УНИВЕРСИТЕТ ПО АРХИТЕКТУРА,
СТРОИТЕЛСТВО И ГЕОДЕЗИЯ

Михаил Константинов

ЧИСЛЕНИ ОСНОВИ НА МАТЕМАТИЧЕСКИТЕ МЕТОДИ

Магистърска програма
„Изследване на строителните
конструкции”

София
1997

Съдържание

Анотация. В тези записки се излагат някои аспекти на числения анализ, свързани с ефектите от крайната точност на машинната аритметика върху резултатите от изчисленията. Разгледани са последователно елементите на машинната аритметика, числената линейна алгебра и числения математически анализ. Специално внимание е отделено на митовете в изчислителната практика.

Записките са предназначени основно за студентите от Строителния факултет на УАСГ от магистърския курс по програмата „Изследване на строителните конструкции", както и за някои от лекционните курсове по дисциплините „Линейна алгебра и аналитична геометрия", „Математически анализ - 1 част" и „Приложна математика" за инженерните факултети. Те могат да се ползват и от студенти по други математически дисциплини, както и от инженери и специалисти от практиката.

Математически основи на числените методи

Глава 1

Числени компютърни пресмятания

1.1 Основни означения

Използвани са следните означения:

$\mathbb{R}$ (респ. $\mathbb{C}$) – множеството на реалните (респ. комплексните) числа;

$\mathbb{R}^{m\times n}$ – пространството на реалните $m \times n$ матрици ($\mathbb{R}^n = \mathbb{R}^{n\times 1}$);

$A^\top \in \mathbb{R}^{n\times m}$ – транспонираната матрица на матрицата $A \in \mathbb{R}^{m\times n}$;

$\|.\|$ – норма в $\mathbb{R}^n$ или $\mathbb{R}^{m\times n}$.

Обикновено ще използваме евклидовата норма

$$\|x\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$$

на вектора $x = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$.

За матрици най-често се използват спектралната норма $\|A\|_2$ или Фробениусовата норма $\|A\|_\mathrm{F}$, където $A = [a_{ij}] \in \mathbb{R}^{m\times n}$. Спектралната норма $\|A\|_2$ е равна на корен втори от най-голямата собствена стойност на матрицата $A^\top A$ (или на най-голямата сингулярна стойност $\sigma_{\max}(A)$ на A). Нормата на Фробениус се определя от

$$\|A\|_\mathrm{F} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

и е равна на евклидовата норма на mn-вектора $\mathrm{vec}(A)$, получен от „разпъването" на матрицата A по стълбове или по редове.

Ако λ е число, то произведение на вектора x с елементи x_i и λ ще наричаме вектора $\lambda x = x\lambda$ с елементи λx_i. Нека x и y са два n-вектора с елементи x_i и y_i. Сума на векторите x и y е векторът с елементи $x_i + y_i$. Скаларно произведение на векторите x и y е числото

$$x^\top y = (x, y) = x_1y_1 + x_2y_2 + \cdots + x_ny_n.$$

При тези означение имаме

$$\|x\|_2 = \sqrt{(x, x)}.$$

В изчислителната практика се използват също така нормите

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

и

$$\|x\|_\infty = \max\{|x_i| : i = 1, 2, \ldots, n\}$$

които са по-лесни за пресмятане в сравнение с нормата $\|x\|_2$. По-общо, за всяко $p \geq 1$ можем да дефинираме т. нар. „p-норма"

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{1/p}.$$

Ако матрицата $A \in \mathbb{R}^{n\times n}$ е неособена, то числото

$$\|x\|_{A,p} = \|Ax\|_p$$

също е норма на вектора x.

С помощта на векторната p-норма може да се дефинира и p-норма на $m \times n$-матрицата A,

$$\|A\|_p = \max\{\|Ax\|_p : x \in \mathbb{R}^n,\ \|x\|_p = 1\}.$$

Ако A е $m \times n$-матрица с елементи a_{ij} и λ е число, то произведение на A и λ е $m \times n$-матрицата $\lambda A = A\lambda$ с елементи λa_{ij}. Нека A и B са матрици с еднакви размери и елементи a_{ij} и b_{ij} съответно. Тогава сума на матриците A и B е матрицата $A + B$ със същите размери и с елементи $a_{ij} + b_{ij}$.

Ако $A \in \mathbb{R}^{m\times k}$ и $B \in \mathbb{R}^{k\times n}$, то произведение на A и B е матрицата $C = AB \in \mathbb{R}^{m\times n}$ с елементи

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj}.$$

За матричните норми са в сила неравенствата

$$\|\lambda A\| \leq |\lambda|\,\|A\|,$$
$$\|A + B\| \leq \|A\| + \|B\|,$$
$$\|AB\| \leq \|A\|\,\|B\|$$

които често се използват в числения анализ. Също така имаме

$$\|AB\|_{\mathrm{F}} \leq \min\{\|A\|_{\mathrm{F}}\|B\|_2, \|A\|_2\|B\|_{\mathrm{F}}\}.$$

Символът := означава „равно по определение".

1.2 Уводни бележки

Съвременните компютри извършват както *числени*, така и *символни* пресмятания. В първия случай пресмятанията по принцип се извършват с грешки от закръгляне, тъй като всеки компютър е система с краен брой вътрешни състояния и следователно може да оперира с краен (макар и голям) брой числа. Така например числата $\sqrt{2}$ и π могат да се запишат в паметта на компютъра с точност до 16 (или друг краен брой) десетични цифри. При обработката на числена информация компютрите могат да оперират с огромни масиви и да решават задачи, в които междинните и крайни резултати са вектори или матрици с десетки хиляди, милиони или даже милиарди елементи. В частност, възможно е решаването на линейна алгебрична система $Ax = b$, където A матрица $10\,000 \times 10\,000$, намирането на спектъра и пресмятането на детерминантата на подобна матрица и т.н.

При извършване на числени пресмятания относителните грешки при отделните операции обикновено са малки, от порядъка на 10^{-16} или 10^{-17}. Въпреки това техният ефект върху точността на решението може да бъде разрушителен. Същевременно възможно е (по софтуерен път) да се постигне прак-

тически всяка отнапред зададена относителна грешка при извършване на елементарните аритметични операции, например 10^{-100} или даже 10^{-1000}, при което ефектът от грешките от закръгляне да се намали силно. Постигането на такава чудовищна точност, обаче, е свързано с недопустимо намаляване на бързодействието при изпълнението на съответните изчислителни алгоритми.

Символните пресмятания по принцип се извършват точно, но и при тях се срещат трудности. Проблемите тук са два: ниското бързодействие при изпълнение на алгоритмите и ниската размерност на решаваните задачи. Така например, ако имаме символна $n \times n$ матрица A с елементи (символи) a_{ij}, то записът на нейната детерминанта ще изисква използването на $N = n!n$ символа. Като използваме формулата на Стърлинг

$$n! \approx \sqrt{2\pi n}(n/e)^n$$

за голямо n лесно можем да определим за кое $n = n_0$ числото N достига броя 10^{85} на елементарните частици във видимата Вселена (намерете това n_0 и го сравнете с $n = 10\,000$, което не е проблем при определяне на детерминантата на числова матрица по някои от съвременните методи за тази цел).

В тази глава са изложени някои основни факти, свързани с използването на компютри за извършване на математически пресмятания.

При решаване на дадена изчислителна задача с помощта на компютър има три основни фактора, от които зависи точността на пресметнатото решение:

1. Свойствата на изчислителната среда (машинната аритметика).

2. Свойствата на изчислителната задача.

3. Свойствата на изчислителния алгоритъм.

В съответствие със съвременните високи стандарти в областта на научните и инженерни пресмятания, без оценка на фактическата грешка в пресметнатото решение, резултатът не може да се приеме за надежден. Тъй като точното решение е (и по принцип остава) неизвестно, то тази грешка може да се оцени само чрез коректно отчитане на изброените фактори.

1.3 Машинна аритметика

Предполагаме, че читателят е запознат с основните закони на аритметиката и следствията от тях. За разлика от „точните" операции в тази аритметика, аритметичните операции, извършвани от компютъра, са в общия случай само приближени, а и не винаги са изпълними дори в тази приближена форма.

Съвкупността от правила (често пъти не съвсем точно формулирани), по които смята компютърът, се нарича *машинна аритметика.* В машинната аритметика се оперира само с краен (макар и голям) брой числа, образуващи множеството от *машинни числа* $\mathbb{M} \subset \mathbb{R}$. При това на всяко реално число $x \in \mathbb{R}$ се съпоставя машинното число (или машинен аналог) $x^* \in \mathbb{M}$, като в общия случай $x^* \neq x$. Винаги е изпълнено $0^* = 0$, но е

възможно $x^* = 0$ дори ако $x \neq 0$. Възможно е също реалното число $x \in \mathbb{R}$ да няма машинен аналог (това става при твърде големи по модул числа).

На всяка точна аритметична операция

$$\bullet \in \{+, -, \times, /\}$$

която на всеки две числа $x, y \in \mathbb{R}$ съпоставя числото

$$x \bullet y \in \mathbb{R},$$

съответства *машинна операция*, която произвежда числото

$$(x \bullet y)^* \in \mathbb{M},$$

за което се очаква да бъде близко до $x \bullet y$.

В машинна аритметика, и по-точно в машинна аритметика с плаваща точка, са възможни някои доста екзотични явления, например:

- нарушени са асоциативният закон при сумирането $(x + y) + z = x + (y + z)$ и при умножението $(xy)z = x(yz)$, като

$$((x + y)^* + z)^* \neq (x + (y + z)^*)^*,$$
$$((xy)^* z)^* \neq (x(yz)^*)^*;$$

- нарушен е дистрибутивният закон $(x + y)z = xz + yz$, свързващ сумирането с умножението, като

$$((x + y)^* z)^* \neq ((xz)^* + (yz)^*)^*;$$

- равно е на нула произведението на две ненулеви числа с ненулеви машинни приближения, а именно

$$(xy)^* = 0,\ x^* \neq 0,\ y^* \neq 0.$$

Ще подчертаем, че тук става въпрос за *възможно* настъпване на тези явления. В много случаи това не се наблюдава - лошото е, че обикновено не знаем кой именно случай е налице.

Понякога при компютърните пресмятания се случват и други неприятности, както например при знаменития израз

$$x = \frac{(a+1)^2 - 2a - 1}{a^2},\ a > 0.$$

Очевидно $x = 1$. При използване на компютър, обаче, стават странни неща. Ако например на a зададем последователно намаляващи стойности, например $a = 1/10,\ 1/100, \ldots, 10^{-n}, \ldots$, то още при съвсем умерено малки аргументи a, например в диапазона от $a = 10^{-8}$ до $a = 10^{-10}$, ще получим явно неверни резултати, например отрицателни стойности за x. При това числото 10^{-10} съвсем не е малко за който и да е компютър; дори джобните калкулатори отдавна работят с малки числа от порядъка на 10^{-100}. Какво тогава е станало? И дали не трябва да си хвърлим компютъра? Отговорът за щастие е: не. Но за да сме сигурни, че в резултат на компютърните пресмятания се получават разумни резултати, е необходимо да познаваме, поне по принцип, особеностите на машинната аритметика.

В машинна аритметика с плаваща точка и *основа* β реалните числа се записват като произведение на два множителя:

дробна част (или *мантиса*) и цяла *степен* на основата. Подобен запис се използва отдавна в естествените и точните науки. Така например числата (в десетичен запис)

$$3.14159, \quad 1997, \quad -0.001417, \quad 300\,000\,000$$

могат да се запишат като

$$0.314159 \times 10^1, \quad 0.1997 \times 10^4, \quad -0.1417 \times 10^{-2}, \quad 0.3 \times 10^9.$$

В машинна аритметика обикновено основата β е степен на двойката, например $\beta = 2$ или $\beta = 16$. Това се дължи на технически причини: по-лесно е да се конструира надеждно устройство с две устойчиви състояния, отколкото да кажем с 10 такива състояния. Впрочем, има сериозни точностни съображения в полза на аритметика с $\beta = 3$, но скоро едва ли ще дочакаме това да стане - производителите на процесори засега не виждат смисъл да глезят чак дотам потребителите.

В такава аритметика ненулевите машинни числа имат вида

$$x^* = \pm \left(\frac{b_1}{\beta} + \frac{b_2}{\beta^2} + \cdots + \frac{b_T}{\beta^T} \right) \times \beta^E = \pm\mu \times \beta^E$$

където b_i са β-ични цифри, такива че

$$1 \leq b_1 \leq \beta - 1; \ 0 \leq b_k \leq \beta - 1, \ k = 2, \ldots, T$$

а числото E е цяло. Дробната част

$$\mu \in \left[\frac{1}{\beta}, 1 - \frac{1}{\beta^T} \right]$$

се нарича *мантиса* на числото $x^* \in \mathbb{M}$. Числото T се нарича *дължина на машинната дума* (в β-ични цифри).

Другояче казано, числото x^* може да се запише като

$$x^* = \pm 0.b_1 b_2 \dots b_T \times \beta^E,$$

откъдето идва и названието „аритметика с плаваща точка". Числото 0 също е елемент на $\mathbb{M}$.

Съществуват и редица други машинни аритметики, например с фиксирана точка, с дробно-рационално представяне на числата и т.н., на които тук не се спираме.

Обикновено E се изменя в целочисления диапазон от $-F+1$ до F, т.е., взима $2F$ на брой стойности. В този случай имаме машинна аритметика с плаваща точка от тип (β, T, F). При това аритметиката от тип (β, ∞, ∞) е всъщност стандартната точна аритметика.

