflow". Нека отбележим, че някои компютри заместват "underflow" с нула.

За да илюстрираме ефектите от грешките от закръгляване, нека разгледаме един хипотетичен компютър, който използва десетична бройна система и представя числата с плаваща точка чрез 1-цифрена експонента със знак и 3-цифрена мантиса.

Най-малкото положително число, което можем да представим в този компютър, е 0.100×10^{-9}, а следващото по големина число е 0.101×10^{-9}. Всяко друго число между тези две трябва да бъде апроксимирано. Това ни дава максимална грешка от закръгляване 0.5×10^{-12}.

Най-голямото число, което можем да представим, е 0.999×10^9, докато следващото по-малко число е 0.998×10^9, което дава максимална грешка 0.5×10^6.

Вижда се, че грешката съществено зависи от големината на числата, които апроксимираме. Затова е по-смислено да говорим за относителната вместо за абсолютната грешка. Може да се покаже, че тя е под 5×10^{-3}, т.е. под 0.5%. Да разгледаме следния пример, който ще ни покаже защо относителната грешка е по-добрия показател за точността на приближението. Ясно е, че абсолютна грешка, равна на 1, при число от порядъка на 10^8 е, по принцип, много по-пренебрежима, отколкото грешка от 0.001 при число от порядъка на 10^{-2}. Относителните грешки в този случай са съответно 10^{-8} и 0.1.

Изобщо, при работа в система числа с плаваща точка с p значещи цифри, може да се покаже, че за относителната грешка ε_r е в сила

$$|\varepsilon|_r \leq \mathbf{0.5 \times 10^{-p}} =: \boldsymbol{\varepsilon}.$$

При работа с числа с двойна точност (double), имаме $p \approx 16$, а с единична (float) – $p \approx 7$. Като резултат от грешките от закръгляване, дори фундаменталните асоциативни и дистрибутивни закони на алгебрата може и да не са в сила при числени пресмятания. Да разгледаме следните примери:

- Асоциативност на събирането

  $$a + (b + c) = (a + b) + c.$$

  Нека $a = 0.456 \times 10^{-2}$, $b = 0.123 \times 10^{0}$, $c = -0.128 \times 10^{0}$. Тогава

  $$(a + b) + c = 0.128 \times 10^{0} - 0.128 \times 10^{0} = 0,$$
  $$a + (b + c) = 0.456 \times 10^{-2} - 0.500 \times 10^{-2} = -0.440 \times 10^{-3}.$$

  Очевидно първият резултат не е верен и причината за това е **събирането на голямо с малко число**. Можем да разгледаме и още по-показателен пример за този проблем – ако съберем 0.100×10^{0} с 0.100×10^{-3}, резултатът е 0.100×10^{0}, т.е. все едно не сме извършили събирането!

- Асоциативност на умножението

  $$a \times (b \times c) = (a \times b) \times c.$$

  При стойности $a = 10^{-6}$, $b = 10^{-6}$, $c = 10^{8}$ лявата страна на асоциативния закон дава верен резултат. При използване на дясната страна обаче, при изчисленията ще се получи "underflow". Виждаме, че дори при работата с числа, които могат да бъдат представени точно, не сме застраховани от наличието на тази грешка. С други думи **за това как ще протече изпълнението на един алгоритъм сериозно влияние може да има редът, в който се изпълняват операциите**.

Горните примери ни показват, че всяка аритметична операция, която извършваме, би могла да въведе грешка. Вместо да работим с точната стойност на $a \odot b$, където $\odot$ е някоя от операциите събиране, изваждане, умножение, деление, работим с

$$fl(a \odot b) = (a \odot b)(1 + \delta)$$

и δ е съответната относителна грешка, която е ограничена от $|\delta| < \epsilon$. Както казахме, числените методи се базират на голям брой аритметични операции, така че това е нещо, което не можем да пренебрегнем при тяхното използване.

За да илюстрираме ефекта на грешките от закръгляване, нека разгледаме следния пример.

Задача 2. Даден е алгебричният полином

$$p(x) = (x-2)^9 = x^9 - 18x^8 + x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512.$$

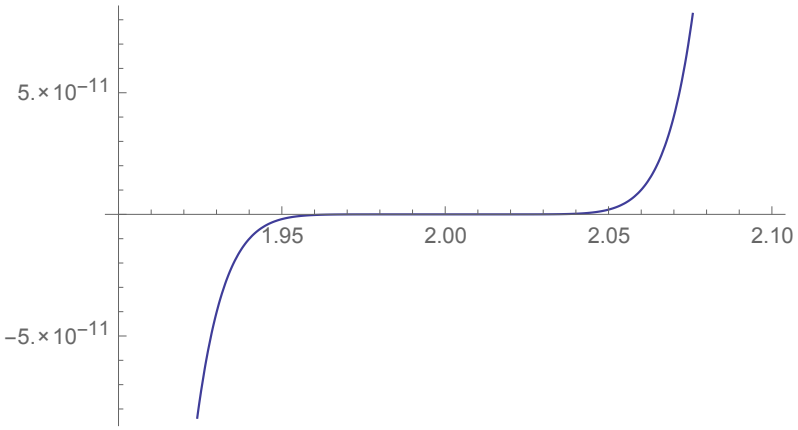
Да се построи неговата графика, като за пресмятане на стойностите му в точката x се използва

а)$p(x) = (x-2)^9$

б)$p(x) = x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$.

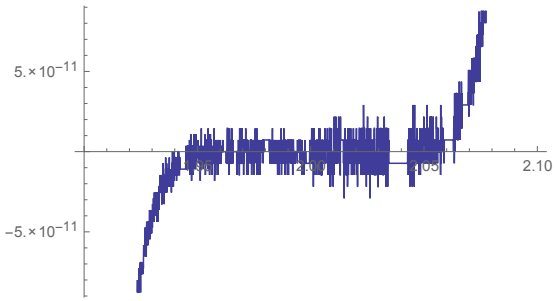
Решение. Решението на а) прилагаме по-долу.

```
f[x_]:=(x-2)^9
Plot[f[x],{x,1.9,2.1}]
```



За б) имаме.

```
p[x_]:=-512+2304 x-4608 x^2+5376 x^3-4032 x^4+2016 x^5-672 x^6+144 x^7-18 x^8+x^9
Plot[p[x],{x,1.9,2.1}]
```



Очевидно във втория случай резултатът е по-лош. Причината е в големия брой аритметични операции, които извършваме при него. Грешките от закръгляване водят до появилия се "шум". □

Вземайки предвид казаното дотук, **числените методи, които използваме трябва да са такива, че грешките от закръгляване да не водят до драстично изменение на резултата. Такива методи се наричат устойчиви.**

Глава 2

Директни методи за решаване на системи линейни алгебрични уравнения

Основна задача на числените методи на линейната алгебра е решаването на системи линейни уравнения

$$A\mathbf{x} = \mathbf{b}. \tag{2.1}$$

Съществуват както директни методи (ще разгледаме модификации на метода на Гаус), така и итерационни методи (тръгвайки от начално приближение $\mathbf{x}_0$, построяваме по някакво правило редицата от последователни приближения $\mathbf{x}_0$, $\mathbf{x}_1,\ldots,\mathbf{x}_n,\ldots$, която да клони към точното решение на системата (2.1)).

Важно

Когато изучаваме един числен метод, следва да се спрем на следните въпроси:

- описание и имплементация на алгоритъма;
- реализуемост – в кои случаи методът приключва успешно работа и връща резултат;
- устойчивост – какъв е ефектът на грешките от закръгляване; казваме, че един метод е неустойчив, ако малки грешки от входните данни (от закръгляване) водят до големи грешки в крайния резултат;
- "скорост" на алгоритъма – можем да разглеждаме този въпрос като брой операции и като скорост на сходимост при итерационните методи;
- оценка на грешката – ако нямаме оценка за точността на резултата, който получаваме, то той би бил абсолютно неизползваем.

2.1 Метод на Гаус и негови модификации

2.1.1 Метод на Гаус

Методът на Гаус е класическият метод за решаване на системи линейни уравнения. При него оригиналната система се свежда до такава с горна триъгълна матрица (прав ход на алгоритъма). Получената система вече може да бъде лесно решена, като от последното уравнение намерим последното неизвестно, след това от предпоследното уравнение – предпоследното неизвестно и т.н. Ще илюстрираме метода със следния пример.

Задача 3. 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:

  1. На първата стъпка ($i = 1$) към втория ред прибавяме първия, умножен по $1/2$, а към третия прибавяме първия, умножен по $-1/4$. Получаваме матрицата

  $$[A^{(1)}|\mathbf{b}^{(1)}] = \left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & -3 & -3/2 & -10.5 \\ 0 & -3/2 & 15/4 & 14.25 \end{array}\right].$$

  2. На втората стъпка ($i = 2$) към третия ред прибавяме втория, умножен по $1/2$. Получаваме окончателно горната триъгълна матрица

  $$[A^{(2)}|\mathbf{b}^{(2)}] = \left[\begin{array}{ccc|c} 4 & -2 & 1 & 11 \\ 0 & 3 & -3/2 & -10.5 \\ 0 & 0 & 3 & 9 \end{array}\right].$$

- Обратен ход на алгоритъма:

  1. $i = 3$: $x_3 = 3$.
  2. $i = 2$: $x_2 = \frac{-10.5+1.5\times 3}{3} = -2$.
  3. $i = 1$: $x_1 = \frac{11-1\times 3+2\times(-2)}{4} = 1$.

□

Да запишем алгоритъма в общ вид.

1. **Прав ход на алгоритъма** (получаваме горна триъгълна матрица):

   За $i = \overline{1, n-1}$ (на i-тата стъпка от алгоритъма правим всички елементи от i-тия стълб под диагоналния равни на 0)

   За $j = \overline{i+1, n}$ от j-тия ред вадим i-тия, умножен по $l_j^{(i)} = \frac{a_{ji}^{(i-1)}}{a_{ii}^{(i-1)}}$:

$$a_{jk}^{(i)} = a_{jk}^{(i-1)} - l_j^{(i)} a_{ik}^{(i-1)},\ k = \overline{i, n}, \tag{2.2}$$

$$b_j^{(i)} = b_j^{(i-1)} - l_j^{(i)} b_i^{(i-1)}. \tag{2.3}$$

2. **Обратен ход на алгоритъма** (намираме неизвестните):

   За $i = n, n-1, \ldots, 1$ намираме x_i по формулата

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}. \tag{2.4}$$

Прилагаме една примерна имплементация на метода на Гаус на Wolfram Mathematica.

In[14]:=
```
(*Input data*)
A = {{4, -2, 1}, {-2, 4, -2}, {1, -2, 4}};
b = {11, -16, 17};
n = Length[A];
(*Forward elimination*)
For[i = 1, i ≤ n, i++,
 For[j = i + 1, j ≤ n, j++,
  l = A[[j, i]] / A[[i, i]];
  A[[j, i]] = 0;
  For[k = i + 1, k ≤ n, k++,
   A[[j, k]] = A[[j, k]] - l * A[[i, k]];
  ];
  b[[j]] = b[[j]] - l * b[[i]];
 ]
]
(*Backward substitution*)
x = Table[0, {i, n}] ; (*We initilize a list,
in which we shall keep the values of the unknowns*)
For[i = n, i ≥ 1, i--,
 x[[i]] = (b[[i]] - Sum[A[[i, j]] x[[j]], {j, i + 1, n}]) / A[[i, i]]

]
x
```

Методът на Гаус обаче 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

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

На първата стъпка от алгоритъма получаваме матрицата

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

Нека работим в система от числа с плаваща точка с 16-цифрена мантиса. С други думи, можем да запазим в паметта 16 значещи цифри. 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, което може да бъде представено в разглежданата система от числа с плаваща точка. Нека приемем, че това е -10^{20}. Тогава, вместо матрицата $A^{(1)}$, в паметта ще се запази матрицата

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

Единствената грешка от закръгляване е в елемента a_{22}, като относителната грешка е от порядъка на 10^{-20}. Да видим обаче до какво води тази изключително малка грешка. Лесно се вижда, че решението на системата с разширена матрица $[\overline{A}^{(1)}|b^{(1)}]$ 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;
- 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 3: 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 преобразования** – 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".

2.1.2 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.

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:

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

Сега алгоритъмът продължава без проблеми:

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

Дори закръглявайки числата в последната матрица, заради работа с числа с плаваща точка, 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.

Задача 4. Да се направи правият ход на метода на Гаус с частичен избор на главния елемент за матрицата

$$\left[\begin{array}{ccc} 2 & 2 & 4 \\ 4 & 8 & 24 \\ 1 & 1 & 5 \end{array}\right].$$

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:

$$\left[\begin{array}{ccc} 2 & 2 & 4 \\ \boxed{4} & 8 & 24 \\ 1 & 1 & 5 \end{array}\right] \to \left[\begin{array}{ccc} \boxed{4} & 8 & 24 \\ 2 & 2 & 4 \\ 1 & 1 & 5 \end{array}\right] \to \left[\begin{array}{ccc} \boxed{4} & 8 & 24 \\ 0 & -2 & -8 \\ 0 & -1 & -1 \end{array}\right].$$

$$\left[\begin{array}{ccc} 4 & 8 & 24 \\ 0 & \boxed{\text{-2}} & -8 \\ 0 & -1 & -1 \end{array}\right] \to \left[\begin{array}{ccc} 4 & 8 & 24 \\ 0 & \boxed{\text{-2}} & -8 \\ 0 & 0 & 3 \end{array}\right].$$

