MATICOVÉ ALGORITMY

Jan Lánský

Eva Ulrychová

Maticové algoritmy

Matrix Algorithms

The publisher

CURRICULUM, CZ-142 00 Praha, Cholupická 39, Czech Republic
e-mail: phcurriculum@yahoo.com

The authors

RNDr. Jan Lánský, Ph.D.
Vysoká škola finanční a správní, Estonská 500/3, 101 00 Praha 10
zizelevak@gmail.com
Affiliation: The University of Finance and Administration, Prague, Czech Republic

RNDr. Eva Ulrychová, Ph.D.
Vysoká škola finanční a správní, Estonská 500/3, 101 00 Praha 10
ulrychova@mail.vsfs.cz
Affiliation: The University of Finance and Administration, Prague, Czech Republic

The reviewers

RNDr. Hana Hladíková, Ph.D.
Ing. Vladimír Nulíček, CSc.
RNDr. Martin Pergel, Ph.D.

Online presentation: www.csrggroup.org, online catalogue NKP
ISBN 978-80-904948-8-6

OBSAH

PŘEDMLUVA

Znalost matematiky je jedním ze základních pilířů informatických studijních oborů na vysokých školách. V této knize představujeme maticové algoritmy, které bývají součástí náplně všech kurzů lineární algebry (vyučované často v rámci základního kurzu matematiky), proto by kniha mohla sloužit jako literatura pro odpovídající předmět na různých školách zaměřených na výuku informatiky. Kniha obsahuje základní témata s výjimkou problematiky vlastních čísel a vlastních vektorů – z informatického hlediska zpracování algoritmů na hledání vlastních čísel matice obecného řádu musí využívat postupů numerické matematiky přesahujících rámec knihy.

Znalost základů programování (někdy bývá označováno jako algoritmizace) je pro informatické obory zásadní a bývá vyučována v prvním ročníku bakalářského studia. V úvodním kurzu programování je třeba získat schopnost formalizace daného postupu pomocí vhodného použití cyklu, podmínek a přiřazení. Následuje schopnost správně zanořit dva, tři cykly a schopnost zdrojový kód správně rozdělovat na samostatné funkční bloky. Maticové algoritmy jsou přirozeným zdrojem příkladů k procvičování těchto programátorských schopností. Algoritmy probírané v jednotlivých kapitolách této knihy mají vzrůstající obtížnost měřenou počtem a vzájemným uspořádáním použitých programátorských konstrukcí. Tuto knihu lze použít i jako doplňkovou literaturu pro základní kurz programování či algoritmizace.

Kniha je strukturována do dvojic kapitol – po každé kapitole zpracovávající tematický celek z matematiky následuje odpovídající kapitola z informatiky. Liché kapitoly jsou tedy psány z pohledu matematika (E. Ulrychová), sudé z pohledu informatika (J. Lánský).

Cílem knihy je vyložit látku formou přístupnou všem studentům a nezatěžovat čtenáře nadbytečnými pojmy (jejichž řádné pochopení navíc činí studentům značné potíže). Z tohoto důvodu je výklad koncipován bez užití obecně definovaného pojmu vektorový prostor a základních pojmů s ním spojených (lineární kombinace vektorů, lineární nezávislost vektorů apod.).

Pro lepší srozumitelnost jsou někdy použity poněkud volnější formulace, proto ani výklad v matematických částech nemá standardní strukturu "definice, věta, důkaz". Formulace tvrzení je co nejjednodušší, zároveň je někdy upozorňováno na časté chyby či možné chybné interpretace. Formální zápis je často nahrazen či doplněn slovní formulací, kterou jsou studenti schopni nejen snáze akceptovat, ale následně sami vyjádřit lépe než s využitím formálního zápisu. Je přizpůsoben i jazyk užívaný při formulacích definovaných pojmů a tvrzení.

Látka je vysvětlována s využitím četných příkladů, v některých případech jsou kromě standardních řešených příkladů zařazeny i jednoduché aplikační příklady, které ukazují možné využití vykládaných pojmů či procesů. I když se nemusí nutně jednat o aplikace v praxi, poukazují příklady určitým způsobem na souvislost matematiky s reálným životem. Tyto příklady jsou voleny tak, aby jejich řešení nevyžadovalo téměř žádné znalosti z jiných vědních oborů či z jiných matematických odvětví (výjimku tvoří příklad odkazující k základům teorie grafů).

V závěru každé kapitoly jsou zařazeny úlohy k procvičení. V matematických částech je to kromě praktické části (neřešených příkladů s výsledky) i část teoretická – řada tvrzení, o jejichž správnosti mají studenti rozhodnout. Na tomto cvičení si studenti mohou nejen zopakovat vyloženou látku a ověřit si, že ji správně pochopili, mohou si zároveň uvědomit nutnost přesného formulování matematických tvrzení. Toto cvičení tak může sloužit i k nácviku precizního vyjadřování. Správné výsledky jsou uvedeny ve většině případů i s odůvodněním, proč chybné či nepřesně formulované tvrzení neplatí.

V informatických částech knihy se snažíme o věrné zachycení podstaty algoritmů představených v matematických částech knihy. Nesnažíme se provádět optimalizace, které by sice zrychlily běh programu, ale byly by v rozporu s matematickou částí knihy. Neprovádíme ani optimalizace algoritmů zasahující do sféry numerické matematiky.

Ve většině programovacích jazyků bývá zvykem pole o n prvcích indexovat hodnotami 0 až $n-1$. V matematice bývá zvykem pole o n prvcích indexovat hodnotami 1 až n. Analogická situace platí pro matice – matice je dvojrozměrné pole. Informatik musí při použití indexu pole uvažovat, zda se právě nachází v matematickém nebo informatickém světě. My jsme se rozhodli v této knize pro řešení, které je z hlediska matematika velmi nezvyklé, ale z hlediska informatika naprosto přirozené. Pole indexujeme hodnotami 0 až $n-1$ v obou částech knihy, jak v části matematické, tak části informatické. Výhodou tohoto přístupu je, že informatik nemusí při indexování polí opouštět svůj svět. Nevýhoda se projeví ve chvíli, kdy informatik bude chtít svým poznáním maticových algoritmů přesáhnout meze této knihy a bude se muset vyrovnat s matematickým světem indexování polí.

Zdrojové kódy jsou programovány v jazyku C. Tento jazyk je základem většiny v současné době komerčně používaných jazyků. Zároveň se jedná o plnohodnotný jazyk používaný v praxi. Jazyk C má řadu složitých syntaktických konstrukcí. My jsme se rozhodli používat pouze konstrukce základní, které budou začátečnickému programátorovi blízké. V pokročilých partiích knihy jsou na ukázku předvedeny některé složitější konstrukce s vysvětlením jejich převodu na konstrukce základní. Zaměřením knihy jsou v prvé řadě maticové algoritmy. V oblasti základů programování může být tato kniha použita jen jako doplněk výuky. Cílem knihy není čtenáře naučit programovací jazyk C, ten je pouze použit jako prostředek. Nezabýváme se například problematikou struktur nebo klíčového slova `const`.

V rámci zjednodušení zdrojových kódů jsme se rozhodli nepoužívat cykly `while`, ale používáme pouze cykly `for`. Cyklus `for` dává čtenáři lepší představu o počtu provedených kroků cyklu, respektive o jejich horním odhadu. V některých situacích bývá cyklus předčasně ukončen.

Ve funkcích neprovádíme některé testy vstupních parametrů. Tyto testy by se musely opakovat na začátku každé funkce a zbytečně by zamlžovaly vysvětlovaný algoritmus. Předpokládáme, že nám uživatel jako vstup nezadal místo matice nulový ukazatel, že nám nezadal záporné rozměry matice atd. Testujeme pouze, zda matice je požadovaného typu (například čtvercová, regulární) a provádíme testování dalších parametrů funkce, pokud tyto požadavky vycházejí ze specifických požadavků algoritmu – například při výměně dvou řádku matice testujeme, zda se tyto řádky nacházejí v matici.

Doufáme, že se tato kniha stane užitečným pomocníkem při studiu základů lineární algebry a programování.

Jan Lánský a Eva Ulrychová
Vysoká škola finanční a správní

1. VEKTORY

1.1 Vektory – základní pojmy

V celé této knize bude písmeno $\mathbb{R}$ značit množinu všech reálných čísel, písmeno $\mathbb{N}$ množinu všech přirozených čísel. Budeme pracovat s reálnými čísly a jejich uspořádanými dvojicemi, trojicemi atd. – obecně n-ticemi (kde n je přirozené číslo). Připomínáme, že uspořádanou n-ticí reálných čísel rozumíme výčet n reálných čísel, kde záleží na jejich pořadí. Tyto n-tice budeme zapisovat do kulatých závorek a jednotlivá čísla oddělovat čárkami. Např. $(-3, 5)$ je uspořádaná dvojice, $(\frac{1}{2}, 0, -4)$ je uspořádaná trojice atd. Protože v uspořádaných n-ticích záleží na pořadí, je $(-3, 5) \neq (5, -3)$, $(\frac{1}{2}, 0, -4) \neq (-4, \frac{1}{2}, 0)$ atd. Každé reálné číslo lze považovat za uspořádanou n-tici pro $n = 1$.

Z hlediska programování je vhodnější číslovat jednotlivé prvky v obecném zápisu uspořádané n-tice nikoliv od 1 (jak bývá běžné), ale od 0, takže např. uspořádanou dvojici budeme místo běžného značení (a_1, a_2) značit (a_0, a_1) apod.

Pojem 1.1 (vektor, složky vektoru)

Pod pojmem (n-složkový) vektor (ozn. např. $\bar{a}$, $\bar{b}, ...$) budeme rozumět uspořádanou n-tici reálných čísel:

$$\bar{a} = (a_0, a_1, ..., a_{n-1}).$$

Čísla $a_0, a_1, ..., a_{n-1}$ se nazývají složky vektoru $\bar{a}$. Takový vektor se někdy také nazývá aritmetický vektor (pojem vektor může být chápán i mnohem obecněji[1]).

Pojem 1.2 (nulový vektor)

Vektor, jehož všechny složky jsou rovny nule, se nazývá nulový vektor (ozn. $\bar{o}$):

$$\bar{o} = (0, 0, ..., 0).$$

1.2 Aritmetické operace s vektory

Pojem 1.3 (rovnost vektorů)

Dva n-složkové vektory jsou si rovny, jestliže se rovnají všechny jejich odpovídající složky:

je-li $\bar{a} = (a_0, a_1, ..., a_{n-1})$, $\bar{b} = (b_0, b_1, ..., b_{n-1})$, pak

$$\bar{a} = \bar{b}, \quad \text{právě když} \quad a_0 = b_0,\ a_1 = b_1,\ \dots,\ a_{n-1} = b_{n-1}.$$

Pojem 1.4 (násobek vektoru reálným číslem (konstantou))

Při násobení vektoru reálným číslem vynásobíme každou jeho složku tímto reálným číslem:

$$k \cdot \bar{a} = k \cdot (a_0, a_1, ..., a_{n-1}) = (k \cdot a_0, k \cdot a_1, ..., k \cdot a_{n-1}).$$

Pojem 1.5 (součet vektorů)

Při sčítání dvou (či více) vektorů o stejném počtu složek sečteme odpovídající složky vektorů:

$$\bar{a} + \bar{b} = (a_0, a_1, ..., a_{n-1}) + (b_0, b_1, ..., b_{n-1}) = (a_0 + b_0, a_1 + b_1, ..., a_{n-1} + b_{n-1}).$$

Poznámka (úmluva: operace s vektory)

Součet vektorů je definován pouze pro vektory o stejném počtu složek. Pro zjednodušení textu budeme dále předpokládat, že kdykoliv sčítáme vektory, je toto sčítání možné (tedy že sčítané vektory mají stejný počet složek) a nebudeme vždy tuto skutečnost explicitně uvádět. Obdobně nebudeme vždy uvádět, že násobíme vektory reálnými čísly. Tečku označující násobení můžeme vynechávat – budeme často místo $k \cdot \bar{a}$ psát pouze $k\bar{a}$.

[1] Viz např. v [3].

Odčítání vektoru $\bar{b}$ od vektoru $\bar{a}$ považujeme za přičítání minus jedna násobku vektoru $\bar{b}$ k vektoru $\bar{a}$:

$$\bar{a} - \bar{b} = \bar{a} + (-1) \cdot \bar{b}.$$

Příklad 1.1

$(1,1,1) \neq (1,1,1,1)$ (vektory nemají stejný počet složek),

$0 \cdot (1,2,3) = (0 \cdot 1, 0 \cdot 2, 0 \cdot 3) = (0,0,0)$,

$3 \cdot (-1,2,5) = (3 \cdot (-1), 3 \cdot 2, 3 \cdot 5) = (-3,6,15)$,

$(\frac{5}{2}, \frac{-1}{2}, 4) = (\frac{1}{2} \cdot 5, \frac{1}{2} \cdot (-1), \frac{1}{2} \cdot 8) = \frac{1}{2}(5,-1,8)$,

$(2,-1,5) + (4,3,-2) = (2+4, -1+3, 5+(-2)) = (6,2,3)$,

$(7,3) - (8,2) = (7-8, 3-2) = (-1,1)$,

$(4,-1) + (0,0) = (4+0, -1+0) = (4,-1)$,

$2 \cdot (3,0,-1) - 3 \cdot (1,2,-4) + (-3,6,-10) = (6,0,-2) - (3,6,-12) + (-3,6,-10) = (0,0,0)$,

$(1,2) + (3,4,5)$ nedefinováno (vektory nemají stejný počet složek).

Příklad 1.2 (ilustrace použití vektorů a jejich součtu)

Podnik A vyrábí čtyři výrobky; označme denní produkci (počet kusů) prvního výrobku v_0, druhého výrobku v_1, třetího v_2 a čtvrtého v_3. Pak tuto informaci lze zapsat pomocí vektoru $\bar{v} = (v_0, v_1, v_2, v_3)$. Vyrábí-li podnik B tytéž výrobky v množství analogicky zapsaném jako vektor $\bar{u} = (u_0, u_1, u_2, u_3)$, pak součet vektorů $\bar{v} + \bar{u} = (v_0 + u_0, v_1 + u_1, v_2 + u_2, v_3 + u_3)$ udává celkovou denní produkci jednotlivých výrobků v obou podnicích dohromady.

Příklad 1.3 (ilustrace použití vektorů a jejich násobku reálným číslem)

Zapíšeme-li ceny n výrobků uváděné v korunách jako vektor $\bar{p} = (p_0, p_1, ..., p_{n-1})$, pak při kurzu koruna ku jiné měně např. 1:15 lze ceny těchto výrobků v jiné měně vyjádřit vektorem $15\bar{p} = 15(p_0, p_1, ..., p_{n-1})$.

Pojem 1.6 (aritmetický vektorový prostor $\mathbb{R}_n$)

Množinu všech uspořádaných n-tic reálných čísel, pro které je výše uvedeným způsobem definováno sčítání a násobení reálným číslem, označíme $\mathbb{R}_n$ (tato množina se nazývá (aritmetický) vektorový prostor; jedná se o speciální případ tzv. vektorového prostoru[2]).

Poznámka (zápis $\bar{a} \in \mathbb{R}_n$)

Pojem aritmetický vektorový prostor nebudeme příliš používat, ale zápisem např. $\bar{a}, \bar{b} \in \mathbb{R}_n$ bude stručně řečeno, že vektory $\bar{a}$ a $\bar{b}$ jsou oba n-složkové.

Poznámka (řádkový a sloupcový vektor)

Aritmetické vektory jakožto uspořádané n-tice můžeme chápat i jako sloupce: $\bar{a} = \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_{n-1} \end{pmatrix}$.

Sčítáme-li vektory, musíme všechny sčítané vektory považovat buď za řádkové, nebo všechny za sloupcové. Tuto skutečnost nebudeme dále explicitně uvádět. Pokud to nebude nutné, nebudeme v dalším textu ani rozlišovat, jedná-li se o vektory řádkové či sloupcové.

Aritmetické operace s vektory mají následující vlastnosti:

Pravidlo 1.1 *(vlastnosti aritmetických operací)*

Jsou-li $\bar{a}, \bar{b}, \bar{c} \in \mathbb{R}_n$ *(n-složkové vektory) a* $k, l \in \mathbb{R}$ *(čísla), pak platí*

(1) $\bar{a} + \bar{b} = \bar{b} + \bar{a}$,

(2) $(\bar{a} + \bar{b}) + \bar{c} = \bar{a} + (\bar{b} + \bar{c})$,

[2]Viz např. v [3].

(3) $k(\bar{a}+\bar{b}) = k\bar{a} + k\bar{b}$,

(4) $(k+l)\bar{a} = k\bar{a} + l\bar{a}$,

(5) $k(l\bar{a}) = (kl)\bar{a}$.

1.3 Skalární součin vektorů

Pro dva vektory o stejném počtu složek definujeme jejich skalární součin:

Pojem 1.7 (skalární součin vektorů)

Skalární součin vektorů[3] $\bar{a} = (a_0, a_1, ..., a_{n-1})$ a $\bar{b} = (b_0, b_1, ..., b_{n-1})$ (označíme $\bar{a} \bullet \bar{b}$) definujeme jako číslo:

$$\bar{a} \bullet \bar{b} = (a_0, a_1, ..., a_{n-1}) \bullet (b_0, b_1, ..., b_{n-1}) = a_0b_0 + a_1b_1 + ... + a_{n-1}b_{n-1}.$$

P ř í k l a d 1.4

$(3, -1, 2) \bullet (1, 4, -3) = 3 \cdot 1 + (-1) \cdot 4 + 2 \cdot (-3) = -7$

$(2, 1) \bullet (-1, 2) = 0$

$(1, 2) \bullet (-2, 1, 0)$ nedefinováno.

Poznámka (skalární součin je číslo)

Zdůrazňujeme, že skalární součin vektorů je *číslo*, nikoliv vektor.

Poznámka (úmluva: skalární součin)

Skalární součin je definován pouze pro vektory o stejném počtu složek; pro zjednodušení textu nebudeme při používání skalárního součinu tuto skutečnost vždy uvádět.

P ř í k l a d 1.5

Zákazník chce koupit 6 kusů pečiva po 3 Kč, 4 litry nápoje po 15 Kč/l a 10 vajec po 2 Kč.

Cenu, kterou za celý nákup zaplatí, lze vyjádřit pomocí skalárního součinu vektorů. Počet jednotlivých nakupovaných položek lze zapsat pomocí vektoru $\bar{p} = (6, 4, 10)$ a ceny zboží pomocí vektoru $\bar{c} = (3, 15, 2)$. Cena celého nákupu bude $6 \cdot 3 + 4 \cdot 15 + 10 \cdot 2$ Kč, což lze vyjádřit jako skalární součin

$$\bar{p} \bullet \bar{c} = (6,\ 4,\ 10) \bullet (3,\ 15,\ 2) = 6 \cdot 3 + 4 \cdot 15 + 10 \cdot 2 = 98.$$

Zákazník zaplatí za nákup 98Kč.