Нека да пресметнем колко числа съдържа множеството $\mathbb{M}$. Броят на положителните числа се получава като съобразим, че b_1 може да приема $\beta - 1$ стойности, а всяко от $(T-1)$-те числа $b_2, \dots, b_T$ може да приема β стойности. Така на всяко фиксирано E съответстват $(\beta-1)\beta^{T-1}$ числа, т.е., имаме общо

$$N = 2F(\beta - 1)\beta^{T-1}$$

положителни числа и още толкова отрицателни числа в $\mathbb{M}$. Като прибавим и нулата намираме, че множеството $\mathbb{M}$ има

$$|\mathbb{M}| = 1 + 2N = 1 + 4F(\beta - 1)\beta^{T-1}$$

елемента. Това число може да ни изглежда много голямо, но то все пак е крайно. Именно обстоятелството, че множеството $\mathbb{M}$

е крайно, е главната причина за особеностите на машинната аритметика.

Ще отбележим също, че в $\mathbb{M}$ има най-малко положително число $X_{\min}$, което отговаря на $b_1 = 1$, $b_2 = \cdots = b_T = 0$ и $E = -F + 1$:

$$X_{\min} = \frac{1}{\beta} \times \beta^{-F+1} = \beta^{-F}.$$

Аналогично, най-голямото машинно число $X_{\max}$ се получава при $b_1 = b_2 = \cdots = b_T = \beta - 1$ и $E = F$, т.е.,

$$X_{\max} = \left(\frac{\beta - 1}{\beta} + \frac{\beta - 1}{\beta^2} + \cdots + \frac{\beta - 1}{\beta^T} \right) \times \beta^F = \beta^F \left(1 - \beta^{-T} \right).$$

Така имаме

$$X_{\min} X_{\max} = 1 - \beta^{-T} \approx 1.$$

Реалните числа x, за които

$$X_{\min} \leq |x| \leq X_{\max}$$

са в *стандартния диапазон* на машинната аритметика. Числото 0 също се причислява към стандартния диапазон на машинната аритметика.

Дадено ненулево число x от стандартния диапазон на машинната аритметика се представя чрез най-близкото до него машинно число x^* (ако x е по средата между две съседни машинни числа, то x^* се избира равно на по-голямото по модул). Числото x^* се нарича *машинен аналог* на x. Машинният аналог на нулата е самата тя. Също така, ако x е в стандартния диапазон на машинната аритметика, то $x^* = x$ точно когато $x \in \mathbb{M}$.

Тази операция се нарича *закръгляне* и при нея се прави относителна грешка, удовлетворяваща неравенството

$$\frac{|x^* - x|}{|x|} \leq \text{eps} := \frac{\beta^{1-T}}{2}.$$

Числото eps се нарича *мярка на закръгляне* на машинната аритметика (използват се и термините *относителна работна точност* и даже *машинен епсилон*) и е най-важната характеристика на тази аритметика.

Числото eps може да се дефинира (приблизително) и като най-малкото положително машинно число ε, за което е изпълнено машинното неравенство

$$1 + \varepsilon > 1.$$

Двойката $(\text{eps}, X_{\max})$ характеризира машинната аритметика с достатъчна за практиката точност и в този смисъл можем да говорим за машинна аритметика от тип $(\text{eps}, X_{\max})$. Колкото по-малка е мярката на закръгляне eps и колкото по-голямо е максималното положително машинно число $X_{\max}$, толкова повече машинната артиметика е по-близка до точната (стандартна) аритметика. Така от формална гледна точка точната аритметика се оказва граница на една редица от машинни аритметики с $\text{eps} \to 0$ и $X_{\max} \to \infty$, или накратко като машинна артиметика от тип $(0, \infty)$.

Като специални трябва да се разгледат случаите, когато някое число x (фигуриращо в данните на задачата или получено като междинен или краен резултат от компютърни прес-

мятания) е извън стандартния диапазон на машинната аритметика.

Ако $|x| < X_{\min}$, то числото се закръгля към 0, т.е., $x^* = 0$. Това явление се нарича *препълване на порядъка отдолу* (на английски *underflow*) и е опасно, макар и не фатално. Опасността идва от факта, че в някоя следваща операция x^* може да се окаже знаменател и тогава ще стигнем до аварийната ситуация *делене на нула* (*division by zero*), при която изчисленията се прекратяват. Ако обаче на някоя от следващите стъпки x^* се прибави към някое ненулево машинно число, тази неприятност може да се избегне. Интересно е, че при препълване на порядъка отдолу относителната грешка от закръгляне е точно 100 процента.

Ако $|x| > X_{\max}$ настъпва явлението *препълване на порядъка отгоре* (*overflow*), при което изчисленията се прекратяват. Това е най-тежката ситуация при изпълнение на даден алгоритъм, при която (обикновено) няма разумен автоматичен изход и се налага намеса на човек-оператор в работата на компютъра. При някои компютърни системи се допускат до няколко препълвания на порядъка отгоре преди да настъпи прекратяване на изчисленията. Във всеки случай, обаче, препълването отгоре трябва да се избягва на всяка цена. Един от начините за това е началните и междинни данни да се мащабират. Пример за такова мащабиране е даден по-долу.

Да видим сега какво се случва при машинното изпълнение на дадена аритметична операция $\bullet$ върху две реални ненулеви числа x, y от стандартния диапазон на машинната аритмети-

ка, при което резултатът $x \bullet y$ също е ненулево число от стандартния диапазон. В този случай за повечето компютри е в сила следната *основна хипотеза* на машинната аритметика:

Резултатът $(x \bullet y)^*$ *от машинното изпълнение на операцията* $\bullet$ *удовлетворява равенството*

$$(x \bullet y)^* = (x \bullet y)(1 + \delta),$$

където модулът на относителната грешка δ *не надвишава* $C\mathrm{eps}$, *а* $C \leq 3$ *е положителна константа.*

Следователно машинните аритметични операции се извършват с малка относителна грешка от порядъка на мярката на закръгляне eps. Разбира се, в редица случаи аритметичните операции се изпълняват точно, при което $\delta = 0$.

Така на практика машинната аритметика се характеризира с константите eps, $X_{\min}$ и $X_{\max}$. Като вземем предвид, че последните две числа са приблизително реципрочни, стигаме до извода, че машинната аритметика се определя само от две константи, а именно eps и $X_{\max}$. Така например за компилатора MS FORTRAN имаме

$$\mathrm{eps} \approx 2.22 \times 10^{-16}, \ X_{\max} \approx 4.49 \times 10^{307}.$$

В такава аритметика ще получим

$$1 + (10^{17} - 10^{17}) = 1$$

(вярно!), но

$$(1 + 10^{17}) - 10^{17} = 0$$

(много грешно!).

Също така правилно ще пресметнем

$$10^{200}(10^{200}10^{-150}) = 10^{200}10^{50} = 10^{250}$$

но няма да можем да пресметнем въобще същия израз при друг ред на умножаване

$$(10^{200}10^{200})10^{-150}$$

тъй като числото в скобата надхвърля $X_{\max}$ и порядъкът се препълва отгоре.

И накрая, машинното произведение на положителните числа $x = 10^{-150}$ и $y = 10^{-170}$ с положителни машинни аналози x^* и y^* е равно на нула: $(xy)^* = 0$.

Читателят трябва добре да различава числата eps и $X_{\min}$. Те и двете са малки, но

$$\frac{\text{eps}}{X_{\min}} \gg 1$$

като отношението $\text{eps}/X_{\min}$ действително е огромно. За разгледаните по-горе конкретни параметри на машинна аритметика имаме

$$\frac{\text{eps}}{X_{\min}} \approx 10^{292}.$$

Трудно е да си представим толкова голямо число. Например, броят на елементарните частици в известната ни Вселена е „само" около 10^{85} (!).

Положителните машинни числа образуват крайната N-членна редица

$$X_{\min} = X_1 < X_2 < \cdots < X_{N-1} < X_N = X_{\max}$$

като е изпълнено съотношението

$$X_{k+1} = X_k(1 + \text{eps}).$$

Разликата

$$X_{k+1} - X_k = X_k\text{eps}$$

между две последователни машинни числа може да се окаже много голямо число. Така например при $k = N - 1$ имаме

$$X_N - X_{N-1} \approx 10^{292}.$$

Следователно множеството

$$\mathbb{M} = \{-X_N, -X_{N-1}, \ldots, -X_2, -X_1, 0, X_1, X_2, \ldots, X_{N-1}, X_N\}$$

от машинни числа е доста „рехаво", особено в левия и десния си край (надяваме се, че читателят ще ни прости тази терминологична фриволност).

Разпространено е мнението, че евентуална голяма относителна грешка в резултата от машинните пресмятания се дължи на голям брой малки относителни грешки в отделните елементарни машинни операции (решаването дори на прости изчислителни задачи може да изисква огромен брой аритметични операции). Това, обаче, често е погрешно. Вярно е по-скоро обратното. При извършване на голям брой аритметични операции с компютър грешките обикновено не се натрупват еднопосочно, а даже имат тенденция взаимно да се унищожават. А наличието на голяма грешка в крайния резултат най-често се дължи на малък брой елементарни операции ((или даже на

една единствена такава операция!), при което е настъпила катастрофална загуба на точност.

Да разгледаме сега една от най-големите опасности при извършване на пресмятания в машинна аритметика, виновна обикновено за настъпилата изчислителна катастрофа. Това е така нареченото *катастрофално взаимно унищожение* (*catastrophic cancelation*), което настъпва при изваждане на близки числа. При това грешката от самото машинно изваждане (ако изобщо има такава) обикновено е без всякакво значение. Катастрофата се дължи на загубата на информация при изваждане на еднаквите цифри в левите рязряди на съответните мантиси.

Ще поясним тези съображения с пример. Нека са дадени двете много точни числа x и y с десетичен запис

$$\begin{aligned} x &= 0.123456789\xi \\ y &= 0.123456789\eta. \end{aligned}$$

Предполагаме, че първите девет значещи цифри $1, 2, \ldots, 9$ в двата записа са верни, а десетата (ξ или η) е съмнителна. Ще подчертаем, че това е направо чудовищна точност в инженерните науки, където три-четири-пет верни значещи цифри обикновено се смятат за добро постижение. Нека сега да извадим едно от друго тези две много точни числа. Получаваме

$$z = x - y = \pm\, 0.000000000\zeta, \;\; \zeta = |\xi - \eta|.$$

В резултата **няма нито една сигурна значеща цифра (!)**, т.е., ние се изхитрихме от две невероятно точни числа само с

една операция да получим едно съвсем неточно число. Отново подчертаваме, че причина за това не е машинното изваждане (при $\beta = 10$ то се извършва изобщо без грешки от закръгляне като $(x-y)^* = x-y$), а *загубата на информация* при изваждане на цифрите в левите разряди.

Избягването на това явление, заедно с предотвратяването на препълванията на порядъка, е първа грижа при организация на изчисленията в машинна аритметика. Така например изчисляването на израза

$$x = \sqrt{a^2 + b^2} - |b|$$

не трябва да става директно по указаната формула, защото при малко a/b ще настъпи взаимно унищожение. Поради това следва да използваме еквивалентния израз

$$x = \frac{a^2}{\sqrt{a^2 + b^2} + |b|}$$

при който взаимното унищожение е невъзможно.

С катастрофално взаимно унищожение се обясняват и грешките при пресмятане на израза

$$x = \frac{(a+1)^2 - 2a - 1}{a^2},\ a > 0.$$

В машинна аритметика с eps $\approx 10^{-16}$ (а именно такава аритметика е реализирана в повечето компютри) при $a < 10^{-8}$ в числителя имаме изваждане на две много близки числа $(a+1)^2$ и $2a+1$ с разлика $a^2 < 10^{-16}$. При това резултатът от изваждането може да не съдържа нито една вярна значеща цифра (!). Естествено, същото се отнася и за крайния резултат.

Ще отбележим накрая, че известната от училищния курс по математика формула

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

за корените на квадратното уравнение

$$ax^2 + bx + c = 0$$

е един много лош начин за пресмятането на $x_{1,2}$ в машинна аритметика поради опасността от катастрофално взаимно унищожение при пресмятане на един от корените. Може даже да се твърди, че непосредственото използване на тази формула е най-лошият начин за решаване на квадратно уравнение в машинна аритметика. Куриозното е, че именно тази формула се дава често като пример в задачите по програмиране без всякакво предупреждение. Впрочем, да се състави една прилична компютърна програма за решаване на квадратно уравнение далеч не е проста работа.

Препълването на порядъка както нагоре, така и надолу, може да се избегне чрез подходящо мащабиране на входните данни и на междинните резултати. Ще поясним това с пример.

Нека е необходимо да пресметнем нормата

$$x = \|a\| = \sqrt{a_1^2 + a_2^2}$$

на вектора $a = [a_1, a_2]^\top$ от $\mathbb{R}^2$ с компоненти a_1 и a_2. При решаване на тази проста задача пряко по горната формула лесно може да настъпи препълване на порядъка дори когато

данните a_1, a_2 и резултатът x са ненулеви числа в стандартния диапазон на машинната аритметика, например

$$X_{\min} < |a_1|, |a_2|, |x| < X_{\max}.$$

Нека първо предположим, че a_1 е сравнително голямо число, така че $a_1^2 > X_{\max}$. Това би се случило на стандартен персонален компютър при $|a_1| > 10^{155}$. В този случай настъпва препълване на порядъка отгоре и пресмятанията се прекратяват. Алгоритъмът не произвежда резултат, макар че и данните и резултатът могат да са твърде далеч от опасните граници на стандартния диапазон.