□

Да разгледаме имплементацията на метода:

```
In[21]:= Gauss[AInput_, bInput_] := (
  A = AInput;
  b = bInput;
  n = Length[A];
  (*Forward elimination*)
  For[i = 1, i ≤ n, i++,
   (*Find the index of the row with maximal element*)
   maxIndex = i;
   For[j = i + 1, j ≤ n, j++,
    If[Abs[A[[j, i]]] > Abs[A[[maxIndex, i]]],
     maxIndex = j]
   ];
   (*Change the rows with indices i and maxIndex*)
   For[j = i, j ≤ n, j++,
    temp = A[[i, j]];
    A[[i, j]] = A[[maxIndex, j]];
    A[[maxIndex, j]] = temp;
   ];
   temp = b[[i]];
   b[[i]] = b[[maxIndex]];
   b[[maxIndex]] = temp;
   (*Eliminate the elements under the main diagonal in the i-th column*)
   For[j = i + 1, j ≤ n, j++,
    l = A[[j, i]] / A[[i, i]];
    A[[j, i]] = 0;
    For[k = i + 1, k ≤ n, k++,
     A[[j, k]] = A[[j, k]] - l * A[[i, k]];
    ];
    b[[j]] = b[[j]] - l * b[[i]];
   ]
  ];
  (*Backward substitution*)
  x = Table[0, {i, n}] ;
  (*We create a list, in which we shall keep the values of the unknowns*)
  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"**.

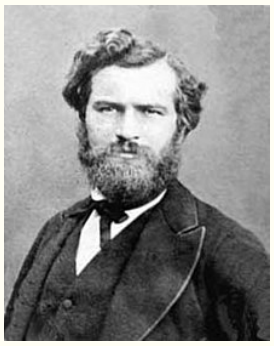
Забележка. Тук няма да се спираме на метода на Гаус с (пълен) избор на главния елемент, тъй като неговото програмно реализиране е по-сложно, а подобряването на устойчивостта в сравнение с метода на Гаус с частичен избор на главния елемент е много малко. Затова на практика се използва най-вече методът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

2.1.3 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.

2.1.4 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

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

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

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

Окончателно получаваме

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

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. 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

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.

2.2 LU декомпозиция

В редица случаи е удобно **дадена матрица да се разложи на произведение от матрици** по определен начин. Например диагонализирането на една матрица, т.е. представянето ѝ във вида

$$A = T\Lambda T^{-1},$$

където Λ е диагоналната матрица от собствените стойности на матрицата A, а стълбовете на T са съответстващите им собствени вектори ни дава удобен начин за повдигането на матрица на дадена степен. Имаме

$$A^n = \underbrace{(T\Lambda\cancel{T^{-1}})(\cancel{T}\Lambda\cancel{T^{-1}})(\cancel{T}\Lambda\cancel{T^{-1}})\dots(\cancel{T}\Lambda T^{-1})}_{n \text{ пъти}} = T\Lambda^n T^{-1}.$$

Последното може да бъде лесно пресметнато, тъй като, за да повдигнем диагоналната матрица Λ на n-та степен трябва просто да повдигнем диагоналните елементи на съответната степен и не е необходимо да се правят n на брой матрични умножения, какъвто щеше да бъде случаят, ако директно бяхме повдигнали A^n.

Сега ще се спрем на една друга важна декомпозиция и ще коментираме случаите, когато тя е полезна. Ясно е, че ако една линейна система има триъгълна матрица, то нейното решаване е непосредствено – от всяко уравнение намираме едно неизвестно. Тогава, ако можем дадена матрица A да представим във вида

$$A = LU,$$

където L и U са съответно долна триъгълна и горна триъгълна матрици, то решаването на системата $Ax = b$ е еквивалентно на решаването на системата $LUx = b$ и се свежда до последователното решаване на системите $Ly = b$ и $Ux = y$, които имат триъгълни матрици.

2.2.1 Предварителни сведения от линейната алгебра

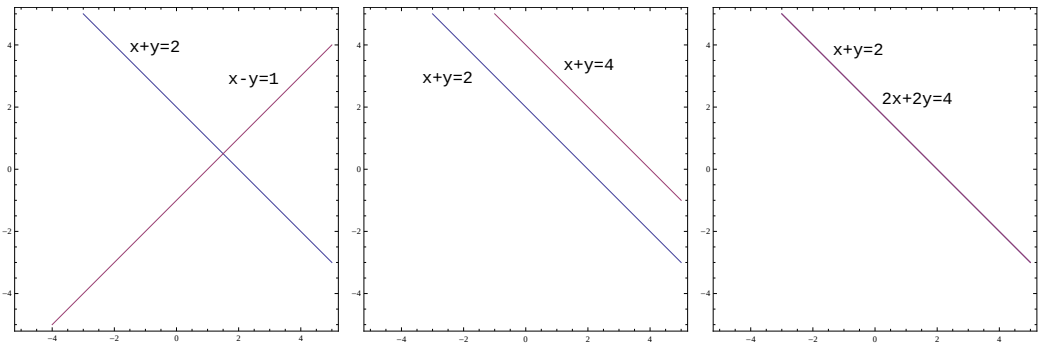
Преди да покажем как може да се намери исканото разлагане на матрицата A, нека припомним някои факти от линейната алгебра. Впрочем не всичко, приведено тук, ще бъде пряко свързано с въпроса за намиране на LU-разлагане, но ще се възползваме от случая да припомним някои важни факти.

Често е полезно да имаме геометрична интерпретация на даден аналитичен обект. Ще разгледаме две геометрични представяния на дадена линейна система. Първо, ще коментираме "row picture" за линейна система

В 2D можем да визуализираме всяко от уравненията на линейната система (т.е. всеки ред на системата), като изобразим всички точки, чиито координати удовлетворяват уравненията в декартова координатна система. Както знаем, те отговарят на прави в равнината.

Пример 7. Съществуват точно три възможни взаимни положения на две прави в равнината. Да разгледаме следните примери:

$$\begin{aligned} x + y &= 2 \\ x - y &= 1 \end{aligned} \qquad \begin{aligned} x + y &= 2 \\ x + y &= 4 \end{aligned} \qquad \begin{aligned} x + y &= 2 \\ 2x + 2y &= 4 \end{aligned}$$



В първия случай двете прави се пресичат в точка. Координатите ѝ съответстват на единственото решение на системата.

На втората фигура правите са успоредни и следователно системата няма решение.

В третия случай двете прави съвпадат, т.е. системата има безброй много решения (по-точно права от решения).

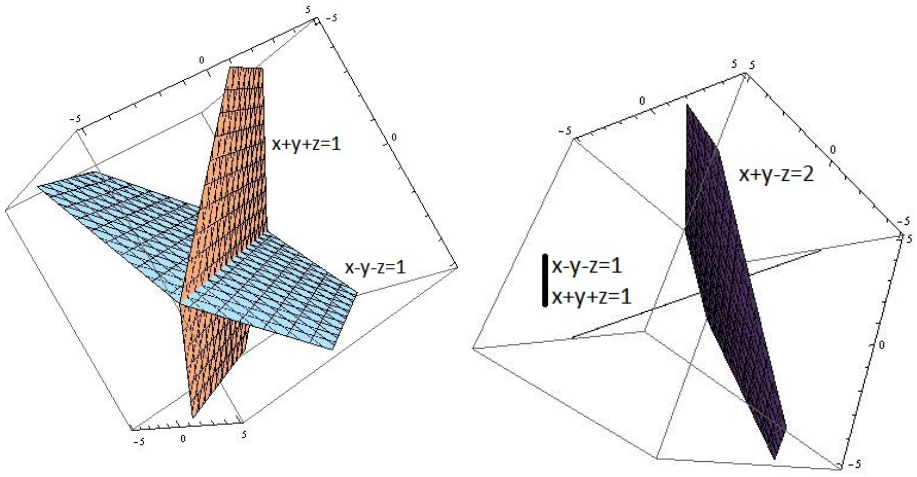
Преминавайки към 3D, row picture за всяко уравнение е равнина в тримерното пространство.

"Хубавият" случай е, когато системата има единствено решение. Това се получава, когато първите две равнини се пресичат в права, която пресича третата равнина в точка.

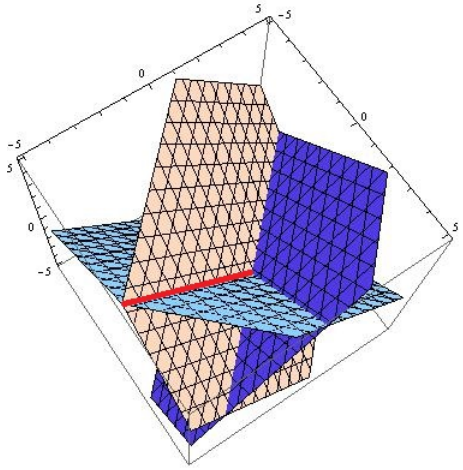
Пример 8. Разглеждаме системата

$$x + y + z = 1$$
$$x - y - z = 1$$
$$x + y - z = 2$$

На лявата фигура по-долу са изобразени първите две равнини, които се пресичат в права. На дясната фигура е показана тази права, която пресича третата равнина в точка.



Пълният row picture, състоящ се от трите равнини е илюстриран по-долу. Пресечницата на първите две равнини е означена в червено.



Задача 5. Опишете всички възможности за това как три равнини са разположени в пространството.

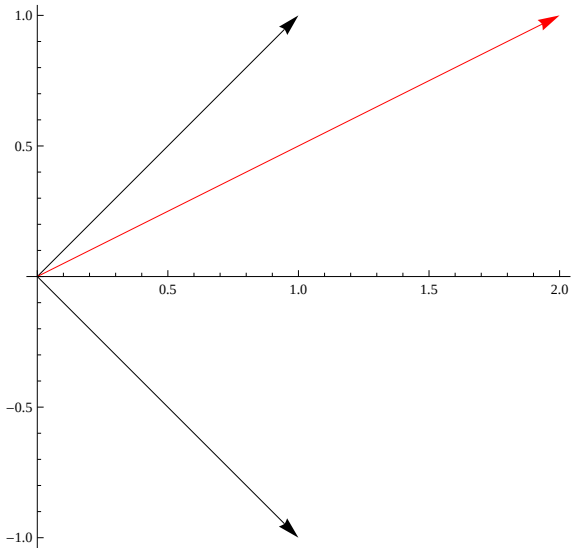
Както виждаме, row picture е относително трудно да се визуализира в 3D, но отново дава ценна информация за геометрията на линейните системи. В повече измерения обаче тази илюстрация не може да помогне.

Оказва се, че в много случаи column picture създава по-добра интуиция. Нека първо разгледаме отново системите от Пример 7.

Пример 9. Работейки стълб по стълб, можем да запишем трите системи от Пример 7, както следва:

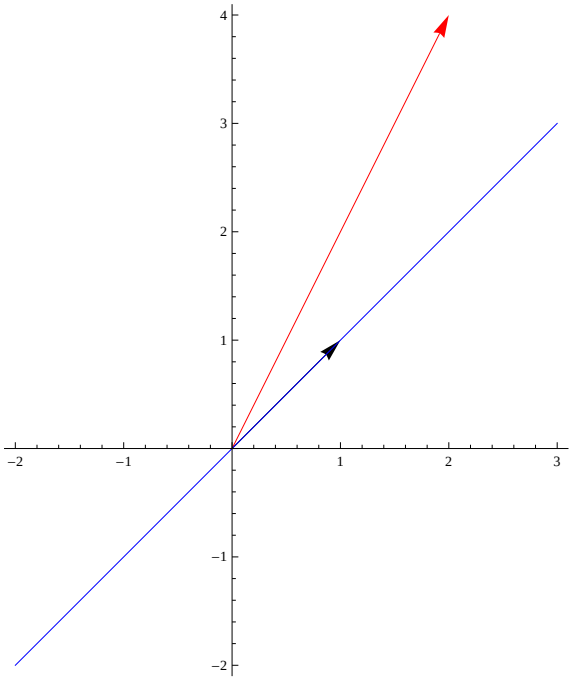
- $$\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ -1 \end{bmatrix} y = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

  Търсим числата x и y така, че линейната комбинация на вектор-стълбовете в лявата страна (означени в черно) да е равна на дясната страна (означена в червено).



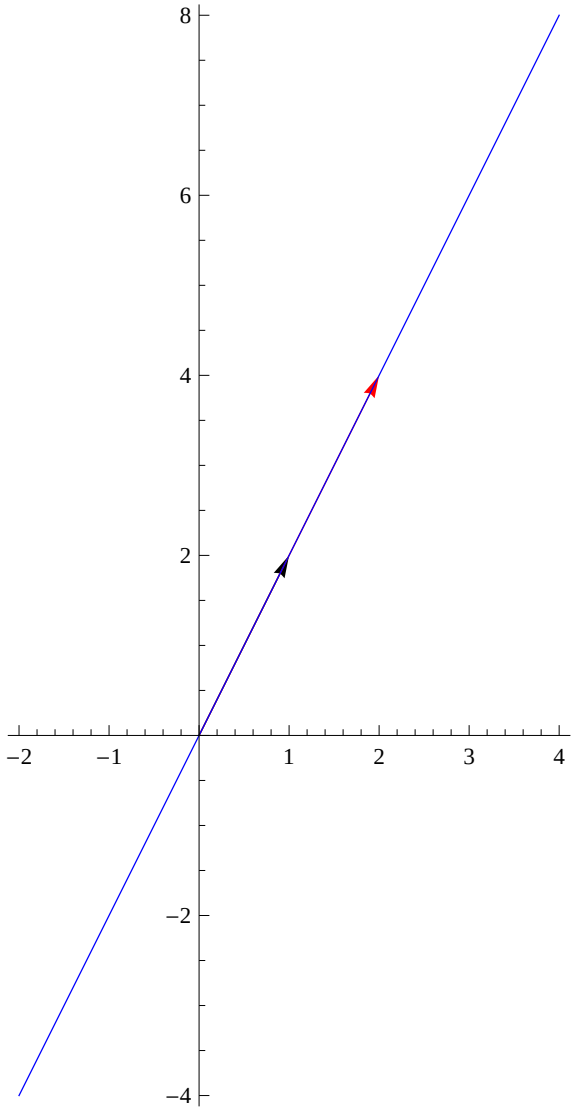
- Във втория пример двата вектора в лявата страна лежат на една права Тогава всички техни линейни комбинации ще останат на същата права (означена в синьо). Следователно системата няма решение, тъй като нито една линейна комбинация не може да е равна на червения вектор.

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$



- В третия пример векторите в лявата страна отново са равни, но дясната лежи на правата, определена от тях. Следователно системата има решение. По-точно тя има безброй много решения, тъй като, независимо колко далеч по правата "сме се придвижили" с първия вектор, то вторият може "да ни върне" до желаната точка.

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 2 \end{bmatrix} y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$



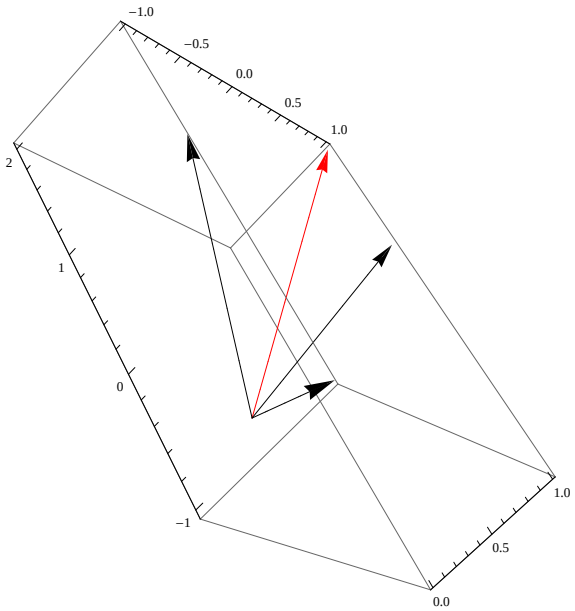
Важно

Умножавайки дадена матрица отдясно с вектор-стълб, получаваме линейна комбинация на стълбовете на матрицата.

Предимството на column picture в сравнение с row picture е, че първото може непосредствено да се пренесе и в 3D, практически без промени. Идеята е толкова проста, че можем почти да "визуализираме" column picture и в 4 и повече измерения.

Пример 10. Да разгледаме системата

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



В този случай векторите в лявата страна сочат в три независими посоки и пораждат цялото $\mathbb{R}^3$ (казано иначе, движейки се в тези посоки, можем да стигнем до всяка точка в пространството). Следователно системата има единствено решение за всяка дясна страна.

Важно

Дадена линейна алгебрична система може да бъде визуализирана с нейните row picture и column picture.
Работейки по редове, търсим пресечните точки на хиперравнините в n-мерното пространство, определени от всяко уравнение.
Работейки по стълбове, търсим линейната комбинация на вектор-стълбовете на матрицата на системата, равна на вектор-стълба на десните страни.

1. Умножение отдясно (умножение по стълбове)

   На база на казаното по-горе, можем да видим и какъв е ефектът на това да умножим една матрица отдясно с друга:

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ B_1 & B_2 & B_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ C_1 & C_2 & C_3 \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

   В този случай за стълбовете на резултата е в сила

$$C_i = a_{1i}B_1 + a_{2i}B_2 + a_{3i}B_3, \ i = \overline{1,3}.$$

Важно

С други думи, ако умножим матрицата B отдясно с A, то i-тият стълб на резултата се получава като линейна комбинация на стълбовете на B. Коефициентите в тази линейна комбинация са елементите от i-тия стълб на матрицата A.

2. Умножение с матрица отляво (умножение по редове).

Да разгледаме следното матрично равенство:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \cdots B_1 \cdots \\ \cdots B_2 \cdots \\ \cdots B_3 \cdots \end{bmatrix} = \begin{bmatrix} \cdots C_1 \cdots \\ \cdots C_2 \cdots \\ \cdots C_3 \cdots \end{bmatrix}.$$

В сила е следната зависимост:

$$C_i = a_{i1}B_1 + a_{i2}B_2 + a_{i3}B_3, \ i = \overline{1,3}.$$

Важно

С други думи, ако умножим матрицата B отляво с A, то i-тият ред на резултата се получава като линейна комбинация на редовете на B. Коефициентите в тази линейна комбинация са елементите от i-тия ред на матрицата A.

За простота разгледахме случая на матрици 3×3. Обобщението за произволни матрици е тривиално.

2.2.2 Алгоритъм

И така, вече сме готови да видим как може да се намери исканото разлагане. Ще го илюстрираме с пример. Да разгледаме матрицата

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}.$$

Извършваме правия ход на метода на Гаус и получаваме горна триъгълна матрица:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} =: U.$$

С други думи, методът на Гаус **трансформира** матрицата A до горната триъгълна матрица U. Но това е една линейна трансформация, следователно тя може да се представи чрез умножението на A с дадена матрица отляво, т.е. $L^{-1}A = U$, където L^{-1} е матрицата на въпросната трансформация. Тогава е в сила $A = LU$. Въпросът е как да намерим матрицата L (която, както ще видим, е долна триъгълна матрица, т.е. ни дава исканото разлагане).

- На първата стъпка от метода на Гаус матрицата A се преобразува до $A^{(1)}$ и нека матрицата на това преобразование е L_1^{-1} ($L_1^{-1}A = A^{(1)}$). Първият ред на матрицата $A^{(1)}$ е равен на първия ред на матрицата A:

$$A_1^{(1)} = 1 \times A_1 + 0 \times A_2 + 0 \times A_3. \tag{2.5}$$

Вторият ред на $A^{(1)}$ се получава, като от втория ред на A извадим първия, умножен по 1, т.е.

$$A_2^{(1)} = -1 \times A_1 + 1 \times A_2 + 0 \times A_3. \tag{2.6}$$

Аналогично

$$A_3^{(1)} = -1 \times A_1 + 0 \times A_2 + 1 \times A_3. \tag{2.7}$$

Вземайки предвид (2.5), (2.6), (2.7) и това, което казахме за умножение с матрица отляво, е ясно, че $A^{(1)} = L_1^{-1}A$, където

$$L_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

- Аналогично $U = L_2^{-1}A^{(1)}$, където

$$L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}.$$

Тогава $(L_2^{-1}L_1^{-1})A = U$ и следователно $A = L_1L_2U$.

- Следващият въпрос е как да намерим обратните матрици на L_1^{-1} и L_2^{-1}. Ако разглеждаме матрицата L_1^{-1} като трансформация, можем лесно да отговорим на този въпрос. Това е трансформация, която приложена върху дадена матрица X вади първия ѝ ред от втория. Обратната трансформация тогава трябва да прибавя първия ред на X към втория. Аналогично обратната трансформация трябва да прибавя първия ред на X към третия:

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

С други думи, всички елементи под главния диагонал сменят знаците си. Аналогично

$$L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

- Лесно се вижда, че

$$L := L_1 L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

И така, приведеният пример илюстрира факта, че за да разложим матрицата A във вида $A = LU$, трябва да извършим правия ход на метода на Гаус и така получаваме матрицата U.

Важно

Матрицата L е долна триъгълна матрица, чиито елементи a_{ij} под главния диагонал са коефициентите, с които умножаваме j-тия ред, за да го извадим от i-тия.

Да разгледаме още една задача, с която да затвърдим казаното.

Задача 6. Да се намери LU декомпозицията на матрицата

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix}$$

Решение. Правият ход на алгоритъма ни дава

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 8 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{bmatrix} =: U.$$

На първата стъпка от втория ред вадим първия, умножен по 2, а от третия вадим първия, умножен по -1. На втората стъпка от третия ред вадим втория, умножен по -1. Тогава матрицата L има вида

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

С други думи,

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

□

Можем лесно да модифицираме програмата, която предложихме за метода на Гаус, като в една нова матрица L, която в началото е единичната матрица и на всяка стъпка запазваме коефициентите, с които i-тият ред се умножава, за да се извади от следващите. Прилагаме примерна имплементация на Mathematica.