P ř í k l a d 1.6

K vektoru $\bar{a} = (-7, 2)$ nalezneme nenulový vektor $\bar{b}$ takový, že $\bar{a} \bullet \bar{b} = 0$.

Vektor $\bar{b}$ nalezneme snadno: stačí vzít $\bar{b} = (2, 7)$ nebo jakýkoliv jeho nenulový násobek.

Poznámka (nulový skalární součin)

Obecně: hledáme-li k vektoru $\bar{a} = (a_0, a_1)$ vektor $\bar{b}$ takový, že $\bar{a} \bullet \bar{b} = 0$, pak stačí vzít $\bar{b} = (a_1, -a_0)$ nebo jakýkoliv jeho násobek.

Skalární součin má následující vlastnosti:

Pravidlo 1.2 *(vlastnosti skalárního součinu)*

Jsou-li $\bar{a}, \bar{b}, \bar{c} \in \mathbb{R}_n$ *(n-složkové vektory) a* $k \in \mathbb{R}$ *(číslo), pak platí*

(1) $\bar{a} \bullet \bar{b} = \bar{b} \bullet \bar{a}$,

(2) $(k\bar{a}) \bullet \bar{b} = k(\bar{a} \bullet \bar{b})$,

(3) $\bar{a} \bullet (\bar{b} + \bar{c}) = \bar{a} \bullet \bar{b} + \bar{a} \bullet \bar{c}$,

(4) $\bar{a} \bullet \bar{a} \geq 0$; $\bar{a} \bullet \bar{a} = 0$ *právě když* $\bar{a} = \bar{o}$.

[3] Skalární součin lze zavést i obecněji – viz např. v [3].

1.4 Úlohy k procvičení

$\boxed{A}$ **Teoretická část**

Posuďte pravdivost následujících tvrzení:

a) Součet nenulových vektorů je vždy nenulový vektor.

b) Přičtením nulového vektoru se daný vektor nezmění.

c) Vynásobením vektoru nulou vznikne nulový vektor.

d) Skalární součin dvou vektorů s kladnými složkami je vektor s kladnými složkami.

e) Skalární součin dvou vektorů s kladnými složkami je kladné číslo.

f) Skalární součin vektorů je nezáporné číslo.

g) Skalární součin nulového vektoru s libovol. vektorem (o stejném počtu složek) je nulový vektor.

h) Skalární součin nulového vektoru s libovolným vektorem (o stejném počtu složek) je roven nule.

i) Skalární součin nenulových vektorů je vždy nenulové číslo.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N ($\bar{a} + (-\bar{a}) = \bar{o}$ nebo např. $(1,-1) + (2,0) + (-3,1) = (0,0)$)

b) P

c) P

d) N (skalární součin není vektor)

e) P

f) N (např. $(-2,5) \bullet (3,1) = -1$)

g) N (skalární součin není vektor)

h) P

i) N (např. $(2,-3) \bullet (3,2) = 0$)

$\boxed{B}$ **Praktická část**

$\boxed{1}$ Pro vektory $\bar{a} = (-1,2,4)$, $\bar{b} = (1,-4,-3)$, $\bar{c} = (1,2,-6)$ vypočítejte:

a) $3\bar{a} + \bar{b} + \bar{c}$

b) $3\bar{a} + 2\bar{b} + \bar{c}$

c) $0 \cdot \bar{a} + 0 \cdot \bar{b} + 0 \cdot \bar{c}$

$\boxed{2}$ Z následujících vztahů vypočítejte x, y.

a) $3\bar{a} - 2\bar{b} = \bar{c}$, kde $\bar{a} = (x,1,2)$, $\bar{b} = (3,y,-1)$, $\bar{c} = (0,-5,8)$

b) $2\bar{a} + \bar{b} = 3\bar{c}$, kde $\bar{a} = (-1,x,3)$, $\bar{b} = (y,-2,5)$, $\bar{c} = (1,4,3)$

$\boxed{3}$ Na vektorech $\bar{a} = (-1,2,0)$, $\bar{b} = (3,1,-4)$, $\bar{c} = (-2,5,1)$ a číslech $k = 3$, $l = 4$ demonstrujte vlastnosti (1) – (5) aritmetických operací s vektory.

$\boxed{4}$ Vypočítejte skalární součiny vektorů $\bar{a}$, $\bar{b}$:

a) $\bar{a} = (-1,2)$, $\bar{b} = (3,4)$

b) $\bar{a} = (2,-3)$, $\bar{b} = (4,1,5)$

c) $\bar{a} = (1,-2,1)$, $\bar{b} = (2,3,4)$

d) $\bar{a} = (-3,1,-2)$, $\bar{b} = (1,1,1)$

$\boxed{5}$ Na vektorech $\bar{a} = (3,2,-1)$, $\bar{b} = (-1,2,4)$, $\bar{c} = (2,-3,1)$ demonstrujte vlastnosti (1) – (4) skalárního součinu.

$\boxed{6}$ K vektoru $\bar{a} = (5,-4)$ nalezněte všechny vektory $\bar{b}$ takové, že $\bar{a} \bullet \bar{b} = 0$.

Výsledky $\boxed{B}$

$\boxed{1}$ a) $(-1, 4, 3)$, b) $(0, 0, 0)$, c) $(0, 0, 0)$

$\boxed{2}$ a) $x = 2$, $y = 4$, b) řešení neexistuje

$\boxed{3}$ Platí. (1) $\bar{a} + \bar{b} = (-1, 2, 0) + (3, 1, -4) = (2, 3, -4)$, $\bar{b} + \bar{a} = (3, 1, -4) + (-1, 2, 0) = (2, 3, -4)$,
(2) $(\bar{a} + \bar{b}) + \bar{c} = (2, 3, -4) + (-2, 5, 1) = (0, 8, -3)$, $\bar{a} + (\bar{b} + \bar{c}) = (-1, 2, 0) + (1, 6, -3) = (0, 8, -3)$ atd.

$\boxed{4}$ a) 5, b) nelze, c) 0, d) -4

$\boxed{5}$ Platí. (1) $\bar{a} \bullet \bar{a} = (3, 2, -1) \bullet (3, 2, -1) = 14 > 0$,
(2) $\bar{a} \bullet \bar{b} = (3, 2, -1) \bullet (-1, 2, 4) = -3$, $\bar{b} \bullet \bar{a} = (-1, 2, 4) \bullet (3, 2, -1) = -3$ atd.

$\boxed{6}$ a) $(4, 5)$ nebo jakýkoliv jeho násobek

2. VEKTORY Z POHLEDU INFORMATIKA

2.1 Reprezentace vektoru

V této knize pracujeme s vektory reálných čísel o n složkách (viz Pojem 1.1). Vektor o n složkách je v procedurálních programovacích jazycích nejvýhodnější reprezentovat pomocí datové struktury pole délky n.

Pojem 2.1 (pole, index)

Pole p délky n je souvislý kus paměti, ve kterém je uloženo n prvků stejného datového typu, v našem případě reálného datového typu. K jednotlivým prvkům pole lze přistupovat na základě znalosti pořadového čísla prvku v rámci pole, toto pořadové číslo se nazývá index.

Pojem 2.2 (reprezentace vektoru pomocí pole)

Pole p délky n reprezentuje n složkový vektor v, pokud každý prvek pole p s indexem i reprezentuje i-tou složkou vektoru v. Prvek pole reprezentuje složku vektoru, pokud obsahuje číselnou hodnotu shodnou (až na chybu reprezentace reálných čísel v počítači) s číselnou hodnotou složky vektoru.

Poznámka (reprezentace reálných čísel)

Reálné číslo nejde standardními prostředky programovacího jazyka reprezentovat přesně, ale pouze přibližně. Reálné datové typy, které se k reprezentaci reálných čísel používají, mají předem určený počet platných cifer, které jsou schopny reprezentovat. Například reálný typ `double` má obvykle přesnost 15 až 17 platných cifer[4]. Pokud reálné číslo obsahuje větší počet platných cifer nebo má nekonečný desetinný rozvoj (např. Eulerovo číslo e), při jeho reprezentaci pomocí reálného datového typu dojde ke snížení přesnosti na počet cifer, které je reálný datový typ schopen reprezentovat.

Poznámka (indexování pole od nuly)

Ve většině procedurálních programovacích jazyků je index prvního prvku pole délky n roven hodnotě 0 a index posledního prvku pole je roven hodnotě $n-1$. Z tohoto důvodu v celé knize číslujeme jednotlivé složky vektorů (a matic) také od hodnoty 0 (a nikoliv od hodnoty 1, jak bývá zvykem ve většině odborné literatury). Pro prvky pole a složky vektorů používáme označení 0. prvek pole (složka vektoru) až $(n-1)$-ní prvek pole (složka vektoru). Toto netradiční značení má tu výhodu, že se snadno přepisují postupy uvedené v teoretické části knihy do zdrojových kódů programovacího jazyka. Nevýhoda tohoto značení se projeví při studiu jiné odborné literatury, kdy je nutné mezi naším značením a většinovým značením provést transformaci.

Zdrojové kódy uváděné v této knize jsou v syntaxi jazyka C. V jazyce C máme na výběr mezi statickou a dynamickou alokací pole. Statická alokace je vhodná pro pole, která vytváříme jako dočasná a jejich platnost je omezena na funkci, ve které byly vytvořeny. Výhodou statické alokace je snazší práce než s alokací dynamickou.

Pojem 2.3 (statická alokace pole)

Statická alokace pole se provádí následující deklarací. Nejprve je uveden datový typ jednotlivých prvků pole, který je následován názvem proměnné (tj. název pole). Za názvem proměnné je v hranatých závorkách uzavřena délka pole, která musí být celočíselným výrazem s kladnou hodnotou a musí být známa v době kompilace programu. Nepovinně může následovat znaménko `=`, které umožňuje pole při inicializaci vyplnit předem danými hodnotami. Hodnoty se uvádějí za znaménkem `=` uzavřené ve složených závorkách a od sebe jsou oddělené čárkou.

Zdrojový kód 2.1 (statická alokace pole)

```
00 int main ()
01 {
02   double staticky_vektor[5] =  {1,0,6,3,2};
03   double x = staticky_vektor[2];
04   return 0;
05 }
```

[4] Podrobnější informace o problematice najdeme ve standardu IEEE Std 754-2008.

Ve Zdrojovém kódu 2.1 na řádku `01` je ukázka statické alokace pole reálných čísel (datový typ `double`) pojmenovaného `staticky_vektor`. Pole je délky `5` a obsahuje hodnoty `1`, `0`, `6`, `3`, `2`. Toto pole reprezentuje vektor $v = (1, 0, 6, 3, 2)$. Na řádku `03` je příkaz na zjištění hodnoty druhého prvku v poli a uložení této hodnoty do právě deklarované proměnné `x`. Hodnota proměnné `x` je `6`.

Statickou alokací pole se dále v této knize nebudeme zabývat, veškeré prováděné alokace budou pouze dynamické.

Dynamická alokace je nutná pro pole, které chceme využít i v jiné funkci, než ve které bylo vytvořeno. Dynamickou alokaci využijeme vždy, pokud pole vracíme jako návratovou hodnotu nebo voláme funkci, která má toto pole jako svůj parametr. Termín vracíme pole jako návratovou hodnotu je nepřesný, ve skutečnosti vracíme pouze ukazatel na první prvek pole, analogicky pro předávání pole jako parametru funkci, opět se předává odkaz na první prvek.

Pojem 2.4 (dynamická alokace pole)

Pro dynamickou alokaci pole je nejprve nutné mít deklarovanou proměnnou typu ukazatel na datový typ, který je shodný s datovým typem prvků pole. Do tohoto ukazatele přiřadíme výsledek volání funkce `malloc` volané s parametrem udávajícím velikost paměti nutné pro alokaci pole. Velikost paměti se spočte vynásobením počtu prvků pole s velikostí jednoho prvku pole získanou pomocí operátoru `sizeof`. Pokud používáme C++ kompilátor, je ještě nutné výsledek volání funkce `malloc` přetypovat na ukazatel na datový typ, který je shodný s datovým typem prvků pole. Ukazatel nyní bude ukazovat na první prvek vzniklého pole.

Ve Zdrojovém kódu 2.2 nalezneme funkce `vytvor_vektor` a `smaz_vektor`. Funkce `vytvor_vektor` by měla být použita, kdykoliv chceme v našem programu pracovat s novým vektorem. Pokud nebudeme nějaký vektor dále potřebovat, měla by být využita funkce `smaz_vektor`. Poté se již nebudeme muset starat o problematiku dynamických alokací, která je technicky náročnou partií programovacího jazyka C.

Parametr `n` funkce `vytvor_vektor` udává počet složek vektoru. Funkce vrací návratovou hodnotu dynamicky alokovaného pole schopného reprezentovat `n` složkový vektor. Přesněji, je vrácen ukazatel na první prvek tohoto pole. Hodnoty jednotlivých složek vráceného vektoru nejsou definované, před použitím vektoru pro další výpočty je nutné tyto hodnoty definovat. Na řádku `02` je uvedena deklarace proměnné typu ukazatel na reálné číslo. Na řádku `03` probíhá dynamická alokace pole popsaná v Pojmu 2.4, včetně přetypování výsledku volání funkce `malloc` na ukazatel na reálné číslo. Na řádku `04` testujeme, zda se nám podařilo alokovat paměť potřebnou pro dané pole. Pokud se paměť nepodařilo alokovat, voláme funkci `chyba` s parametrem typu řetězec popisujícím vzniklý problém. Funkce `chyba` může být implementována například tak, že textový parametr vypíše na obrazovku a ukončí program.

Na řádcích `07` až `10` Zdrojového kódu 2.2 nalezneme funkci `smaz_vektor`. Tato funkce uvolní paměť, která byla vyhrazena pro pole `vektor`, které je prvním parametrem funkce. Druhý parametr funkce `n` udává velikost pole. Velikost pole není nikde v těle funkce využita a funkce by tento parametr ani mít nemusela. Více viz Poznámka o délce pole. Na řádku `09` voláme funkci, která uvolní paměť, která byla vyhrazena pro pole `vektor`. Od této chvíle je vektor zrušený a nemůžeme dále číst nebo měnit hodnoty jeho složek. Celá funkce `smaz_vektor` by šla nahradit samotným řádkem `09`, ale z důvodu dodržení jednotného názvosloví funkcí pracujících s vektory jsme to neudělali.

Zdrojový kód 2.2 (dynamická alokace a dealokace pole)

```
00 double * vytvor_vektor(int n)
01 {
02   double * vektor;
03   vektor = (double*)malloc(n*sizeof(double));
04   if (!vektor) chyba("Nedostatek pameti pro alokaci vektoru.\n");
05   return vektor;
06 }
07 void smaz_vektor(double * vektor, int n)
08 {
09   free(vektor);
10 }
```

Poznámka (délka pole)

V jazyku C při práci s polem musíme znát nejen název pole, ale také délku pole. Pokud délku pole v rámci programu zapomeneme, nemáme možnost ji zjistit (s výjimkou jednoho špinavého triku, který nebudeme raději ani uvádět). Proto doporučujeme použití jednotného přístupu pro práci s vektory. Vstupním parametrem jakékoliv funkce bude vždy nejen pole, ale i jeho délka, a to i v případech, kdy se nám zdá parametr délky zbytečný (viz funkce `smaz_vektor`).

Pojem 2.5 (reprezentace vektoru pomocí struktury)

Další možnou reprezentací vektoru je struktura, která bude mít dvě položky. První položka bude obsahovat ukazatel na dynamicky alokované pole, které reprezentuje daný vektor. Druhou položkou bude počet složek vektoru.

Ve Zdrojovém kódu 2.3 vidíme deklaraci struktury `vektor`, jak je popsána v Pojmu 2.5. Na řádku `02` je deklarována položka `data` typu ukazatel na reálné číslo, která bude obsahovat ukazatel na dynamicky alokované pole. Na řádku `03` je položka `n` udávající počet složek vektoru. Za deklarací struktury nesmíme zapomenout napsat středník – řádek `04`. Pro plnohodnotnou práci se strukturou vektor by bylo nutné ještě vytvořit obdobu funkce `vytvor_vektor` (viz Zdrojový kód 2.2).

Zdrojový kód 2.3 (struktura `vektor`)

```
00 struct vektor
01 {
02   double * data;
03   int n;
04 };
```

Používáním struktury vektor nám odpadne starost s pamatováním si délky pole. V případě potřeby délku pole zjistíme z položky struktury `n`. Zjednoduší se hlavičky funkcí, kdy místo dvou parametrů (ukazatele na první prvek pole a délky pole) budeme předávat pouze jeden.

Zdrojový kód 2.4 demonstruje stinnou stránku používání struktury `vektor`. Při reprezentaci vektoru pomocí dynamicky alokovaného pole je přístup k jednotlivým složkám vektoru intuitivní pomocí operátoru hranatých závorek (viz řádek `00`). Při reprezentaci vektoru pomocí struktury `vektor` musíme pro přístup k jednotlivým složkám vektoru nejprve provést přístup k položce data a teprve na ni vyvoláme operátor hranatých závorek (viz řádek `01`). Tento postup ztěžuje orientaci ve zdrojovém kódu. Z tohoto důvodu jsme se rozhodli v celé knize používat pro reprezentaci vektoru dynamicky alokované pole a nikoliv strukturu `vektor`.

Zdrojový kód 2.4 (přístup ke složkám vektoru)

```
00 v[2] = 5;
01 w->data[2] = 5;
```

Poznámka (řídké vektory)

V praxi se někdy používají vektory o velkém počtu složek (tisíce, desítky tisíc), z nichž většina má nulovou hodnotu. Takovéto vektory se nazývají řídké vektory a místo reprezentace pomocí pole je pro ně výhodnější reprezentace například pomocí spojového seznamu dvojic. Nenulová složka vektoru je reprezentována dvojicí, která obsahuje pořadové číslo dané složky a její hodnotu. Řídkými vektory se dále v této knize nebudeme zabývat.

2.2 Základní funkce pro vektory

V předchozím textu jsme se seznámili se způsoby, jak reprezentovat vektor v programovacím jazyku. Dohodli jsme se, že pro reprezentaci vektoru budeme používat dynamickou alokaci pole. Nyní přestaneme rozlišovat mezi pojmy vektor a reprezentace vektoru, s výjimkou situací, kdy bude potřeba tyto dva pojmy odlišit.

Ve Zdrojovém kódu 2.2 byla představena funkce `vytvor_vektor`, která alokuje prostor pro nový vektor, který můžeme vyplnit požadovanými daty. Ve Zdrojovém kódu 2.5 tuto funkci využijeme k vytvoření vektoru $v = (5, 2, 0, -4)$. Nejprve na řádku `00` deklarujeme proměnnou `v` typu ukazatel na reálné číslo. Na

řádku 01 pomocí funkce vytvor_vektor alokujeme prostor pro 4-složkový vektor. Na řádku 02 dosadíme do jednotlivých složek vektoru požadované hodnoty (do 0. složky hodnotu 5, do 1. složky hodnotu 2, do 2. složky hodnotu 0 a do 3. složky hodnotu -4. Z našeho pohledu se proměnná v stala vektorem $v = (5, 2, 0, -4)$.