От друга страна, ако числата a_1 и a_2 са сравнително малки, така че $a_1^2 < X_{\min}$ и $a_2^2 < X_{\min}$, то ще настъпи двукратно препълване на порядъка отдолу и алгоритъмът ще произведе резултат $x^* = 0$ с относителна грешка от 100 процента. Това би се случило на стандартен персонален компютър и процесор на Интел при $0 < |a_1|, |a_2| < 10^{-155}$. Допълнителна (а може би и основна) неприятност тук е, че впоследствие може да се стигне до делене на нула, при което изчисленията се прекратяват.

И двете разгледани по-горе опасности могат да се избегнат чрез предварително *мащабиране на данните*. Да разгледаме следния алгоритъм

$$\begin{aligned} k &= |a_1| + |a_2| \\ b_1 &= \frac{a_1}{k},\ b_2 = \frac{a_2}{k} \\ \|b\| &= \sqrt{b_1^2 + b_2^2} \end{aligned}$$

$$x \quad = \quad k\|b\|.$$

Лесно е да се провери, че тук на практика се избягва препълването и в двете посоки при условие, че и данните и резултатът са в стандартния диапазон на машинната аритметика.

1.4 Изчислителни задачи

Независимо от своята конкретна природа, повечето изчислителни задачи могат да се формулират по следния начин. Даден е набор от m числа $a = (a_1, \ldots, a_m)$, които се интерпретират като *начални* (или *входни*) *данни* на задачата, и правило (или функция) f, което на всяко a от зададено множество $\mathcal{A}$ съпоставя набор от n числа $x = (x_1, \ldots, x_n)$, които се интерпретират като *резултат* (или *изходни данни*).

Удобно е да считаме a и x за вектори от $\mathbb{R}^m$ и $\mathbb{R}^n$ съответно. Така f може да се разглежда като функция, изобразяваща множеството $\mathcal{A} \subset \mathbb{R}^m$ в $\mathbb{R}^n$, а изчислителната задача формално се идентифицира с двойката (f, a). По-нататък ще приемем, че функцията f е непрекъсната.

За изчислителната задача се използва и означението $x = f(a)$. Често се налага да се работи не с един, а с няколко набора от данни a, например $a \in \mathcal{A}_1$, където A_1 е зададено подмножество на $\mathcal{A}$. В частност, възможно е $\mathcal{A}_1 = \mathcal{A}$. Тук говорим за *семейство* от изчислителни задачи $(f, \mathcal{A}_1)$. И така, изчислителната задача се свежда до пресмятане на резултата x по зададено a:

$$a \mapsto x = f(a), \ a \in \mathcal{A}.$$

Често изчислителните задачи се задават неявно, например чрез уравнение от вида

$$F(a, x) = 0$$

където

$$F : \mathcal{A} \times \mathbb{R}^n \to \mathbb{R}^n$$

е зададена функция. При достатъчно общи и естествени предположения относно F се оказва, че в околност $\mathcal{N}_a$ на точката $a_0 \in \mathcal{A}$, такава че

$$F(a_0, x_0) = 0$$

за някое $x_0 \in \mathbb{R}^n$, е определена непрекъсната функция f, удовлетворяваща

$$x_0 = f(a_0).$$

Така на всеки набор от данни $a \in \mathcal{N}_a$ се съпоставя резултатът $x = f(a)$.

Аналогично се формулират изчислителни задачи с комплексни входни и изходни данни. Впрочем, всяка такава задача лесно може да се преформулира като задача с реални данни.

Да разгледаме като пример задачата за решаване на линейното векторно алгебрично уравнение

$$Ax = b$$

където $A \in \mathbb{R}^{n \times n}$ е зададена неособена матрица и $b \in \mathbb{R}^n$ е зададен вектор. Тук елементите на A и b формират $(n^2 + n)$-вектор на данните, а x е n-векторът на изходните данни

(или резултата). Подмножеството $\mathcal{A}$ е определено от условието за обратимост на матрицата A.

Като втори пример да разгледаме алгебричното уравнение

$$x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n = 0$$

където a_i са зададени реални константи (елементи на вектора на данните a). Тук в качеството на резултат можем да приемем n-те корена (в общия случай комплексни) $x_1, \ldots, x_n$ на уравнението.

Една от най-важните характеристики на една изчислителна задача е нейната *чуствителност*. Чуствителността се измерва с изменението в резултата при изменение в данните. Интуитивно, *една задача е чуствителна, ако малки изменения в данните водят до големи изменения в резултата.* Естествено, понятията „малко" и „голямо" тук са относителни и се разбират в контекста на конкретната машинна аритметика.

Нека е дадена изчислителната задача $x = f(a)$ и нека δa е (малко) изменение в данните, такова че $a + \delta a \in \mathcal{A}$. На това изменение отговаря изменение δx в резултата,

$$x + \delta x = f(a + \delta a).$$

Оттук

$$\delta x = f(a + \delta a) - f(a).$$

Природата на смущенията в данните може да е най-разнообразна. Смущенията може да се дължат на измервателни или параметрични грешки при определяне на математическия модел на дадено явление или обект, както и на грешки от закръгляне при

пресмятания в машинна аритметика. Така например още при записване на данните a в паметта на компютъра ние получаваме един нов набор a^*, който представлява машинен аналог на първоначалните данни a. При това съгласно правилото за закръгляне, когато числата a_i са в стандартния диапазон на машинната аритметика, имаме

$$a_i^* = a_i(1+\delta_i) = a_i + \delta a_i,\ \delta a_i = a_i\delta_i$$

където $|\delta_i| \leq \text{eps}$ и eps е мярката на закръгляне на машинната аритметика. Така

$$|\delta a_i| \leq |a_i|\text{eps}$$

и

$$a^* = a + \delta a,\ \|\delta a\| \leq \|a\|\text{eps}.$$

В този случай смущенията в данните са неизбежни, произтичащи от самия характер на машинната аритметика. Естествено, в хода на изчисленията се въвеждат и допълнителни грешки. Целта на добре организирания изчислителен процес е, ако е възможно, тези допълнителни грешки да не надхвърлят значително грешките, породени от началното закръгляне на данните в паметта на компютъра.

Да предположим, че при зададено $r > 0$ и всяко δa с $\|\delta a\| \leq r$ е изпълнено неравенството

$$\|\delta x\| \leq K\|\delta a\|$$

където

$$K = K(f, a, r)$$

е величина, независеща от δa. Тогава задачата се нарича *регулярна*. Константата

$$L = L(f, a) = \lim_{r \to 0} K(f, a, r)$$

се нарича *абсолютно число на обусловеност* на задачата (f, a).

В случай, че подобна оценка за $\|\Delta x\|$ не съществува, задачата е *сингулярна*.

Освен в термините на абсолютни изменения, чуствителността на една изчислителна задача може да изследва и чрез относителните изменения в данните и в резултата. Това е възможно, ако данните a и резултатът x са ненулеви вектори. Да положим

$$\delta_a = \frac{\|\delta a\|}{\|a\|}, \ \delta_x = \frac{\|\delta x\|}{\|x\|}.$$

На практика се предпочита работа именно с относителните изменения (или грешки). В този случай за регулярните задачи имаме оценката

$$\delta_x \leq k\delta_a, \ \delta_a \leq \frac{r}{\|a\|}$$

където

$$k := k(f, a, r) = K(f, a, r)\frac{\|a\|}{\|x\|} = K(f, a, r)\frac{\|a\|}{\|f(a)\|}.$$

Числото

$$\ell = \ell(f, a) = \lim_{r \to 0} k(f, a, r) = L(f, a)\frac{\|a\|}{\|f(a)\|}$$

се нарича *относително число на обусловеност* на задачата (f, a).

Например, задачата за решаване на едно алгебрично уравнение от степен n е регулярна, ако то има n на брой различни корена, и сингулярна, ако то има кратни корени. В частност, решаването на уравнението

$$x^2 + a_1 x + a_2 = x^2 - 2x + 1 = 0$$

с кратни корени

$$x_1 = x_2 = 1$$

е сингулярна задача. Ако смутим малко свободния член $a_2 = 1$ до $1 - \varepsilon$, където

$$0 < \varepsilon \ll 1$$

то получаваме смутеното уравнение

$$(x - 1)^2 = \varepsilon$$

с корени

$$x^*_{1,2} = 1 \pm \sqrt{\varepsilon}$$

т.е.,

$$|\delta x_i| = \sqrt{\varepsilon}.$$

В случая не съществува линейна оценка за големината на смущението в решението като функция на смущението ε в данните.

Сингулярните задачи са изключително чуствителни спрямо изменения в данните и поради това решаването им в машинна аритметика обикновено е свързано с големи грешки от закръгляне.

Дори когато една задача е регулярна, то тя все още може да е силно чуствителна в контекста на конкретна машинна аритметика, характеризираща се с мярка на закръгляне eps. Такива задачи се наричат *лошо обусловени* в дадената машинна аритметика. Ще подчертаем, че понятието „обусловеност" има смисъл не изобщо, а само в дадена машинна артиметика.

Да разгледаме регулярната изчислителна задача (f, a), характеризираща се с относителното число на обусловеност ℓ. Задачата се нарича *много добре обусловена*, ако

$$\operatorname{eps} \ell \ll 1$$

и *много лошо обусловена*, ако

$$\operatorname{eps} \ell \approx 1.$$

В последния случай, при решаване на задачата в машинна аритметика с мярка за закръгляне eps, в полученото решение може да няма нито една вярна значеща цифра. Това е изчислителна катастрофа, която следва категорично да се избягва.

Възможна е една по-детайлна класификация на чуствителността на регулярните изчислителни задачи в контекста на дадена машинна аритметика. Задачата се нарича:

- *лошо обусловена*, ако $\ell \geq \operatorname{eps}^{-2/3}$;
- *умерено обусловена*, ако $\operatorname{eps}^{-1/3} \leq \ell < \operatorname{eps}^{-2/3}$;
- *добре обусловена*, ако $\ell < \operatorname{eps}^{-1/3}$.

Съществува едно полезно евристично правило, според което при $\ell \mathrm{eps} < 1$ в изчисленото решение има приблизително

$$-\log_{10}(\ell \mathrm{eps})$$

верни значещи цифри. Нека например $l = 10^{10}$ и $\mathrm{eps} = 10^{-16}$. Тогава можем да очакваме

$$-\log(10^{10}10^{-16}) = -(-6) = 6$$

верни значещи цифри в решението.

1.5 Изчислителни алгоритми

При решаване на една изчислителна задача с помощта на компютър се извършва (последователно или паралелно) редица от елементарни изчислителни операции и в частност редица от елементарни аритметични операции. Тази именно последователност от елементарни операции се асоциира с понятието *изчислителен алгоритъм.* [1]

Другояче казано, всеки алгоритъм определя някаква (точна или приближена) декомпозиция на функцията f чрез зададен брой (например p) стъпки, която може да се опише така. На първата стъпка от данните $a^{(0)} = a$ чрез една аритметична операция f_1 се преминава към междинния резултат $a^{(1)}$,

$$a^{(1)} = f_1(a^{(0)})$$

[1] Точно определение на понятието „алгоритъм" тук не се дава.

като векторите $a^{(1)}$ и $a^{(0)}$ могат да са с различни размери. Аналогично, на k-тата стъпка от междинния резултат $a^{(k-1)}$ чрез една аритметична операция f_k се получава следващия междинен резултат

$$a^{(k)} = f_k(a^{(k-1)}).$$

На последната k-та стъпка получаваме крайния резултат

$$\widehat{x} = a^{(p)} = f_p(a^{(p-1)}).$$

При това алгоритъмът е (теоретично) точен, ако $\widehat{x} = x$, или (теоретично) приближен, ако $\widehat{x} \neq x$. В последния случай съществува оценка от вида

$$\|\widehat{x} - x\| \leq \eta_p$$

където $\eta_p \to 0$ при $p \to \infty$. Намирането на конкретния вид на величината η_p като функция на p, чрез която се оценява сходимостта на алгоритъма, е предмет на класическия числен анализ.

Ще отбележим, че дотук изобщо не стана въпрос в каква изчислителна среда се изпълнява даден алгоритъм, т.е., дали той се реализира в точна или в машинна аритметика.

Два алгоритъма се наричат *еквивалентни*, ако в точна аритметика те водят до един и същ резултат.

Например задачата

$$x = f(a) = f(a_1, a_2) = a_1^2 - a_2^2$$

може да се реши чрез следните еквивалентни алгоритми:

1. Алгоритъм 1:

$$u = a_1^2,$$
$$v = a_2^2,$$
$$x = u - v.$$

2. Алгоритъм 2:

$$u = a_1 - a_2,$$
$$v = a_1 + a_2,$$
$$x = uv.$$

Два еквивалентни алгоритъма, обаче, могат да имат съвършено различно поведение при изпълнението им в машинна аритметика. Така например алгоритъм 1 е по-точен при някои съотношения между числата a_1 и a_2, докато алгоритъм 2 е по-точен в останалите случаи. Така може да се конструира един усъвършенстван алгоритъм 3, който превключва алгоритмите 1 и 2 в зависимост от това в коя област са данните $a = (a_1, a_2)$. По този начин с цената на известно забавяне на изпълнението на алгоритъма се постига по-висока точност в крайния резултат.

Разгледаната в предишния раздел опасност от взаимно унищожение предлага пример за еквивалентни алгоритми, от които единият

$$x = \sqrt{a^2 + b^2} - |b|$$

е лош, а другият

$$x = \frac{a^2}{\sqrt{a^2 + b^2} + |b|}$$

е добър, което е постигнато с малко хитрост и една аритметична операция (делене) в повече.