```
In[8]:= LU[A_] :=
 (U = A;
  n = Length[A];
  L = Table[0, {n}, {n}];
  For[i = 1, i ≤ n, i++,
   L[[i, i]] = 1
  ];
  For[i = 1, i ≤ n, i++,
   For[j = i + 1, j ≤ n, j++,
    l = U[[j, i]] / U[[i, i]];
    L[[j, i]] = l;
    U[[j, i]] = 0;
    For[k = i + 1, k ≤ n, k++,
     U[[j, k]] = U[[j, k]] - l * U[[i, k]];
    ]
   ]
  ];
  {L, U}
 )
```

Както казахме, имайки LU декомпозицията на дадена матрица, можем лесно да решим системата $LU\mathbf{x} = \mathbf{b}$ на две стъпки.

Първо, решаваме системата $L\mathbf{y} = \mathbf{b}$, която има долна триъгълна матрица. i-тото уравнение на тази система има вида

$$l_{i1}y_1 + \cdots + l_{ii}y_i = b_i,$$

откъдето получаваме

$$y_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}y_j}{l_{ii}}, \quad i = 1, \ldots, n.$$

След това решаваме системата $U\mathbf{x} = \mathbf{y}$, която има горна триъгълна матрица. Имаме

$$u_{ii}x_i + \cdots + u_{in}x_n = y_i$$

и следователно

$$x_i = \frac{y_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}, \quad i = n, \ldots, 1.$$

Прилагаме примерна имплементация на обратния ход на алгоритъма.

```
In[1]:= LUSolve[L_, U_, b_] := (
  n = Length[b];
  y = Table[0, {i, n}] ;
  For[i = 1, i ≤ n, i++,
   y[[i]] = (b[[i]] - Sum[L[[i, j]] y[[j]], {j, 1, i - 1}])/L[[i, i]]
   ];
  x = Table[0, {i, n}] ;
  For[i = n, i ≥ 1, i--,
   x[[i]] = (y[[i]] - Sum[U[[i, j]] x[[j]], {j, i + 1, n}])/U[[i, i]]
   ];
  x
  )
```

Важно

На база на казаното, правенето на LU-разлагане реално не е нов начин за решаване на една линейна алгебрична система, а по-скоро друг начин за записване на метода на Гаус. Матрицата L съдържа информацията от правия ход на метода, а матрицата U съдържа резултата от правия ход.

2.2.3 Кога е полезно използването на LU-декомпозицията?

Както казахме, ако сме разложили матрицата A във вида $A = LU$, системата $A\mathbf{x} = \mathbf{b}$ може да се реши със сложност $O(n^2)$. Самото разлагане на матрицата обаче става по метода на Гаус. С други думи, ако решим системата директно по метода на Гаус, ще направим дори по-малко операции, защото иначе след метода на Гаус (за да разложим A) ще трябва да решим две системи с триъгълна матрица.

Важно

Предимството на това разлагане обаче идва, ако трябва да решим много системи с една и съща матрица. Ако искаме да решим n системи по метода на Гаус, това означава да направим $n \times O(n^3) = O(n^4)$ операции, докато, ако първо разложим матрицата (за $O(n^3)$ операции) и после решим n-те системи за $n \times O(n^2) = O(n^3)$ операции, в крайна сметка ще сме направили $O(n^3)$ операции.

Интересно

В таблицата по-долу е приведено сравнение между порядъка операции, които трябва да се извършат, за да се реши дадена система с n уравнения по метода на Гаус (броени са само умножения/деления, които са по-бавни операции и са $n^3/3 + O(n^2)$) и като се решат двете системи с триъгълни матрици след намирането на LU-разлагане:

n	$n^3/3$	$2n^2$	% Reduction
10	$3.\overline{3} \times 10^2$	2×10^2	40
100	$3.\overline{3} \times 10^5$	2×10^4	94
1000	$3.\overline{3} \times 10^8$	2×10^6	99.4

Да разгледаме няколко примера, които илюстрират защо би ни било полезно да решаваме много линейни алгебрични системи с една и съща матрица, но различни десни страни.

Пример 11. Както казахме, линейни алгебрични системи възникват директно при моделирането на физически системи, които са в равновесно състояние, т.е. не се изменят във времето/пространството. Нека разгледаме отново системата от пример 1. За удобство ще я запишем отново тук:

$$\begin{aligned} 6c_1 - c_3 &= 5c_{01}, \\ -3c_1 + 3c_2 &= 0, \\ -c_2 + 9c_3 &= 8c_{03}, \\ -c_2 - 8c_3 + 11c_4 - 2c_5 &= 0, \\ -3c_1 - c_2 + 4c_5 &= 0, \end{aligned}$$

където c_{01} и c_{03} са концентрациите на разглежданото вещество в разтворите, вливани в първия и третия реактор. **С други думи, десните страни описват това, с което системата бива стимулирана.** В разглеждания пример имахме $c_{01} = 10$ и $c_{03} = 20$. Решавайки горната система, можем да намерим равновесната концентрация във всеки от реакторите.

Разбира се, моделирането често има за цел да се симулират различни сценарии и евентуално процесът да се оптимизира. Нека например имаме възможност да избираме между разтвори с концентрации 10, 20, 30, 40, 50 които да вливаме. Тогава можем да пресметнем равновесната концентрация за всяка комбинация от разтвори, като първо намерим LU-разлагането на матрицата на системата и след това използваме реализирания по-горе метод LUSolve за решаването на системата с всяка дясна страна:

```
In[7]:= A = {{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}} // N;
    luA = LU[A];
    concentrations = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
    For[input1 = 1, input1 ≤ Length[concentrations], input1++,
     For[input2 = 1, input2 ≤ Length[concentrations], input2++,
      result = LUSolve[luA[[1]], luA[[2]],
        {5 concentrations[[input1]], 0, 8 concentrations[[input2]], 0, 0}];
      Print["Input 1:", concentrations[[input1]], "Input 2:",
       concentrations[[input2]], ". Output concentrations:", result]
     ]
    ]
```

Част от резултата от изпълнението на горния код е приведен по-долу:

```
Input 1:10Input 2:10. Output concentrations:
{10., 10., 10., 10., 10.}

Input 1:10Input 2:20. Output concentrations:
{11.5094, 11.5094, 19.0566, 16.9983, 11.5094}

Input 1:10Input 2:30. Output concentrations:
{13.0189, 13.0189, 28.1132, 23.9966, 13.0189}

Input 1:10Input 2:40. Output concentrations:
{14.5283, 14.5283, 37.1698, 30.9949, 14.5283}

Input 1:10Input 2:50. Output concentrations:
{16.0377, 16.0377, 46.2264, 37.9931, 16.0377}

Input 1:20Input 2:10. Output concentrations:
{18.4906, 18.4906, 10.9434, 13.0017, 18.4906}

Input 1:20Input 2:20. Output concentrations:
{20., 20., 20., 20., 20.}
```

На база на резултатите експериментаторът може да вземе решение за това кой ще бъде оптималният вариант за разглеждания процес.

Пример 12. Още един важен случай, в който се налага решаването на много системи с една и съща матрица, е намирането на обратна матрица. Нека е дадена матрицата A. Нейната обратна е такава матрица, че

$$A \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Тогава стълбовете на матрицата A^{-1} са решенията на системите

$$A\mathbf{x}_i = \mathbf{e}_i,\ i = \overline{1,n}$$

където e_i е i-тият единичен вектор и за решаването им е удобно матрицата A да бъде разложена във вида $A = LU$. Нека разгледаме примерна функция, намираща обратната матрица на дадена несингулярна квадратна матрица.