Zdrojový kód 2.5 (vektor v =(5,2,0,-4))

```
00 double * v;
01 v = vytvor_vektor(4);
02 v[0]=5; v[1]=2; v[2]=0; v[3]=-4;
```

Ve Zdrojovém kódu 2.6 je implementována funkce, která vytvoří nulový vektor (viz Pojem 1.2) o zadaném počtu složek. Parametr funkce n je požadovaný počet složek vytvářeného vektoru. Na řádku 02 deklarujeme proměnnou i, která bude řídící proměnnou cyklu. Na řádku 03 deklarujeme proměnnou vektor, která bude uchovávat právě vytvářený vektor. Na řádku 04 alokujeme prostor pro n-složkový vektor. Na řádku 05 se nachází hlavička cyklu s řídící proměnnou i, která nabývá hodnot od 0 do n-1. Na řádku 06 se nachází tělo cyklu, ve kterém složce vektoru s indexem i přiřazujeme hodnotu 0. Na řádku 07 vracíme návratovou hodnotu funkce, v našem případě vzniklý nulový vektor.

Zdrojový kód 2.6 (nulový vektor)

```
00 double * nulovy_vektor(int n)
01 {
02   int i;
03   double * vektor;
04   vektor = vytvor_vektor(n);
05   for (i=0; i< n; i++)
06     vektor[i]=0;
07   return vektor;
08 }
```

Při práci s vektory pomocí programovacího jazyka je výhodné mít funkci, která vypíše požadovaný vektor na obrazovku (případně do souboru). Ve Zdrojovém kódu 2.7 je implementována funkce pro výpis vektoru na obrazovku. Vektor je vypisován po jednotlivých složkách za pomocí cyklu. K výpisu jednotlivých složek (řádek 04) použijeme knihovní funkci printf, která nabízí velké množství možností pro výpis různých datových typů. My si pouze povíme, že číslo 9 značí celkový počet vypisovaných cifer daného čísla a číslo 2 značí počet vypisovaných cifer za desetinnou čárkou. Po skončení cyklu je celý vektor vypsán a my provedeme odřádkování (řádek 05). Případný další vypisovaný obsah bude začínat na novém řádku.

Zdrojový kód 2.7 (výpis vektoru)

```
00 void vypis_vektor(double * vektor, int n)
01 {
02   int i;
03   for (i=0; i<n; i++)
04     printf("%9.2f ", vektor[i]);
05   printf("\n");
06 }
```

Poznámka (const)

Zkušenější programátory jistě napadlo, proč ve Zdrojovém kódu 2.7 není v hlavičce funkce (řádek 00) uvedeno klíčové slovo const před výrazem double * vektor. Uvedením tohoto klíčového slova prohlašujeme, že nebudeme modifikovat hodnoty daného parametru v průběhu běhu funkce. Přesněji, říkáme, že nebudeme přepisovat hodnoty jednotlivých složek vektoru. Funkce, která vektor vypisuje na obrazovku, by jistě svůj parametr modifikovat neměla. Je to významný bezpečnostní prvek, který nás chrání proti zbytečným chybám, například před prohozením levé a pravé strany operátoru přiřazení. Bohužel v případě matic, které budou v následujících kapitolách, tato problematika svou složitostí překračuje pojetí této

knihy. Proto nebudeme tuto problematiku řešit ani u vektorů.

Při programování nastává potřeba otestovat funkčnost zdrojového kódu na nějakých vstupních hodnotách. Někdy potřebujeme otestovat funkčnost pro konkrétní hodnoty (viz Zdrojový kód 2.5), jindy naopak požadujeme, aby vstupní hodnoty byly pokud možno co nejvíce náhodné. K tomuto účelu slouží funkce `nahodny_vektor`, popsaná ve Zdrojovém kódu 2.8. Význam většiny zdrojového kódu je nám znám z předchozích odstavců, kde byl vysvětlen pro jiné funkce. Zajímavý je pouze řádek `06`, na kterém voláme knihovní funkci `rand`. Tato funkce vrací náhodné celé číslo z intervalu `0` až `32767`. V některých překladačích může být horní hranice intervalu i vyšší. Pomocí operátoru zbytek po celočíselném dělení `%` toto číslo převedeme na náhodné celé číslo z intervalu `0` až `8`. Následně odečtením konstanty `4` získáme náhodné celé číslo z intervalu `-4` až `4`.

Zdrojový kód 2.8 (náhodný vektor)

```
00 double * nahodny_vektor(int n)
01 {
02   double * vektor;
03   int i;
04   vektor = vytvor_vektor(n);
05   for (i=0; i<n; i++)
06     vektor[i] = rand()%9 - 4;
07   return vektor;
08 }
```

Poznámka (náhodná čísla)

Volba, jaká náhodná čísla chceme používat, je čistě otázkou vkusu. My doporučujeme volit spíše nižší čísla, aby se dala dobře vypisovat na obrazovku i v případě, že na ně bude aplikována posloupnost aritmetických operací (zejména operací násobení).

2.3 Aritmetické operace s vektory

U dvou vektorů o stejném počtu složek můžeme zjišťovat jejich rovnost (viz Pojem 1.3). Ve Zdrojovém kódu 2.9 je implementována funkce, která rozhoduje, zda dva vektory `u` a `v` jsou si rovny. Parametrem funkce jsou vektor `u` s počtem složek `n_u` a vektor `v` s počtem složek `n_v`. Pokud vektory nemají shodný počet složek, je zavolána uživatelská funkce `chyba`. V cyklu se porovnávají odpovídající složky obou vektorů. Pokud nastane u některé dvojice složek nerovnost, funkce v tento moment končí a vrací hodnotu `0`, vektory se nerovnají. Pokud proběhne celý cyklus až do konce, musely se nutně všechny složky obou vektorů rovnat. V tomto případě funkce vrací hodnotu `1`, vektory se rovnají.

Zdrojový kód 2.9 (rovnost vektorů)

```
00 int rovnost_vektoru(double * u, int n_u, double * v, int n_v)
01 {
02   int i;
03   if (n_u != n_v) chyba("Vektory museji mit stejny pocet slozek.");
04   for (i=0; i<n_u; i++)
05     if (u[i] != v[i]) return 0;
06   return 1;
07 }
```

Poznámka (přesnost aritmetických operací)

V Poznámce o přesnosti reprezentace reálných čísel jsme zmínili, že reálná čísla lze při počítačovém zpracování reprezentovat pouze s předem danou přesností. Představme si následující experiment. Číslo `1` vydělíme stokrát po sobě číslem `7` (například pomocí cyklu) a poté ho opětovně stokrát po sobě vynásobíme číslem `7`. Získané číslo nebude `1`, ale na posledních reprezentovaných desetinných místech (cca 15. místo a dále) se bude od čísla `1` lišit. Výsledkem porovnání takto vzniklého čísla s číslem `1` bude nepravda. Z tohoto důvodu nemá funkce na porovnání dvou vektorů příliš velké využití. Dva vektory, které si z matematického hlediska mají být rovny, budou často funkcí prohlášeny za nerovné.

Ve Zdrojovém kódu 2.10 je implementována funkce, která vynásobí vektor konstantou (skalární hodnotou) – viz Pojem 1.4. Parametrem funkce je kromě vektoru `u` a počtu jeho složek `n` rovněž příslušná konstanta `k`, kterou má být vektor vynásoben. Na řádku `04` vytváříme nový vektor `v`, který bude obsahovat výsledek vynásobení vektoru `u` konstantou `k`. V cyklu do jednotlivých složek vektoru `v` přiřazujeme výsledky součinu konstanty `k` a odpovídající složky vektoru `u`.

Zdrojový kód 2.10 (násobení vektoru konstantou)

```
00 double * nasobeni_vektoru_konstantou(double * u, int n, double k)
01 {
02   double * v;
03   int i;
04   v = vytvor_vektor(n);
05   for (i=0; i<n; i++)
06     v[i] = k * u[i];
07   return v;
08 }
```

Poznámka (nový vektor místo modifikace)

Funkci na násobení vektoru konstantou jsme se rozhodli implementovat tak, že vytvoří nový vektor, který je násobkem vektoru původního. Původní vektor zůstane nemodifikován. Funkce by šla rovněž pojmout tak, že by se modifikoval vektor původní. Rozhodnutí, kterou z těchto dvou variant zvolit (případně zda implementovat obě dvě), je závislé na plánovaném využití této funkce.

Zdrojový kód 2.11 (součet vektorů)

```
00 double * soucet_vektoru(double * u, int n_u, double * v, int n_v)
01 {
02   double * soucet;
03   int i;
04   if (n_u != n_v) chyba("Vektory museji mit stejny pocet slozek.");
05   soucet = vytvor_vektor(n_u);
06   for (i=0; i<n_u; i++)
07     soucet[i] = u[i]+v[i];
08   return soucet;
09 }
```

Ve Zdrojovém kódu 2.11 je implementována funkce, která sečte dva vektory o stejném počtu složek (viz Pojem 1.5). Parametry funkce jsou oba dva vektory a počty jejich složek. Pokud se počty složek obou vektorů liší, je vyvolána uživatelská funkce `chyba`. Na řádku `05` vytváříme nový vektor, který bude obsahovat výsledek součtu vstupních vektorů. Vektory se v cyklu sčítají po složkách.

Zdrojový kód 2.12 (skalární součin vektorů)

```
00 double skalarni_soucin_vektoru(double * u, int n_u, double * v, int n_v)
01 {
02   double skalarni_soucin = 0;
03   int i;
04   if (n_u != n_v) chyba("Vektory museji mit stejny pocet slozek.");
05   for (i=0; i<n_u; i++)
06     skalarni_soucin += u[i]*v[i];
07   return skalarni_soucin;
08 }
```

Ve Zdrojovém kódu 2.12 je implementována funkce, která vypočítá skalární součin dvou vektorů `u` a `v` o stejném počtu složek `n_u` (viz Pojem 1.7). Pokud se počty složek obou vektorů liší, je vyvolána uživatelská funkce `chyba`. Na řádku `02` deklarujeme proměnnou `skalarni_soucin`, kterou inicializujeme na hodnotu `0`. V této proměnné budeme uchovávat aktuální spočtenou hodnotu skalárního součinu zatím zpracovaných

částí vektorů. Na řádku se `05` nachází cyklus, jehož řídící proměnná `i` postupně nabývá hodnot od `0` do `n_u-1`. V každém kroku cyklu se vypočítá součin `i` -tých složek vektorů `u` a `v`. Tento součin se přičte k aktuální hodnotě proměnné `skalarni_soucin`, která před začátkem cyklu byla vynulována. Po skončení cyklu je výsledkem funkce hodnota proměnné `skalarni_soucin`.

Poznámka (inicializace proměnné)

Ve Zdrojovém kódu 2.12 na řádku `02` jsme inicializovali proměnnou na hodnotu `0`. Tento krok je velmi důležitý. Pokud by proměnná nebyla inicializována, mohla by obsahovat jakoukoliv hodnotu, a jakékoliv použití této hodnoty by vedlo k nesmyslným výsledkům, případně k běhové chybě programu (v závislosti na použitém překladači).

2.4 Otestujme naše funkce

Zdrojový kód 2.13 bude zatím nejrozsáhlejší kus kódu, který jsme doposud v této knize viděli. Nebojme se, neobsahuje žádné obtížné myšlenkové kroky. Dal by se rozdělit do čtyř částí: direktivy, funkce `chyba`, funkce `ukazka_vektory` a funkce `main`.

Zdrojový kód 2.13 (ukázka práce s vektory)

```
00 #include <malloc.h>
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <time.h>
04 void chyba (char * chybove_hlaseni)
05 {
06   printf("%s",chybove_hlaseni);
07   exit(-1);
08 }
09 void ukazka_vektory(void)
10 {
11   double * u, * v, * w;
12   double s;
13   srand((unsigned int)time(NULL));
14   u = nahodny_vektor(7);
15   v = nasobeni_vektoru_konstantou(u,7,2);
16   w = soucet_vektoru(u,7,u,7);
17   vypis_vektor(u,7);
18   vypis_vektor(v,7);
19   vypis_vektor(w,7);
20   printf("Jsou v a w si rovny?: %d\n",rovnost_vektoru(v,7,w,7));
21   s = skalarni_soucin_vektoru(u,7,v,7);
22   printf("Skalarni soucin vektoru u a v je: %9.2f\n",s);
23   smaz_vektor(u,7);
24   smaz_vektor(v,7);
25   smaz_vektor(w,7);
26 }
27 int main()
28 {
29   ukazka_vektory();
30   return 0;
31 }
```

Direktivy preprocesoru (řádky `00` až `03`) říkají, jaké knihovny chceme v programu používat. Knihovna je skupina funkcí řešících podobnou problematiku. Toto je sice velmi zjednodušené vysvětlení, ale pro naše účely je plně postačující. V předchozích textu jsme několikrát o nějaké funkci prohlásili, že je to funkce knihovní. Pokud chceme knihovní funkci v programu použít, musíme pomocí direktivy `#include` programu oznámit, že chceme využívat knihovnu, ve které se tato funkce nachází. Pro účely naší knihy postačí na úplném začátku každého programu uvést tyto čtyři řádky.

Na řádcích `04` až `08` se nachází uživatelská funkce chyba, jejíž volání jsme několikrát viděli v předchozích zdrojových kódech. Tato funkce vypíše na obrazovku chybové hlášení, které ji bylo předáno parametrem funkce a následně ukončí běh programu.

Mezi řádky `08` a `09` musíme vložit zdrojové kódy všech uživatelských funkcí, které ve zbytku programu použijeme. Funkce se musejí rovněž nacházet ve správném pořadí. Příkladem správného pořadí je to pořadí, v jakém se funkce objevují v této knize. Pokud nějakou funkci zapomeneme do zdrojového kódu vložit, kompilátor nám oznámí chybové hlášení, ve kterém bude název chybějící funkce uveden.

Nyní přeskočíme na poslední část zdrojového kódu, na funkci `main`. Tato funkce by ve slušně napsaných programech měla zpracovávat pouze parametry programu získané z příkazového řádku při spuštění programu (v našem případě je nevyužíváme) a poté by měla volat jednu funkci, které vykoná celou funkčnost programu. V našem případě se jedná o funkci `ukazka_vektory`.

Vrátíme se k jádru zdrojového kódu, funkci `ukazka_vektory`. Z deklarace proměnných vidíme, že ve funkci budeme využívat tři vektory `u`, `v`, `w`. Zajímavý je řádek `13`, který souvisí s generováním náhodných čísel v celém programu. Pokud tento řádek odstraníme, budou se při každém spuštění programu generovat ta samá náhodná čísla, například vektor `u` bude mít vždy stejné hodnoty jednotlivých složek. Pokud řádek `13` v programu ponecháme, bude mít vektor `u` po každém spuštění programu jinou hodnotu.

Na řádku `14` vytváříme náhodný vektor `u`, který na řádku `15` násobíme konstantou `2` a výsledek tohoto násobení ukládáme do vektoru `v`. Na řádku `16` sčítáme vektor u sám se sebou a výsledek sčítání vektorů ukládáme do vektoru `w`. Na řádku `20` testujeme, zda si jsou vektory `v` a `w` rovny. Z matematického hlediska by rovnost vektorů měla nastat, z důvodů nepřesnosti reprezentace reálných čísel v počítači a nepřesnosti aritmetických operací rovnost nastat nemusí. Úmyslně jsme vybrali takový příklad, ve kterém se provedlo minimum aritmetických operací, a předpokládáme, že rovnost nastane s vysokou pravděpodobností.

Ve funkci jsme dále na řádku `21` vypočítali skalární součin vektorů `u` a `v`. Na posledních řádcích funkce jsme smazali všechny tři vektory, se kterými jsme ve funkci pracovali.

2.5 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 2. Jazyk C, který je vybrán pro tuto knihu, je společným předkem většiny v dnešní době používaných programovacích jazyků. Z tohoto důvodu by neměly při plnění tohoto úkolu nastat neřešitelné situace. Pokud si vyberete pro implementaci jazyk C, získáváte pochvalu autorů této knihy.

2 Najděte dva takové vektory, které se z matematického hlediska rovnají, ale funkce `rovnost_vektoru` je prohlásí za nerovné.

3 Pomocí programu na příkladech náhodných vektorů demonstrujte všechny body Pravidla 1.1. Pokud platnost nenastane, vzniklou situaci vysvětlete.

4 Pomocí programu na příkladech náhodných vektorů demonstrujte všechny body Pravidla 1.2. Pokud platnost nenastane, vzniklou situaci vysvětlete.

5 Pokud patříte mezi zkušené programátory, implementujte všechny funkce uvedené v kapitole 2 pomocí reprezentace vektoru pomocí struktury (viz Pojem 2.5). Posuďte, se kterou implementací (reprezentace vektoru pomocí pole nebo pomocí struktury) se Vám lépe pracuje.

3. MATICE

3.1 Matice – základní pojmy

Matice je v podstatě "tabulka" čísel; řádky i sloupce v této "tabulce" budeme pro účely programování podobně jako u vektorů číslovat od 0 (nikoliv od 1, jak je jinak běžné). Má-li "tabulka" např. tři sloupce, budou očíslovány 0,1,2 – začínáme tedy nultým (nikoliv prvním) sloupcem (stejně tak pro řádky).

Pojem 3.1 (matice, prvek matice)

Maticí typu $m \times n$ rozumíme schéma (tabulku) $m \cdot n$ reálných čísel – prvků matice – uspořádaných do m řádků a n sloupců. Prvky matice označujeme dvojím indexem – první udává řádek a druhý udává sloupec, ve kterém se prvek nachází. Prvek a_{ij} je tedy prvek v i-tém řádku a j-tém sloupci; říkáme také, že se prvek a_{ij} nachází na pozici (i, j). Matice značíme velkými písmeny a schéma prvků zapisujeme do závorek:

$$A = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,n-1} \\ a_{10} & a_{11} & \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} \end{pmatrix}.$$

Někdy budeme pro matice používat stručný zápis $A = (a_{ij})$, případně s vyznačením typu matice: $A = (a_{ij})_{m\times n}$ nebo jen $A_{m\times n}$.

Poznámka (indexy)

V případě složitějšího zápisu řádkového či sloupcového indexu oddělujeme od sebe řádkový a sloupcový index čárkou – tak, jak vidíme ve výše uvedené matici.

Příklad 3.1

Matice

$$A = \begin{pmatrix} -3 & 2 & 0 & 1 \\ 4 & -5 & 9 & -2 \\ -8 & 3 & -6 & 7 \end{pmatrix}$$

je matice typu 3×4. Raději znovu připomínáme, že řádky i sloupce číslujeme od nuly, takže např. číslo 2 je v nultém řádku a prvním sloupci, tedy $a_{01} = 2$; dále např. $a_{12} = 9$, $a_{23} = 7$ atd.