През последните 50-тина години [2] се констатира, че редица класически числени методи и алгоритми имат доста неприятно поведение в машинна аритметика и в частност водят до недопустимо големи грешки дори и при добре обусловени задачи. Това наложи както преработка на маса класически изчислителни средства, така и създаването на нови, пригодени специално за компютърни пресмятания.

Да разгледаме накратко свойствата на изчислителните алгоритми при изпълнението им в машинна аритметика с мярка за закръгляне eps. Да означим с x^* резултата, произведен от описания по-горе алгоритъм

$$a = a^{(0)} \mapsto a^{(1)} \mapsto \cdots \mapsto a^{(p)} = x$$

за който предполагаме, че е теоретично точен (т.е., че алгоритъмът би произвел точния резултат $x = f(a)$ при изпълнението му в точна аритметика с eps $= 0$). Ще предполагаме, че както данните, така и междинните и крайни резултати са числа от стандартния диапазон на машинната аритметика. Това означава, че пресмятанията се извършват без препълвания на порядъка.

Изчислителният алгоритъм се нарича *устойчив*, ако произведеният резултат x^* е близък до точния резултат $f(a^*)$ на

[2] За рожденна дата на съвременните цифрови компютри се приема февруари 1946.

близка задача ($a^* \approx a$) в следния смисъл:

$$\begin{aligned} \|x^* - f(a^*)\| &\leq \operatorname{eps} R\|x\|, \\ \|a^* - a\| &\leq \operatorname{eps} S\|a\| \end{aligned}$$

където R и S са константи, характеризиращи алгоритъма.

Приложен към добре обусловена задача, един числено устойчив алгоритъм по принцип произвежда точен резултат. Ако, обаче, задачата е лошо обусловена, дори и числено устойчив алгоритъм може да произведе резултат, обременен с големи грешки.

От изложените по-горе резултати непосредствено се извежда една много елегантна оценка на грешката в решението на дадена регулярна задача. С точност до малки от първи ред относно eps включително имаме

$$\begin{aligned} \|x^* - x\| &= \|x^* - f(a^*) + f(a^*) - f(a)\| \\ &\leq \|x^* - f(a^*)\| + \|f(a^*) - f(a)\| \\ &\leq \operatorname{eps} R\|x\| + L\|a^* - a\| \\ &\leq \operatorname{eps} R\|x\| + \operatorname{eps} SL\|a\|. \end{aligned}$$

Оттук за относителната грешка в изчисленото решение получаваме

$$\frac{\|x^* - x\|}{\|x\|} \leq \operatorname{eps} R + \operatorname{eps} SL\frac{\|a\|}{\|x\|} = \operatorname{eps}(R + \ell S).$$

Вижда се как оценката за относителната грешка зависи от трите фундаментални фактора, влияещи върху изчислителния процес:

- машинната аритметика чрез мярката на закръгляне eps;
- изчислителната задача чрез относителното число на обусловеност ℓ;
- изчислителния алгоритъм чрез константите R и S.

Резултатът и грешката естествено зависят и от $X_{\max}$, но косвено, чрез условието за избягване на препълвания на порядъка в хода на изпълнение на алгоритъма.

Ще резюмираме ефектите при решаване на дадена задача $x = f(a)$ в машинна аритметика в зависимост от свойствата на задачата и на алгоритъма.

1. *Добре обусловена задача и числено устойчив алгоритъм.* Тъй като алгоритъмът е устойчив, изчисленото решение x^* е близко до $f(a^*)$, където a^* е близко до точните данни a^*. От добрата обусловеност следва, че $f(a^*)$ е близко до точното решение $f(a)$. Следователно и изчисленото решение е също близко до точното решение. Това е идеалният случай за извършване на числени математически пресмятания с помощта на компютър.

2. *Лошо обусловена задача и числено устойчив алгоритъм.* Както в предходния случай, пресметнатото решение x^* е близко до точното решение $f(a^*)$ на близка задача, тъй като a^* е близко до a. За съжаление тук поради лошата обусловеност може да се окаже, че $f(a^*)$ не е близко то точното решение $f(a)$, а оттам – че и пресметнатото

решение x^* е далеч от точното решение x. Ако алгоритъмът е надежден и сигнализира за лоша обусловеност и възможни големи грешки, ползвателят може да вземе мерки и най-малкото да има едно на ум, че има проблеми с точността. Впрочем, често границите на грешката, предсказана от оценките за обусловеност, не се достигат и е възможно полученият резултат все пак да е приемлив. Във всеки случай използването на машинна аритметика с по-малка мярка на закръгляне ще подобри точността на изчисленото решение.

3. *Добре обусловена задача и числено неустойчив алгоритъм.* Тук пресметнатото решение x^* не е непременно близко до решението на която и да е близка задача. Затова въпреки добрата обусловеност на задачата x^* може да е далеч от точното решение x. Разбира се, може да се окаже че x^* и x са близки, но за това няма никаква гаранция.

4. *Лошо обусловена задача и числено неустойчив алгоритъм.* Това е изчислителна катастрофа. Вероятно изчисленото решение x^* е далеч от точното. Възможно е в x^* да няма вярна значеща цифра или даже верен алгебричен знак. Опитите да се използва машинна аритметика с по-малка мярка на закръгляне едва ли ще оправят нещата.

1.6 Задачи

1.1 Докажете, че за мантисата μ при машинното представяне на числата са в сила неравенствата

$$\frac{1}{\beta} \leq \mu \ \leq 1 - \frac{1}{\beta^T}.$$

1.2 Напишете всички положителни машинни числа в аритметика с основа $\beta = 2$, дължина на думата $T = 3$, $F = 5$, и ги нанесете върху числовата ос. Определете eps и $X_{\max}$. Сега получихте ли представа как изглежда едно множество от машинни числа? [3]

1.3 Извършете някои аритметични операции с реални числа от стандартния диапазон на машинната аритметика от пример 1.2. Коментирайте резултатите.

Забележка. При умножаване на две числа те първо се закръглят, мантисите им се умножават, степените им се сумират и накрая полученият резултат се нормализира и закръгля. При събиране на две числа те първо се закръглят, степените им се изравняват към по-голямата, вмъква се съответният брой нули между β–ичната точка и първата ненулева β–ична цифра на по-малкото по абсолютна стойност събираемо, мантисите се сумират и накрая резултатът се нормализира и закръгля.

1.4 Напишете програма за пресмятане на експонентата

$$\exp(a) = \sum_{k=0}^{\infty} \frac{a^k}{k!} = 1 + a + \frac{a^2}{2!} + \cdots \tag{1.1}$$

[3] Признавам, че когато за пръв път видях картинката на едно такова множество останах леко шокиран.

която сумира членовете на реда докато поредното събираемо не измени изчислената преди него сума (този стопиращ критерий непременно ще сработи в машинна аритметика). Пресметнете резултата за $a = -5, -10, -20$. За сведение,

$$\begin{aligned}\exp(-5) &\approx 6.74 \times 10^{-3},\\ \exp(-10) &\approx 4.54 \times 10^{-5},\\ \exp(-20) &\approx 2.06 \times 10^{-9}.\end{aligned}$$

На какво се дължи настъпилата изчислителна катастрофа? Как работи програмата при $a > 0$? Ако отговорите правилно на тези два въпроса няма да е трудно да напишете надеждна програма за пресмятане на експонентата както за положителни, така и за отрицателни стойности на аргумента.

1.5 Подобрете програмата от задача 1.4 като използвате равенството

$$\exp(a) = (\exp(a/m))^m,\ m \neq 0$$

така, че винаги да пресмятате реда (1.1) при положителен аргумент, по-малък от 1.

1.6 (това е една задача за професионалисти!) Напишете алгоритъм и програма за решаване на квадратното уравнение

$$ax^2 + bx + c = 0$$

които да гарантират, че ако данните a, b, c са в стандартния диапазон на машинната аритметика, грешката в изчислените корени няма да надвишава $C\mathrm{eps}$, където константата C не

зависи от мярката на закръгляне eps. Тествайте вашия алгоритъм и го сравнете с наивния алгоритъм, основан на пряко използване на формулите за решаване на квадратно уравнение.

1.7 Напишете програма, която да генерира редицата $\{x_n\}$ по рекурентната формула

$$x_n = 0.5\, x_{n-1} - 0.75\, x_{n-2},\ n \geq 2,\ x_0 = 2,\ x_1 = 1.$$

Сравнете резултата с точния резултат

$$x_n = 2^{1-n},\ n \geq 1.$$

За кои стойности на n се получава „взрив" на грешката?

1.8 Програмирайте знаменитите изрази $x_m(a)$, където

$$\begin{aligned} x_2(a) &= \frac{(a+1)^2 - 2a - 1}{a^2}, \\ x_3(a) &= \frac{(a+1)^3 - 3a^2 - 3a - 1}{a^3}, \\ x_4(a) &= \frac{(a+1)^4 - 4a^3 - 6a^2 - 4a - 1}{a^4} \end{aligned}$$

и ги пресметнете на Вашия компютър за $a = 10^{-p}$, $p = 0, 1, 2, \ldots$. При някакво p, зависещо от m, настъпва взрив на грешката и изчислителна катастрофа.[4] Намерете тази зависимост на p от $m = 2, 3, 4$ и я обяснете в термините на мярката на закръгляне eps в използваната машинна артиметика. Очаквам да получите нещо като

$$pm \approx \log_{10}\left(\frac{1}{\text{eps}}\right),$$

[4] Както вече предупредихме, не бързайте да си изхвърляте компютъра, повредата вероятно не е в него.

което при стандартен процесор на Intel за PC дава $pm \approx 16$.

Глава 2

Елементи на числената линейна алгебра

В тази глава ще разгледаме решаването на някои основни задачи на линейната алгебра в машинна аритметика с мярка на закръгляне eps.

2.1 Операции с матрици

Нека A и B са две матрици с еднакви размери. При сумиране на тези матрици в машинна аритметика вместо точната сума $C = A + B$ се получава една изобщо различна от C матрица $C^* = (A + B)^*$. За елементите на C е в сила

$$c_{ij}^* = c_{ij}(1 + \delta_{ij}),\ |\delta_{ij}| \leq \text{eps}.$$

Оттук

$$|c_{ij}^* - c_{ij}| = |c_{ij}\delta_{ij}| \leq \text{eps}|c_{ij}|$$

и

$$\|(A+B)^* - (A+B)\|_{\text{F}} \leq \text{eps}\|A+B\|_{\text{F}}.$$

Следователно при $A + B \neq 0$ имаме

$$\frac{\|(A+B)^* - (A+B)\|_{\text{F}}}{\|A+B\|_{\text{F}}} \leq \text{eps}.$$

Получената оценка показва, че сумирането на две матрици в машинна аритметика се прави с малка относителна грешка от порядъка на мярката на закръгляне eps.

Да разгледаме сега пресмятането на произведението на две матрици в машинна аритметика.

Нека първо са дадени два неортогонални вектора $a, b \in \mathbb{R}^n$. Може да се покаже, че пресмятането на скаларното произведение $a^\top b$ става с относителна грешка, за която е в сила оценката

$$\frac{|(a^\top b)^* - a^\top b|}{|a^\top b|} \leq \text{eps}\frac{n\|a\|\,\|b\|}{|a^\top b|}.$$

Така относителната грешка при пресмятане на скаларно произведение в машинна аритметика не е ограничена и в частност може да е недопустимо голяма при

$$n\frac{\|a\|\,\|b\|}{|a^\top b|} \gg 1.$$

С използване на това неравенство лесно се показва, че за произведението на две матрици $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$ с $AB \neq 0$

в машинна аритметика е в сила следната оценка за относителната грешка

$$\frac{\|(AB)^* - AB\|_{\mathrm{F}}}{\|AB\|_{\mathrm{F}}} \leq \mathrm{eps}\, n\, \frac{\|A\|_f \|B\|_{\mathrm{F}}}{\|AB\|_{\mathrm{F}}}.$$

Вижда се, че относителната грешка може да бъде много голяма, тъй като множителят

$$\frac{\|A\|_{\mathrm{F}} \|B\|_{\mathrm{F}}}{\|AB\|_{\mathrm{F}}}$$

не е ограничен отгоре (знаменателят $\|AB\|_{\mathrm{F}}$ може да е произволно малък).

Има обаче един много важен частен случай, когато машинното произведение на две матрици се пресмята с малка относителна грешка. Нека например матрицата $A = [a_1, \ldots, a_n]$ е с ортонормирани стълбове $a_i \in \mathbb{R}^m$, т.е., $\|a_i\| = 1$ и $a_i^\top a_j = 0$ при $i \neq j$. Тогава

$$\|A\|_{\mathrm{F}} = \sqrt{n},\ \|AB\|_{\mathrm{F}} = \|B\|_{\mathrm{F}}$$

и следователно

$$\frac{\|(AB)^* - AB\|_{\mathrm{F}}}{\|AB\|_{\mathrm{F}}} \leq \mathrm{eps}\, n^{3/2}.$$

Това е много окуражителен резултат, който показва, че умножаването с ортогонална матрица не е опасно в числено отношение. Впрочем, това е намерило отражение в правилото, че *надеждните в числено отношение матрични алгоритми*

включват само ортогонални, респективно унитарни, преобразувания. Известно изключение тук прави алгоритъмът на Гаус за решаване на линейни алгебрични уравнения.