```
In[2]:= InverseMatrix[A_] := (
          {L, U} = LU[A];
          n = Length[A];
          AInverse = Table[0, {n}, {n}];
          For[k = 1, k ≤ n, k++,
           b = Table[If[l == k, 1, 0], {l, 1, n}];
           AInverse[[k]] = LUSolve[L, U, b];
          ];
          Transpose[AInverse]
         )
```

Пример 13. Пресмятането на детерминантата на матрица също става непосредствено след разлагането на матрицата A. Имаме $\det(A) = \det(L)\det(U)$.

Интересно

Повечето комерсиални софтуерни пакети за решаване на линейни алгебрични системи не прилагат метода на Гаус директно, а първо правят LU-разлагане. Разбира се, както вече казахме, обикновено методът на Гаус се прилага с частичен избор на главния елемент. В този случай LU-разлагането може да се обобщи и да се получи декомпозиция от вида $A = P^{-1}LU$, където P е пермутационна матрица, т.е. описва смените на редове в правия ход на метода на Гаус с частичен избор на главния елемент.

2.3 Числени методи за системи със специална структура

Както отбелязахме във въведението, често на практика матриците, които се получават при решаването на дадена реална задача, не са произволни, а имат някаква специална структура, например:

- симетрични матрици – $A = A^T$;
- тридиагонални матрици – матрици, които имат ненулеви елементи само по главния диагонал и диагоналите под и над него;
- разредени матрици – матрици с много нулеви елементи.

Когато матрицата на дадена система има специална структура, тя може да се използва, за да получим решението на системата по-ефективно (с по-малка времева сложност).

2.3.1 Метод на Холецки за разлагане на симетрични и положително определени матрици

В случая, когато матрицата A е симетрична и положително определена, се оказва, че разлагането на горна и долна триъгълна матрица, което разгледахме

в секция 2.2 може да стане по-лесно. Матрицата може да бъде разложена във вида

$$A = LL^T,$$

където L е долна триъгълна матрица, т.е. има вида

$$L = \begin{bmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix}.$$

Това е т.нар. разлагане на Холецки.

Интересно: Андре-Луи Холецки

Андре–Луи Холецки (1777-1855) е френски топограф, математик и военен. Известен е основно с откриването на декомпозицията на матрица, с която се занимаваме в настоящата секция, и която той използвам в своята геодезическа дейност. Артилерийски офицер, той загива в битка няколко месеца преди края на първата световна война. Неговото откритие става публично след смъртта му, благодарение на друг офицер от френската армия, комендант Беноа.

Това дали една матрица е положително-определена може да се установи, като се използва критерият на Силвестър.

Твърдение: Критерий на Силвестър

Една матрица е положително определена, когато всичките ѝ главни минори са положителни, т.е.

$$\Delta_1 = |a_{11}| > 0,\ \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \ldots,$$

$$\Delta_i = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1i} \\ a_{21} & a_{22} & \cdots & a_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} \end{vmatrix} > 0, \ldots, \Delta_n = \det A_n > 0$$

Нека разгледаме следния пример:

Задача 7. Да се провери, че матрицата

$$A = \begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix}$$

е симетрична и положително-определена и да се разложи по метода на Холецки.

Решение. Очевидно матрицата е симетрична. Нека се уверим, че тя е и положително-определена. За тази цел прилагаме критерия на Силвестър. Главните минори са

$$\Delta_1 = |4| = 4 > 0,\ \Delta_2 = \begin{vmatrix} 4 & 12 \\ 12 & 37 \end{vmatrix} = 4 > 0, \Delta_3 = \begin{vmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{vmatrix} = 36 > 0.$$

Следователно матрицата действително е положително-определена.

Търсим разлагането във вида

$$\begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_3 3 \end{bmatrix}.$$

Умножавайки първия ред на L с първия стълб на L^T, можем да намерим l_{11}. Получаваме

$$l_{11}^2 = 4 \Rightarrow l_{11} = 2.$$

След това можем да намерим останалите елементи от първия стълб на L, умножавайки съответно втория и третия ред на L с първия стълб на L^T:

$$l_{21}l_{11} = 12 \Rightarrow l_{21} = 6,$$
$$l_{31}l_{11} = -16 \Rightarrow l_{31} = -8.$$

Преминаваме към втория стълб, като отново първо намираме диагоналния елемент и след това елементите под него:

$$l_{21}^2 + l_{22}^2 = 37 \Rightarrow l_{22} = 1,$$
$$l_{31}l_{21} + l_{32}l_{22} = -43 \Rightarrow l_{32} = 5.$$

Накрая за последния диагонален елемент имаме

$$l_{31}^2 + l_{32}^2 + l_{33}^2 = 98 \Rightarrow l_{33} = 3.$$

Окончателно получихме матрицата

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{bmatrix}$$

□

В общия случай, аналогично, последователно се попълват първият, вторият и т.н. стълб, започвайки от диагоналния елемент:

$$for\ k = \overline{1,n}:$$
$$l_{kk} = \sqrt{a_{kk} - l_{k1}^2 - \cdots - l_{k,k-1}^2},$$
$$l_{jk} = \frac{a_{kj} - \sum_{i=1}^{k-1} l_{ki}l_{ji}}{l_{kk}}, \quad j = \overline{k+1,n}.$$

За извеждането на горните формули виж учебника.

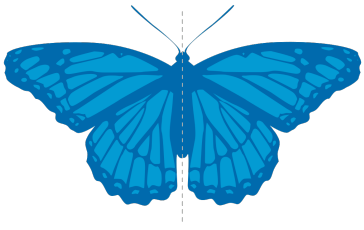
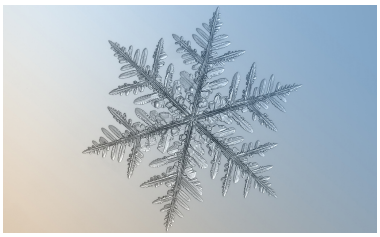
Важно

Може да се покаже, че методът на Холецки има сложност $\frac{1}{3}n^3 + O(n^2)$ операции. Следователно, когато матрицата на дадена система е симетрична и положително определена, е целесъобразно да се използва методът на Холецки за разлагане на матрицата, откъдето системата да се реши, както обяснихме в секция 2.2. Това ще направи решаването на системата по-бързо, отколкото ако използваме метода на Гаус.

Прилагаме примерна имплементация на метода в системата Mathematica:

```
In[1]:= Cholesky[A_] := (
  n = Length[A];
  L = Table[0, {i, n}, {j, n}];
  For[k = 1, k ≤ n, k++, (*Iterate over the columns*)
   (*Find the diagonal element*)
   L[[k, k]] = √(A[[k, k]] - Sum[L[[k, p]]^2, {p, 1, k - 1}]) ;
   For[j = k + 1, j ≤ n, j++, (*Find all the elements below the main diagonal*)
    L[[j, k]] = (A[[k, j]] - Sum[L[[k, i]] L[[j, i]], {i, 1, k - 1}])/L[[k, k]]
   ]
  ];
  L
 )
```

Въпросът за работата със симетрични матрици е много важен на практика, тъй като много задачи, които възникват при разглеждането на реални проблеми водят до линейни алгебрични системи с такива матрици. Симетрията се среща навсякъде в природата и ето защо е естествено, че тя намира своето отражение в математическите модели и се среща постоянно в математически задачи.


Тук ще се спрем като пример на една от много важните задачи, където намират приложение методите за симетрични матрици – задачата за намиране на приближение по метода на най-малките квадрати.

Метод на най-малките квадрати

Един от най-често използваните на практика числени методи е методът на най-малките квадрати. Нека са дадени точките $(x_1, y_1), \ldots, (x_s, y_s)$. Търсим полином от степен, ненадминаваща n, който да е възможно "най-близо" до точките. Можем да формулираме задачата и по следния начин. Искаме вектора от грешките, които се получават във всяка от дадените точки, да е възможно

"най-малък", т.е. да има най-малка Евклидова дължина. Векторът на грешките има вида

$$\mathbf{r} := \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_s \end{bmatrix} = \begin{bmatrix} p(x_1) \\ p(x_2) \\ \vdots \\ p(x_s) \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_s \end{bmatrix}.$$

Нека запишем полинома $p(x)$ във вида

$$p(x) = (1, x, \ldots, x^n)(a_0, a_1, \ldots, a_n)^T,$$

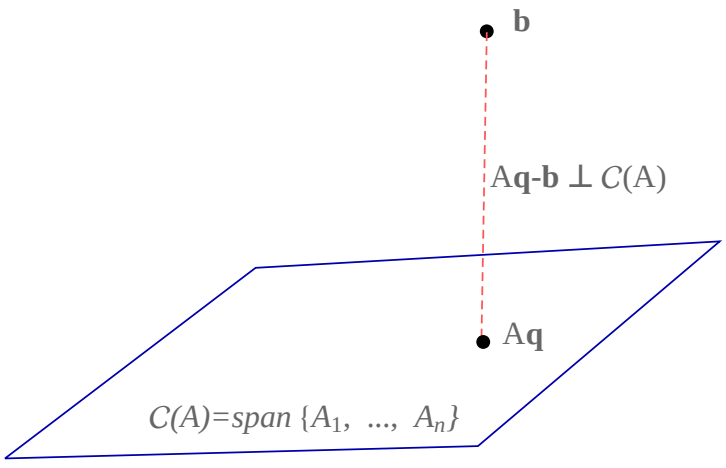
където $a_0, a_1, \ldots, a_n$ са коефициентите на полинома. Тогава получаваме

$$\mathbf{r} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_s & \cdots & x_s^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_s \end{bmatrix} =: A\mathbf{q} - \mathbf{b}. \tag{2.8}$$

Може да се покаже, че въпросът за минимизирането на $\|A\mathbf{q} - \mathbf{b}\|_2$ е еквивалентен на този за решаването на системата (така наречените **нормални уравнения**)

$$A^T A\mathbf{q} = A^T\mathbf{b}. \tag{2.9}$$

С други думи, нашата задача е да определим $\mathbf{q} \in \mathbb{R}^{n+1}$ така, че $A\mathbf{q} \in \mathcal{C}(A)$ да е възможно най-близко до $\mathbf{b}$ (в смисъла на $\|\cdot\|_2$-нормата). Интуитивно е ясно (а в курса "Числени методи на анализа" е и доказвано), че $\mathbf{q}$ е оптимално решение т.с.т.к. $A\mathbf{q}$ е ортогоналната проекция на $\mathbf{b}$ в $\mathcal{C}(A)$:



С други думи, трябва да определим $\mathbf{q}$ така, че да е в сила

$$(\mathbf{b} - A\mathbf{q}) \cdot A_i = 0,\ i = \overline{1, n},$$

където A_i са стълбовете на A. Последното условие е същото като

$$A^T(\mathbf{b} - A\mathbf{q}) = \mathbf{0}$$

или

$$A^T A\mathbf{q} = A^T\mathbf{b}.$$

Получихме т.нар. **нормални уравнения**. Да забележим, че ако A има пълен ранг по стълбове (засега ще се ограничим до този случай, тъй като той е най-важен на практика), матрицата $A^T A$ е квадратна, обратима, симетрична и положително определена.

Твърдение: Нормални уравнения за решаване на задача по метода на най-малките квадрати

Оптималното решение на преопределената система

$$A\mathbf{q} = \mathbf{b},$$

т.е. решението за което е минимална Евклидовата дължина (втората норма) на грешката – $\|A\mathbf{q} - \mathbf{b}\|_2$, е решението на системата

$$A^T A\mathbf{q} = A^T b.$$

Матрицата $A^T A$ на тази система е квадратна, симетрична и положително-определена.

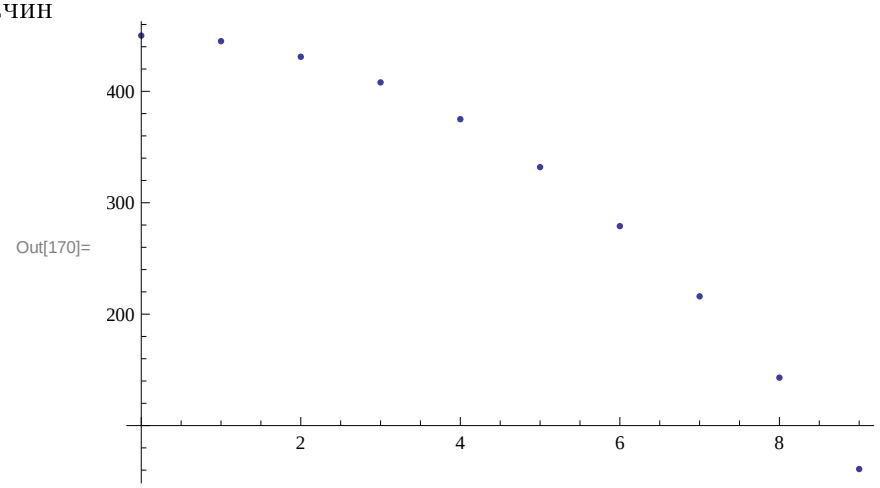
Да разгледаме следния пример.

Задача 8. Тяло е пуснато от височина $450m$. Неговата височина е измервана през интервали от 1 сек. Данните са систематизирани в следната таблица:

t, sec	0	1	2	3	4	5	6	7	8	9
h, m	450	445	431	408	375	332	279	216	143	61

Да се намери подходяща функция, описваща процеса.

Решение. Нека първо да изобразим точките. Графиката изглежда по следния начин



Точките изглежда да лежат върху парабола. Затова ще търсим функцията, моделираща процеса, във вида $f(x) = a_0 + a_1 x + a_2 x^2$. За да определим по метода

на най-малките квадрати коефициентите на полинома, вземайки предвид (2.8) и (2.9), трябва да решим системата

$$A^T A\mathbf{q} = A^T\mathbf{b},$$

където

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \end{bmatrix},\ \mathbf{q} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} 450 \\ 445 \\ 431 \\ 408 \\ 375 \\ 332 \\ 279 \\ 216 \\ 143 \\ 61 \end{bmatrix}.$$

Нека означим $B := A^T A$, $\mathbf{c} := A^T\mathbf{b}$. Тогава системата, която трябва да решим, е $B\mathbf{x} = \mathbf{c}$. Първо, ще разложим матрицата B, която е симетрична и положително определена по метода на Холецки:

```
x = Range[0, 9];
A = Table[{1, x[[i]], x[[i]]^2}, {i, 1, 10}];
y = {{450, 445, 431, 408, 375, 332, 279, 216, 143, 61}} // Transpose;
B = Transpose[A].A;
c = Transpose[A].b;
L = Cholesky[B]
```

В резултат от изпълнението на горния код получаваме, че матрицата L има вида

$$L = \begin{bmatrix} \sqrt{10} & 0 & 0 \\ 9\sqrt{\frac{5}{2}} & \sqrt{\frac{165}{2}} & 0 \\ 57\sqrt{\frac{5}{2}} & 9\sqrt{\frac{165}{2}} & 4\sqrt{33} \end{bmatrix}.$$

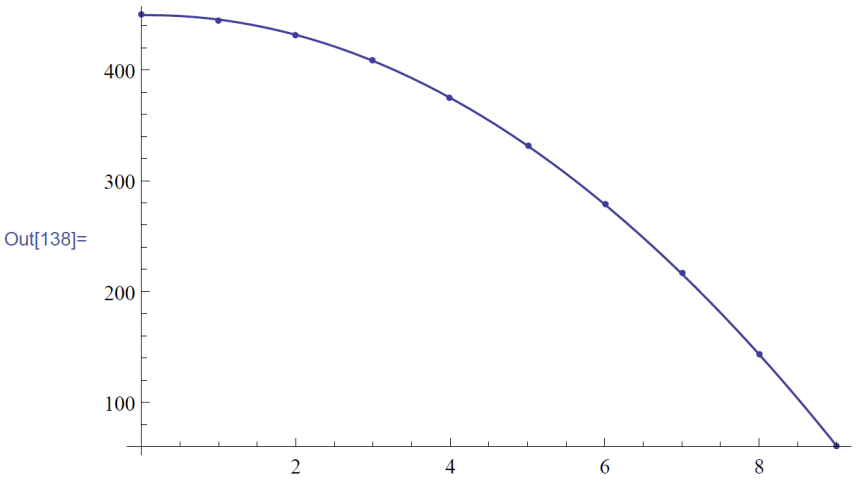
Остава да решим системата $LL^T\mathbf{q} = \mathbf{c}$ на две стъпки, както обяснихме в параграфа, посветен на LU-декомпозицията. За тази цел използваме написаната от нас по-рано функция LUSolve.