Poznámka (aritmetický vektor jako speciální případ matice)

Aritmetický řádkový vektor $\bar{a} = (a_0, a_1, ..., a_{n-1})$ lze považovat za matici typu $1 \times n$. Aritmetický sloupcový vektor $\bar{a} = \begin{pmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{n-1} \end{pmatrix}$ lze považovat za matici typu $n \times 1$.

Pojem 3.2 (čtvercová matice, obdélníková matice)

Matice typu $n \times n$, tj. matice, která má stejný počet řádků jako sloupců, se nazývá čtvercová matice (řádu n). Ostatní matice se nazývají obdélníkové.

Pojem 3.3 (nulová matice)

Matice, jejíž všechny prvky jsou rovny nule, se nazývá nulová matice (ozn. O).

Příklad 3.2

Matice $B = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 0 & -6 \\ 1 & -3 & 0 \end{pmatrix}$ je čtvercová matice řádu tři. Matice $O = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ je obdélníková nulová matice typu 2×3.

Pojem 3.4 (transponovaná matice)

Matice, která vznikne z matice A záměnou řádků za sloupce (při zachování pořadí), se nazývá matice transponovaná k matici A a značí se A^T. Je-li matice A typu $m \times n$, pak matice A^T je typu $n \times m$.

Matice A je maticí transponovanou k matici A^T; matice A a A^T jsou tedy navzájem transponované. Zřejmě platí[5] $(A^T)^T = A$.

Příklad 3.3

$$A = \begin{pmatrix} -7 & 2 & 4 & 3 \\ 6 & -4 & 3 & -2 \\ -1 & 1 & -6 & 8 \end{pmatrix}, \qquad A^T = \begin{pmatrix} -7 & 6 & -1 \\ 2 & -4 & 1 \\ 4 & 3 & -6 \\ 3 & -2 & 8 \end{pmatrix}.$$

Poznámka (vztah řádkového a sloupcového vektoru)

Řádkový a odpovídající sloupcový aritmetický vektor lze považovat za navzájem transponované matice.

Příklad 3.4 (ilustrace použití matic)

Pomocí matic lze zapsat údaje, které se běžně zapisují do tabulek – počet kusů zboží ve skladech, počet osob s určitou vlastností v různých lokalitách nebo různých časových obdobích apod. Význam transponované matice si lze snadno představit – údaje z tabulky se místo "po řádcích" čtou "po sloupcích". Porovnejme např. řazení údajů v následujících tabulkách, udávajících počty snídaní, obědů a večeří v jistém stravovacím zařízení během pracovního týdne:

	Po	Út	St	Čt	Pá
snídaně	40	75	72	68	60
obědy	77	95	93	87	71
večeře	54	75	70	64	–

Tabulka 3.1. Počty snídaní, obědů a večeří

	snídaně	obědy	večeře
Po	40	77	54
Út	75	95	75
St	72	93	70
Čt	68	87	64
Pá	60	71	–

Tabulka 3.2. počty snídaní, obědů a večeří

Tabulkám 3.1 a 3.2 odpovídají navzájem transponované matice typu 3×5, resp. 5×3:

$$A = \begin{pmatrix} 40 & 75 & 72 & 68 & 60 \\ 77 & 95 & 93 & 87 & 71 \\ 54 & 75 & 70 & 64 & 0 \end{pmatrix}, \qquad A^T = \begin{pmatrix} 40 & 77 & 54 \\ 75 & 95 & 75 \\ 72 & 93 & 70 \\ 68 & 87 & 64 \\ 60 & 71 & 0 \end{pmatrix}.$$

Pojem 3.5 (symetrická matice)

Čtvercová matice A, pro kterou platí[5] $A = A^T$, se nazývá symetrická. Čtvercová matice A řádu n je symetrická, právě když $a_{ij} = a_{ji}$ pro všechna $i, j = 0, ..., n-1$.

Příklad 3.5

Matice

$$A = \begin{pmatrix} -2 & 4 & 6 \\ 4 & -3 & 0 \\ 6 & 0 & -5 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 3 \\ 0 & 4 & 0 \\ 3 & 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 0 \\ 0 & 4 \end{pmatrix}$$

jsou symetrické.

Pojem 3.6 (diagonála matice, diagonální prvek)

Prvky $a_{00}, a_{11}, a_{22}...$ se nazývají diagonální prvky, tvoří tzv. (hlavní) diagonálu matice.

Diagonála matice

typu 3×3:

$$\begin{pmatrix} \mathbf{a_{00}} & a_{01} & a_{02} \\ a_{10} & \mathbf{a_{11}} & a_{12} \\ a_{20} & a_{21} & \mathbf{a_{22}} \end{pmatrix}$$

typu 2×3:

$$\begin{pmatrix} \mathbf{a_{00}} & a_{01} & a_{02} \\ a_{10} & \mathbf{a_{11}} & a_{12} \end{pmatrix}$$

typu 3×2:

$$\begin{pmatrix} \mathbf{a_{00}} & a_{01} \\ a_{10} & \mathbf{a_{11}} \\ a_{20} & a_{21} \end{pmatrix}$$

[5] Dvě matice jsou si rovny, jsou-li stejného typu a rovnají-li se prvky na odpovídajících si pozicích – viz Pojem 3.10.

Poznámka (symetrie podle diagonály)

Symetrická matice je (jakožto čtverec) symetrická podle hlavní diagonály.

Pojem 3.7 (trojúhelníková matice)

Čtvercová matice, která má pod diagonálou všechny prvky rovny nule, se nazývá horní trojúhelníková matice.

Čtvercová matice, která má nad diagonálou všechny prvky rovny nule, se nazývá dolní trojúhelníková matice[6].

Poznámka (úmluva: trojúhelníková matice)

Protože budeme téměř výhradně pracovat s horní trojúhelníkovou maticí, nebudeme pro stručnost přívlastek "horní" uvádět. V případě, že bude potřeba odlišit horní a dolní trojúhelníkovou matici, uvedeme tuto skutečnost explicitně.

Pojem 3.8 (diagonální matice)

Čtvercová matice, která má nad i pod diagonálou všechny prvky rovny nule, se nazývá diagonální matice[7].

Pojem 3.9 (jednotková matice)

Speciálním případem diagonální matice je jednotková matice – čtvercová matice, která má na diagonále samé jedničky a mimo diagonálu samé nuly. Jednotkovou matici budeme značit I, případně s vyznačením řádu, např. $I_{3\times 3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Poznámka (nulová a jednotková matice)

Pozor – zatímco matice $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ je nulová matice, matice $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ není jednotková!

Příklad 3.6

Matice

$$A = \begin{pmatrix} 2 & 4 & 7 \\ 0 & -3 & 2 \\ 0 & 0 & 5 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 7 & 0 \\ 0 & 6 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E = \begin{pmatrix} 6 & -1 & 4 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

jsou trojúhelníkové, matice C a D jsou navíc i diagonální (žádná z matic není jednotková).

Poznámka (velké matice)

V praxi se často vyskytují matice o velkém počtu řádků i sloupců. I prvky matic mohou být velká čísla.

Příklad 3.7

Černobílý digitální obraz se skládá např. z 340×280 bodů (pixelů), každý bod má jeden z 256 odstínů šedi (očíslovaných od 0 do 255). Tyto údaje lze zapsat maticí 340×280, jejíž prvky jsou čísla od 0 do 255.

3.2 Aritmetické operace s maticemi

Podobně jako u aritmetických vektorů zavedeme rovnost matic, násobek matice reálným číslem a součet matic.

Pojem 3.10 (rovnost matic)

Dvě matice jsou si rovny, jsou-li stejného typu a rovnají-li se prvky na odpovídajících si pozicích, tzn. pro matice $A = (a_{ij})$, $B = (b_{ij})$ typu $m \times n$ je

$$A = B \quad \text{právě když} \quad a_{00} = b_{00},\ a_{01} = b_{01},\ \ldots,\ a_{m-1,n-1} = b_{m-1,n-1}.$$

[6] Pojem trojúhelníková matice není v literatuře zcela jednotný. Někdy se připouští i obdélníková matice s nulovými prvky pod (resp. nad) diagonálou. V některé literatuře je požadována nenulovost diagonálních prvků.

[7] Pojem diagonální matice není v literatuře zcela jednotný. Někdy se připouští i obdélníková matice, jejíž všechny mimodiagonální prvky jsou nulové.

Pojem 3.11 (násobek matice reálným číslem (konstantou))

Při násobení matic reálným číslem vynásobíme každý prvek matice tímto číslem:

$$c \cdot A = c \cdot (a_{ij})_{m\times n} = (c \cdot a_{ij})_{m\times n}, \quad c \in \mathbb{R}.$$

Pojem 3.12 (součet matic)

Při sčítání matic stejného typu sečteme prvky na stejných pozicích:

$$A + B = (a_{ij})_{m\times n} + (b_{ij})_{m\times n} = (a_{ij} + b_{ij})_{m\times n}.$$

V dalším textu nebudeme důsledně typy matic uvádět – není-li typ uveden, předpokládáme při sčítání matice stejného typu.

Příklad 3.8

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \neq \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{(matice nejsou stejného typu)},$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix} \neq \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{(matice nejsou stejného typu, nulový řádek hraje roli)},$$

$$2 \cdot \begin{pmatrix} 1 & 3 \\ -4 & 7 \end{pmatrix} = \begin{pmatrix} 2\cdot 1 & 2\cdot 3 \\ 2\cdot(-4) & 2\cdot 7 \end{pmatrix} = \begin{pmatrix} 2 & 6 \\ -8 & 14 \end{pmatrix},$$

$$\begin{pmatrix} \frac{5}{2} & \frac{3}{2} & 6 \\ 2 & -\frac{1}{2} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\cdot 5 & \frac{1}{2}\cdot 3 & \frac{1}{2}\cdot 12 \\ \frac{1}{2}\cdot 4 & \frac{1}{2}\cdot(-1) & \frac{1}{2}\cdot 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 5 & 3 & 12 \\ 4 & -1 & 0 \end{pmatrix},$$

$$0 \cdot \begin{pmatrix} 1 & 0 & -3 \\ 2 & -4 & 6 \end{pmatrix} = \begin{pmatrix} 0\cdot 1 & 0\cdot 0 & 0\cdot(-3) \\ 0\cdot 2 & 0\cdot(-4) & 0\cdot 6 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 1 & -2 & 3 \\ -4 & 0 & 5 \end{pmatrix} + \begin{pmatrix} 4 & -3 & 2 \\ -1 & -5 & 0 \end{pmatrix} = \begin{pmatrix} 5 & -5 & 5 \\ -5 & -5 & 5 \end{pmatrix} = 5 \cdot \begin{pmatrix} 1 & -1 & 1 \\ -1 & -1 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad \text{(přičtením nulové matice se matice nezmění)},$$

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

(přičtením k-násobku jednotkové matice se v původní matici zvětší diagonální prvky o k, ostatní prvky zůstanou beze změny),

$$\begin{pmatrix} 5 & 2 \\ -4 & 9 \end{pmatrix} + \begin{pmatrix} 1 & 0 & -3 \\ 2 & -4 & 6 \end{pmatrix} \qquad \text{nedefinováno.}$$

Příklad 3.9 (ilustrace použití matic, jejich součtu a násobku reálným číslem)

V prodejně s porcelánem nabízejí určitý typ hlubokých, mělkých a dezertních talířů ve čtyřech barevných variantách – modré (M), zelené (Z), červené (Č) a oranžové (O). Každý měsíc doplňují zásoby o potřebný počet kusů. Tabulka 3.3 udává počet objednaných kusů v měsíci lednu:

	M	Z	Č	O
hluboké	80	70	60	80
mělké	70	60	50	90
dezertní	50	40	30	30

Tabulka 3.3. Počet talířů

Údaje zapsané v této tabulce lze reprezentovat maticí: $L = \begin{pmatrix} 80 & 70 & 60 & 80 \\ 70 & 60 & 50 & 90 \\ 50 & 40 & 30 & 30 \end{pmatrix}$.

Pro velký zájem o toto zboží objednali v únoru dvojnásobek veškerého zboží oproti lednu:

$$U = 2L = \begin{pmatrix} 160 & 140 & 120 & 160 \\ 140 & 120 & 100 & 180 \\ 100 & 80 & 60 & 60 \end{pmatrix}. \qquad \text{V březnu objednali:} \quad B = \begin{pmatrix} 90 & 80 & 90 & 70 \\ 70 & 90 & 60 & 80 \\ 60 & 40 & 50 & 30 \end{pmatrix}.$$

Za 1. čtvrtletí tedy celkem objednali množství popsané maticí $L + U + B$, tj.:

$$\begin{pmatrix} 80+160+90 & 70+140+80 & 60+120+90 & 80+160+70 \\ 70+140+70 & 60+120+90 & 50+100+60 & 90+180+80 \\ 50+100+60 & 40+80+40 & 30+60+50 & 30+60+30 \end{pmatrix} = \begin{pmatrix} 330 & 290 & 270 & 310 \\ 280 & 270 & 210 & 350 \\ 210 & 160 & 140 & 120 \end{pmatrix}.$$

Průměrné měsíční objednávky v 1. čtvrtletí jsou popsány maticí (zaokrouhleno)

$$P = \frac{1}{3}(L + U + B) = \begin{pmatrix} 110 & 97 & 90 & 103 \\ 93 & 90 & 70 & 117 \\ 70 & 53 & 47 & 40 \end{pmatrix}.$$

Maticový zápis je přehledný – z jednotlivých matic lze snadno vyčíst potřebné údaje, např. že z uvedeného zboží byl největší zájem o mělké oranžové talíře, z barev byla nejpreferovanější modrá (součty ve sloupcích matice $L + U + B$ jsou: 0. sloupec (modrá): 820, 1. sloupec (zelená): 720, 2. sloupec (červená): 620, 3. sloupec (oranžová): 780); nejžádanější byly hluboké talíře (součty v řádcích matice $L + U + B$ jsou: 0. řádek (hluboké talíře): 1200, 1. řádek (mělké talíře): 1100, 2. řádek (dezertní talíře): 630).

Aritmetické operace s maticemi mají následující vlastnosti:

Pravidlo 3.1 *(vlastnosti operací s maticemi)*

Jsou-li A, B, C matice stejného typu a k, $l \in \mathbb{R}$ (čísla), pak platí

(1) $A + B = B + A$,

(2) $(A + B) + C = A + (B + C)$,

(3) $k(A + B) = kA + kB$,

(4) $(k + l)A = kA + lA$,

(5) $k(lA) = (kl)A$,

(6) $(A + B)^T = A^T + B^T$,

(7) $(kA)^T = kA^T$.

3.3 Násobení matic

Zavedeme ještě součin matic. Protože způsob násobení matic by se mohl zdát uměle komplikovaný, uvedeme nejprve následující příklad, který by zavedení násobení matic mohl trochu přiblížit.

P ř í k l a d 3.10

Osoby A, B, C hodlají zakoupit zboží z_0, z_1, z_2, z_3, každá z osob v jiném množství. Celý nákup může každá z osob zrealizovat buď v obchodě O_0 nebo v O_1. Který obchod bude pro kterou osobu výhodnější?

V tabulce 3.4 je uvedeno, kolik kusů zboží z_1, z_2, z_3, z_4 každá z osob požaduje, tabulka 3.5 uvádí ceny zboží v obchodech O_0, O_1:

	z_0	z_1	z_2	z_3
A	6	5	3	1
B	3	6	2	2
C	3	4	3	1

Tab. 3.4. Požadované množství zboží

	O_0	O_1
z_0	1,50	1,00
z_1	2,00	2,50
z_2	5,00	4,50
z_3	16,00	17,00

Tab. 3.5. Ceny v obchodech O_0 a O_1

Údaje z tabulek je možno reprezentovat maticemi:
matice

$$P = \begin{pmatrix} 6 & 5 & 3 & 1 \\ 3 & 6 & 2 & 2 \\ 3 & 4 & 3 & 1 \end{pmatrix}$$

udává, kolik jednotlivé osoby požadují kusů zboží;
matice

$$C = \begin{pmatrix} 1,50 & 1 \\ 2 & 2,50 \\ 5 & 4,50 \\ 16 & 17 \end{pmatrix}$$

udává ceny zboží.
Z tabulek je zřejmé, že např. osoba A zaplatí v obchodě O_0:

$$6 \cdot 1,50 + 5 \cdot 2,00 + 3 \cdot 5,00 + 1 \cdot 16,00 = 50$$

a v obchodě O_1 :

$$6 \cdot 1,00 + 5 \cdot 2,50 + 3 \cdot 4,50 + 1 \cdot 17,00 = 49,$$

podobně pro osoby B, C.
Částku zaplacenou osobou A v obchodě O_0 lze zapsat jako skalární součin vektoru

$$\bar{p}_0 = (6,\ 5,\ 3,\ 1),$$

udávajícího počet kusů zboží požadovaný osobou A (0. řádek matice P) a vektoru

$$\bar{c}_0 = (1,50,\ 2,00,\ 5,00,\ 16,00),$$

udávajícího ceny zboží v obchodě O_0 (0. sloupec matice C):

$$\bar{p}_0 \bullet \bar{c}_0 = (6,\ 5,\ 3,\ 1) \bullet (1,50,\ 2,00,\ 5,00,\ 16,00) = 6 \cdot 1,50 + 5 \cdot 2,00 + 3 \cdot 5,00 + 1 \cdot 16,00 = 50,$$

částku zaplacenou osobou A v obchodě O_1 jako skalární součin

$$\bar{p}_0 \bullet \bar{c}_1 = (6,\ 5,\ 3,\ 1) \bullet (1,00,\ 2,50,\ 4,50,\ 17,00) = 6 \cdot 1,00 + 5 \cdot 2,50 + 3 \cdot 4,50 + 1 \cdot 17,00 = 49,$$

podobně pro osoby B, C:

$$\bar{p}_1 \bullet \bar{c}_0 = (3,\ 6,\ 2,\ 2) \bullet (1,50,\ 2,00,\ 5,00,\ 16,00) = 3 \cdot 1,50 + 6 \cdot 2,00 + 2 \cdot 5,00 + 2 \cdot 16,00 = 58,50$$

(částka zaplacená osobou B v obchodě O_0) atd.
Zapíšeme výsledky těchto skalárních součinů do matice, a to tak, že skalární součin $\bar{p}_0 \bullet \bar{c}_0$ napíšeme do nultého řádku a nultého sloupce, skalární součin $\bar{p}_0 \bullet \bar{c}_1$ napíšeme do nultého řádku a prvního sloupce, skalární součin $\bar{p}_1 \bullet \bar{c}_0$ napíšeme do prvního řádku a nultého sloupce atd. Dostaneme matici

$$R = \begin{pmatrix} 50 & 49 \\ 58,50 & 61 \\ 43,50 & 43,50 \end{pmatrix}.$$

Např. nultý řádek matice R vyjadřuje částku, kterou zaplatí osoba A v obchodě O_0 (prvek r_{00}) a v obchodě O_1 (prvek r_{01}), atd.
Jak je vidět, je pro osobu A výhodnější nakoupit v obchodě O_1, pro osobu B v O_0 a osoba C by zaplatila stejně v obchodě O_0 jako v obchodě O_1.