Важен аспект на числените компютърни пресмятания, който се развива активно през последните 15 години, са изчислителните алгоритми, гарантиращи *максимална възможна точност* в дадена машинна аритметика. Така например такъв алгоритъм гарантира, че пресметнатото скаларно произведение на два вектора $a, b \in \mathbb{R}^n$ е най-близкото то това произведение машинно число. Това е един неподобрим резултат, защото такава грешка от закръгляне се прави при записване на скаларното произведение в паметта на компютъра. Изобщо, алгоритмите с максимална точност решават някои сравнително прости задачи (като пресмятане на скаларно произведение, респективно на произведение на две матрици) с грешки, съизмерими с грешките при закръгляне на резултата към най–близкото машинно число.

Интересно е, че при съвременните персонални компютри ефектът на пресмятане с максимална точност се получава донякъде автоматично, тъй като аритметичните операции в процесора се извършват с по-висока точност, отколкото е резултантната работна точност. Повече подробности по този въпрос са дадени в [3].

2.2 Линейни алгебрични уравнения

В тази точка ще използваме само спектралната матрична норма, т.е., приемаме $\|\cdot\| = \|\cdot\|_2$.

Да разгледаме решаването на линейното алгебрично уравнение

$$Ax = b$$

в машинна аритметика с мярка за закръгляне eps. Ще предполагаме, че $A \in \mathbb{R}^{n\times n}$ е неособена матрица, $b \in \mathbb{R}^n$ е зададен ненулев вектор и $x \in \mathbb{R}^n$ е търсеното решение.

2.2.1 Чуствителност

Ще изследваме чуствителността на задачата като зададем смущения δA, δb в данните A, b, такива че

$$\|\delta A\| < \frac{1}{\|A^{-1}\|}.$$

Това неравенство гарантира, че смутената матрица $A + \delta A$ също е неособена. Нека $x + \delta x$ е единственото решение на смутеното уравнение

$$(A + \delta A)(x + \delta x) = b + \delta b.$$

След разкриване на скобите получаваме

$$\delta x = A^{-1}(\delta b - \delta A x) - A^{-1}\delta A \delta x$$

и

$$\|\delta x\| \leq \|A^{-1}\|(\|\delta b\| + \|\delta A\|\,\|x\|) + \|A^{-1}\|\|\delta A\|\,\|\delta x\|.$$

Тъй като $\|A^{-1}\|\,\|\delta A\| < 1$ имаме

$$\|\delta x\| \leq \frac{\|A^{-1}\|(\|\delta b\| + \|\delta A\|\,\|x\|)}{1 - \|A^{-1}\|\,\|\delta A\|}.$$

Оценката за чуствителността става особено елегантна, ако се премине към относителни смущения

$$\delta_x := \frac{\|\delta x\|}{\|x\|},\ \delta_A := \frac{\|\delta A\|}{\|A\|},\ \delta_b := \frac{\|\delta b\|}{\|b\|}.$$

Имаме

$$\delta_x \leq \frac{c(\eta\delta_b + \delta_A)}{1 - c\delta_A} \tag{2.1}$$

където

$$\begin{aligned} c &= c(A) = \mathrm{cond}(A) := \|A\|\,\|A^{-1}\| = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \\ \eta &= \eta(A,b) := \frac{\|b\|}{\|A\|\,\|x\|} = \frac{\|b\|}{\|A\|\,\|A^{-1}b\|}. \end{aligned}$$

Тъй като от $Ax = b$ следва

$$\|b\| \leq \|A\|\,\|x\|$$

то $\eta \leq 1$. Така получаваме по-простата и широко разпространена, но по принцип по-груба оценка

$$\delta_x \leq \frac{c(\delta_b + \delta_A)}{1 - c\delta_A}. \tag{2.2}$$

От друга страна имаме

$$\|A^{-1}b\| \leq \|A^{-1}\|\,\|b\|$$

откъдето

$$\frac{1}{\|A^{-1}b\|} \geq \frac{1}{\|A^{-1}\|\,\|b\|}$$

и $\eta \geq 1/c$. Окончателно

$$\frac{1}{\mathrm{cond}(A)} \leq \eta \leq 1.$$

Долната граница за η се достига когато b е десен сингулярен вектор на матрицата A^{-1}, съответстващ на максималната ѝ сингулярна стойност $\|A^{-1}\|$. В този случай не е разумно да се заменя η с 1. Нещо повече, оценката (2.1) с $\eta < 1$ може да е произволен брой пъти по-точна от оценката (2.2), в която е положено $\eta = 1$.

Да разгледаме случая, когато смущенията са породени от грешки от закръгляне при записване на данните в паметта на компютъра. Тогава можем да приемем, че

$$\delta_A, \delta_b \leq \mathrm{eps}$$

откъдето

$$\delta_x \leq \frac{c(1+\eta)\mathrm{eps}}{1 - c\,\mathrm{eps}}.$$

Когато $c\,\mathrm{eps} \ll 1$ имаме приближеното неравенство

$$\delta_x \leq c(1+\eta)\mathrm{eps}.$$

Така величината

$$\kappa := c(1+\eta) \leq 2c$$

може да се приеме като оценка на числото на обусловеност на уравнението.

2.2.2 Грешки

Най-разпространените методи за числено решаване на уравнението са методът на Гаус и методът на QR декомпозицията. И при двата метода е в сила следната оценка за относителната грешка в изчисленото решение $x^* \in \mathbb{R}^n$:

$$\delta = \frac{\|x^* - x\|}{\|x\|} \leq \operatorname{eps} \psi(n) \operatorname{cond}(A),\ \operatorname{cond}(A) := \|A\|\,\|A^{-1}\|$$

където $\psi(n)$ е полином на n от степен, ненадвишаваща 3. Горната оценка обикновено е силно песимистична, т.е., тя значително преоценява действителната грешка. Според едно практическо правило когато

$$\operatorname{eps} \operatorname{cond}(A) < 1$$

то в изчисленото решение вероятно има

$$[-\log_{10}(\operatorname{eps} \operatorname{cond}(A))]$$

верни значещи цифри, където $[u]$ е цялата част на числото $u > 0$.

Да разгледаме сега задачата за най-малки квадрати

$$Ax \approx b,\ A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m,\ x \in \mathbb{R}^n,\ m \gg n$$

която се формулира като оптимизационна задача за намиране на x, така че

$$\|Ax - b\| = \min.$$

Когато $\mathrm{rank}(A) < n$ решението на задачата не е единствено и можем да поставим допълнителното условие

$$\|x\| = \min .$$

Сега вече задачата има единствено решение. Наименованието на задачата идва от историята и се основава на наблюдението, че всъщност се минимизира сумата

$$\|r\|^2 = r_1^2 + \cdots + r_m^2$$

от квадратите на компонентите на т. нар. остатъчен вектор $r = Ax - b$.

Теоретически решението може да се пресметне от условието, че векторът $Ax - b$ трябва да е ортогонален на подпространството $\mathrm{Rg}(A)$, разпънато от стълбовете на A. Това дава

$$A^\top (Ax - b) = 0$$

или

$$A^\top Ax = A^\top b.$$

Това е т. нар. *нормална форма* на задачата за най-малки квадрати. На практика формирането на симетричната матрица $A^\top A$ се избягва защото може да доведе до загуба на ранг при извършване на пресмятанията в машинна аритметика. Действително, нека пресмятанията се извършват в машинна аритметика с мярка на закръгляне eps. Да разгледаме матрицата

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

където

$$\text{eps} \leq \varepsilon < \sqrt{\text{eps}}.$$

Матрицата A е от пълен ранг 2, който добре „се разпознава" от съвременните компютърни системи за матрични пресмятания.

Ако формираме матрицата

$$A^\top A = \begin{bmatrix} 1 & 1 \\ 1 & 1+\varepsilon^2 \end{bmatrix}$$

виждаме, че тя е също от ранг 2, но след записването ѝ в паметта на компютъра елементът в позиция (2,2) се закръгля до 1. Така машинната матрица

$$(A^\top A)^* = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

вече е от ранг 1 и нормалната система може даже да няма решение! Такъв ще бъде например случаят когато $b = [1, 1, 0]^\top$.

Да разгледаме решението на задачата с матрица A от пълен ранг n по метода на QR декомпозицията.

Нека

$$A = QR$$

е QR декомпозицията на A, където матрицата Q е ортогонална ($Q^\top Q = I_m$), а матрицата R има вида

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

Тук $R_1 \in \mathbb{R}^{n\times n}$ е горна триъгълна неособена матрица. Ако означим

$$c := Q^\top = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},\ c_1 \in \mathbb{R}^n$$

то имаме

$$\|Ax - b\| = \|QRx - b\| = \|Q(Rx - c)\| = \|Rx - c\| = \min.$$

Тъй като

$$\|Rx - c\|^2 = \|R_1 x - c_1\|^2 + \|c_2\|^2$$

то следва да определим $x = [x_1, \ldots, x_n]^\top$ като решение на уравнението

$$R_1 x = c_1 = [\gamma_1, \ldots, \gamma_n]^\top.$$

Понеже матрицата $R_1 = [\rho_{ij}]$ е горна триъгълна, това уравнение се решава директно чрез обратно заместване

$$\begin{aligned} x_n &= \frac{\gamma_n}{\rho_{nn}}, \\ x_{n-1} &= \frac{\gamma_{n-1} - \rho_{n-1,n} x_n}{\rho_{n-1,n-1}}, \\ &\cdots, \\ x_1 &= \frac{\gamma_1 - \rho_{12} x_2 - \cdots - \rho_{1n} x_n}{\rho_{11}}. \end{aligned}$$

При използване на тази изчислителна схема оценката за грешката в изчисленото решение x^* е

$$\delta := \frac{\|x^* - x\|}{\|x\|} \leq 6mnC\mathrm{cond}(A)\mathrm{eps},$$
$$C := 1 + \mathrm{cond}(A)\frac{\mathrm{cond}(A)\|Ax - b\| + \|b\|}{\|A\|\,\|x\|}$$

където

$$\mathrm{cond}(A) := \|A\|\,\|A^\dagger\|$$

и $A^\dagger \in \mathbb{R}^{n\times m}$ е псевдообратната матрица на матрицата A.

2.3 Задача за собствени стойности

Да разгледаме задачата за пресмятане на спектъра на матрицата $A \in \mathbb{R}^{n\times n}$. Нека λ е проста собствена стойност на A, на която съответстват нормираните десен u и ляв v собствени вектори, т.е.,

$$\begin{aligned} Au &= \lambda u,\ \|u\| = 1, \\ v^\top A &= \lambda v^\top, \|v\| = 1. \end{aligned}$$

2.3.1 Чуствителност

Нека δA е смущение в A, на което отговарят смущения $\delta\lambda$ в собствената стойност λ и δu, δv в собствените вектори u, v съответно. Имаме

$$\begin{aligned} (A+\delta A)(u+\delta u) &= (\lambda+\delta\lambda)(u+\delta u), \\ (A+\delta A)^\top(v+\delta v) &= (\lambda+\delta\lambda)(v+\delta v). \end{aligned}$$

С точност до малки от първи ред включително е в сила

$$(A-\lambda I_n)\delta u + \delta A u = \delta\lambda u.$$

Умножаваме това равенство отляво с $v^\top$ с отчитане на факта, че v е ляв собствен вектор на A, т.е.,

$$v^\top(A-\lambda I_n) = 0.$$

В резултат получаваме

$$v^\top \delta A u = (v^\top u)\delta\lambda.$$

Понеже λ е проста собствена стойност векторите u и v са линейно независими (неколинеарни) и $v^\top u \neq 0$. Оттук

$$\delta\lambda = \frac{v^\top \delta A u}{v^\top u}$$

и

$$|\delta\lambda| \leq \frac{\|\delta A\| \, \|u\| \, \|v\|}{|v^\top u|} = \frac{\|\delta A\|}{|v^\top u|}.$$

Така величината

$$K_\lambda := \frac{1}{|v^\top u|}$$

е абсолютното число на обусловеност на собствената стойност λ.

Когато $\lambda \neq 0$ имаме и оценка в термините на относителни смущения както следва

$$\delta_\lambda \leq k_\lambda \delta_A$$

където

$$\delta_\lambda := \frac{|\delta\lambda|}{|\lambda|}, \;\; \delta_A := \frac{\|\delta A\|}{\|A\|}$$

а

$$k_\lambda := K_\lambda \frac{\|A\|}{|\lambda|}$$

е относителното число на обусловеност на собствената стойност λ.

Числата на обусловеност на собствените стойности се наричат още спектрални числа на обусловеност на матрицата A.

2.3.2 Грешки

Когато λ^* е пресметнатата собствена стойност на матрицата A по QR метода в машинна аритметика с мярка на закръгляне eps, то в сила е оценката

$$\frac{|\lambda^* - \lambda|}{\|A\|} \leq \text{eps}\, \psi(n) K_\lambda$$

където $\psi(n)$ е полином от степен, ненадвишаваща 2.

Когато собствената стойност е k-кратна ($k > 1$) задачата е сингулярна, потенциално е силно чувствителна и е възможно в изчисленото решение да има абсолютни грешки от порядъка на

$$C\,(\text{eps})^{1/k}$$

където $C \geq 1$ е константа. Същевременно относителните грешки могат да се окажат недопустимо големи.