```
LUSolve[L, Transpose[L], c]
```

Окончателно получихме $\mathbf{q} = (449.364, 0.962121, -4.90152)^T$.

Нека построим графиките на полинома $P(x) = \mathbf{q}\cdot(1, x, x^2)$ и дадените точки в една координатна система, за да визуализираме резултата.

```
In[44]:= p[x_] := {{1, x, x^2}}.q;
         plot1 = Plot[p[x], {x, 0, 9}]
         plot2 = ListPlot[Table[{x[[i]], b[[i, 1]]}, {i, 1, Length[x]}]];
         Show[plot1, plot2]
```



□

Интересно

Изучавайки света около нас, ние искаме да намерим зависимости между различни величини. За да се изследва дадено явление или даден процес, е необходимо да се направят експерименти, които да дадат информация за него. След това данните от тези експерименти се използват, за да се създаде математически модел, който ги описва.

Твърди се, че Галилео Галилей открил закона за свободното падане на телата, като правил експерименти от наклонената кула в Пиза. Той пускал тела от нея, като така стигнал до хипотезата, че изминатият път е пропорционален на квадрата на времето.

2.3.2 Метод на дясната прогонка за решаване на системи с тридиагонална матрица

Важен клас системи са тези с тридиагонална матрица, т.е. тези, които имат ненулеви елементи само по главния диагонал, под него и над него:

$$\begin{bmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & * & * & * & \\ & & & & * & * \end{bmatrix}.$$

Нека запишем в общ вид системата, зададена с тридиагонална матрица, по следния начин:

$$\begin{array}{rcrcrcll} & - & C_1x_1 & + & B_1x_2 & = & F_1, & \\ A_ix_{i-1} & - & C_ix_i & + & B_ix_{i+1} & = & F_i, & i=2,\ldots,n-1 \\ A_nx_{n-1} & - & C_nx_n & & & = & F_n. & \end{array}$$

Тогава от първото уравнение можем да изразим първото неизвестно чрез второто:

$$x_1 = \frac{B_1}{C_1}x_2 - \frac{F_1}{C_1} =: \alpha_2x_2 + \beta_2.$$

Така, ако сме изразили $x_{i-1} = \alpha_ix_i+\beta_i$, можем да го заместим в i-тото уравнение и оттам да изразим x_i чрез x_{i+1}:

$$x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}. \tag{2.10}$$

Коефициентите α_i и β_i се определят от формулите

$$\alpha_{i+1} = \frac{B_i}{C_i - A_i\alpha_i},\ \beta_{i+1} = \frac{A_i\beta_i - F_i}{C_i - A_i\alpha_i},\ i = 1,\ldots,n-1.$$

Накрая, заместваики $x_{n-1} = \alpha_nx_n + \beta_n$ в последното уравнение, получаваме

$$x_n = \frac{A_n\beta_n - F_n}{C_n - A_n\alpha_n}.$$

Тогава от (2.10) можем да изразим последователно $x_{n-1}, x_{n-2}, \ldots, x_1$.

Задача 9. Като се използва методът на дясната прогонка, да се реши системата

$$\begin{array}{rcrcrcrcl} 3x_1 & - & x_2 & & & & & = & 3, \\ -x_1 & + & 3x_2 & - & x_3 & & & = & 0, \\ & & -x_2 & + & 3x_3 & - & x_4 & = & 0, \\ & & & & -x_3 & + & 3x_4 & = & 0. \end{array}$$

Решение. От първото уравнение изразяваме

$$x_1 = \frac{1}{3}x_2 + 1.$$

Заместваики горното във второто уравнение, получаваме

$$-\frac{1}{3}x_2 - 1 + 3x_2 - x_3 = 0 \Longrightarrow x_2 = \frac{3}{8}x_3 + \frac{3}{8}.$$

□

Продължавайки аналогично, получаваме

$$-\frac{3}{8}x_3 - \frac{3}{8} + 3x_3 - x_4 = 0 \Longrightarrow x_3 = \frac{8}{21}x_4 + \frac{1}{7}$$

и окончателно

$$-\frac{8}{21}x_4 - \frac{1}{7} + 3x_4 = 20 \Longrightarrow x_4 = \frac{423}{55}.$$

Сега, връщайки се назад, имаме

$$x_3 = \frac{8}{21}x_4 + \frac{1}{7} = \frac{169}{55},\ x_2 = \frac{3}{8}x_3 + \frac{3}{8} = \frac{84}{55},\ x_1 = \frac{1}{3}x_2 + 1 = \frac{83}{55}.$$

Прилагаме примерна имплементация на метода:

```
In[41]:= A = {{3, -1, 0, 0}, {-1, 3, -1, 0}, {0, -1, 3, -1}, {0, 0, -1, 3}} // N;
         b = {3, 0, 0, 20} // N;
         (*Find the coefficients α_i, β_i*)
         n = Length[A];
         α = Table[0, {i, n}];
         β = Table[0, {i, n}];
         x = Table[0, {i, n}];
         α[[2]] = A[[1, 2]] / (-A[[1, 1]]); (*C_1=-A[[1,1]]*)
         β[[2]] = -b[[1]] / (-A[[1, 1]]);
         For[i = 2, i < n, i++,
          α[[i + 1]] = A[[i, i + 1]] / (-A[[i, i]] - A[[i, i - 1]] α[[i]]); (*C_i=-A[[i,i]]*)
          β[[i + 1]] = (A[[i, i - 1]] β[[i]] - b[[i]]) / (-A[[i, i]] - A[[i, i - 1]] α[[i]]);
         ]
         x[[n]] = (A[[n, n - 1]] β[[n]] - b[[n]]) / (-A[[n, n]] - A[[n, n - 1]] α[[n]]);
         For[i = n - 1, i ≥ 1, i--,
          x[[i]] = α[[i + 1]] x[[i + 1]] + β[[i + 1]]
         ]
         x
Out[51]= {1.50909, 1.52727, 3.07273, 7.69091}
```

Допълнителни задачи 1

Задачи за решаване на ръка

Задача 10. В системата числа с плаваща точка, която дефинирахме (с 3-цифрена мантиса и 1-цифрена експонента със знак, десетична бройна система), направете пресмятанията, илюстриращи факта, че дистрибутивният закон

$$a \times (b + c) = a \times b + a \times c$$

може и да не е в сила при числени пресмятания. За целта използвайте стойностите $a = 0.200 \times 10^1$, $b = -0.600 \times 10^0$, $c = 0.602 \times 10^0$.

Задача 11. Използвайки метода на Гаус с частичен избор на главния елемент, решете системата

$$\begin{aligned} u + 4v + 2w &= -2, \\ -2u - 8v + 3w &= 32, \\ v + w &= 1. \end{aligned}$$

Напишете пермутационните матрици P_1 и P_2, които описват смените на редове на първата и втората стъпка от алгоритъма.

Намерете матрицата $\overline{A} = PA$, чиято LU-декомпозиция бихме намерили по класическия метод на Гаус.

Задача 12. Намерете LU-декомпозицията на матрицата

$$A = \begin{bmatrix} 2 & 3 & 3 \\ 0 & 5 & 7 \\ 6 & 9 & 8 \end{bmatrix}.$$

Като използвате тази декомпозиция, решете системите $Ax = b$ с десни страни $b = (2, 2, 5)^T$ и $b = (1, 1, -3)^T$.

Какво печелим от LU-декомпозицията при решаването на тези системи?

Задача 13. Дадена е матрицата

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 4 \\ 1 & 4 & 8 \end{bmatrix}.$$

Коя е матрицата L_1^{-1}, която трансформира A в $A^{(1)}$ – матрицата, получена след първата стъпка на метода на Гаус,

$$A^{(1)} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 3 & 3 \\ 0 & 3 & 7 \end{bmatrix}?$$

Коя е обратната на L_1^{-1} матрица?

Задача 14. Като използвате метода на Холецки, пресметнете $\mathbf{x} = A^{-1}\mathbf{b}$, $\det A$ и A^{-1}:

$$[A|\mathbf{b}] = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 2 & 5 & 8 & -1 \\ 3 & 8 & 14 & -3 \end{array}\right]$$

Задача 15. Да се докаже, че ако A е обратима диагонална/триъгълна матрица, то и A^{-1} е от същия вид.

Задача 16. Да се докаже, че методът на Холецки за разлагане на симетрични и положително определени матрици има сложност $1/3n^3 + O(n^2)$ аритметични операции.

Задача 17. Дадено е диференциалното уравнение

$$u''(x) + u(x) = x,\ x \in (0, 1)$$
$$u(0) = u(1) = 0.$$

Да се реши приближено, като интервалът $[0, 1]$ се раздели с равноотдалечени възли на подинтервали със стъпка $h = 0.01$ и производните в него се апроксимират във всеки възел (вж. записките). Получената линейна система да се реши с подходящ числен метод.

Програмиране

Задача 18. Напишете програма на език за програмиране, който знаете (C++/ Java/ FORTRAN и др.), която намира корените на квадратното уравнение

$$x^2 + 3000.001x + 3,$$

използвайки стандартната формула за целта.

- Направете експеримент, като използвате променливи с единична (float) и двойна точност (double).
- Сравнете с точните стойности на корените: -0.001 и -3000.
- Обяснете коя е причината за появилите се грешки.
- Потърсете в интернет информация за устойчиви по отношение на грешките от закръгляване методи за решаване на квадратни уравнения. Имплементирайте някой от тях.

Забележка. Matlab и системите за компютърна алгебра не са подходящи за целите на задачата, тъй като в тях не задаваме явно типа на данните.

Задача 19. Да се напише функция $hilbertMatrix[n\_]$ в Mathematica, която генерира матрицата

$$\begin{bmatrix} 1 & 1/2 & \cdots & 1/n \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n) \end{bmatrix}.$$

Задача 20. Да се модифицира програмата, реализираща метода на Гаус, за да се реализира методът на Гаус-Жордан без избор на главния елемент (вж. записките). Разликата между двата метода 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.

Задача 21. Напишете програма, която намира обратната на дадена симетрична матрица, като използва метода на Холецки за получаване на триъгълното разлагане. Сравнете времената за реализираната функция и функцията $InverseMatrix[A\_]$ (вж. изпратения файл LU.nb), като използвате вградената в Mathematica функция $TimeUsed$.

Задача 22. Решете задачи 9,11, като напишете подходящи програми в системата Mathematica.

Задача 23. По метода на най-малките квадрати да се намери подходяща функция, която приближава таблицата

x	3	4	5	7	8	9	11	12
y	1.6	3.6	4.4	3.4	2.2	2.8	3.8	4.6

Да се илюстрира графично.

Глава 3

Итерационни методи за решаване на системи линейни алгебрични уравнения

Основен проблем на директните методи, които разгледахме за решаване на системата

$$A\overline{x} = \overline{b},$$

е тяхното бързодействие. За големи системи уравнения тяхното прилагане е нецелесъобразно, а често пъти – и невъзможно. На практика, от друга страна, възникват системи с десетки хиляди и дори милиони уравнения. За системи с голяма размерност обикновено се използват т.нар. итерационни методи. Идеята при тях е следната. Започвайки с дадено начално приближение $\overline{x}_0$, построяваме редицата $\overline{x}_0, \overline{x}_1, \overline{x}_2, \ldots$ от последователни приближения, която да бъде сходяща към решението на системата. Тогава очевидно е важен въпросът за **сходимостта на съответните числени методи**.

3.1 Метод на простата итерация

Нека е дадена системата

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ & & \cdots\cdots\cdots\cdots\cdots & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}$$

Ако от първото уравнение изразим x_1, от второто – x_2 и т.н. получаваме еквивалентната система

$$x_i = -\sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j + \frac{b_i}{a_{ii}}, \quad i = \overline{1, n}. \tag{3.1}$$

Ако имаме дадено приближение, заместваики с него в дясната страна на (3.1) и използвайки съответните равенства, ще получим нови стойности на $x_1, \ldots, x_n$.

И така, формулите за построяване на редицата от последователни приближения имат вида

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}, \quad i = \overline{1,n}. \tag{3.2}$$

Задача 24. Да се приведе системата

$$\begin{array}{rcrcrcl} 4x_1 & - & x_2 & + & x_3 & = & 12, \\ -x_1 & + & 4x_2 & - & 2x_3 & = & -1, \\ x_1 & - & 2x_2 & + & 4x_3 & = & 5 \end{array} \tag{3.3}$$

във вид, удобен за прилагане на метода на простата итерация. Да се направят две итерации, започвайки с начално приближение $\overline{x}_0 = (0,0,0)^T$.

Решение. От първото уравнение изразяваме x_1 чрез другите две неизвестни, от второто – x_2, а от третото – x_3. Получаваме еквивалентната система

$$\begin{array}{rcrcrcl} x_1 & = & \frac{1}{4}x_2 & - & \frac{1}{4}x_3 & + & 3, \\ x_2 & = & \frac{1}{4}x_1 & + & \frac{1}{2}x_3 & - & \frac{1}{4}, \\ x_3 & = & -\frac{1}{4}x_1 & + & \frac{1}{2}x_2 & + & \frac{5}{4}. \end{array}$$

Тогава итерационният процес се построява по формулите

$$\begin{array}{rcrcrcl} x_1^{(k+1)} & = & \frac{1}{4}x_2^{(k)} & - & \frac{1}{4}x_3^{(k)} & + & 3, \\ x_2^{(k+1)} & = & \frac{1}{4}x_1^{(k)} & + & \frac{1}{2}x_3^{(k)} & - & \frac{1}{4}, \\ x_3^{(k+1)} & = & -\frac{1}{4}x_1^{(k)} & + & \frac{1}{2}x_2^{(k)} & + & \frac{5}{4}. \end{array} \tag{3.4}$$

Тогава, започвайки от началното приближение $\overline{x}_0 = (0,0,0)^T$, получаваме

$$\begin{aligned} x_1^{(1)} &= \frac{1}{4} \times 0 - \frac{1}{4} \times 0 + 3 = 3, \\ x_2^{(1)} &= \frac{1}{4} \times 0 + \frac{1}{2} \times 0 - \frac{1}{4} = -\frac{1}{4}, \\ x_3^{(1)} &= -\frac{1}{4} \times 0 + \frac{1}{2} \times 0 + \frac{5}{4} = \frac{5}{4}. \end{aligned}$$

За второто приближение $\overline{x}_2$ получаваме

$$\begin{aligned} x_1^{(2)} &= \frac{1}{4} \times \left(-\frac{1}{4}\right) - \frac{1}{4} \times \frac{5}{4} + 3 = \frac{21}{8}, \\ x_2^{(2)} &= \frac{1}{4} \times 3 + \frac{1}{2} \times \frac{5}{4} - \frac{1}{4} = \frac{9}{8}, \\ x_3^{(2)} &= -\frac{1}{4} \times 3 + \frac{1}{2} \times \left(-\frac{1}{4}\right) + \frac{5}{4} = \frac{3}{8}. \end{aligned}$$

□

Разбира се, възниква въпросът за сходимостта на така построения итерационен процес. Отговор на този въпрос дава следното твърдение.

Твърдение

Итерационният процес

$$\overline{x}_{k+1} = B\overline{x}_k + \overline{d} \tag{3.5}$$

е сходящ за произволно начално приближение тогава и само тогава, когато всички собствени стойности на матрицата B са по модул по малки от 1.

Задача 25. Да се изследва сходимостта на метода на простата итерация за системата (3.3).

Решение. Първо, да запишем итерационния процес във вида (3.5). Използвайки (3.4), получаваме

$$\overline{x}_{k+1} = \underbrace{\begin{bmatrix} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & 0 \end{bmatrix}}_{B} \overline{x}_k + \underbrace{\begin{bmatrix} 3 \\ -\frac{1}{4} \\ \frac{5}{4} \end{bmatrix}}_{\overline{d}}$$

Собствените стойности на матрицата B са решенията на уравнението

$$\det(B - \lambda I) = \begin{vmatrix} -\lambda & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\lambda & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\lambda \end{vmatrix} = 0,$$

т.е. са $1/2$ и $(-1 \pm \sqrt{3})/4$ и са по модул по-малки от 1. Следователно за произволно начално приближение методът на простата итерация е сходящ за дадената система. □

Задача 26. Да се намерят стойностите на реалния параметър a, за които методът на простата итерация е сходящ при всяко начално приближение за системата

$$\begin{aligned} ax + y &= 1, \\ x + y + z &= 1, \\ y + az &= 1. \end{aligned}$$

Решение. Ясно е, че при $a = 0$ системата е неопределена. Нека $a \neq 0$. Отново ще запишем итерационния процес във вида (3.5). Имаме

$$\begin{aligned} x^{(k+1)} &= -\tfrac{1}{a}y^{(k)} + \tfrac{1}{a}, \\ y^{(k+1)} &= -x^{(k)} - z^{(k)} + 1, \\ z^{(k+1)} &= -\tfrac{1}{a}y^{(k)} + \tfrac{1}{a}, \end{aligned}$$

т.е. матрицата B има вида

$$B = \begin{bmatrix} 0 & -\frac{1}{a} & 0 \\ -1 & 0 & -1 \\ 0 & -\frac{1}{a} & 0 \end{bmatrix}.$$

Нейният характеристичен полином е

$$\det(B-\lambda I)=\begin{vmatrix} -\lambda & -\frac{1}{a} & 0 \\ -1 & -\lambda & -1 \\ 0 & -\frac{1}{a} & -\lambda \end{vmatrix} = -\lambda\left(\lambda^2-\frac{2}{a}\right).$$

Следователно собствените стойности на B са $\lambda_1=0$, и $\lambda_{2,3}=\pm\sqrt{\frac{2}{a}}$.

- Първи случай: $a>0$. Ясно е, че в този случай условието за сходимост е изпълнено т.с.т.к $a>2$.
- Втори случай: $a<0$. Имаме $|\pm\sqrt{2/a}|=|i\sqrt{2/-a}|<1 \iff 2/-a<1$, т.е. $a<-2$.

Окончателно получихме, че методът е сходящ при произволно начално приближение, точно когато $a\in(-\infty,-2)\cup(2,+\infty)$. □

Теоретично, точното решение "се достига" при $n\to\infty$. Разбира се, на практика ние не можем да направим безброй много итерации. Възниква необходимостта от **критерий, по който да спираме разглеждания итерационен процес**, когато сме достигнали решение, което е достатъчно близо (в някакъв смисъл) до точното решение.

Говорейки за "близо", ясно е, че този критерий трябва да е свързан с понятията норма и разстояние. Да припомним някои често използвани норми и породените от тях разстояния. Нека $x\in\mathbb{R}^n$. Тогава равномерната (максимум) норма и породеното от нея разстояние се дефинират с

$$\|\overline{x}\|_\infty=\max_{i=\overline{1,n}}|x_i|,\quad \rho(\overline{x},\overline{y})=\|\overline{x}-\overline{y}\|_\infty=\max_{i=\overline{1,n}}|x_i-y_i|.$$

Евклидовата норма и евклидовото разстояние са съответно

$$\|\overline{x}\|_2=\sqrt{x_1^2+\cdots+x_n^2},\quad \rho(\overline{x},\overline{y})=\|\overline{x}-\overline{y}\|_2=\left\{\sum_{i=1}^n(x_i-y_i)^2\right\}^{1/2}.$$

Тогава можем да спираме итерационния процес, когато две последователни приближения са достатъчно близо едно до друго в смисъла на някое разстояние (обикновено се използва относително разстояние), т.е. когато

$$\varepsilon_r:=\frac{\|\overline{x}^{(k+1)}-\overline{x}^{(k)}\|}{\|\overline{x}^{(k)}\|}<\varepsilon_{max},$$

където ε_{max} е отнапред зададена желана точност. Ако правенето на нови итерации не променя съществено резултата, можем да считаме, че сме стигнали с достатъчно добра точност до точното решение.

Вече сме готови да разгледаме имплементацията на метода в Mathematica:

```
A = {{4, -1, 1}, {-1, 4, -2}, {1, -2, 4}} // N;
b = {12, -1, 5} // N;
n = Length[A];
ϵ = 0.001;
maxIter = 100;
iter = 1;
xOld = Table[0, {i, n}];
xNew = Table[0, {i, n}];
For[i = 1, i ≤ n, i++, (*Compute the first approximation x1*)
  xNew[[i]] = 1/A[[i, i]] (-Sum[A[[i, j]] xOld[[j]], {j, 1, i - 1}] -
      Sum[A[[i, j]] xOld[[j]], {j, i + 1, n}]) + b[[i]]/A[[i, i]]
 ];
(*Compute the successive approximations
 until the desired accuracy is reached or
 the maximum number of iterations is reached*)
While[Norm[xNew - xOld] / Norm[xNew] ≥ ϵ && iter < maxIter,
 xOld = xNew;
 For[i = 1, i ≤ n, i++,
  xNew[[i]] = 1/A[[i, i]] (-Sum[A[[i, j]] xOld[[j]], {j, 1, i - 1}] -
      Sum[A[[i, j]] xOld[[j]], {j, i + 1, n}]) + b[[i]]/A[[i, i]]
 ];
 iter++
]
xNew
iter
```

Резултатът от изпълнението на горния код е

```
Out[11]= {3.00045, 0.999379, 1.00062}

Out[12]= 19
```

Желаната точност е постигната за 19 итерации. Намереното решение с точност до четвъртия знак след десетичната запетая е $\overline{x} = (3.0005, 0.9994, 1.0006)^T$. Ако сравним с точното решение, което е $\overline{\xi} = (3, 1, 1)^T$, виждаме, че получената точност действително отговаря на зададената (0.1%).

Задача 27. Да се изследва сходимостта на метода на простата итерация за системата

$$\begin{aligned} x_1 &- 5x_2 = -4, \\ 7x_1 &- x_2 = 6. \end{aligned}$$

на базата на Твърдение 3.5. Да се направят първите 20 итерации на метода на простата итерация върху тази система при начално приближение $\overline{x}_0 = (0, 0)^T$.

Решение. Проверката на условието оставяме за самостоятелна работа. Може да се покаже, че то не е изпълнено, т.е. не за всяко начално приближение методът

ще бъде сходящ. Обърнете внимание, че в този случай твърдението не ни дава отговор на въпроса дали съществуват начални приближения, за които методът е сходящ, и кои са те.

За началното приближение $\overline{x}_0 = (0,0)^T$, използвайки вече реализираната програма със съответните входни данни, получаваме

```
Out[59]= {-2.75855 × 10^15, -2.75855 × 10^15}
```

Резултатът показва, че методът е разходящ.

□

3.2 Метод на Зайдел

Методът на Зайдел е модификация на метода на простата итерация. При него на $k+1$-вата итерация приближението се намира по формулите

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}, \quad i = \overline{1,n}. \tag{3.6}$$

С други думи, се използват и всички вече намерени на $k+1$-вата стъпка координати на вектора $\overline{x}_{k+1}$, т.е. "най-добрата" в дадения момент информация за съответните координати.

Имплементацията в Mathematica е следната:

```
A = {{4, -1, 1}, {-1, 4, -2}, {1, -2, 4}} // N;
b = {12, -1, 5} // N;
n = Length[A];
ϵ = 0.001;
maxIter = 100;
iter = 1;
xOld = Table[0, {i, n}];
xNew = Table[0, {i, n}];
For[i = 1, i ≤ n, i++,
  xNew[[i]] = 1/A[[i, i]] (-Sum[A[[i, j]] xOld[[j]], {j, 1, i - 1}] -
      Sum[A[[i, j]] xOld[[j]], {j, i + 1, n}]) + b[[i]]/A[[i, i]]
 ];
While[Norm[xNew - xOld] / Norm[xNew] ≥ ϵ && iter < maxIter,
 xOld = xNew;
 For[i = 1, i ≤ n, i++,
  xNew[[i]] = 1/A[[i, i]] (-Sum[A[[i, j]] xNew[[j]], {j, 1, i - 1}] -
      Sum[A[[i, j]] xOld[[j]], {j, i + 1, n}]) + b[[i]]/A[[i, i]]
 ];
 iter++
]
xNew
iter
```

Резултатът от изпълнението на горния код е следният:

```
Out[35]= {3.00015, 1.00017, 1.00005}

Out[36]= 7
```

Желаната точност е постигната за 7 итерации, което е по-бързо от метода на простата итерация.

3.3 Метод на спрегнатия градиент

Глава 4

Някои практически въпроси, свързани с прилагането на числените методи за решаване на линейни алгебрични системи

4.1 Сравнение на бързодействието на методите

Нека разгледаме системата $Ax = b$, където матрицата A е от ред n и има вида

$$A = \begin{bmatrix} 2.2 & 1 & & & & & \\ 1 & 2.2 & 1 & & & & \\ & 1 & 2.2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 2.2 & 1 & \\ & & & & 1 & 2.2 & \end{bmatrix},$$

а дясната страна е $b = (1, 1, \ldots, 1)^T$.

Матрицата A е избрана така, че да могат да бъдат приложени всички изучени методи, с цел да се направи сравнение на тяхното бързодействие за системи с различна размерност n. Времето в секунди, за което всеки от тях решава системата, е дадено в таблицата по-долу. Всички числени експерименти са направени в системата Wolfram Mathematica 8 на персонален компютър с процесор Intel Core i5-6200U, 2.4GHz.

n	Гаус (с част. избор) $2/3n^3 + O(n^2)$	Холецки $1/3n^3 + O(n^2)$	Прогонка $O(n)$	Якоби $O(n^2)$	Зайдел $O(n^2)$
50	0.14	0.062	<0.01	0.125 (51 ит.)	0.047 (17 ит.)
100	1.078	0.328	<0.01	0.766 (79 ит.)	0.14 (15 ит.)
200	8.422	2.328	<0.01	2.937 (80 ит.)	0.516 (14 ит.)
400	66.297	18.172	<0.01	11.985 (80 ит.)	1.734 (12 ит.)
800	613.594	147.25	0.015	47.238 (81 ит.)	6.234(11 ит.)
1600	над 1 час	1379.73	0.03	190.766 (81 ит.)	23.078 (10 ит.)

Както може да се очаква, методът на дясната прогонка е чувствително по-бърз от останалите, което го прави приложим за системи с много голяма размерност, имащи подходяща структура.

Останалите методи обаче имат сравнимо бързодействие за системи с малка размерност. За $n = 50, 100, 200$ методът на Холецки е дори по-бърз от метода на Якоби, който има сложност от по-нисък ред.

За системи с малка размерност директните методи са за предпочитане. Те имат сравнимо бързодействие, но са значително по-надеждни. **За системи с голяма размерност обаче виждаме, че итерационните методи имат съществено преимущество.** Това може да се обясни с факта, че броят итерации не се променя много при увеличаване размерността на системата.

4.2 Число на обусловеност. Априорни и апостериорни оценки на грешката.

Както видяхме, неустойчивостта на даден метод може да доведе до съществени грешки в крайния резултат. Причината за това е, че алгоритъмът "увеличава" грешките от закръгляване. Използването на друг, устойчив, метод ще доведе до получаването на верен резултат.

Съществува обаче и друга причина, поради която резултатите от числено решаване на дадена линейна система могат да бъдат лоши. Това е т.нар. обусловеност на задачата. Това понятие е свързано с грешката, до която грешката във входните данни може да доведе в резултата.

Ще изведемм оценка на грешката при решаването на една линейна алгебрична система, ако променим малко входните данни. Нека x е решението на оригиналната система, а $\hat{x}$ е решението на системата с променени входни данни, т.е.

$$\begin{aligned}(A + \Delta A)\hat{x} &= b,\\ Ax &= b.\end{aligned}$$

Вадейки второто уравнение от първото, за грешката получаваме

$$\begin{aligned}A(\hat{x} - x) &= -\Delta A\hat{x},\\ \hat{x} - x &= -A^{-1}\Delta A\hat{x}.\end{aligned}$$

Тогава за абсолютната грешка, измерена в дадена норма, е в сила

$$\varepsilon_a := \|\hat{x} - x\| = \|A^{-1}A\hat{x}\| \leq \|A^{-1}\|\|A\|\|\hat{x}\|.$$

За да изразим относителната грешка, делим двете страни на $\|\hat{x}\|$ и получаваме

$$\varepsilon_r := \frac{\|x - \hat{x}\|}{\|\hat{x}\|} \leq \underbrace{\|A^{-1}\|\|A\|}_{cond(A)} \frac{\|\Delta A\|}{\|A\|}. \tag{4.1}$$

И така, виждаме от горната оценка, че дори незначителна промяна във входните данни, $\|\Delta A\|/\|A\|$, може да доведе до голяма грешка в резултата, ако $cond(A)$ е голямо. Числото $cond(A)$ се нарича число на обусловеност за матрицата A.

Да обърнем внимание, че обусловеността е свойство на решаваната задача, а не на метода, с който я решаваме.

Да пресметнем числото на обусловеност за матрицата на Хилберт от ред 20:

```
H = HilbertMatrix[20];
N[Norm[Inverse[H], 2] Norm[H, 2]]
```

Out[10]= 2.45216×10^{28}

Вземайки предвид последния резултат и оценката (4.1), можем да очакваме, че работейки в компютърна аритметика, решаването на система с матрица на Хилберт, ще бъде лошо обусловена задача и получените грешки ще са съществени. За да илюстрираме този факт, нека първо решим системата $Hx = b$ за $b = (1, 1, \ldots, 1)^T$ символно, използвайки вградената функция LinearSolve в Mathematica:

In[18]:=
```
b = Table[1, {20}];
LinearSolve[H, b]
```

Out[19]=
```
{-20, 7980, -790 020, 34 321 980, -823 727 520, 12 355 912 800,
 -124 932 007 200, 894 921 112 800, -4 698 335 842 200, 18 503 322 637 800,
 -55 509 967 913 400, 127 994 058 246 600, -227 544 992 438 400,
 311 023 037 001 600, -323 717 854 838 400, 251 780 553 763 200,
 -141 626 561 491 800, 54 396 360 988 200, -12 759 640 231 800, 1 378 465 288 200}
```

Да обърнем внимание, че тъй като не използваме числа с плаваща точка във входните данни, а работим символно, Mathematica намира точното решение на системата.

Нека сега решим същата система, като единствено по главния диагонал на матрицата H добави 10^{-15}, което е сравнимо с машинната грешка:

In[22]:=
```
H1 = H + DiagonalMatrix[Table[10^-15, {20}]];
LinearSolve[H1, b] // N
```

Out[23]=
```
{4.18461, -428.767, 9520.28, -66 286.5, -19 407.5, 2.19682 × 10^6,
 -1.01854 × 10^7, 1.79997 × 10^7, -4.97211 × 10^6, -1.87141 × 10^7,
 3.77987 × 10^6, 2.21275 × 10^7, 7.80635 × 10^6, -1.97735 × 10^7, -2.34981 × 10^7,
 5.52176 × 10^6, 3.27829 × 10^7, 1.19906 × 10^7, -4.52997 × 10^7, 1.83143 × 10^7}
```

Отново сме работили символно, т.е. Mathematica е върнала точното решение на променената система. Виждаме, че решението няма нищо общо с решението на системата $Hx = b$. От друга страна, ние сме променили входните данни със стойности, сравними с машинната грешка, т.е. такива грешки са неизбежни заради закръгляването. Тогава можем да очакваме, че числено решаване на системата $Hx = b$ няма да бъде добро. Действително:

In[13]:=
```
LinearSolve[N[H], Table[1, {20}]]
```

```
LinearSolve::luc : Result for LinearSolve of badly conditioned matrix {≪1≫} may contain significant numerical error
```

Out[13]=
```
{-9.92855, 1073.33, -21 305.6, -18 047.6, 3.8623 × 10^6,
 -4.18194 × 10^7, 2.09821 × 10^8, -5.51336 × 10^8, 6.27157 × 10^8, 3.06172 × 10^8,
 -1.50994 × 10^9, 6.47086 × 10^8, 1.62751 × 10^9, -8.38695 × 10^8, -3.11475 × 10^9,
 4.47977 × 10^9, -1.8183 × 10^9, -4.77868 × 10^8, 5.73674 × 10^8, -1.22314 × 10^8}
```

Работейки числено, Mathematica връща съвършено различен резултат, като извежда съобщение, че системата е лошо обусловена.

Ако задачата е лошо обусловена, от нито един числен метод не може да очакваме добър резултат. Численият метод работи със закръглените входни данни и дори да реши абсолютно точно тази задача, резултатът можем да очакваме да е съществено различен от решението на оригиналната задача (т.е. без да са закръглени входните данни). Ето защо, единствената възможност е задачата да се запише в еквивалентна форма, в която ще бъде по-добре

обусловена.

Глава 5

Числени методи за намиране на собствени стойности и собствени вектори на матрица

Дефиниция

Казваме, че числото λ е собствена стойност на матрицата A, ако съществува вектор $\overline{x}$, за който $A\overline{x} = \lambda\overline{x}$, т.е. изображението на вектора $\overline{x}$ е вектор, колинеарен на дадения.

Твърдение

Собствените стойности на матрицата A са решенията на уравнението $\det(A - \lambda I) = 0$, където I е единичната матрица.

Доказателство. От $\overline{x} = \lambda\overline{x}$ следва, че хомогенната система $(A - \lambda I)\overline{x} = 0$ има поне едно ненулево решение, $\overline{x}$, и следователно матрицата $A - \lambda I$ е особена, т.е. има нулева детерминанта. □

Собствените стойности са съществена характеристика на линейния оператор, задаващ се със съответната матрица. Също така, тяхното намиране е свързано с редица важни задачи в математиката, в т.ч. някои разлагания на матрици, решаване на линейни ОДУ, изследвания за устойчивост, обусловеност и др.

5.1 Метод на Данилевски

Методът на Данилевски е директен метод за намиране на собствените стойности на дадена матрица.

Основна идея, на която се базират директните методи, е да приведем оригиналната матрица в такава, за която задачата (в случая за намиране на собствените стойности) е лесно решима. При това трябва да извършваме такива преобразования, че задачата за оригиналната матрица и тази, получена след преобразованията, да имат едни и същи решения.

Ако една матрица е в нормална форма на Фробениус, т.е.

$$A = \begin{bmatrix} p_1 & p_2 & \cdots & p_{n-1} & p_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix},$$

то нейният характеристичен полином може лесно да бъде намерен, като развием $\det(A - \lambda I)$ по първия ред, и собствените стойности на A са решенията на уравнението

$$p(x) \equiv \lambda^n - p_1\lambda^{n-1} - \cdots - p_n = 0.$$

Ще илюстрираме метода със следния пример.

Задача 28. Да се намерят собствените стойности на матрицата

$$A = \begin{bmatrix} -1 & 3 & -2 \\ 1 & 1 & -1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Решение. На първата стъпка ще приведем последния ред във вид, съответстващ на нормалната форма на Фробениус. За тази цел втория стълб ще прибавим, умножен с -1, към първия. Това преобразование съответства на умножението отдясно на матрицата A с матрицата

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

За да бъде това преобразование на подобие, трябва да умножим отляво с нейната обратна

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Така, след първата стъпка получаваме

$$A^{(1)} = M_1^{-1}AM_1 = \begin{bmatrix} -4 & 3 & -2 \\ -4 & 4 & -3 \\ 0 & 1 & 0 \end{bmatrix}.$$

На втората стъпка остава да приведем и втория ред на матрицата в подходящия вид. За тази цел прибавяме първия стълб към втория, прибавяме първия стълб, умножен с -3/4 към третия и делим първия стълб на -4. Това преобразование можем да извършим, като умножим матрицата $A^{(1)}$ отдясно с

$$M_2 = \begin{bmatrix} -1/4 & 1 & -3/4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

За да направим преобразование на подобие, отляво умножаваме с обратната матрица

$$M_2^{-1} = \begin{bmatrix} -4 & 4 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Окончателно получаваме матрицата

$$A^{(2)} = M_2^{-1} A^{(1)} M_2 = \begin{bmatrix} 0 & 1 & -4 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

която е подобна на A, т.е. има същите собствени стойности като нея, и е в нормална форма на Фробениус. Нейният характеристичен полином е $\lambda^3 - \lambda + 4$ и следователно собствените стойности на A са 1.79632 и $0.898161 \pm 1.19167i$. □

Привеждаме примерна имплементация на метода на Данилевски.

```
In[1]:= A = {{-1, 3, -2}, {1, 1, -1}, {1, 1, 0}};
        n = Length[A];
        For[i = n, i ≥ 2, i--,
         M = IdentityMatrix[n];
         For[j = 1, j ≤ i - 2, j++,
          M[[i - 1, j]] = -A[[i, j]] / A[[i, i - 1]];
         ];
         For[j = i, j ≤ n, j++,
          M[[i - 1, j]] = -A[[i, j]] / A[[i, i - 1]];
         ];
         M[[i - 1, i - 1]] = 1 / A[[i, i - 1]];
         A = Inverse[M].A.M
        ]
        p[λ_] = λ^n;
        For[i = 1, i ≤ n, i++,
         p[λ_] = p[λ] - A[[1, i]] λ^(n-i)
        ]
        p[λ]
```

5.2 Методи, свързани с подпространствата на Крилов

Нека A е матрица $n \times n$ и нека $\mathbf{x}_0 \in \mathbb{R}^n$. Линейното подпространство на $\mathbb{R}^n$, породено от векторите $\mathbf{x}_0, A\mathbf{x}_0, \ldots, A^k\mathbf{x}_0 \quad (k \leq n)$, се нарича подпространство на Крилов от ред k. Ще означаваме

$$\mathcal{K}_k := span\{\mathbf{x}_0, A\mathbf{x}_0, \ldots, A^k\mathbf{x}_0\}.$$

Нека векторите $\mathbf{x}_0, \ldots, A^m\mathbf{x}_0$ са линейно-независими за някое $m < n$ и нека $\mathcal{K}_{m+1} = \mathcal{K}_m$.

Правейки едни и същи преобразования с базисните елементи на $\mathcal{K}_k$ и π_k, $k \leq m$ ще получаваме съответстващи си елементи на двете пространства (т.е. с едни и същи координати по отношение на тези базиси).

Ясно е, че векторите $\mathbf{x}_0, \ldots, A^{m+1}\mathbf{x}_0$ са линейно-зависими. Може да се докаже, че ако

$$A^{m+1}\mathbf{x}_0 + \alpha_1 A^m \mathbf{x}_0 + \cdots + \alpha_{m+1}\mathbf{x}_0 = 0, \tag{5.1}$$

тогава решенията на уравнението

$$\lambda^{m+1} + \alpha_1 \lambda^m + \cdots + \alpha_{m+1} = 0 \tag{5.2}$$

са собствени стойности на A. В частност, ако $m+1 = n$, тогава те са всички собствени стойности на A.

5.2.1 Метод на Крилов

Методът на Крилов имплементира непосредствено казаното дотук. Ако намерим линейната комбинация (5.1), която е равна на 0, то ние автоматично сме намерили делител на характеристичния полином – лявата страна на (5.2). Начинът, по който това се имплементира на практика, илюстрираме със следващия пример.

Задача 29. Нека

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

По метода на Крилов да се намери характеристичния полином (или негов делител), като се избере $\mathbf{x}_0 = (1,1,0)^T$.

Решение. Разглеждаме векторите $\mathbf{x}_0, A\mathbf{x}_0, A^2\mathbf{x}_0, A^3\mathbf{x}_0$, където

$$A\mathbf{x}_0 = (1,0,-1)^T,\ A^2\mathbf{x}_0 = (2,-1,-1)^T, A^3\mathbf{x}_0 = (3,-3,0)^T.$$

Търсим тяхната линейна комбинация, която е равна на 0. Тогава съответната линейна комбинация на $1, \lambda, \lambda^2, \lambda^3$ ще е търсеният характеристичен полином. За тази цел разглеждаме матрицата

$$\left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 1 & 0 & -1 & \lambda \\ 2 & -1 & -1 & \lambda^2 \\ 3 & -3 & 0 & \lambda^3 \end{array}\right]. \tag{5.3}$$

Правейки преобразования с лявата ѝ страна така, че в последния ред да получим нулевия вектор, то същите преобразования в дясната страна ще ни дадат характеристичния полином, който търсим.

Получаваме последователно

$$\left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 1 & 0 & -1 & \lambda \\ 2 & -1 & -1 & \lambda^2 \\ 3 & -3 & 0 & \lambda^3 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 0 & -1 & -1 & \lambda - 1 \\ 0 & -3 & -1 & \lambda^2 - 2 \\ 0 & -6 & 0 & \lambda^3 - 3 \end{array}\right]$$

$$\to \left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 0 & -1 & -1 & \lambda - 1 \\ 0 & 0 & 2 & \lambda^2 - 3\lambda + 1 \\ 0 & 0 & 6 & \lambda^3 - 6\lambda + 3 \end{array}\right]$$

$$\to \left[\begin{array}{ccc|c} 1 & 1 & 0 & 1 \\ 0 & -1 & -1 & \lambda - 1 \\ 0 & -3 & -1 & \lambda^2 - 2 \\ 0 & 0 & 0 & \lambda^3 - 3\lambda^2 + 3\lambda \end{array}\right].$$

Следователно характеристичният полином на A е $\lambda^3 - 3\lambda^2 + 3\lambda$. □

Забележка. Ако рангът на матрицата е по-малък от n, тогава ще получим делител на характеристичния полином. Другите делители ще получим, като започнем с друго начално приближение.

5.2.2 Метод на Ланцош (метод на ортогонализацията за симетрични матрици)

Ако разгледаме по-внимателно операциите, които направихме с матрицата (5.3), можем да забележим следното. Ако разглеждаме операциите само в първите два реда на матрицата, това са всъщност линейни операции в подпространството на Крилов $\mathcal{K}_1$ (в лявата страна) и съответни операции в пространството от линейните полиноми π_1 (в дясната страна). Операциите в първите три реда са всъщност линейни операции в $\mathcal{K}_2$ и π_2 и т.н. Пространствата са зададени със следните базисни елементи, чиито линейни комбинации ние намираме – $\mathcal{K}_k$ е зададено с $\mathbf{x}_0, A\mathbf{x}_0, \ldots, A^k\mathbf{x}_0$, а π_k е зададено с базиса $1, \lambda, \ldots, \lambda^k$. Разбира се, ние можем да работим с други базиси на пространствата $\mathcal{K}_k$ и със съответните базиси на π_k. Най-добрият избор, разбира се, би бил да вземем ортогонални базиси. **Основното предимство е, че това прави алгоритъма по-устойчив от числена гледна точка**.

Ще илюстрираме горната идея на база на същия пример, който използвахме за метода на Крилов.

Задача 30. Нека

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

По метода на Ланцош да се намери характеристичния полином (или негов делител) при начален вектор $\mathbf{x}_0 = (1, 1, 0)^T$.

Решение. Последователно ще ортогонализираме базисите на подпространствата на Крилов и ще намерим съответните елементи на полиномиалните пространства.

1. Пространството на Крилов $\mathcal{K}_0$ се определя от $\mathbf{x}_0$, съответно пространството π_0 е определено от константния полином 1. Тук имаме само по един елемент в базисите и няма какво да ортогонализираме.

2. Имаме $\mathcal{K}_1 = span\{\mathbf{x}_0, A\mathbf{x}_0\}$, съответно $\pi_1 = span\{1, \lambda\}$. Имаме $A\mathbf{x}_0 = (1, 0, -1)^T$. Ортогонализираме базиса, като избираме вектор $\mathbf{x}_1 = A\mathbf{x}_0 + \alpha\mathbf{x}_0$, който е ортогонален на $\mathbf{x}_0$:

$$(A\mathbf{x}_0 + \alpha\mathbf{x}_0, \mathbf{x}_0) = 0.$$

Следователно

$$\alpha = -\frac{(A\mathbf{x}_0, \mathbf{x}_0)}{(\mathbf{x}_0, \mathbf{x}_0)} = -\frac{1}{2},$$

т.е.

$$\mathbf{x}_1 = (1, 0, -1)^T - \frac{1}{2}(1, 1, 0)^T = \left(\frac{1}{2}, -\frac{1}{2}, -1\right)^T.$$

Така получихме ортогонален базис:

$$\mathcal{K}_1 = span\left\{\mathbf{x}_0 = (1, 1, 0)^T, \mathbf{x}_1 = \left(\frac{1}{2}, -\frac{1}{2}, -1\right)^T\right\}.$$

Съответстващият втори базисен елемент на е $\pi_1 = \lambda + \alpha.1$. Тогава получаваме следното съответстващо представяне:

$$\pi_1 = span\left\{1, \lambda - \frac{1}{2}\right\}.$$

3. Имаме $\mathcal{K}_2 = span\{\mathbf{x}_0, \mathbf{x}_1, A\mathbf{x}_1\}$ и съответно $\pi_2 = span\left\{1, \lambda - \frac{1}{2}, \lambda^2 - \frac{1}{2}\lambda\right\}$. Имаме $A\mathbf{x}_1 = \left(\frac{3}{2}, -1, -\frac{1}{2}\right)^T$. Ортогонализираме базиса, като избираме вектор

$$\mathbf{x}_2 = A\mathbf{x}_1 + \alpha\mathbf{x}_1 + \beta\mathbf{x}_0,$$

който да е ортогонален на първите два:

$$(\mathbf{x}_0, \mathbf{x}_2) = 0 \Longrightarrow \beta = -\frac{(A\mathbf{x}_1, \mathbf{x}_0)}{(\mathbf{x}_0, \mathbf{x}_0)} = -\frac{1}{4},$$
$$(\mathbf{x}_1, \mathbf{x}_2) = 0 \Longrightarrow \alpha = -\frac{(A\mathbf{x}_1, \mathbf{x}_1)}{(\mathbf{x}_1, \mathbf{x}_1)} = -\frac{7}{6},$$

т.е. $\mathbf{x}_2 = \left(\frac{2}{3}, -\frac{2}{3}, \frac{2}{3}\right)^T$. Съответстващият трети базисен елемент на π_2 е

$$\left(\lambda^2 - \frac{1}{2}\lambda\right) + \alpha\left(\lambda - \frac{1}{2}\right) + \beta.1 = \lambda^2 - \frac{5}{3}\lambda + \frac{1}{3}.$$

4. Имаме $\mathcal{K}_3 = span\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, A\mathbf{x}_2\}$. Да обърнем внимание, че четирите вектора в последното са линейно-зависими, т.е. при ортогонализацията ще получим линейна комбинация, която е равна на нула (а точно това е целта ни! – вж. (5.1) и (5.2)). Имаме $A\mathbf{x}_2 = \left(0, -\frac{4}{3}, \frac{4}{3}\right)^T$.

Търсим

$$\mathbf{x}_3 = A\mathbf{x}_2 + \alpha\mathbf{x}_2 + \beta\mathbf{x}_1 + \gamma\mathbf{x}_0,$$

ортогонален на първите три (забележете, че това е задължително нулевия вектор, тъй като това е вектор от $\mathbb{R}^3$, който е ортогонален на $\mathcal{K}_2 \equiv \mathbb{R}^3$):

$$\alpha = -\frac{(A\mathbf{x}_2, \mathbf{x}_2)}{(\mathbf{x}_2, \mathbf{x}_2)} = -\frac{4}{3},\ \beta = -\frac{(A\mathbf{x}_2, \mathbf{x}_1)}{(\mathbf{x}_1, \mathbf{x}_1)} = \frac{4}{9},\ \gamma = -\frac{(A\mathbf{x}_2, \mathbf{x}_0)}{(\mathbf{x}_0, \mathbf{x}_0)} = \frac{2}{3}.$$

Тогава търсеният характеристичен полином е

$$\left(\lambda^3 - \frac{5}{3}\lambda^2 + \frac{1}{3}\lambda\right) - \frac{4}{3}\left(\lambda^2 - \frac{5}{3}\lambda + \frac{1}{3}\right) + \frac{4}{9}\left(\lambda - \frac{1}{2}\right) + \frac{2}{3} = \lambda^3 - 3\lambda^2 + 3\lambda.$$

□

Изхождайки от последния пример е ясно, че на k-тата стъпка от алгоритъма търсим $k-1$ коефициента:

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{x}_{k-1} + \cdots + \alpha_0\mathbf{x}_0,$$

където

$$\alpha_j = -\frac{(A\mathbf{x}_{k-1}, \mathbf{x}_j)}{(\mathbf{x}_j, \mathbf{x}_j)},\quad j = \overline{0, k-1}.$$

Оказва се, че ако матрицата е симетрична, нещата се опростяват значително, тъй като може да се докаже, че $\alpha_0 = \cdots = \alpha_{k-3} = 0$. Ще приведем примерна имплементация на метода за симетрични матрици. По-точно, ще напишем скрипт, който по подаден начален вектор намира характеристичния полином или негов делител