Výše uvedeným způsobem je definován součin matic $P \cdot C$ (v tomto pořadí – jak uvidíme dále, na pořadí násobených matic záleží): $R = P \cdot C$.

Nyní už zavedeme součin matic obecně.

Dvě matice lze vynásobit pouze tehdy, má-li druhá matice tolik řádků, kolik má první sloupců: je-li matice A typu $m \times n$, matice B typu $n \times p$, pak lze vypočítat součin $A \cdot B$; výsledkem bude matice (ozn. C) typu $m \times p$:

$$A_{m\times n} \cdot B_{n\times p} = C_{m\times p}.$$

Označíme-li prvky matice $C = A \cdot B$ jako c_{ij}, pak prvek c_{ij} bude skalárním součinem i-tého řádku matice A a j-tého sloupce matice B:

$$A \cdot B = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,n-1} \\ \dots & \dots & \dots & \dots \\ \mathbf{a_{i0}} & \mathbf{a_{i1}} & \dots & \mathbf{a_{i,n-1}} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,0} & \dots & a_{m-1,n-1} \end{pmatrix} \cdot \begin{pmatrix} b_{00} & \dots & \mathbf{b_{0j}} & \dots & b_{0,p-1} \\ b_{10} & \dots & \mathbf{b_{1j}} & \dots & b_{1,p-1} \\ \dots & \dots & \dots & \dots & \dots \\ b_{n-1,0} & \dots & \mathbf{b_{n-1,j}} & \dots & b_{n-1,p-1} \end{pmatrix},$$

tedy $c_{ij} = a_{i0}b_{0j} + a_{i1}b_{1j} + \dots + a_{i,n-1}b_{n-1,j}$.

Pojem 3.13 (součin matic)

Součinem matic $A = (a_{ij})$ typu $m \times n$ a $B = (b_{ij})$ typu $n \times p$ bude matice $C = (c_{ij})$ typu $m \times p$, kde c_{ij} je skalárním součinem i-tého řádku matice A a j-tého sloupce matice B:

$$A_{m\times n} \cdot B_{n\times p} = C_{m\times p}, \quad \text{kde} \quad c_{ij} = a_{i0}b_{0j} + a_{i1}b_{1j} + \dots + a_{i,n-1}b_{n-1,j}.$$

Poznámka (součin matic)

Při počítání součinu matic vždy bereme *řádky první matice* a *sloupce druhé matice*.

Příklad 3.11

Vypočítáme součin $A \cdot B = C$ pro matice:

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -3 & 4 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} -1 & 5 & 2 & 3 \\ -6 & 0 & 1 & 4 \\ 3 & -2 & -4 & -1 \end{pmatrix}.$$

Matice A je typu 2×3, matice B typu 3×4, lze tedy provést součin $A \cdot B = C$. Matice C bude typu 2×4. Prvek c_{00} vypočítáme jako skalární součin 0. řádku matice A a 0. sloupce matice B: $c_{00} = (2, -1, 0) \bullet (-1, -6, 3) = 2 \cdot (-1) + (-1) \cdot (-6) + 0 \cdot 3 = 4$ (doporučujeme ukazovat si tyto součiny přímo v maticích), prvek c_{01} vypočítáme jako skalární součin 0. řádku matice A a 1. sloupce matice B: $c_{01} = (2, -1, 0) \bullet (5, 0, -2) = 2 \cdot 5 + (-1) \cdot 0 + 0 \cdot (-2) = 10$ atd., až poslední prvek c_{13} vypočítáme jako skalární součin 1. řádku matice A a 3. sloupce matice B: $c_{13} = (-3, 4, 1) \bullet (3, 4, -1) = -3 \cdot 3 + 4 \cdot 4 + 1 \cdot (-1) = 6$:

$$C = A \cdot B = \begin{pmatrix} 2 & -1 & 0 \\ -3 & 4 & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 & 5 & 2 & 3 \\ -6 & 0 & 1 & 4 \\ 3 & -2 & -4 & -1 \end{pmatrix} =$$

$$= \begin{pmatrix} -2+6+0, & 10+0+0, & 4-1+0 & 6-4+0 \\ 3-24+3, & -15+0-2, & -6+4-4 & -9+16-1 \end{pmatrix} = \begin{pmatrix} 4 & 10 & 3 & 2 \\ -18 & -17 & -6 & 6 \end{pmatrix}.$$

Příklad 3.12

Pro matice A, B z předchozího příkladu nelze provést součin $B \cdot A$, neboť matice B má 4 sloupce a matice A má 2 řádky (čtenářům doporučujeme zkusit si matice v tomto pořadí vynásobit – zjistí, že příslušné skalární součiny řádku první matice a sloupce druhé matice nelze provést – vektory nemají stejný počet složek).

Příklad 3.13

Matice

$$A = \begin{pmatrix} -4 & 2 & 0 \\ -5 & 2 & 7 \end{pmatrix}, \quad B = \begin{pmatrix} 8 & 4 & 5 \\ -3 & 0 & 5 \end{pmatrix}$$

nelze vynásobit ani v pořadí $A \cdot B$, ani v pořadí $B \cdot A$. Stejný typ matic nezaručuje možnost jejich násobení.

Poznámka (matice vhodných typů)

V dalším textu budeme matice, jejichž součin lze v předepsaném pořadí provést (tzn. matice, u kterých je počet sloupců první matice roven počtu řádků druhé matice), nazývat *matice vhodných typů*.

Poznámka (součin řádkového a sloupcového vektoru)

Při násobení aritmetických vektorů (jakožto speciálního případu matic) je nutno rozlišovat řádkové a sloupcové vektory. Je-li v součinu první vektor chápaný jako řádek (tj. matice typu $1 \times n$) a druhý jako sloupec (tj. matice typu $n \times 1$), je výsledkem součinu matice typu 1×1 (tj. číslo; odpovídá skalárnímu součinu). Je-li první vektor sloupcový (tj. matice typu $n \times 1$) a druhý řádkový (tj. matice typu $1 \times n$), je výsledkem součinu matice typu $n \times n$. V případě, že je první vektor sloupcový a druhý řádkový, lze násobit i vektory o nestejném počtu složek – je-li první vektor typu $n \times 1$ a druhý typu $1 \times m$, je výsledkem součinu matice typu $n \times m$.

Poznámka (úmluva: součin matic)

V dalším textu budeme tečky označující součin většinou vynechávat – místo $A \cdot B$ tak budeme psát pouze AB.

Je zřejmé, že násobení matic není komutativní, tzn. obecně neplatí $AB = BA$. Jak jsme již viděli, ne každé dvě matice jsou násobitelné v obou pořadích $A \cdot B$ i $B \cdot A$; v některých případech, kdy to možné je, jsou ale matice AB a BA různých typů, a nemůže tedy platit $AB = BA$ (např. je-li matice A typu 2×3 a matice B typu 3×2, pak AB je typu 2×2 a BA je typu 3×3). Pro čtvercové matice A, B stejného řádu je možné provést oba součiny AB i BA a tyto součiny jsou stejného řádu, avšak rovnost $AB = BA$ i v tomto případě platí pouze výjimečně.

P ř í k l a d 3.14

$$A = \begin{pmatrix} -1 & 2 & 0 \\ 3 & 1 & -2 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & -3 \\ 1 & 4 \\ -1 & 0 \end{pmatrix},$$

$$A_{2\times3} \cdot B_{3\times2} = \begin{pmatrix} 0 & 11 \\ 9 & -5 \end{pmatrix}_{2\times2}, \qquad B_{3\times2} \cdot A_{2\times3} = \begin{pmatrix} -11 & 1 & 6 \\ 11 & 6 & -8 \\ 1 & -2 & 0 \end{pmatrix}_{3\times3}, \qquad A \cdot B \neq B \cdot A\,.$$

$$C = \begin{pmatrix} 2 & -1 \\ -3 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 3 & -4 \\ -1 & -2 \end{pmatrix},$$

$$C_{2\times2} \cdot D_{2\times2} = \begin{pmatrix} 7 & -6 \\ -10 & 10 \end{pmatrix}_{2\times2}, \qquad D_{2\times2} \cdot C_{2\times2} = \begin{pmatrix} 18 & -7 \\ 4 & -1 \end{pmatrix}_{2\times2}, \qquad C \cdot D \neq D \cdot C\,.$$

$$E = \begin{pmatrix} 1 & -2 \\ 1 & 8 \end{pmatrix}, \qquad F = \begin{pmatrix} 3 & 2 \\ -1 & -4 \end{pmatrix},$$

$$E_{2\times2} \cdot F_{2\times2} = \begin{pmatrix} 5 & 10 \\ -5 & -30 \end{pmatrix}_{2\times2}, \qquad F_{2\times2} \cdot E_{2\times2} = \begin{pmatrix} 5 & 10 \\ -5 & -30 \end{pmatrix}_{2\times2}, \qquad E \cdot F = F \cdot E\,.$$

P ř í k l a d 3.15

Vynásobíme danou matici A jednotkovou maticí vhodného typu:

a)

$$A = \begin{pmatrix} -5 & 3 & 12 \\ 2 & -7 & 4 \end{pmatrix}$$

$$A_{2\times3} \cdot I_{3\times3} = \begin{pmatrix} -5 & 3 & 12 \\ 2 & -7 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -5 & 3 & 12 \\ 2 & -7 & 4 \end{pmatrix} = A,$$

$$I_{2\times2} \cdot A_{2\times3} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -5 & 3 & 12 \\ 2 & -7 & 4 \end{pmatrix} = \begin{pmatrix} -5 & 3 & 12 \\ 2 & -7 & 4 \end{pmatrix} = A.$$

b)

$$A = \begin{pmatrix} 5 & -3 \\ 4 & 7 \end{pmatrix}$$

$$I_{2\times 2}\cdot A_{2\times 2}=\begin{pmatrix}1&0\\0&1\end{pmatrix}\cdot\begin{pmatrix}5&-3\\4&7\end{pmatrix}=\begin{pmatrix}5&-3\\4&7\end{pmatrix}=A.$$
$$A_{2\times 2}\cdot I_{2\times 2}=\begin{pmatrix}5&-3\\4&7\end{pmatrix}\cdot\begin{pmatrix}1&0\\0&1\end{pmatrix}=\begin{pmatrix}5&-3\\4&7\end{pmatrix}=A.$$

Poznámka (jednotková matice v roli jedničky)

V předchozím příkladu jsme viděli, že vynásobením matice A jednotkovou maticí vhodného typu se matice A nezmění. Tento výsledek není náhodný – platí $AI=A$, $IA=A$ (matice I ovšem může být pokaždé jiného typu – viz a) v předchozím příkladu). Jednotková matice má tedy při násobení matic podobnou roli jako jednička při násobení čísel (obdobně nulová matice má vlastnosti odpovídající číslu nula). Pro čtvercovou matici A navíc platí: $AI=IA$ (zde je matice I na obou stranách rovnosti stejná).

Příklad 3.16

Pro matice A, B z Příkladu 3.11 vypočítáme součin $B^T\cdot A^T$ a porovnáme s maticí $A\cdot B$:

$$B^T=\begin{pmatrix}-1&-6&3\\5&0&-2\\2&1&-4\\3&4&-1\end{pmatrix},\quad A^T=\begin{pmatrix}2&-3\\-1&4\\0&1\end{pmatrix},\quad B^T\cdot A^T=\begin{pmatrix}4&-18\\10&-17\\3&-6\\2&6\end{pmatrix}.$$

Porovnáním výsledku s maticí $A\cdot B$ zjistíme, že platí $B^T\cdot A^T=(A\cdot B)^T$.

Výsledek z předchozího příkladu není náhodný. Rovnost $(A\cdot B)^T=B^T\cdot A^T$ platí pro libovolné matice vhodných typů.

Násobení matic má následující vlastnosti:

Pravidlo 3.2 *(vlastnosti násobení matic)*

Jsou-li A, B, C, I matice vhodných typů a $k\in\mathbb{R}$ (číslo), pak platí

(1) *Násobení matic není komutativní.*

(2) *I – jednotková matice;* $A_{m\times n}\cdot I_{n\times n}=A_{m\times n}$, $I_{m\times m}\cdot A_{m\times n}=A_{m\times n}$,
A – čtvercová: $AI=IA$

(3) $kAB=AkB=ABk$,

(4) $A(BC)=(AB)C$,

(5) $A(B+C)=AB+AC$,

(6) $(A+B)C=AC+BC$,

(7) $(AB)^T=B^TA^T$,

(8) *matice A^TA a AA^T jsou symetrické.*

Poznámka (mocnina matice)

Podobně jako u čísel definujeme mocninu čtvercové matice: $A^0=I$, $A^2=A\cdot A$, $A^3=A\cdot A\cdot A$ atd. Výslovně zdůrazňujeme, že např. A^2 neobdržíme umocněním všech prvků matice A na druhou!

Příklad 3.17

Uvažujme autobusové spojení mezi lokalitami 0, 1, 2, 3, 4, přičemž nemusí být spojení mezi jakýmikoliv dvěma lokalitami. Definujeme matici A typu 5×5 o prvcích a_{ij} následujícím způsobem: položíme $a_{ii}=0$ a pro $i\neq j$ položíme $a_{ij}=1$, je-li spojení mezi i a j, a $a_{ij}=0$, není-li spojení mezi i a j. Matice A je zřejmě symetrická, neboť uvažujeme oboustranné spojení mezi i a j.

Například z matice

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

lze vyčíst, že mezi lokalitami 0 a 1 existuje spojení (neboť $a_{01}=a_{10}=1$), zatímco mezi lokalitami 0 a 2 nikoliv (neboť $a_{02}=a_{20}=0$).

Matice

$$A^2 = A \cdot A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 2 & 1 & 1 \\ 1 & 3 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 4 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

pak uvádí, kolika způsoby lze dorazit z lokality i do lokality j (nebo obráceně) s použitím dvou autobusů, tj. s jedním přestupem (nebereme přitom ohled na časovou návaznost spojů). Označíme-li prvky matice A^2 jako $(a^2)_{ij}$, pak např. prvek $(a^2)_{13} = 2$ udává, že z lokality 1 do lokality 3 (nebo obráceně) lze s jedním přestupem dorazit dvěma způsoby. Protože je

$$(a^2)_{13} = a_{10}a_{03} + a_{11}a_{13} + a_{12}a_{23} + a_{13}a_{33} + a_{14}a_{43} = 1 + 0 + 1 + 0 + 0,$$

a protože součin $a_{1k}a_{k3}$ může být roven jedné jen v případě, že $a_{1k} = 1$ a současně $a_{k3} = 1$ (a tedy existuje spojení mezi 1 a k i mezi k a 3), znamená to, že z lokality 1 do lokality 3 lze dorazit buď s přestupem v lokalitě 0 (neboť $a_{10}a_{03} = 1$) nebo v lokalitě 2 (neboť $a_{12}a_{23} = 1$), zatímco např. s přestupem v lokalitě 4 to možné není (neboť $a_{14}a_{43} = 0$).

Počet spojení mezi jednotlivými lokalitami s maximálně jedním přestupem je vyjádřen maticí

$$A + A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 2 & 1 & 2 & 1 & 1 \\ 1 & 3 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 4 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & 2 & 1 \\ 2 & 3 & 2 & 3 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 2 & 3 & 2 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Vidíme, že např. mezi lokalitami 1 a 3 jsou 3 možnosti – přímé spojení (neboť $a_{13} = 1$) a 2 možnosti s přestupem (neboť $(a^2)_{13} = 2$).

Uvažujme navíc i vlakové spojení mezi lokalitami 0 – 4, analogicky vyjádřené maticí

$$V = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

Pak matice

$$AV = A \cdot V = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 2 & 1 & 2 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

vyjadřuje, kolika způsoby lze z jedné lokality do druhé dorazit s jedním přestupem, a to nejdřív autobusem a pak vlakem. Označme prvky matice AV jako $(av)_{ij}$. Vidíme např., že $(av)_{34} = 2$ – jsou tedy 2 možnosti spojení z lokality 3 do lokality 4 (nikoliv obráceně – záleží na pořadí autobus-vlak). Protože je např.

$$(av)_{34} = a_{30}v_{04} + a_{31}v_{14} + a_{32}v_{24} + a_{33}v_{34} + a_{34}v_{44} = 0 + 1 + 1 + 0 + 0,$$

lze z lokality 3 do lokality 4 jet buď s přestupem v lokalitě 1 nebo 2.

Podobně matice

$$VA = V \cdot A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix}$$

vyjadřuje počet možností, jak z jedné lokality do druhé dorazit s jedním přestupem, a to nejdřív vlakem a pak autobusem. Všimněme si, že zatímco v opačném pořadí (autobus-vlak) byly 2 možnosti spojení z 3 do 4, v tomto případě (vlak-autobus) spojení mezi 3 a 4 není.

Matice AV a VA nejsou symetrické, ale platí $AV = (VA)^T$ (spojení z i do j v pořadí autobus-vlak je pro nás totéž jako spojení z j do i v pořadí vlak-autobus).

Počet možných spojů z jedné lokality do druhé (nebo obráceně) s maximálně jedním přestupem při použití libovolného dopravních prostředku (autobus, vlak) je dán maticí

$$A + V + A^2 + V^2 + AV + VA = \begin{pmatrix} 6 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 5 & 3 \\ 4 & 4 & 4 & 5 & 3 \\ 4 & 5 & 5 & 7 & 3 \\ 4 & 3 & 3 & 3 & 3 \end{pmatrix}.$$

Např. z lokality 1 do lokality 3 je 5 možností spojení. (přímé, dvěma autobusy s přestupem v 0 nebo 2, nejdřív autobusem a pak vlakem s přestupem v 0, nejdřív vlakem a pak autobusem s přestupem v 4).