Действително, да разгледаме $n \times n$ матрицата

$$A = \begin{bmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & \lambda \end{bmatrix}$$

която има n–кратна собствена стойност λ. Да смутим матрицата до $A + E$, където E има единствен ненулев елемент ε в позиция $(n, 1)$. Тогава смутената матрица $A + E$ има характеристично уравнение

$$(z - \lambda)^n = (-1)^n \varepsilon$$

и собствени стойности от вида

$$z = \lambda + (-1)^n \varepsilon^{1/n}.$$

Така смущение в матрицата с норма $|\varepsilon|$ предизвиква смущение в собствените стойности с големина $|\varepsilon|^{1/n}$. Нека например $n = 16$, $\lambda = 0.01$ и $\varepsilon = 10^{-16}$, което е от порядъка на грешката от закръгляне в този случай. Тогава изменението в собствените стойности е с големина 0.1. Така за матрица със съвсем скромен размер грешките от закръгляне предизвикаха 1000 на сто относително изменение в собствените стойности.

2.4 Пресмятане на ранг на матрица

Пресмятането на ранга на дадена матрица $A \in \mathbb{R}^{m\times n}$ в машинна аритметика с мярка на закръгляне eps е деликатна задача. Действително, ако рангът $r = \mathrm{rank}(A)$ на A не е пълен, т.е., $r < p := \min\{m, n\}$, то произволно малко смущение δA в A може да вдигне ранга до p:

$$\mathrm{rank}(A + \delta A) = p.$$

Действително, нека декомпозицията по сингулярни стойности на матрицата A е

$$A = USV^\top,$$
$$S = S(A) = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0),\ \sigma_1 \geq \cdots \geq \sigma_r$$

където U и V са ортогонални матрици,

$$U^\top U = I_m,\ V^\top V = I_n,$$

и

$$\sigma_i = \sigma_i(A)$$

са сингулярните стойности на A.

Смущението

$$\delta A = U\mathrm{diag}(0, \ldots, 0, \eta_{r+1}, \ldots, \eta_p)V^\top$$

с

$$\eta_{r+1} = \ldots = \eta_p = \frac{\varepsilon}{\sqrt{p-r}}$$

има норма на Фробениус ε, където $\varepsilon > 0$ може да е произволно малко. Същевременно лесно се проверява, че смутената матрица $A + \delta A$ е от пълен ранг p.

Така показахме, че рангът на дадена матрица е неустойчива нагоре числова характеристика. От друга страна още при записване на матрицата в паметта на компютъра ние получаваме закръглената матрица A^*, такава че

$$\|A^* - A\| \leq \mathrm{eps}\|A\|.$$

По този начин закръглената матрица може да се окаже от пълен ранг дори когато първоначалната матрица не е.

Така впрочем достигнахме до парадоксалната ситуация рангът на една плътна матрица с големи размери да се окаже непознаваема величина. Действително, тъй като матрицата по условие е голяма, рангът й може да се търси само с компютър. Но същият този ранг по принцип се разрушава в момента на записване на матрицата в паметта на компютъра. Има един

радикален изход от това положение и той е да признаем, че „теоретичният" ранг на матрица (определен например като броя на линейно независимите й редове или стълбове) няма смисъл при големи плътни матрици (!)

Най-надеждният начин за пресмятане на ранга на една матрица е чрез броя на нейните ненулеви сингулярни стойности. Една от причините за това е, че тези стойности са слабо чувствителни относно смущения в матрицата. Като се използва декомпозицията по сингулярни стойности на A и унитарната инвариантност на матричните норми $\|\cdot\| = \|\cdot\|_2$ и $\|\cdot\|_{\mathrm{F}}$ може да се докаже следното неравенство (известно като теорема на Мирски):

$$\|S(A+\delta A) - S(A)\| \leq \|\delta A\|.$$

В частност имаме

$$\sqrt{\sum_{i=1}^{p}(\sigma_i(A+\delta A) - \sigma_i(A))^2} \leq \|\delta A\|_{\mathrm{F}}$$

и

$$\max\{|\sigma_i(A+\delta A) - \sigma_i(A)| : i = 1, \ldots, p\} \leq \|\delta A\|_2.$$

Следователно сингулярните стойности на всяка матрица са много добре обусловени с абсолютно число на обусловеност 1.

Да предположим, че

$$\sigma_1^* \geq \sigma_2^* \geq \cdots \geq \sigma_p^*$$

са пресметнатите в машинна аритметика сингулярни стойности на матрицата A. Ако сред тях има малки, то може да се

очаква, че някои от тях се дължат на грешки от закръгляне, а не на действителния ранг на първоначалната матрица A. Когато сингулярните числа са групирани в два сравнително добре обособени кластера: „големи" и „малки", то задачата е лесна – можем да приемем, че рангът на матрицата е равен на броя на големите сингулярни стойности. Значително по-трудно е да оценим ранга в случай, че сингулярните стойности формират редица от плавно намаляващи числа, например $\sigma_i^* \approx 10^{-i}$. Така възниква задачата да отсеем придобитите ненулеви сингулярни стойности, които изкуствено (вероятно) са повишили ранга на матрицата. За целта се въвежда понятието *числен ранг* на матрица.

Нека $\tau > 0$ е зададен (малък) числов праг. Ще казваме, че матрицата A е от числен ранг s (с точност до τ), ако тя има s на брой сингулярни числа, по-големи или равни на $\tau\|A\|$, а останалите й ненулеви сингулярни числа (ако има такива) са по-малки от $\tau\|A\|$, т.е.

$$\sigma_s \geq \mathrm{eps}\|A\|,\ \sigma_{s+1} < \tau\|A\|.$$

В реални условия ние не знаем точно сингулярните стойности σ_i на A, а разполагаме само с изчислените сингулярни стойности σ_i^*. Поради това за числен ранг на матрицата A в машинна аритметика с мярка на закръгляне eps се приема броят на числата σ_i^*, по-големи от $\mathrm{eps}\,\sigma_1^*$. Има и по-консервативни оценители на ранга, при които в качеството на долна граница за сингулярните стойности, които носят ранг, се приема $\mathrm{eps}\,n\,\sigma_1^*$ или даже $\mathrm{eps}\,n^2\,\sigma_1^*$, където n е редът на матрицата.

2.5 Задачи

2.1 Намерете в явен вид числата на обусловеност на неособената матрица

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{F}^{2\times 2}$$

където $\mathbb{F} = \mathbb{R}$ или $\mathbb{F} = \mathbb{C}$, в нормата на Фробениус и в спектралната норма.

2.2 Ако A и B са неособени матрици с еднакви размери докажете тъждеството

$$B^{-1} - A^{-1} = B^{-1}(A - B)A^{-1}.$$

2.3 Ако A е неособена матрица и δA е смущение в A, такова че

$$\|A\|\,\|\delta A\| < 1$$

покажете, че са в сила оценките

$$\frac{\|(A+\delta A)^{-1} - A^{-1}\|}{\|A^{-1}\|} \le \frac{c\,\delta_A}{1 - c\,\delta_A} = \frac{\|A^{-1}\|\,\|\delta A\|}{1 - \|A^{-1}\|\,\|\delta A\|},$$
$$\frac{\|(A+\delta A)^{-1} - A^{-1}\|}{\|(A+\delta A)^{-1}\|} \le c\,\delta_A = \|A^{-1}\|\,\|\delta A\|.$$

За първата оценка използвайте представената по-горе техника като считате x и b за матрици и приемете $b = I_n$. Изведете втората оценка непосредствено с използване на тъждеството от задача 2.2.

2.4 Нека $|A| = [|a_{ij}|]$ е матричният модул на матрицата A, т.е., матрицата, съставена от модулите на елементите на A.

Разгледайте уравнението $Ax = b$ и неговата смутена версия

$$(A + \delta A)(x + \delta x) = b + \delta b$$

с неособена $n \times n$ матрица A. Докажете, че ако собствените стойности на матрицата

$$|A^{-1}|\,|\delta A|$$

са по модул по-малки от 1, то е в сила неравенството

$$|\delta x| \preceq \left(I_n - |A^{-1}|\,|\delta A|\right)|A^{-1}|(|\delta b| + |\delta A|\,|x|)$$

което се разбира покомпонентно.

2.5 Нека x^* е приближено решение на уравнението

$$Ax = b$$

с квадратна матрица A. Определете при какви най-малки смущения δA, δb векторът x^* е точно решение на смутеното уравнение с данни $A + \delta A$, $b + \delta b$, т.е.,

$$(A + \delta A)x^* = b + \delta b.$$

За целта намерете т. нар. *обратна грешка*

$$\min\{\|[\delta A, \delta b]\|_{\mathrm{F}} : (A + \delta A)x^* = b + \delta b\}$$

като решите задачата за най-малки квадрати

$$\delta A x^* - \delta b = b - Ax^*$$

относно неизвестните δA, δb.

2.6 Покажете, че за неособената $n \times n$ матрица B и ненулевия n–вектор y са в сила зависимостите

$$\sigma_{\min}(B)\|y\| \leq \|By\| \leq \sigma_{\max}(B)\|y\|$$

и

$$\frac{1}{\|B\|\,\|y\|} \leq \frac{1}{\|By\|} \leq \frac{\|B^{-1}\|}{\|y\|}.$$

Ще припомним, че работим в 2–нормата, при което

$$\sigma_{\min}(B) = \frac{1}{\|B^{-1}\|},\ \sigma_{\max}(B) = \|B\|.$$

2.7 Разгледайте линейното алгебрично уравнение $Ax = b$ с квадратна неособена матрица A под влиянието на смущения δA, δb, такива че матрицата $A + \delta A$ е неособена и $\|\delta b\| < \|b\|$. Покажете, че векторът $b + \delta b \neq 0$ и решението

$$x + \delta x = (A + \delta A)^{-1}(b + \delta b)$$

на смутената система са ненулеви. Обосновете оценката

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \frac{c(\delta_A + \delta_b)}{1 - \delta_b}.$$

Може ли да получите по-добра оценка от този вид?

Глава 3

Елементи на числения математически анализ

3.1 Пресмятане на стойностите на функция

Пресмятането на стойностите на дадена функция

$$x \mapsto y = f(x)$$

в машинна аритметика може да е свързано със значителни трудности и големи грешки. Да означим с y^* резултата за y, пресметнат в машинна аритметика от тип $(\mathrm{eps}, X_{\max})$. Нека за простота x и y са скалари. Случаят на векторна функция на векторен аргумент се разглежда аналогично. Ще предполагаме също, че функцията f е диференцируема в околност на аргумента x.

Два са факторите, от които зависят абсолютната

$$E := |y^* - y|$$

и относителната

$$e := \frac{|y^* - y|}{|y|}$$

грешки (последната има смисъл само при $y \neq 0$). Ще разглеждаме тези фактори при предположение, че

$$X_{\min} < |x|, |y| < X_{\max}.$$

Първият фактор е свързан с обстоятелството, че ако аргументът x не е машинно число, то в машинна аритметика се използва неговата закръглена стойност

$$x^* = x(1 + \varepsilon), \ |\varepsilon| \leq \text{eps}.$$

Така още при записване на x в паметта на компютъра се правят абсолютна грешка

$$|x^* - x| \leq \text{eps}\,|x|$$

и относителна грешка

$$\frac{|x^* - x|}{|x|} \leq \text{eps}.$$

Следователно в най-добрия случай може да очакваме, че относителната грешка в пресметнатия резултат няма да надхвърля значително мярката на закръгляне eps.

Вторият източник на грешки е закръглянето при изпълнение на аритметичните операции за пресмятане на $f(x^*)$ от входните данни x^*. Да означим с $f^*(x^*)$ резултата от машинното изпълнение на алгоритъма $x^* \mapsto f^*(x^*)$. Тогава имаме

$$y^* = f^*(x^*).$$

За абсолютната грешка получаваме

$$\begin{aligned} E &= |f^*(x^*) - f(x)| = |f^*(x^*) - f(x^*) + f(x^*) - f(x)| \\ &\leq |f(x^*) - f(x)| + |f^*(x^*) - f(x^*)| = E_1 + E_2 \end{aligned}$$

където E_1 и E_2 са оценки за приноса на горните два източника в абсолютната грешка E.

С точност до малки от първи ред относно eps имаме

$$E_1 \leq |f'(x)|\,|x^* - x| \leq \mathrm{eps}|f'(x)|\,|x|$$

и

$$E_2 \leq \mathrm{eps}\,\Gamma$$

където константата $\Gamma > 0$ зависи от функцията f, аргумента x и от използвания алгоритъм.

Оттук получаваме

$$E \leq \mathrm{eps}(|f'(x)|\,|x| + \Gamma)$$

и

$$e \leq \mathrm{eps}(\kappa(f,x) + \gamma),\ y = f(x) \neq 0.$$

Тук

$$\kappa(f,x) := \frac{|f'(x)|\,|x|}{|f(x)|}$$

е относителното число на обусловеност при пресмятане на f в точката x, а

$$\gamma := \frac{\Gamma}{|f(x)|}.$$

Ще отбележим, че докато $\kappa(f, x)$ е характеристика само на задачата (т.е., на функцията f и аргумента x), то γ зависи и от избрания изчислителен алгоритъм.

От горния анализ следва правилото

- *Относителната грешка е при пресмятане на стойността $f(x)$ на функцията f в точката x може да е голяма когато модулите на аргумента $|x|$ и/или на производната $|f'(x)|$ са големи, или когато стойността $f(x)$ на функцията е малка.*

3.2 Числено диференциране

Диференцирането на функция

$$f : (a, b) \to \mathbb{R}$$

в дадена точка $x \in (a, b)$ е деликатна операция, която често се натъква на теоретични и числени трудности. Тук накратко ще разгледаме въпроса за числено диференциране, което се състои в намиране на приближение на производната $f'(x)$ при извършване на операциите в машинна аритметика с мярка на закръгляне eps. Числената апроксимация на производната се извършва на два етапа.