```
In[1]:= A = {{1, 0, -1}, {0, 1, 0}, {-1, 0, 1}} // N;
        xPrev = {{1}, {1}, {0}};
        polyPrev = 1;
        α = -Transpose[xPrev].(A.xPrev) / Transpose[xPrev].xPrev // Flatten;
        xCurr = A.xPrev + α[[1]] xPrev ;
        polyCurr = λ + α[[1]] polyPrev;
        iter = 1;
        While[Norm[xCurr] > 0.00001,
         xPrevPrev = xPrev;
         polyPrevPrev = polyPrev;
         xPrev = xCurr;
         polyPrev = polyCurr;
         α = -Transpose[xPrev].(A.xPrev) / Transpose[xPrev].xPrev // Flatten;
         β = -Transpose[xPrevPrev].(A.xPrev) / Transpose[xPrevPrev].xPrevPrev // Flatten;
         xCurr = A.xPrev + α[[1]] xPrev + β[[1]] xPrevPrev;
         polyCurr = λ * polyPrev + α[[1]] polyPrev + β[[1]] polyPrevPrev // Expand;
         iter++
        ]
```

5.2.3 Метод на Ланцош (метод на биортогонализацията за несиметрични матрици)

За несиметрични матрици може да се получи подобно опростяване на това, което имаме при метода на ортогонализацията за симетрични матрици. За тази цел обаче трябва да се построят две редици взаимно-ортогонални вектори, започвайки с начални вектори $\mathbf{x}_0$ и $\mathbf{y}_0$. Първата редица са вектори в пространствата $\mathcal{K}_{\|} = span\{\mathbf{x}_0, A\mathbf{x}_0 \ldots, A^k\mathbf{x}_0\}$, а другата – в пространствата $\bar{\mathcal{K}}_k = span\{\mathbf{y}_0, A^T\mathbf{y}_0 \ldots, (A^T)^k\mathbf{y}_0\}$. Тук се използва фактът, че собствените стойности на A и на A^T са едни и същи.

Задача 31. Нека

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

По метода на Ланцош да се намери характеристичния полином (или негов делител) при начални вектори $\mathbf{x}_0 = (1,1,0)^T$, $\mathbf{y}_0 = (0,1,0)^T$.

Решение. Построяваме последователно векторите по аналогичен начин на това, което направихме в предишния параграф. Ще бележим с червено съответните елементи на пространствата от полиноми. Всички вектори по-долу ще интерпретираме като вектор-стълбове. Ще приведем изчисленията съвсем накратко. Те използват факта, че $(\mathbf{x}_i, \mathbf{y}_j) = 0$ за $i \neq j$.

- $$\left|\begin{array}{ll} A\mathbf{x}_0 = (1,0,-1) & \lambda \\ \mathbf{x}_1 = A\mathbf{x}_0 + \alpha\mathbf{x}_0 & \\ \alpha = -\frac{(A\mathbf{x}_0,\mathbf{y}_0)}{(\mathbf{x}_0,\mathbf{y}_0)} = 0 & \\ \mathbf{x}_1 = (1,0,1) & \lambda \end{array}\right| \left|\begin{array}{ll} A^T\mathbf{y}_0 = (-1,1,0) & \lambda \\ \mathbf{y}_1 = A^T\mathbf{y}_0 + \alpha\mathbf{y}_0 & \\ \alpha = -\frac{(A^T\mathbf{y}_0,\mathbf{x}_0)}{(\mathbf{x}_0,\mathbf{y}_0)} = 0 & \\ \mathbf{y}_1 = (-1,1,0) & \lambda \end{array}\right|$$

- $$\left|\begin{array}{ll} A\mathbf{x}_1 = (2,-1,-1) & \lambda^2 \\ \mathbf{x}_2 = A\mathbf{x}_1 + \alpha\mathbf{x}_1 + \beta\mathbf{x}_0 & \\ \alpha = -\frac{(A\mathbf{x}_1,\mathbf{y}_1)}{(\mathbf{x}_1,\mathbf{y}_1)} = -3 & \\ \beta = -\frac{(A\mathbf{x}_1,\mathbf{y}_0)}{(\mathbf{x}_0,\mathbf{y}_0)} = 1 & \\ \mathbf{x}_2 = (0,0,2) & \lambda^2 - 3\lambda + 1 \end{array}\right| \left|\begin{array}{ll} A^T\mathbf{y}_1 = (-2,1,1) & \lambda^2 \\ \mathbf{y}_2 = A^T\mathbf{y}_1 + \alpha\mathbf{y}_1 + \beta\mathbf{y}_0 & \\ \alpha = -\frac{(A^T\mathbf{y}_1,\mathbf{x}_1)}{(\mathbf{x}_1,\mathbf{y}_1)} = -3 & \\ \beta = -\frac{(A^T\mathbf{y}_1,\mathbf{x}_0)}{(\mathbf{x}_0,\mathbf{y}_0)} = 1 & \\ \mathbf{y}_2 = (1,-1,1) & \lambda^2 - 3\lambda + 1 \end{array}\right|$$

- $$\left|\begin{array}{ll} A\mathbf{x}_2 = (-2,0,2) & \lambda^3 - 3\lambda^2 + \lambda \\ \mathbf{x}_3 = A\mathbf{x}_2 + \alpha\mathbf{x}_2 + \beta\mathbf{x}_1 + \gamma\mathbf{x}_0 & \\ \alpha = -\frac{(A\mathbf{x}_2,\mathbf{y}_2)}{(\mathbf{x}_2,\mathbf{y}_2)} = 0 & \\ \beta = -\frac{(A\mathbf{x}_2,\mathbf{y}_1)}{(\mathbf{x}_1,\mathbf{y}_1)} = 2 & \\ \gamma = -\frac{(A\mathbf{x}_2,\mathbf{y}_0)}{(\mathbf{x}_0,\mathbf{y}_0)} = 0 & \\ \mathbf{x}_3 = (0,0,0) & (\lambda^3 - 3\lambda^2 + \lambda) + 2\lambda \end{array}\right|$$

  Да обърнем внимание, че изчисляването на γ не беше необходимо. Направихме го, само за да покажем, че резултатът действително е 0.

  Отново за характеристичния полином получихме $\lambda^3 - 3\lambda^2 + 3\lambda$.

□

Допълнителни задачи 2

Задача 32. Дадена е системата

$$\begin{array}{rcrcrcl} 3x_1 & + & x_2 & - & x_3 & = & 1, \\ & & 2x_2 & + & x_3 & = & 2, \\ x_1 & - & 2x_2 & + & 3x_3 & = & 3. \end{array}$$

1. Приведете системата във вид, удобен за прилагане на метода на простата итерация.
2. Изследвайте сходимостта на метода на простата итерация за дадената система.
3. Можете ли да приложите някое от достатъчните условия, изведени в лекционния курс, за да покажете, че методът е сходящ за всяко начално приближение?
4. Изследвайте сходимостта на метода на Зайдел за дадената система.
5. Направете на ръка по две итерации по метода на простата итерация и по метода на Зайдел за начално приближение $(0,0,0)^T$.
6. Сравнете времето, за което схожда методът на простата итерация и методът на Зайдел, като в цикъл пуснете решаването на системата 10000 пъти (в противен случай системата ще бъде решена за много малко време, което не може да бъде регистрирано).

Задача 33. Да се имплементира методът на Крилов.

Задача 34. С методите на Крилов, Ланцош и Данилевски да се намерят характеристичните полиноми на матриците

$$\begin{bmatrix} 4 & 2 & 2 \\ 2 & 5 & 1 \\ 2 & 1 & 6 \end{bmatrix}, \quad \begin{bmatrix} 3 & 1 & 5 \\ 1 & 3 & 5 \\ 5 & 5 & -1 \end{bmatrix}$$

- на ръка;
- като се използват написани от вас програми в Mathematica.

Задача 35. Имплементирайте степенния метод, описан по-долу.

Степенният метод е **итерационен метод** за намиране на най-голямата по модул собствена стойност на дадена квадратна матрица A.

Вземайки за **начално приближение произволен ненулев вектор**, $c^{(0)}$, правим итерации по формулата

$$c^{(k+1)} = \frac{A.c^{(k)}}{\|A.c^{(k)}\|}.$$

Итерационният процес се прекъсва тогава, когато две последователни приближения имат приблизително равни координати, т.е. когато

$$1 - \varepsilon < \left| \frac{c_1^{(k+1)}}{c_1^{(k)}} \right| < 1 + \varepsilon.$$

Най-голямата собствена стойност на матрицата A се задава с формулата

$$\frac{c^T.A.c}{c^T.c}.$$

Задача 36. Имплементирайте метода на Якоби, описан по-долу.

Методът на Якоби е **итерационен метод** за приближеното намиране на собствените стойности на една **симетрична матрица** A. При него се правят **преобразования на подобие**

$$A_{new} = MA_{old}M^{(-1)},$$

където матрицата на трансформация M има следния вид:

$$M = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & \cos\phi & \cdots & -\sin\phi & & \\ & & \vdots & & \vdots & & \\ & & \sin\phi & \cdots & \cos\phi & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}.$$

Тук

$$\varphi = \frac{1}{2}\arctan\frac{2a_{ij}}{a_{ii} - a_{jj}},$$

а i и j са индексите на най-големия по абсолютна стойност извъндиагонален елемент.

Итерациите се правят, **докато матрицата стане "почти диагонална"**, т.е. докато сумата от квадратите на извъндиагоналните елементи

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{ij}^2$$

стане по-малка от ε.