3.4 Úlohy k procvičení

$\boxed{A}$ Teoretická část

Posuďte pravdivost následujících tvrzení:

a) Pro libovolné matice A, B stejného typu platí: $(A + B)^T = A^T + B^T$.

b) Platí: $(A^T)^T = A$.

c) Pro libovolné matice A, B vhodných typů platí: $(AB)^T = A^T B^T$.

d) Pro libovolné matice A, B vhodných typů platí: $(AB)^T = B^T A^T$.

e) Matice lze sčítat jen tehdy, jsou-li stejného typu.

f) Matice lze násobit jen tehdy, jsou-li stejného typu.

g) Sčítání matic je komutativní.

h) Násobení matic je komutativní.

i) Pro všechny matice A, B vhodných typů platí: $AB \neq BA$.

j) Aby mohlo být pro matice A, B splněno $AB = BA$, musí být obě matice čtvercové stejného řádu.

k) Aby bylo možné násobit matice A, B v obou pořadích (tj. AB i BA), je nutné, aby obě matice A, B byly čtvercové stejného řádu.

l) Přičtením jednotkové matice k matici A stejného řádu se v matici A změní pouze prvky na diagonále.

m) Vynásobením matice jednotkovou maticí vhodného typu se matice nezmění.

n) Rovnost $A \cdot I_{n\times n} = I_{n\times n} \cdot A$ platí pouze pro matici A typu $n \times n$.

o) Nelze spolu násobit matici a aritmetický vektor.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) P

b) P

c) N (platí $(AB)^T = B^T A^T$)

d) P

e) P

f) N (součin AB lze provést pouze pro $A_{m\times n}$ a $B_{n\times p}$)

g) P

h) N (např. $A = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$, $AB = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$, $BA = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$)

i) N (násobení není komutativní, ale pro některé matice je $AB = BA$)

j) P (AB lze provést, je-li $A_{m\times n}$ a $B_{n\times p}$; pak pro $BA = B_{n\times p}A_{m\times n}$ musí být $p = m$, tedy $A_{m\times n}$ a $B_{n\times m}$. Pak je $(AB)_{m\times m}$ a $(BA)_{n\times n}$. Aby mohlo být $AB = BA$, musí být výsledné matice stejného řádu, tzn. $m = n$.)

k) N (může být $A_{m\times n}$ a $B_{n\times m}$)

l) P

m) P

n) P ($A_{m\times n} \cdot I_{n\times n} = A_{m\times m}$; $I_{n\times n} \cdot A_{n\times m} = A_{n\times m}$. Aby se $AI = IA$, musí být $m = n$.)

o) N (aritmetický vektor $\bar{x}$ lze chápat jako matici typu $1 \times n$ – lze násobit $\bar{x}_{1\times n} \cdot A_{n\times m}$, resp. jako matici typu $n \times 1$ – lze násobit $A_{m\times n} \cdot \bar{x}_{n\times 1}$)

B Praktická část

1 Určete typy daných matic a vypište diagonální prvky jednotlivých matic:

a) $\begin{pmatrix} 1 & -2 & 2 & 2 \\ 2 & -3 & 7 & 6 \\ -3 & 9 & 3 & 2 \\ 1 & -1 & 5 & 5 \end{pmatrix}$ b) $\begin{pmatrix} -3 & -3 & 1 & 1 \\ 5 & 6 & -6 & -2 \\ 1 & 0 & 4 & 0 \\ 2 & 3 & -5 & -1 \\ -1 & -3 & 9 & 1 \end{pmatrix}$ c) $\begin{pmatrix} 1 & -2 & -1 & -2 & -3 \\ 3 & -6 & 1 & 6 & 5 \\ -2 & 4 & 0 & 2 & 1 \end{pmatrix}$ d) $\begin{pmatrix} -2 & 1 \\ 8 & -4 \\ -6 & 3 \\ 12 & -6 \end{pmatrix}$

e) $\begin{pmatrix} -1 & 3 & 0 \\ 2 & -5 & 4 \\ 3 & -7 & 5 \\ 1 & 0 & 10 \end{pmatrix}$ f) $\begin{pmatrix} 1 & 3 & 2 \\ 3 & 1 & 5 \\ 2 & 5 & 4 \end{pmatrix}$ g) $\begin{pmatrix} 0 & 0 & 3 & 3 & 5 \\ 0 & 0 & 1 & -5 & -7 \\ 0 & 0 & 5 & 2 & 4 \\ 0 & 0 & 7 & 1 & 2 \end{pmatrix}$ h) $\begin{pmatrix} 4 & 2 & 7 \\ 3 & 1 & 5 \\ 5 & 3 & 9 \\ 2 & 0 & 3 \\ -2 & 8 & 1 \end{pmatrix}$

i) $\begin{pmatrix} 1 \\ 4 \\ 5 \end{pmatrix}$ j) $\begin{pmatrix} 3 & -3 & 6 & 3 \end{pmatrix}$

2 Na daných maticích A, B, C a číslech $k = 2$, $l = -3$ demonstrujte vlastnosti (1) – (5) aritmetických operací s maticemi:

$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & 4 & 5 \end{pmatrix}$, $B = \begin{pmatrix} 3 & -1 & 2 \\ -2 & 0 & 4 \end{pmatrix}$, $C = \begin{pmatrix} 2 & 3 & 1 \\ -1 & 1 & 2 \end{pmatrix}$.

3 Vypočítejte z následujících vztahů x, y, u, v.

a) $3\begin{pmatrix} 2 & -1 \\ x & 3 \end{pmatrix} - 2\begin{pmatrix} y & 2 \\ 3 & -1 \end{pmatrix} = \begin{pmatrix} 4 & u-10 \\ 3 & 10-v \end{pmatrix}$

b) $2\begin{pmatrix} x & u \\ -y & 2v \end{pmatrix} + \begin{pmatrix} 2y & 4v \\ -2x & 2u \end{pmatrix} = \begin{pmatrix} 2 & -10 \\ -2 & -10 \end{pmatrix}$

4 K maticím z úlohy 1 napište matice transponované a uveďte jejich typy.

5 Vypočítejte součiny matic A a B (pokud to lze).

a) $A = \begin{pmatrix} 5 & -2 & 4 \\ 3 & -3 & 3 \\ -2 & 1 & 0 \end{pmatrix}$, $B = \begin{pmatrix} -1 & 2 & 0 \\ 3 & -4 & 1 \\ 2 & 1 & -2 \end{pmatrix}$ b) $A = \begin{pmatrix} 1 & -2 \\ 3 & -4 \end{pmatrix}$, $B = \begin{pmatrix} 2 & -3 \\ -1 & 1 \\ 0 & 2 \end{pmatrix}$

c) $A = \begin{pmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$, $B = \begin{pmatrix} 1 & -2 \\ 1 & -2 \\ 1 & -2 \end{pmatrix}$ d) $A = \begin{pmatrix} 1 & -2 & 3 \\ 2 & 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 4 & 1 & -2 \\ -1 & 3 & 2 \end{pmatrix}$

6 Pro matice z úlohy 5 vypočítejte součiny matic A^T a B^T(pokud to lze) a porovnejte s výsledky úlohy 5.

7 Na daných maticích A, B, C a číslech $k = 2$, $l = 3$ demonstrujte vlastnosti (1) – (7) násobení matic:

$A = \begin{pmatrix} 1 & 2 \\ -3 & 4 \end{pmatrix}$, $B = \begin{pmatrix} 2 & -1 \\ 0 & 3 \end{pmatrix}$, $C = \begin{pmatrix} 3 & -1 \\ 2 & 1 \end{pmatrix}$.

Výsledky B

1 a) 4×4, $a_{00} = 1$, $a_{11} = -3$, $a_{22} = 3$, $a_{33} = 5$

b) 5×4, $a_{00} = -3$, $a_{11} = 6$, $a_{22} = 4$, $a_{33} = -1$

c) 3×5, $a_{00} = 1$, $a_{11} = -6$, $a_{22} = 0$

d) 4×2, $a_{00} = -2$, $a_{11} = -4$

e) 4×3, $a_{00} = -1$, $a_{11} = -5$, $a_{22} = 5$

f) 3×3, $a_{00} = 1$, $a_{11} = 1$, $a_{22} = 4$

g) 4×5, $a_{00} = 0$, $a_{11} = 0$, $a_{22} = 5$, $a_{33} = 1$

h) 5×3, $a_{00} = 4$, $a_{11} = 1$, $a_{22} = 9$

i) 3×1, $a_{00} = 1$

j) 1×4, $a_{00} = 3$

2 Platí.

3 a) $x = 3$, $y = 1$, $u = 3$, $v = -1$, b) $x = 1 - t$, $y = t$, $u = -5 - 2s$, $v = s$; s, $t \in \mathbb{R}$

4 a) $\begin{pmatrix} 1 & 2 & -3 & 1 \\ -2 & -3 & 9 & -1 \\ 2 & 7 & 3 & 5 \\ 2 & 6 & 2 & 5 \end{pmatrix}$ b) $\begin{pmatrix} -3 & 5 & 1 & 2 & -1 \\ -3 & 6 & 0 & 3 & -3 \\ 1 & -6 & 4 & -5 & 9 \\ 1 & -2 & 0 & -1 & 1 \end{pmatrix}$ c) $\begin{pmatrix} 1 & 3 & -2 \\ -2 & -6 & 4 \\ -1 & 1 & 0 \\ -2 & 6 & 2 \\ -3 & 5 & 1 \end{pmatrix}$ d) $\begin{pmatrix} -2 & 8 & -6 & 12 \\ 1 & -4 & 3 & -6 \end{pmatrix}$

e) $\begin{pmatrix} -1 & 2 & 3 & 1 \\ 3 & -5 & -7 & 0 \\ 0 & 4 & 5 & 10 \end{pmatrix}$ f) $\begin{pmatrix} 1 & 3 & 2 \\ 3 & 1 & 5 \\ 2 & 5 & 4 \end{pmatrix}$ g) $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 1 & 5 & 7 \\ 3 & -5 & 2 & 1 \\ 5 & -7 & 4 & 2 \end{pmatrix}$ h) $\begin{pmatrix} 4 & 3 & 5 & 2 & -2 \\ 2 & 1 & 3 & 0 & 8 \\ 7 & 5 & 9 & 3 & 1 \end{pmatrix}$

i) $\begin{pmatrix} 1 & 4 & 5 \end{pmatrix}$ j) $\begin{pmatrix} 3 \\ -3 \\ 6 \\ 3 \end{pmatrix}$

typy matic: a) 4×4, b) 4×5, c) 5×3, d) 2×4, e) 3×4, f) 3×3, g) 5×4, h) 3×5, i) 1×3, j) 4×1

5 a) $AB = \begin{pmatrix} -3 & 22 & -10 \\ -6 & 21 & -9 \\ 5 & -8 & 1 \end{pmatrix}$, $BA = \begin{pmatrix} 1 & -4 & 2 \\ 1 & 7 & 0 \\ 17 & -9 & 11 \end{pmatrix}$, b) AB nelze, $BA = \begin{pmatrix} -7 & 8 \\ 2 & -2 \\ 6 & -8 \end{pmatrix}$,

c) $AB = \begin{pmatrix} 6 & -12 \\ 9 & -18 \end{pmatrix}$, $BA = \begin{pmatrix} -4 & -4 & -4 \\ -4 & -4 & -4 \\ -4 & -4 & -4 \end{pmatrix}$, d) nelze ani AB ani BA

6 Platí $A^T B^T = (BA)^T$, resp. $B^T A^T = (AB)^T$.

7 Platí.

4. MATICE Z POHLEDU INFORMATIKA

4.1 Reprezentace matice

V této knize pracujeme s maticemi typu $m \times n$, jejichž prvky jsou reálná čísla uspořádána do m řádků a n sloupců (viz Pojem 3.1). V procedurálních programovacích jazycích existují čtyři základní způsoby reprezentace matice: pomocí jednorozměrného pole, dvojrozměrného pole, pole polí nebo struktury obsahující pole polí.

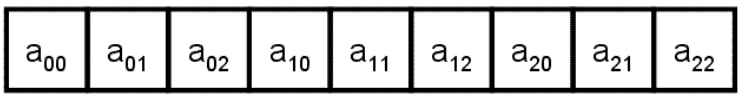
Pojem 4.1 (reprezentace matice pomocí jednorozměrného pole)

(Jednorozměrné) pole p délky $m \cdot n$ reprezentuje matici A typu $m \times n$, pokud každý prvek pole p s indexem $i \cdot n + j$ reprezentuje prvek matice A nacházející se v i-tém řádku a j-tém sloupci. Prvek pole reprezentuje prvek matice, pokud obsahuje číselnou hodnotu shodnou (až na chybu reprezentace reálných čísel v počítači) s číselnou hodnotou prvku matice.

Na tomto místě by mohla být zopakována Poznámka o reprezentaci reálných čísel nacházející se za Pojmem 2.2. (Jednorozměrné) pole je definováno v Pojmu 2.1.

Poznámka (umístění prvků matice v jednorozměrném poli)

Prvky matice jsou v poli umístěny postupně po celých řádcích. Za posledním prvkem prvního řádku matice se nachází první prvek druhého řádku matice, atd. Na obrázku 4.1 vidíme umístění prvků matice A typu 3×3 v paměti při její reprezentaci pomocí jednorozměrného pole.



Obr. 4.1. Reprezentace matice pomocí jednorozměrného pole

Poznámka (nevýhody a výhody reprezentace matice pomocí jednorozměrného pole)

Reprezentaci matice pomocí jednorozměrného pole nedoporučujeme používat. Tento způsob je jednoduchý z hlediska alokace, ale složitý z hlediska přístupu k jednotlivým prvkům matice. Alokaci provádíme pouze jednou pro každou matici, přístupy k jednotlivým prvkům matice provádíme často. Při přístupu k prvkům matice je nutné vypočítávat pozice prvků v poli na základě jejich umístění v matici. Tento výpočet je nepřehledný a bývá častým zdrojem chyb, a to i v případě, že je prováděn pomocí funkce nebo makra. V případě statické alokace je výhodnější použít reprezentaci matice pomocí dvojrozměrného pole (viz Pojem 4.3). V případě dynamické alokace je výhodnější použít reprezentaci matice pomocí pole polí (viz Pojem 4.5).

Pojem 4.2 (dvojrozměrné pole, index)

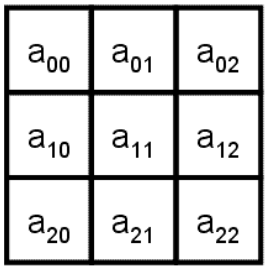
Dvojrozměrné pole p s rozměry m a n je souvislý kus paměti, ve kterém je uloženo $m \cdot n$ prvků stejného datového typu, v našem případě reálného datového typu. K jednotlivým prvkům pole lze přistupovat na základě znalosti pořadových čísel prvku v rámci jednotlivých rozměrů, tato pořadové čísla se nazývají index prvního rozměru i a index druhého rozměru j. Platí $0 \leq i < m$, $0 \leq j < n$. Uspořádaná dvojice indexu prvního rozměru a indexu druhého rozměru (i, j) se nazývá index.

Pojem 4.3 (reprezentace matice pomocí dvojrozměrného pole)

Dvojrozměrné pole p s rozměry m a n reprezentuje matici A typu $m \times n$, pokud každý prvek pole p s indexem (i, j) reprezentuje prvek matice A nacházející se v i-tém řádku a j-tém sloupci. Prvek pole reprezentuje prvek matice, pokud obsahuje číselnou hodnotu shodnou (až na chybu reprezentace reálných čísel v počítači) s číselnou hodnotou prvku matice.

Poznámka (umístění prvků matice v dvojrozměrném poli)

Prvky matice jsou v dvojrozměrném poli umístěny podobně jako jsou umístěny v matici. Dvojrozměrné pole bývá z tohoto důvodu často v literatuře označováno jako matice. Na obrázku 4.2 vidíme umístění prvků matice A typu 3 x 3 v paměti při její reprezentaci pomocí dvojrozměrného pole.



Obr. 4.2. Reprezentace matice pomocí dvojrozměrného pole

Zdrojový kód 4.1 demonstruje reprezentaci matice M typu 3×4 pomocí jednorozměrného pole `A` délky `12` (řádek `02`) a pomocí dvojrozměrného pole B s rozměry `3` a `4` (řádek `03`). V obou dvou případech se jedná o reprezentaci matice pomocí statické alokace (viz Pojem 2.3). V případě reprezentace matice pomocí dvojrozměrného pole je dynamická alokace (viz Pojem 2.4) vyloučena. V případě reprezentace matice pomocí jednorozměrného pole je dynamická alokace možná (viz Zdrojový kód 2.2).

Deklarace dvojrozměrného pole `B` (řádek `03`) je následující. Za uvedením datového typu double se nachází název pole `B` následovaný jednotlivými rozměry pole uzavřenými v hranatých závorkách, první rozměr `3` a druhý rozměr `4`.

Na řádku `04` zajistíme, že prvek pole `A` s indexem `9` (výsledek výrazu `2*4+1`) reprezentuje prvek matice M s hodnotou 7 nacházející se ve 2. řádku a 1. sloupci matice. Na řádku `05` zajistíme, že prvek pole `B` s indexem `(2,1)` reprezentuje prvek matice M s hodnotou 7 nacházející se ve 2. řádku a 1. sloupci matice.

Zdrojový kód 4.1 (alokace matice pomocí jednorozměrného a dvojrozměrného pole)

```
00 int main ()
01 {
02   double A[12];
03   double B[3][4];
04   A[2*4+1] = 7;
05   B[2][1] = 7;
06 }
```

Pojem 4.4 (pole polí, index)

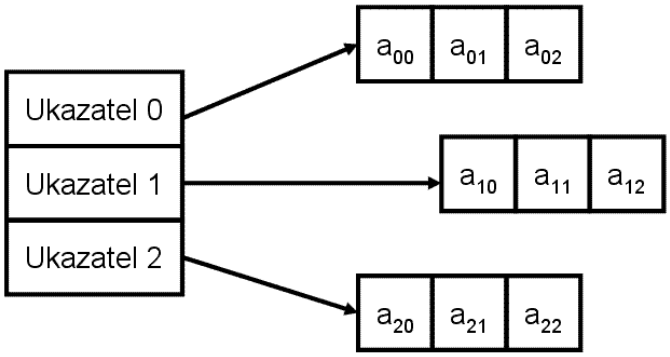
Pole polí p s rozměry m a n je datová struktura, ve které je uloženo $m \cdot n$ prvků stejného datového typu t, v našem případě reálného datového typu. Pole polí p obsahuje pole ukazatelů u délky m a m polí v_0 až v_{m-1} délky n obsahujících prvky datového typu t. Prvek pole ukazatelů u s indexem i ukazuje na pole v_i. Pozice prvku v rámci pole v_i je označena indexem j. K jednotlivým prvkům pole polí p lze přistupovat na základě znalosti jejich indexu (i, j), který je uspořádanou dvojicí indexu v poli ukazatelů i a indexu v poli prvků j.

Pojem 4.5 (reprezentace matice pomocí pole polí)

Pole polí p s rozměry m a n reprezentuje matici A typu $m \times n$, pokud každý prvek pole polí p s indexem (i, j) reprezentuje prvek matice A nacházející se v i-tém řádku a j-tém sloupci. Prvek pole reprezentuje prvek matice, pokud obsahuje číselnou hodnotu shodnou (až na chybu reprezentace reálných čísel v počítači) s číselnou hodnotou prvku matice.

Poznámka (umístění prvků matice v poli polí)

Prvky matice jsou v poli polí umístěny podobně jako jsou umístěny v matici, pouze jednotlivé řádky pole polí se mohou nacházet v nesouvisejících kusech paměti. Každý řádek je tvořen souvislým kusem paměti. Navíc je zde pole ukazatelů, které uchovává informace o umístění jednotlivých řádků v paměti. Na obrázku 4.3 vidíme umístění prvků matice A typu 3×3 v paměti při jejím reprezentaci pomocí pole polí.