На първия етап се избира приближена формула за пресмятане на $f'(x)$ като отношение на крайни нараствания, например

$$f'(x) \approx \frac{\Delta f(x)}{\Delta x}$$

където вида на нарастванията $\Delta f(x)$ и Δx зависи от избраната приближена формула. При това се внася т. нар. *грешка от апроксимация*, E_T за която е в сила

$$E_T = O(\Delta x^k),\ \Delta x \to 0.$$

Тук k е натурално число, което определя реда на приближената формула.

На втория етап се пресмятат нарастванията $\Delta f(x)$ и Δx така, че грешките от закръгляне да се минимизират и да не се допусне съществено взаимно унищожение в резултата за $\Delta f(x)$ (известно взаимно унищожение е неизбежно при числено пресмятане на производни). В резултат се прави грешка E_R, в която са сумирани ефектите от закръгленията и взаимното унищожение.

Глобалната грешка е

$$E = E_T + E_R$$

и тя зависи от f, x, Δx, от избрания изчислителен алгоритъм и от параметрите на машинната аритметика (предимно от мярката на закръгляне eps). Именно на втория етап се избира такова нарастване за x, при което сумарната грешка е минимална.

По аналогичен начин се пресмятат числено и производни от по-висок ред.

Важно е да се знае, че наивната стратегия просто да се намалява Δx без да се държи сметка за грешките от закръгляне и от взаимно унищожение скоро довежда до изчислителна катастрофа, вж. задачи 3.3 и 3.4.

Като евристично правило за определяне на Δx имаме

$$\Delta x \approx \mathrm{eps}^{\frac{1}{k+1}}.$$

Нека функцията f има достатъчен брой производни в интервала (a, b).

За апроксимация на производните на f в дадена точка x могат да се използват различни подходи. Особено прозрачен е подходът, основан на формулите на Тейлър, които записваме във вида

$$f(x+h) = f(x) + f'(x)h + f''(x+\lambda h)\frac{h^2}{2},\ \lambda \in (0,1),$$
$$f(x-h) = f(x) - f'(x)h + f''(x-\mu h)\frac{h^2}{2},\ \mu \in (0,1).$$

Оттук

$$f'(x) = \frac{f(x+h) - f(x)}{h} - f''(x+\lambda h)\frac{h}{2},$$
$$f'(x) = \frac{f(x) - f(x-h)}{h} + f''(x-\mu h)\frac{h}{2}.$$

Така получаваме формулите за диференциране напред

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

и назад

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}.$$

В двата случая грешката от апроксимация не надхвърля

$$\frac{c_2(f)h}{2}$$

където

$$c_n(f) := \sup\{|f^{(n)}(x)| : x \in (a,b)\}.$$

Една по-точна оценка се получава като приложим формулите на Тейлър във вида

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x+\lambda h)\frac{h^3}{6},\ \lambda \in (0,1), \\ f(x-h) &= f(x) - f'(x)h + f''(x)\frac{h^2}{2} - f'''(x-\mu h)\frac{h^3}{6},\ \mu \in (0,1) \end{aligned}$$

Като от първата зависимост извадим втората и разделим на $2h$ получаваме

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - (f'''(x+\lambda h) + f'''(x-\mu h)\frac{h^2}{12}.$$

Следователно приближената формула

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3.1}$$

се характеризира с грешка от апроксимация, ненадхвърляща

$$\frac{c_3(f)h^2}{6}.$$

Приближеното числено пресмятане на производни се натъква на два типа трудности.

Първо, операцията диференциране не е ограничена, т.е., производната на дадена функция може да е произволно голяма дори когато функцията е равномерно ограничена. В термините на горните оценки това означава, че константите $c_n(f)$ могат да се окажат много големи, което на свой ред изисква много малко нарастване h.

Сега обаче се появява втората трудност – нарастването h не може да се избира прекалено малко защото ще настъпи катастрофално взаимно унищожение. Действително, с намаляване на h величините $f(x+h)$ и $f(x-h)$ стават все по–близки и когато те са с еднакъв знак (което е типичният случай) ще нараства броят на загубените при изваждането $f(x+h)-f(x-h)$ верни значещи цифри. Тъй като величините $f(x \pm h)$ на свой ред са пресметнати с грешки от закръгляне, а резултатът от изваждането се дели на малкото число $2h$ то е ясно, че опасността от изчислителна катастрофа е съвсем реална. И тази катастрофа неизменно настъпва когато не се вземат мерки за нейното предотвратяване.

В [3] е показано, че относителната грешка при пресмятане на производната $f'(x) \neq 0$ по формулата (3.1) в машинна аритметика се оценява отгоре с величината

$$\widehat{e}(h, \mathrm{eps}) := \frac{b_1 h^2}{2} + b_2 \frac{\mathrm{eps}}{h} + 2\mathrm{eps}$$

където

$$b_1 := \frac{c_3(f)}{3|f'(x)|},$$

$$b_2 \quad := \quad |x| + \gamma \frac{|f(x)|}{|f'(x)|}.$$

Функцията $\widehat{e}$ има минимум по h при

$$h = \widehat{h} := \left(\frac{b_2}{b_1} \mathrm{eps} \right)^{1/3} = O(\mathrm{eps}^{1/3}).$$

Оттук следва важният извод, че при извършване на пресмятанията в стандартната машинна аритметика с

$$\mathrm{eps} \approx 10^{-16}$$

оптималното нарастване е от 10^{-5} до 10^{-6}. Използването на по-малко нарастване ще влоши рязко точността на изчисленото решение.

Адаптивен алгоритъм и надеждна компютърна програма SDERF на ФОРТРАН 77 за числено диференциране са разгледани в [3]. Пак там са дадени апроксимационни формули за пресмятане на производни от произволен ред и с произволно малка грешка от апроксимация.

3.3 Числено интегриране

Една от фундаменталните задачи в математическия анализ е пресмятането на определения интеграл

$$I(f) := \int_a^b f(x) \mathrm{d}x \, .$$

Много често това не може или не е целесъобразно да стане по аналитичен път и се налага интегралът да се пресмята числено в машинна аритметика. За целта първо се избира т. нар.

квадратурна формула $Q(f)$, която представлява теоретична апроксимация на определения интеграл. Читателят следва да е запознат с основните квадратурни формули от курса по математически анализ. При извършване на сметките в машинна аритметика вместо $Q(f)$ се получава приближението $Q^*(f)$. Тогава задачата е се пресметне такава апроксимация $Q^*(f)$, че да е изпълнено

$$|Q^*(f) - I(f)| \leq D$$

или

$$\frac{|Q^*(f) - I(f)|}{|I(f)|} \leq d.$$

Тук D и d са зададени горни граници за абсолютната и относителната грешка съответно.

За абсолютната грешка имаме

$$\begin{aligned} &|Q^*(f) - I(f)| = |Q^*(f) - Q(f) + Q(f) - I(f)| \\ &\leq |Q^*(f) - Q(f)| + |Q(f) - I(f)| = E_R + E_T. \end{aligned}$$

Тук E_R е грешката от закръгляне при пресмятане на $Q(f)$ в машинна аритметика, а E_T е грешката от апроксимация, която се получава при приближаване на $I(f)$ чрез $Q(f)$.

Обикновено квадратурната формула се основава на разбиване на интервала J на подинтервали $J_1, \ldots, J_n$ и използване на локални квадратурни формули във всеки подинтервал. В най-простия случай подинтервалите са с еднаква дължина

$$h := \frac{b-a}{n}.$$

Тогава грешката от апроксимация удовлетворява оценка от вида

$$E_T \leq \widehat{E}_T = \widehat{E}_T(h) := c_1 h^k, \ h \to 0$$

където където k зависи от избраната квадратурна схема.

Анализ на общата абсолютна грешка при числено интегриране е даден в [3]. Оценката за тази грешка има вида

$$\begin{aligned} \widehat{E} &= \widehat{E}(h) = c_1(b-a)h^k + \mathrm{eps}(b-a)\left(c_2 + \frac{c_3}{h} + c_4 h^{k-1}\right) \\ &= \widehat{E}_T(h) + \widehat{E}_R(h, \mathrm{eps}) \end{aligned}$$

където $c_1, \ldots, c_4$ са коефициенти, зависещи от метода за апроксимация и от функцията f.

На практика числено интегриране се извършва чрез адаптивни алгоритми, които се приспособяват към поведението на функцията f в различните части на интервала $[a, b]$. Локално адаптивна схема за числено интегриране е предложена в [13], като е приложена и компютърната програма QUANC8 на стандартен ФОРТРАН 77. Глобално адаптивен интеграционен алгоритъм е даден в [3] заедно със съответната програма SQGGK на ФОРТРАН 77. Надеждни алгоритми за числено интегриране са вградени в компютърните системи MATLAB, MATHEMATICA, MAPLE, GAUSS, REDUCE и др., предназначени за числени и символни научни и инженерни пресмятания.

3.4 Задачи

3.1 Напишете алгоритъм, който пресмята стойностите на тригонометричните функции

$$\sin(x),\ \cos(x), \tan(x),\ \cot(x)$$

за всяка стойност на аргумента x чрез съответните стойности

$$\sin(y),\ \cos(y), \tan(y),\ \cot(y)$$

за

$$0 \leq y \leq \frac{\pi}{2}.$$

Направете същото за

$$0 \leq y \leq \frac{\pi}{4}.$$

3.2 Напишете алгоритъм, който пресмята стойностите на показателната функция

$$\exp(x)$$

за всяка стойност на аргумента x чрез съответните стойности

$$\exp(y)$$

за

$$0 \leq y \leq 1.$$

Направете същото за

$$0 \leq y \leq 0.5.$$

3.3 Напишете програма за пресмятане на производната на $\sin(x)$ в точката $x = \pi/4$ като използвате формулата

$$\frac{\sin(\pi/4 + h) - \sqrt{2}/2}{h}$$

за

$$h = 10^{-k},\ k = 1, 2, \ldots.$$

За сведение, точният отговор е $\sqrt{2}/2$. За кое k настъпи изчислителна катастрофа? Сравнете съответното h с eps.

3.4 Напишете програма за пресмятане на производната на $\exp(x)$ в точката $x = 1$ като използвате формулата

$$\frac{\exp(1 + h) - e}{h},\ e = \exp(1) = 2.72\ldots$$

за

$$h = 10^{-k},\ k = 1, 2, \ldots.$$

Точният отговор е e. За кое k настъпи изчислителна катастрофа? Сравнете съответното h с eps.

3.5 Направете аналогични експерименти с други елементарни функции f като използвате приближената формула

$$f'(x) \approx \frac{f(x + h) - f(x)}{h}$$

за намаляваща редица от стойности за h. Въз основа на задачи 3.3, 3.4 и 3.5 формулирайте хипотеза, включваща мярката на закръгляне eps.

3.6 Интегралът

$$I_n = \int_0^1 f(x)\mathrm{d}x := \int_0^1 \frac{\mathrm{d}x}{1 + x^n}$$

може да се пресметне с използване на формулата на Нютън-Лайбниц

$$I = F(1) - F(0)$$

за всяко цяло $n \geq 0$, където F е някоя примитивна на функцията f,

$$F'(x) = \frac{1}{1+x^n}.$$

Така например за малки n имаме

$$I_0 = \int_0^1 \frac{\mathrm{d}x}{2} = \frac{1}{2},$$
$$I_1 = \int_0^1 \frac{\mathrm{d}x}{1+x} = \ln(1+x)|_0^1 = \ln 2 - \ln 1 = \ln 2,$$
$$I_2 = \int_0^1 \frac{\mathrm{d}x}{1+x^2} = \mathrm{arctg}x|_0^1 = \frac{\pi}{4}.$$

При $n \geq 3$, обаче, е по-добре да се обърнем към някои справочник (освен ако интегрирането не ни е хоби). С нарастване на n изразите стават крайно неприятни. За $n = 10$ едва ли и в справочника ще намерим нещо. Пресметнете символно примитивната за $n = 10$, и, ако имате търпение, за $n = 25$ с използване например на MATHEMATICA.

Този пример трябва да ни убеди, че и при наличие на решение в затворена форма някои определени интеграли е по-добре да се решават числено.

Глава 4

Митове в изчислителната практика

4.1 Митове

В изчислителната практика съществуват устойчиви митове, разпространени дори сред някои квалифицирани ползватели на компютри за извършване на научни и инженерни пресмятания. Какво впрочем е научният мит? Това е твърдение, което има известно логическо или историческо основание, но понякога просто не е вярно.

1 *Големите грешки в изчисленото решение се дължат на сумарния ефект на голям брой малки грешки от зак-*

ръгляне, които се правят при отделните аритметични операции.

Това е вярно рядко и само отчасти. Обикновено при извършване на голям брой еднотипни аритметични операции грешките не се натрупват еднопосочно, а имат тенденция взаимно да се компенсират. Ако все пак сме получили голяма грешка в крайния резултат това най-вероятно е станало при изпълнението на малък брой критични операции, при които се е случило изчислително нещастие. Понякога за това стига една единствена операция, при която например е настъпило катастрофално взаимно унищожение на верни разряди и съответно фатална загуба на точност.

Има и един юнашки мит, който показва непознаване на въпроса. Това е твърдение от типа на

2 *Грешките от закръгляне нямат значение, защото са малки (вариант: защото взаимно се унищожават).*

Привържениците на този мит все пак показват, че са чували за съществуването на грешки от закръгляне. Защото има и ползватели, които и хабер си нямат за това явление.[1]

Интересен е следващият мит.

3 *Изваждането на близки числа с помощта на компютър е опасно, защото относителната грешка при изваждането е голяма.*

[1] Те вероятно са по-щастливата част от ползвателите. Но това е до време.