Obr. 4.3. Reprezentace matice pomocí pole polí

Ve Zdrojovém kódu 4.2 se nacházejí funkce `vytvor_matici` a `smaz_matici`. Funkce `vytvor_matici` slouží k reprezentaci matice pomocí pole polí s využitím dynamické alokace. Statická alokace pole polí není proveditelná. Funkce `vytvor_matici` by měla být použita, kdykoliv chceme v našem programu pracovat s novou maticí. Funkce `smaz_matici` by měla být využita, pokud s maticí nebudeme dále pracovat. Poté se již nebudeme muset starat o problematiku dynamických alokací, která je obtížnou partií programovacího jazyka C.

Funkce `vytvor_matici` má dva parametry `m` a `n`. Parametr `m` udává velikost pole ukazatelů nazvaného `matice` (řádek `02`), velikost tohoto pole odpovídá počtu řádků matice. Parametr `n` udává velikosti polí reprezentujících jednotlivé řádky matice, tj. počet sloupců matice. Na řádku `04` je dynamicky alokováno pole ukazatelů na jednotlivé řádky matice. V cyklu (řádek `06`) dynamicky alokujeme jednotlivé řádky matice (řádek `08`). Na řádcích `05` a `09` testujeme úspěšnost provedených alokací a v případě neúspěšné alokace voláme funkci `chyba`. Výsledkem funkce je pole ukazatelů na jednotlivé řádky matice nazvané `matice` reprezentující matici typu $m \times n$

Funkce `smaz_matici` má dva parametry `m` a `n`. Parametr `n` by funkce obsahovat nemusela, ale uvádíme ho z důvodu jednotného přístupu k zadávání matice jako parametru funkce, viz Poznámka o délce pole, která následuje za Zdrojovým kódem 2.2. Při mazání matice musíme nejprve v cyklu uvolnit paměť vyhrazenou pro jednotlivé řádky matice (řádek `17`), následně můžeme uvolnit paměť vyhrazenou pro pole ukazatelů (řádek `18`). Pořadí těchto dvou řádků nelze zaměnit, došlo by k běhové chybě programu.

Zdrojový kód 4.2 (dynamická alokace a dealokace matice pomocí pole polí)

```
00 double ** vytvor_matici(int m, int n)
01 {
02   double ** matice;
03   int i;
04   matice = (double**)malloc(m*sizeof(double*));
05   if (!matice) chyba("Nedostatek pameti pro alokaci matice.\n");
06   for (i=0; i<m; i++)
07   {
08     matice[i] =(double*)malloc(n*sizeof(double));
09     if (!matice[i]) chyba("Nedostatek pameti pro alokaci matice.\n");
10   }
11   return matice;
12 }
13 void smaz_matici(double ** matice, int m, int n)
14 {
15   int i;
16   for (i=0; i<m; i++)
17     free(matice[i]);
18   free(matice);
19 }
```

Pojem 4.6 (reprezentace matice pomocí struktury obsahující pole polí)

Matici lze reprezentovat pomocí struktury, která bude mít tři položky. První položkou bude dynamicky alokované pole polí (viz Pojem 4.4). Další dvě položky budou udávat rozměry tohoto pole.

Ve Zdrojovém kódu 4.3 vidíme deklaraci struktury `matice`, jak je popsána v Pojmu 4.6. Na řádku `02` je deklarována položka `data` typu pole ukazatelů na reálné číslo, která bude obsahovat pole polí reprezentující matici. Na řádku `03` je položka `m` udávající délku pole ukazatelů `data`, tj, počet řádků matice. Na řádku `04` je položka `n` udávající délku polí reprezentujících jednotlivé řádky matice, tj, počet sloupců matice. Za deklarací struktury nesmíme zapomenout napsat středník – řádek `05`. Pro plnohodnotnou práci se strukturou `matice` by bylo nutné ještě vytvořit obdobu funkce `vytvor_matici` (viz Zdrojový kód 4.2).

Zdrojový kód 4.3 (struktura matice)

```
00 struct matice
01 {
02   double ** data;
03   int m;
04   int n;
05 };
```

Matici by šlo reprezentovat i pomocí struktury obsahující jednorozměrné pole, ale vzhledem k nevýhodnosti reprezentace matice pomocí jednorozměrného pole tuto variantu nebudeme uvažovat.

Výhodou používání struktury `matice` je redukce počtu parametrů předávaných funkcím, které s maticí pracují. Místo tří parametrů (ukazatel na pole polí reprezentující matici, počet řádků a počet sloupců) předáváme pouze jediný parametr – strukturu `matice`. Za tuto výhodu platíme daň v podobě zhoršeného přístupu k jednotlivým prvkům matice, jak demonstruje Zdrojový kód 2.4 na příkladu struktury `vektor`.

Ve Zdrojovém kódu 4.4 vidíme srovnání reprezentace matice pomocí pole polí a datové struktury `matice`. Při reprezentaci matice pomocí pole polí je přístup k jednotlivým prvkům matice intuitivní pomocí dvojího vyvolání operátoru hranatých závorek (viz řádek `00`). Při reprezentaci matice pomocí struktury `matice` musíme pro přístup k jednotlivým složkám matice nejprve provést přístup k položce `data` a teprve na ni dvakrát vyvoláme operátor hranatých závorek (viz řádek `01`). Tento postup ztěžuje orientaci ve zdrojovém kódu. Z tohoto důvodu jsme se rozhodli v celé knize používat pro reprezentaci matice dynamicky alokované pole a nikoliv strukturu `matice`.

Zdrojový kód 4.4 (přístup ke složkám matice)

```
00 a[3][0] = 4;
01 b->data[3][0] = 4;
```

Poznámka (výhodnost jednotlivých typů reprezentace matice)

Reprezentace matice pomocí jednorozměrného pole je snadná z hlediska alokace, ale nepříjemná z hlediska přístupu k jednotlivým prvkům, nedoporučujeme ji používat. Reprezentace matice pomocí dvojrozměrného pole je realizovatelná pouze při statické alokaci a lze ji použít pro matice, jejichž typ známe před kompilací programu a které zároveň nejsou parametrem nebo návratovou hodnotou funkce. Reprezentace matice pomocí pole polí je realizovatelná pouze při dynamické alokaci. Dále v této knize budeme pracovat pouze s reprezentací matice pomocí pole polí. Zajímavou variantou je i reprezentace matice pomocí struktury obsahující pole polí, ale my jsme se rozhodli preferovat reprezentaci pomocí pole polí.

Poznámka (řídké matice)

V praxi se někdy používají matice o velkém počtu složek (tisíce, desítky tisíc), z nichž většina má nulovou hodnotu. Takovéto matice se nazývají řídké matice. Řídké matice je možné reprezentovat například pomocí konečných automatů[8].

[8] Jednu z konkrétních implementací najdeme v [10].

4.2 Základní funkce pro matice

V předchozím textu jsme se seznámili se způsoby, jak reprezentovat matici v programovacím jazyku. Dohodli jsme se, že pro reprezentaci matice budeme používat pole polí. Nyní přestaneme rozlišovat mezi pojmy matice a reprezentace matice, s výjimkou situací, kdy bude potřeba tyto dva pojmy odlišit. Pro typ matice $m \times n$ budeme velmi často používat programátorské označení rozměry matice m a n. Pro prvek v i-tém řádku a j-tém sloupci budeme používat pojem prvek s indexem (i, j).

Ve Zdrojovém kódu 4.2 byla představena funkce `vytvor_matici`, která alokuje prostor pro novou matici, který můžeme vyplnit požadovanými daty. Ve Zdrojovém kódu 4.5 tuto funkci využijeme k vytvoření matice A z Příkladu 3.1. Nejprve na řádku `00` deklarujeme proměnnou `A` typu ukazatel na pole ukazatelů na reálné číslo. Na řádku `01` pomocí funkce `vytvor_matici` alokujeme prostor pro matici typu 3×4. Na řádcích `02` až `04` dosadíme do jednotlivých prvků matice požadované hodnoty (každý řádek zdrojového kódu odpovídá jednomu řádku matice). Z našeho pohledu se proměnná `A` stala maticí A z Příkladu 3.1.

Zdrojový kód 4.5 (matice A z Příkladu 3.1)

```
00 double ** A;
01 A = vytvor_matici(3,4);
02 A[0][0]=-3; A[0][1]=2; A[0][2]=0; A[0][3]=1;
03 A[1][0]=4; A[1][1]=-5; A[1][2]=9; A[1][3]=-2;
04 A[2][0]=-8; A[2][1]=3; A[2][2]=-6; A[2][3]=7;
```

Při práci s maticemi pomocí programovacího jazyka je výhodné mít funkci, která vypíše požadovanou matici na obrazovku (případně do souboru). Ve Zdrojovém kódu 4.6 je implementována funkce pro výpis matice na obrazovku. Matice je vypisována pomocí dvou vnořených cyklů. Vnější cyklus (řádek `03`) prochází jednotlivé řádky matice a pro každý řádek za pomocí vnitřního cyklu (řádek `05`) vypíše jeho jednotlivé prvky (řádek `06`). K výpisu jednotlivých prvků matice použijeme knihovní funkci `printf`, bude vypsáno `9` cifer, z toho `2` cifry za desetinnou čárkou. Po skončení vnitřního cyklu je vypsán jeden řádek matice. Provedeme odřádkování (řádek `07`), aby další řádek matice byl vypisován na další řádek obrazovky. Po skončení vnějšího cyklu je celá matice vypsána a my provedeme odřádkování (řádek `09`). Případný další vypisovaný obsah bude začínat na novém řádku obrazovky.

Zdrojový kód 4.6 (výpis matice)

```
00 void vypis_matici(double ** matice, int m, int n)
01 {
02   int i, j;
03   for (i=0; i<m; i++)
04   {
05     for (j=0; j<n; j++)
06       printf("%9.2f ", matice[i][j]);
07     printf("\n");
08   }
09   printf("\n");
10 }
```

Na tomto místě by mohla být zopakována Poznámka o klíčovém slovu `const`, která se nachází za Zdrojovým kódem 2.7.

Poznámka (řídící proměnné cyklu)

V programování bývá zvykem řídící proměnné cyklů pojmenovávat písmeny `i`, `j`, `k` atd. V rámci této knihy jsme tuto konvenci ještě upřesnili. Řídící proměnná cyklu `i` slouží k průchodu jednotlivými řádky matice, řídící proměnná cyklu `j` slouží k průchodu jednotlivými sloupci matice. Pokud je třeba použití více vnořených cyklů v rámci jedné funkce, význam dalších řídících proměnných cyklů není předem stanoven.

Poznámka (rozměry matice jako parametry funkce)

Pokud je vstupem funkce jediná matice, její rozměry jsou označeny parametry `m` (počet řádků matice) a `n` (počet sloupců matice). Pokud je vstupem funkce více matic (obvykle dvě), jsou vstupem funkce

i rozměry jednotlivých matic. Tyto rozměry jsou označeny opět parametry `m` a `n` doplněnými o název matice. Například pro matice `A` a `B` budou jejich rozměry označeny `m_A`, `n_A`, `m_B` a `n_B`.

Při programování nastává potřeba otestovat funkčnost zdrojového kódu na nějakých vstupních hodnotách. Někdy potřebujeme otestovat funkčnost pro konkrétní hodnoty (viz Zdrojový kód 4.5), jindy naopak požadujeme, aby vstupní hodnoty byly pokud možno co nejvíce náhodné. K tomuto účelu slouží funkce `nahodna_matice`, popsaná ve Zdrojovém kódu 4.7. Postupný přístup ke všem prvkům matice je nám umožněn pomocí dvou vnořených cyklů (řádky `05` a `06`). Na řádku `07` do daného prvku matice přiřazujeme náhodné číslo z intervalu `-4` až `4`. Problematika generování náhodných čísel je blíže vysvětlena u Zdrojového kódu 2.8.

Zdrojový kód 4.7 (náhodná matice)

```
00 double ** nahodna_matice(int m, int n)
01 {
02   double ** matice;
03   int i, j;
04   matice = vytvor_matici(m,n);
05   for (i=0; i<m; i++)
06     for (j=0; j<n; j++)
07       matice[i][j] = rand()%9 - 4;
08   return matice;
09 }
```

Při programování existují situace, kdy potřebujeme vytvořit kopii dat, se kterými aktuálně pracujeme. Například pokud chceme na datech vyvolat funkci, která má za úkol data upravit, a zároveň chceme mít uchována původní neupravená data. Funkce, které si v této kapitole představíme, budou tuto situaci řešit vytvořením nové matice, do které uloží modifikovaná data. Původní data zůstanou zachována. Alternativní přístup bude představen v kapitole 6 při odstupňování matic. V této situaci nebudeme mít potřebu uchovávat původní data a proto je budeme modifikovat přímo v rámci volané funkce.

K okopírování matice slouží funkce `kopie_matice` popsaná ve Zdrojovém kódu 4.8. Nejprve si vytvoříme novou matici (řádek `04`). Do jednotlivých prvků nové matice pomocí dvou vnořených cyklů (řádky `05` a `06`) okopírujeme hodnoty prvků původní matice `A` (řádek `07`).

Zdrojový kód 4.8 (kopie matice)

```
00 double ** kopie_matice(double ** A, int m, int n)
01 {
02   double ** matice;
03   int i, j;
04   matice = vytvor_matici(m,n);
05   for (i=0; i<m; i++)
06     for (j=0; j<n; j++)
07       matice[i][j] = A[i][j];
08   return matice;
09 }
```

4.3 Speciální typy matic

Zda je matice čtvercová nebo obdélníková (viz Pojem 3.2) poznáme okamžitě porovnáním rozměrů matice. Nulovou matici (viz Pojem 3.3) implementujeme poměrně snadno, například s využitím Zdrojového kódu 4.8, ve kterém modifikujeme řádek `07` tak, aby každému prvku matice přiřazoval hodnotu `0`.

Zdrojový kód 4.9 popisuje funkci `transponovana_matice`, která vytvoří transponovanou matici (viz Pojem 3.4) k zadané matici. Zdrojový kód je svojí strukturou velmi podobný Zdrojovému kódu 4.8, ve kterém jsme vytvářeli kopii matice. Významové rozdíly jsou pouze na řádcích `04` a `07`. Na řádku `04` vytváříme matici `A_t`, která má oproti původní matici `A` prohozené své rozměry – místo matice typu $m \times n$ vytváříme matici typu $n \times m$. Na řádku `07` ukládáme do nové matice `A_t` hodnoty z původní matice `A`. Hodnota, která se nachází v matici `A` v `i`-tém řádku a `j`-tém sloupci, bude v matici `A_t` uložena v `j`-tém

řádku a `i`-tém sloupci.

Zdrojový kód 4.9 (transponovaná matice)

```
00 double ** transponovana_matice(double ** A, int m, int n)
01 {
02   double ** A_t;
03   int i, j;
04   A_t = vytvor_matici(n,m);
05   for (i=0; i<m; i++)
06     for (j=0; j<n; j++)
07       A_t[j][i] = A[i][j];
08   return A_t;
09 }
```

Zdrojový kód 4.10 popisuje funkci `matice_je_symetricka`, která ověří, zda zadaná matice je symetrická (viz Pojem 3.5). Tuto vlastnost má smysl zkoumat pouze u čtvercových matic. Pro obdélníkové matice by na řádku `03` mohla být volána funkce `chyba`. My jsme se rozhodli pouze pro ukončení funkce s návratovou hodnotou `0`, která značí, že matice není symetrická. Pomocí dvou vnořených cyklů (řádky `04` a `05`) porovnáváme (řádek `06`), zda každému prvku matice s indexem `(i,j)` se rovná prvek matice s indexem `(j,i)`. Pokud rovnost neplatí je funkce ukončena s návratovou hodnotou `0`. Matice je symetrická, pokud úspěšně proběhnou oba dva vnořené cykly, poté je funkce ukončena s návratovou hodnotou `1`.

Zdrojový kód 4.10 (symetrická matice)

```
00 int matice_je_symetricka(double ** A, int m, int n)
01 {
02   int i, j;
03   if (m != n) return 0;
04   for (i=0; i<m; i++)
05     for (j=0; j<n; j++)
06       if (A[i][j] != A[j][i]) return 0;
07   return 1;
08 }
```

Poznámka (optimalizace zdrojového kódu)

Ve Zdrojovém kódu 4.10 na řádku `06` je každý prvek matice s indexem `(i,j)` porovnáván s prvkem s indexem `(j,i)` dvakrát. Jednou se nachází na levé straně nerovnosti, podruhé na pravé straně nerovnosti. Konkrétně u prvku s indexem `(3,5)` nejprve testujeme, zda se nerovná prvku s indexem `(5,3)` pomocí výrazu `A[3][5] !=A[5][3]`. Postupem času se otestuje i výraz `A[5][3] !=A[3][5]`. Pokud bychom na řádku `05` modifikovali inicializaci řídící proměnné cyklu `j=0` na `j = i+1`, vynecháme pro symetrické matice polovinu celého výpočtu. V této knize se z důvodu snahy o větší srozumitelnost základních principů jednotlivých algoritmů nepokoušíme podobné optimalizace do zdrojového kódu zanášet.

Zdrojový kód 4.11 (trojúhelníková matice)

```
00 int matice_je_trojuhelnikova(double ** A, int m, int n)
01 {
02   int i, j;
03   if (m != n) return 0;
04   for (i=0; i<m; i++)
05     for (j=0; j<i; j++)
06       if (A[i][j]) return 0;
07   return 1;
08 }
```

Zdrojový kód 4.11 popisuje funkci `matice_je_trojuhelnikova`, která ověří, zda zadaná matice je (horní) trojúhelníková (viz Pojem 3.7). Pomocí dvou vnořených cyklů (řádky `04` a `05`) procházíme (řádek

06) jednotlivé prvky pod diagonálou a pokud je mezi nimi nalezen nenulový prvek, funkce je ukončena s návratovou hodnotu 0, matice není trojúhelníková. Omezení na prvky pod diagonálou je dáno ukončovací podmínkou vnitřního cyklu `j<i`. Matice je trojúhelníková, pokud úspěšně proběhnou oba dva vnořené cykly, poté je funkce ukončena s návratovou hodnotou 1.

K ověření, že matice je diagonální (viz Pojem 3.8) musíme nejprve ověřit, že matice je čtvercová. Následně musíme projít všechny její prvky pod diagonálou i nad diagonálou. Pokud je některý z těchto prvků nenulový, matice není diagonální. V opačném případě je matice diagonální. K implementaci funkce `matice_je_diagonalni` testující, zda je matice diagonální, lze využít modifikaci Zdrojového kódu 4.12. Postačí vypustit podmínku na řádku 08.

Zdrojový kód 4.12 popisuje funkci `matice_je_jednotkova`, která ověří, zda zadaná matice je jednotková (viz Pojem 3.9). Matice musí být čtvercová (řádek 03). Pomocí dvou vnořených cyklů (řádky 04 a 05) procházíme jednotlivé prvky matice. Podmínka na řádku 07 testuje, zda prvky nacházející se pod a nad diagonálou jsou nulové. Podmínka na řádku 08 testuje, zda diagonální prvky jsou rovny jedné. V případě porušení podmínek uvedených na řádcích 07 a 08 je funkce ukončena s návratovou hodnotou 0, matice není jednotková. Matice je jednotková, pokud úspěšně proběhnou oba dva vnořené cykly, poté je funkce ukončena s návratovou hodnotou 1.