Тук има нещо вярно: относителната грешка при изваждане е голяма (и даже е неограничена когато числата са произволно близки). Но проблемът, както знаем, е друг и е свързан със загуба на информация при вазимното унищожаване на левите разряди в позиционния запис на числата. При това самата грешка от изваждането е почти без значение! И никаква грешка да не се правеше при изваждането, информацията пак щеше да се загуби. Впрочем, при достатъчно близки машинни числа изваждането се прави практически без грешка.

Да разгледаме сега някои митове, свързани с решаването на линейни и нелинейни уравнения, например

$$f(x) = 0.$$

Тук x и $f(x)$ са скалари или вектори с еднакъв размер. За функцията f се предполага, че е непрекъсната.

Нека x^* е пресметнатото решение в машинна аритметика. Векторът $f(x^*)$ се нарича остатъчен вектор, а величината

$$\rho(x^*) := \|f(x^*)\|$$

се нарича *остатък*.

Функцията ρ е непрекъсната, а за точното решение x имаме $\rho(x) = 0$. Тези два факта са породили множество митове, на някои от които ще се спрем.

4 *Точността на пресметнатото решение x^* може да се провери чрез големината на остатъка $\rho(x^*)$ – колкото по-малък е остатъкът, толкова решението е по-точно.*

Този мит си има брат–близнак:

5 *От две приближени решения по–близко до точното е онова решение, което има по-малък остатък.*

Очевидно митове 4 и 5 са еквивалентни. И съответно еднакво неверни.

Лесно се показва, че тези митове пропадат при нелинейни уравнения, например

$$f(x) := x^3 - 23.001x^2 + 143.022x - 121.021 = 0.$$

Уравнението има единствен реален корен $x = 1$. Да разгледаме две приближения на този корен: $x_1 = 0.99$ и $x_2 = 11.00$. Впрочем, само първото може да претендира, че е приближение. Ако пресметнем остатъците констатираме, че

$$\rho(x_1^*) = 1.0022,\ \rho(x_2^*) = 0.1.$$

Така „лошото" решение x_2^* с относителна грешка от 1000 на сто се оказа с 10 пъти по-малък остатък от „доброто" решение x_1^* с относителна грещка от 1 на сто! Нещата, естествено, могат още да загрубеят, вж. задача 4.2.

Читателят вече има идея защо митове 4 и 5 пропадат при нелинейни уравнения. Но може би нещо от тях може да се спаси при линейните уравнения? Уви, и това не е така.

Да разгледаме линейното алгебрично уравнение $Ax = b$, където

$$A = \begin{bmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{bmatrix},\ b = \begin{bmatrix} 0.1440 \\ 0.8642 \end{bmatrix}.$$

Приближеното решение

$$x^* = \begin{bmatrix} 0.9911 \\ 0.4870 \end{bmatrix}$$

дава малък остатък

$$\rho(x^*) = \|Ax - b\| = 0.1414 \times 10^{-7}$$

и според мит 4 би трябвало да е близко до точното. Да, но не, тъй като точното решение е

$$x = \begin{bmatrix} 2 \\ -2 \end{bmatrix}.$$

Виждаме, че в x^* даже няма вярна значеща цифра, което е обяснимо, тъй като относителната грешка тук е

$$\frac{\|x^* - x\|}{\|x\|} = 0.643.$$

Този пример е забележителен с това, че *всички* приближени решения, на които първите три цифри съвпадат с тези на точното решение, дават по-големи остатъци в сравнение с остатъка при x^*!

Този феномен при линейните уравнения е обяснен в [3]. Накратко, това е възможно при уравнения с лошо обусловена матрица A, за която числото на обусловеност

$$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\|$$

е голямо. Това наблюдение впрочем е в сила и при нелинейните уравнения.

Така че има доста ирония в историята с митове 4 и 5: проверката на точността на решението чрез големината на остатъка може да се окаже успешна само ако уравнението е добре обусловено. Но тогава и пресметнатото решение вероятно е точно, така че няма какво да се проверява. Ако обаче уравнението е лошо обусловено и има опасност от големи грешки в изчисленото решение, проверката с остатъка е подвеждаща. Лошото е, че 80 на сто от интелигентните ползватели на компютри за числени пресмятания вярват в тези митове. А и в много от книгите по числени пресмятания няма предупреждения на тази тема.

Има и някои доста „усъвършенствани" митове, които често са даже полезни. Но не винаги – те затова са и митове.

6 *Надежден начин да се провери точността на решението е да се повторят сметките с удвоена (или друга разширена) точност.*

Мит 6 си има и рецептурен вариант.

7 *Ако при повторно извършване на сметките с удвоена (или друга разширена) точност първите няколко цифри в изчислените по двата начина решения съвпаднат, то тези цифри са и верни.*

Защо иначе убедителните твърдения 6 и 7 са все пак митове е обяснено в [3]. Вярно е, че ако се работи с произволно малка мярка за закръгляне eps може да се постигнат произволен брой

верни значещи цифри в резултата. Както впрочем е вярно, че цената за това е намаляване на бързодействието произволен брой пъти, вж. задача 4.3

Опитните ползватели на компютри, а надяваме се вече и читателите на тези записки знаят, че ако малки смущения в данните предизвикват големи смущения в резултата (т.е., ако задачата е силно чуствителна, съответно лошо обусловена), то изчисленото решение е със съмнителна точност. Понякога обаче това иначе вярно наблюдение се обръща като се приема, че ако дадени малки смущения в данните променят слабо резултата, то всичко е наред и точността на пресметнатото решение е приемлива. Така стигаме до поредния мит, причината за който е в думичката „дадени" в предходното изречение. Ако вместо нея беше „всички" нямаше да има проблем. Но и никой не може да опита всички възможни смущения, най–малкото поради липса на време.

8 *Ако проверката на резултата чрез повторното му пресмятане с леко смутени данни покаже малко изменение в решението, това е гаранция за надеждност на изчисленията и за постигната задоволителна точност.*

Контрапример за този мит е даден в [3]. Причината е следната. В някои силно чуствителни задачи има „нечуствителни направления" (при нелинейните задачи това са многообразия), такива че изменението на данните по тези направления променя слабо резултата. Същевременно едно малко изменение на данните по други направления може да промени резултата

значително.

Дали все пак не може да спасим нещо от митове 6–8? Оказва се, че отговорът е почти да. Следващото твърдение не е мит, а полезна евристика.

Ако проверката на резултата чрез няколко набора случайно генерирани малки смущения в данните и няколко машинни аритметики с различни мерки на закръгляне покаже малки смущения в изчисленото решение, то с висока степен на достоверност можем да очакваме, че това изчислено решение е близко до точното.

Разбира се, и на тази евристика може да й се намери контрапример – тя затова е и евристика, а не теорема. Само че читателят едва ли ще срещне такъв пример в изчислителната си практика.

4.2 Задачи

4.1 Решаването на линейното векторно алгебрично уравнение

$$Ax = b,\ A \in \mathbb{R}^{n\times n},\ x, b \in \mathbb{R}^n$$

с неособена матрица $A = [a_{ij}]$ се интерпретира геометрично като намиране на пресечната точка x на множество от n равнини

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i,\ i = 1, \ldots, n$$

в общо положение. В случая $n = 2$ става въпрос за пресичане на две неуспоредни прави. Нека $y, z \in \mathbb{R}^2$ са две приближени

решения на уравнението. Дайте графична илюстрация на факта, че y може да е произволен брой пъти по-близо до точното решение x в сравнение със z, като в същото време остатъкът за x е произволен брой пъти по-голям от остатъка за z. Какво става с числото на обусловеност

$$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\|$$

в този случай?

4.2 Конструирайте нелинейно уравнение с единствено реално решение x и две приближени решения y, z, такива че y е произволно близо до x, но има произволно голям остатък, докато z е произволно далеч от x, но има произволно малък остатък.

4.3 С помощта на системата за математически пресмятания MATHEMATICA (или MAPLE, или GAUSS, или символния тулбокс на MATLAB) пресметнете числото π с 1000, 10000 и 100000 знака и засечете необходимото за това време. Може впрочем и с 500000 хиляди знака, но сигурно няма да имате търпение да го изчакате.

Глава 5

Литературна справка

Съществува обширна литература по числен анализ в контекста на прилагането на изчислителни алгоритми в крайна аритметика. Някои общи проблеми на числените методи и числения анализ в частност са разгледани в [3, 4, 7, 12, 13, 14, 16, 17, 1, 8, 19, 23, 24, 26, 22]. Книгата [22] е посветена на изчислителните методи за линейни управляеми системи, но третира и някои общи постановки на числения анализ. Числените методи в линейната алгебра и теория на матриците са представени в [1, 2, 6, 8, 9, 11, 15, 17, 20].

Добър увод в пертурбационната теория на матриците е книгата [25]. Редица резултати в тази област са представени и в [6, 11, 13, 18, 19, 22].

Диалоговата система SYSLAB, разработена от колектив в състав П. Петков, Н. Христов и автора [5, 10], широко се използва за учебни и научни цели в техническите и икономическите

университети и БАН, още повече че софтуерът за нея се разпространява свободно. Системата се основава на надеждни и числено устойчиви матрични алгоритми.

Особено място в научната литература по числен анализ от гледна точка на крайните аритметики заемат статията [21] и монографиите [27, 28]. Известно е, че Дж. Фон Нойман е един от създателите на съвременните цифрови компютри. По-малко известен е фактът, че още през 1947, заедно с Х. Голдщайн [21], той има сериозни приноси в създаването и на съвременния числен анализ.

Съществен принос в числения анализ област има и Дж. Уилкинсън. В редица работи, вкл. книгите [27, 28], той въвежда понятието за числена устойчивост назад, което дава сериозен тласък в развитието на числения анализ.

Библиография

[1] В. Воеводин. *Вычислительные основы линейной алгебры.* Наука, Москва 1977.

[2] В. Воеводин, Ю. Кузнецов. *Матрицы и вычисления.* Наука, Москва 1984.

[3] Н. Вълчанов, М. Константинов. *Съвременни математически методи за компютърни пресмятания. Част 1: Основи на компютърните пресмятания. Числено диференциране и интегриране.* Студии на БИАП по математически науки, т. 1. ЕЛМА, София 1996.

[4] М. Константинов. Пертурбационен и числен анализ на инженерни изчислителни задачи. *Строителство*, т. 39 (1992), 6, стр. 32–29.

[5] М. Константинов, П. Петков, Н. Христов. *Матрични пресмятания с диалоговата система SYSLAB.* Изд. ВИАС, София 1992.

[6] М. Константинов, Н. Вълчанов. *Съвременни математически методи за компютърни пресмятания. Част 2: Числена линейна алгебра.* Студии на БИАП по математически науки, т. 2. ЕЛМА, София 1997.

[7] И. Молчанов. *Методы решения прикладных задач.* Наукова думка, Киев 1987.

[8] М. Петков. *Числени методи на линейната алгебра.* Наука и изкуство, София 1974.

[9] П. Петков. *Теория на автоматичното регулиране. Част 2.* Изд. ВМЕИ, София 1987.

[10] П. Петков, Н. Христов. *Анализ и синтез на линейни системи за управление със SYSLAB.* Техника, София 1993.

[11] П. Петков, Н. Христов, М. Константинов. Числени методи за матрични изчисления. *Автом., изчисл. техн. и автом. системи*, 8/9 (1987), стр. 50–83.

[12] Б. Сендов, В. Попов. *Числени методи. Част 1 и 2.* Наука и изкуство, София 1996.

[13] Дж. Форсайт, М. Малкълм, К. Молър. *Компютърни методи за математически пресмятания.* Наука и изкуство, София 1986.

[14] G. Dahlquist, A. Bjork. *Numerical Methods.* Prentice Hall, Englewood Cliffs, NJ 1974.

[15] G. Golub, C. Van Loan. *Matrix Computations.* John Hopkins, Baltimore 1989.

[16] P. Henrici. *Elements of Numerical Analysis.* Wiley, New York 1964.

[17] N. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia 1996.

[18] M. Konstantinov, P. Petkov, I. Postlethwaite, D. Gu. *Numerical Issues in Linear Control. Part I.* Techn. Report LUED 93-65, Dept. of Engineering, Leicester University, UK 1993.

[19] M. Konstantinov, P. Petkov, I. Postlethwaite, D. Gu. *Perturbation Analysis in Finite Dimensional Spaces.* Techn. Report LUED 96-18, Dept. of Engineering, Leicester University, UK 1996.

[20] W. Miller, C. Wrathall. *Software and Roundoff Analysis of Matrix Algorithms.* Academic Press, New York 1980.

[21] J. Von Neumann, H. Goldstine. Numerical inverting of matrices of high order. *Bull. AMS*, 53 (1947), pp. 1021–1099.

[22] P. Petkov, N. Christov, M. Konstantinov. *Computational Methods for Linear Control Systems.* Prentice Hall, Englewood Cliffs, NJ 1991.

[23] P. Sterbenz. *Floating Point Computation.* Prentice Hall, Englewood Cliffs, NJ 1974.

[24] J. Stoer, R. Bulirsch. *Introduction to Numerical Analysis.* Springer, New York 1983.

[25] G. Stewart, J. Sun. *Matrix Perturbation Theory.* Academic Press, Boston 1990.

[26] J. Vandergraft. *Introduction to Numerical Computations.* Academic Press, New York 1981.

[27] J. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice Hall, Englewood Cliffs, NJ 1963.

[28] J. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford Univ. Press, Oxford 1963.