Zdrojový kód 4.12 (jednotková matice)

```
00 int matice_je_jednotkova(double ** A, int m, int n)
01 {
02   int i, j;
03   if (m != n) return 0;
04   for (i=0; i<m; i++)
05     for (j=0; j<n; j++)
06     {
07       if ((i != j) && (A[i][j])) return 0;
08       if ((i == j) && (A[i][j]) != 1) return 0;
09     }
10   return 1;
11 }
```

4.4 Aritmetické operace s maticemi

Zdrojový kód 4.13 popisuje funkci `rovnost_matic`, která ověří, zda dvě matice jsou si rovny (viz Pojem 3.10). Funkce má jako své parametry dvě matice `A` a `B` a jejich rozměry. Rozměry matice `A` jsou `m_A` a `n_A`, rozměry matice `B` jsou `m_B` a `n_B`. Pokud matice `A` a `B` nejsou stejného typu (řádek 03), nejsou si rovny, funkce je ukončena s hodnotou 0. Pomocí dvou vnořených cyklů (řádky 04 a 05) procházíme jednotlivé prvky matice `A` a porovnáváme je (řádek 06) s prvky matice `B`, které mají shodný index. Pokud si dvojice odpovídajících prvků není rovna, potom matice si nejsou rovny, funkce je ukončena s hodnotou 0. Pokud úspěšně proběhnou oba dva cykly, matice jsou si rovny, funkce je ukončena s hodnotou 1.

Zdrojový kód 4.13 (rovnost matic)

```
00 int rovnost_matic(double ** A, int m_A, int n_A, double ** B, int m_B, int n_B)
01 {
02   int i, j;
03   if ((m_B != m_A) || (n_B != n_A)) return 0;
04   for (i=0; i<m_A; i++)
05     for (j=0; j<n_A; j++)
06       if (A[i][j] != B[i][j]) return 0;
07   return 1;
08 }
```

Poznámka (přibližná rovnost matic)

V Poznámce o přesnosti aritmetických operací za Zdrojovým kódem 2.9 jsme si vysvětlili, že z důvodu zaokrouhlovacích chyb vznikajících při práci s reálnými čísly v počítači nemá smysl testovat na rovnost

dvě reálná čísla, pokud na jedno z nich byla provedena posloupnost aritmetických operací. Analogicky by nemělo smysl na rovnost testovat dvě reálné matice. Můžeme, ale testovat, zda jsou si matice alespoň přibližně rovny. Pokud se dvě reálná čísla liší nejdříve v desátém desetinném místě (nebo v osmém, dvanáctém – dle naší volby), můžeme tato čísla prohlásit za přibližně rovná. Dvě matice jsou si přibližně rovny, pokud jsou stejného typu a prvky na odpovídajících pozicích matic jsou si přibližně rovny. Jak zjistit, že se dvě reálná čísla x a y liší nejdříve v desátém desetinném místě? Nejprve zjistíme, které z čísel je větší, bez újmy na obecnosti to bude x. Pokud platí $x/y - 1 < 10^{-10}$, poté čísla x a y jsou si přibližně rovna.

Zdrojový kód 4.14 popisuje funkci `nasobeni_matice_konstantou`, která vynásobí zadanou matici `A` konstantou `kst` (viz Pojem 3.11). Výsledkem násobení matice konstantou bude nová matice `B`, původní matice `A` zůstane zachována. Pomocí dvou vnořených cyklů (řádky `05` a `06`) procházíme jednotlivé prvky matice `A` a každý její prvek vynásobíme konstantou `kst`. Výsledek násobení prvku matice `A` konstantou uložíme do odpovídajícího prvku matice `B` (řádek `07`). Výsledkem funkce je nově vzniklá matice `B`.

Zdrojový kód 4.14 (násobení matice konstantou)

```
00 double ** nasobeni_matice_konstantou(double ** A, int m, int n, int kst)
01 {
02   double ** B;
03   int i, j;
04   B = vytvor_matici(m,n);
05   for (i=0; i<m; i++)
06     for (j=0; j<n; j++)
07       B[i][j] = kst * A[i][j];
08   return B;
09}
```

Zdrojový kód 4.15 popisuje funkci `scitani_matic`, která sečte dvě matice `A` a `B` (viz Pojem 3.12). Výsledkem sčítání bude nová matice `C`. Pokud matice `A` a `B` nejsou shodného typu, je zavolána funkce `chyba`. Pomocí dvou vnořených cyklů (řádky `06` a `07`) procházíme jednotlivé prvky matic `A` a `B` a odpovídající prvky těchto matic sečteme. Výsledek uložíme do odpovídajícího prvku matice `C`. Výsledkem funkce je nově vzniklá matice `C`.

Zdrojový kód 4.15 (sčítání matic)

```
00 double ** scitani_matic(double ** A, int m_A, int n_A, double ** B, int m_B, int n_B)
01 {
02   int i, j;
03   double ** C;
04   if ((m_B != m_A) || (n_B != n_A)) chyba("Typy matic musi byt shodne.");
05   C = vytvor_matici(m_A,n_A);
06   for (i=0; i<m_A; i++)
07     for (j=0; j<n_A; j++)
08        C[i][j] = A[i][j] + B[i][j];
09   return C;
20 }
```

4.5 Násobení matic

Zdrojový kód 4.16 popisuje funkci `nasobeni_matic`, která vynásobí dvě matice `A` a `B` (viz Pojem 3.13). Nejprve musíme zkontrolovat, zda počet sloupců matice `A` odpovídá počtu řádků matice `B` (řádek `04`). Následně vytvoříme matici `C` typu `m_A x n_B` (řádek `05`), která bude výsledkem součinu matic `A` a `B`.

Všechny zdrojové kódy, které jsme doposud viděli, si vystačily se dvěma vnořenými cykly. Jeden cyklus procházel řádky matice, druhý procházel sloupce matice. V případě násobení matic je situace složitější. Potřebujeme tři vnořené cykly[9]. První dva vnořené cykly (řádky `06` a `07`) procházejí jednotlivé prvky

[9]Pro matice se stovkami sloupců a řádků je dle [15] výhodnější použít rekurzivní Strassenův algoritmus.

matice `C`. Nejprve je příslušný prvek matice `C` vynulován (řádek `09`). Hodnota prvku matice `C` s indexem `(i,j)` má být výsledkem skalárního součinu `i`-tého řádku matice `A` s `j`-tým sloupcem matice `B`. Tento skalární součin se počítá pomocí třetího vnořeného cyklu (řádek `10`), ve kterém se (řádek `11`) se postupně do prvku matice `C` s indexem `(i,j)` přičítají součiny `k`-té složky `i`-tého řádku matice `A` a `k`-té složky `j`-tého sloupce matice `B`.

Zdrojový kód 4.16 (násobení matic)

```
00 double ** nasobeni_matic(double ** A, int m_A, int n_A, double ** B, int m_B, int n_B)
01 {
02   int i, j, k;
03   double ** C;
04   if (n_A != m_B) chyba("Pocet sloupcu A neodpovida poctu radku B.");
05   C = vytvor_matici(m_A,n_B);
06   for (i=0; i<m_A; i++)
07     for (j=0; j<n_B; j++)
08     {
09       C[i][j] = 0;
10       for (k=0; k<n_A; k++)
11         C[i][j] += A[i][k] * B[k][j];
12     }
13   return C;
14 }
```

Na výpočet skalárního součinu v rámci násobení matic se lze podívat i jiným pohledem. Vraťme se ke Zdrojovému kódu 2.12, ve kterém je popsána funkce počítající skalární součin dvou vektorů. Řádek `02` z kódu 2.12 odpovídá řádku `09` z kódu 4.16. Jedná se o nulování prvku, do kterého má být výsledek skalárního součinu uložen. Řádek `05` z kódu 2.12 odpovídá řádku `10` z kódu 4.16. Jedná se o cyklus, který projde všechny složky vektorů, jejichž skalární součin počítá (v našem případě `i`-tého řádku matice `A` a `j`-tého sloupce matice `B`). Řádek `06` z kódu 2.12 odpovídá řádku `11` z kódu 4.16. Do proměnné, která uchovává výsledek skalárního součinu, postupně přičítáme součiny `k`-tých složek obou vektorů.

Zdrojový kód 4.17 (násobení matic s využitím funkce skalárního součinu)

```
00 double ** nasobeni_matic2(double ** A, int m_A, int n_A, double ** B, int m_B, int n_B)
01 {
02   int i, j;
03   double ** C;
04   double * radek;
05   double * sloupec;
06   if (n_A != m_B) chyba("Pocet sloupcu A neodpovida poctu radku B.");
07   C = vytvor_matici(m_A,n_B);
08   for (i=0; i<m_A; i++)
09     for (j=0; j<n_B; j++)
10     {
11        radek = radek_matice(A,m_A,n_A,i);
12        sloupec = sloupec_matice(B,m_B,n_B,j);
13        C[i][j] = skalarni_soucin_vektoru(radek, n_A, sloupec, m_B);
14        smaz_vektor(radek,n_A);
15        smaz_vektor(sloupec,m_B);
16     }
17   return C;
18 }
```

Zdrojový kód 4.17 popisuje funkci `nasobeni_matic2`, která vynásobí dvě matice `A` a `B` s využitím volání již implementované funkce `skalarni_soucin_vektoru` (viz Zdrojový kód 2.12). Vystačíme si pouze se dvěma zanořenými cykly, třetí cyklus je ukryt uvnitř funkce počítající skalární součin vektorů. Přestože myšlenkově je způsob násobení matic uvedený ve Zdrojovém kódu 4.17 jednodušší než způsob uvedený ve

Zdrojovém kódu 4.16, při praktické implementaci narazíme na několik problémů. Ve vnitřním zanoření cyklu musíme nejprve z matice `A` získat její řádek ve formě vektoru, k tomu si musíme implementovat funkci `radek_matice`. Následně musíme z matice `B` získat její sloupec ve formě vektoru, k tomu si musíme implementovat funkci `sloupec_matice`. Poté, co provedeme skalární součin, musíme oba dva získané vektory smazat. Ve výsledku je Zdrojový kód 4.17 rozsáhlejší na počet řádků zdrojového kódu (včetně pomocných funkcí) než Zdrojový kód 4.16. Rovněž kvůli opakované dynamické alokaci a mazání alokované paměti je i mnohonásobně pomalejší.

4.6 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 4.

2 Implementujte funkci `nulova_matice`, která vytvoří nulovou matici typu $m \times n$ (viz Pojem 3.3).

3 Na příkladech náhodných matic demonstrujte tvrzení $(A^T)^T = A$. Bude třeba implementovat funkci `rovnost_matic`, která rozhodne, zda dvě matice jsou si rovny (viz Pojem 3.10)

4 Implementujte funkci `nahodna_symetricka_matice`, která vytvoří náhodnou symetrickou matici typu $n \times n$ (viz Pojem 3.5). Na příkladech náhodných symetrických matic demonstrujte tvrzení $A = A^T$.

5 Implementujte funkci `nahodna_trojuhelnikova_matice`, která vytvoří náhodnou trojúhelníkovou matici typu $m \times n$ (viz Pojem 3.7). Pomocí funkce `matice_je_trojuhelnikova` vyzkoušejte správnost vytvořené funkce `nahodna_trojuhelnikova_matice`.

6 Implementujte funkci `nahodna_diagonální_matice`, která vytvoří náhodnou diagonální matici typu $n \times n$ (viz Pojem 3.8). Pomocí funkce `matice_je_diagonální` vyzkoušejte správnost vytvořené funkce `nahodna_diagonální_matice`.

7 Implementujte funkci `jednotkova_matice`, která vytvoří jednotkovou matici typu $n \times n$ (viz Pojem 3.9). Pomocí funkce `matice_je_jednotkova` zkuste správnost vytvořené funkce `jednotkova_matice`.

8 Implementujte funkci, která rozhodne, že dvě matice jsou si přibližně rovny (viz Poznámka o přibližné rovnosti matic za Zdrojovým kódem 4.13)

9 Pomocí programu na příkladech náhodných matic demonstrujte všechny body Pravidla 3.1. Pokud platnost nenastane, použijte funkci pro přibližnou rovnost matic místo funkce pro přesnou rovnost matic.

10 Implementujte obě dvě varianty funkce pro násobení matic a rozhodněte, která se vám implementovala lépe.

11 Pomocí programu na příkladech náhodných matic demonstrujte všechny body Pravidla 3.2. Pokud platnost nenastane, použijte funkci pro přibližnou rovnost matic místo funkce pro přesnou rovnost matic.

12 Implementujte funkci, která vypočte k-tou mocninu matice A typu $n \times n$, tj, A^k (viz Poznámka za Pravidlem 3.2). Implementujte funkci, která vypočte součet $M = A^1 + \cdots + A^k$. Rozhodněte, jaké nejmenší k (ve vztahu k zadanému n) nám postačuje, aby M aplikovaná na Příklad 3.17 obsahovala autobusová spojení mezi všemi lokalitami, pokud nebudeme nijak omezovat počet přestupů.

5. HODNOST MATICE

Pojem hodnost matice souvisí s pojmem lineární nezávislost vektorů[10], který jsme pro zjednodušení výkladu v této knize nezavedli. Nicméně pojem hodnost, tak jak ho budeme definovat, pro praktické řešení problémů nepřesahujících rámec této knihy postačí.

5.1 Odstupňovaná matice

Nejdříve se seznámíme s pojmem odstupňovaná matice:

Pojem 5.1 (odstupňovaná matice)

Odstupňovaná matice je taková matice, jejíž každý nenulový řádek (tj. řádek, ve kterém je aspoň jeden nenulový prvek) má zleva aspoň o jednu nulu víc než řádek předchozí. Několik posledních řádků může být nulových (nulový řádek je řádek, jehož všechny prvky jsou rovny nule).

Pojem 5.2 (pivot)

První nenulový prvek (zleva) v každém řádku odstupňované matice se nazývá pivot.

P ř í k l a d 5.1

Matice

$$A=\begin{pmatrix}2 & 5 & 1\\ 0 & -3 & 2\\ 0 & 0 & 9\end{pmatrix},\quad B=\begin{pmatrix}4 & 5 & 0 & -1\\ 0 & 3 & 6 & 2\\ 0 & 0 & 0 & -8\end{pmatrix},\quad C=\begin{pmatrix}0 & 3 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{pmatrix},\quad D=\begin{pmatrix}5 & 2 & -2\\ 0 & 4 & 7\\ 0 & 0 & 1\\ 0 & 0 & 0\end{pmatrix},\quad E=\begin{pmatrix}7 & 0\\ 0 & 0\end{pmatrix}$$

$$F=\begin{pmatrix}1 & 2 & 3 & 4 & -5\\ 0 & 0 & 7 & 2 & 1\\ 0 & 0 & 0 & 0 & 8\end{pmatrix},\qquad G=\begin{pmatrix}0 & 0 & 3\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix},\qquad H=\begin{pmatrix}1 & 2\\ 0 & 1\\ 0 & 0\end{pmatrix},\qquad J=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}$$

jsou odstupňované. Pivoty v matici A jsou prvky 2, -3, 9, v matici B prvky 4, 3, -8, v matici C prvky 3, 1, v matici D prvky 5, 4, 1, v matici E prvek 7, v v matici F prvky 1, 7, 8, matici G prvek 3, v matici H prvky 1, 1, v matici J není žádný pivot.

Matice

$$K=\begin{pmatrix}7 & 2 & 3\\ 0 & -5 & -2\\ 0 & 5 & 3\end{pmatrix},\qquad L=\begin{pmatrix}8 & -1 & 3\\ 0 & 0 & 8\\ 0 & 0 & 2\end{pmatrix},\qquad M=\begin{pmatrix}5 & 2\\ 0 & 4\\ 0 & 3\end{pmatrix},\qquad N=\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}$$

nejsou odstupňované.

Poznámka (vztah odstupňované a trojúhelníkové matice)

Každá čtvercová odstupňovaná matice je trojúhelníková. Ne každá trojúhelníková matice je odstupňovaná (viz matice L, N v předchozím příkladu); trojúhelníková matice s nenulovými prvky na diagonále je odstupňovaná.

5.2 Hodnost matice. Elementární řádkové úpravy

Každou matici lze pomocí určitých úprav převést na odstupňovanou. Ačkoliv můžeme různými úpravami téže matice dospět k různým odstupňovaným maticím, lze ukázat, že počet nenulových řádků ve všech odstupňovaných maticích vzniklých z téže matice těmito úpravami je stejný[11].

Pojem 5.3 (hodnost matice)

Hodností matice A (ozn. $h(A)$ nebo jen h) budeme rozumět počet nenulových řádků v libovolné odstupňované matici, která z matice A vznikne užitím následujících úprav:

(1) záměnou pořadí řádků

(2) vynásobením libovolného řádku nenulovým číslem

[10] Viz např. v [1].

[11] Na základě širší teorie je ukázáno např. v [3].

(3) přičtením násobku řádku k jinému řádku

(4) vynecháním řádku, který je násobkem jiného řádku nebo součtem jiných řádků matice A (obecněji: vynecháním řádku, který je tzv. lineární kombinací ostatních řádků[12]).

Pojem 5.4 (elementární řádkové úpravy)

Úpravy (1), (2), (3) (viz Pojem 5.3) se nazývají elementární řádkové úpravy.

Poznámka (vynechání řádku)

Úprava (4) speciálně znamená, že můžeme v matici vynechat všechny nulové řádky. Pokud jsou v matici dva stejné řádky, je možné jeden z nich vynechat. Tato úprava zjednoduší výpočty při "ručním" počítání, z hlediska programování význam nemá, nebudeme ji proto při výpočtech dále používat (až na výjimky, kde tato úprava napomůže pochopení problému).

V matici lze i vydělit řádek nenulovým číslem, neboť dělení odpovídá násobení převrácenou hodnotou tohoto čísla (viz úprava (2)).

Poznámka (nejednoznačnost odstupňovaného tvaru)

Užitím úprav (1) – (4) lze od dané matice přecházet k jiným maticím, jejichž hodnost bude ale stejná jako hodnost původní matice. Odstupňovaná matice, kterou z dané matice úpravami (1) – (4) obdržíme, není jednoznačná (různými úpravami můžeme dospět k různým odstupňovaným tvarům téže matice). Počet nenulových řádků je však ve všech odstupňovaných maticích vzniklých z téže matice stejný.

Mezi maticemi upravovanými dle (1) – (4) píšeme symbol $\sim$ (matice nalevo a napravo od symbolu $\sim$ mají stejnou hodnost).

V následujícím příkladu se soustředíme na pochopení podstaty pojmu hodnost matice – použijeme zde i úpravu (4), kterou (z důvodu uvedeného výše) jinak při výpočtech používat nebudeme.

P ř í k l a d 5.2

Matice A, B, C, D, E, F G, H, J z Příkladu 1 jsou již odstupňované – jejich hodnost je rovna počtu jejich nenulových řádků: $h(A) = 3$, $h(B) = 3$, $h(C) = 2$, $h(D) = 3$, $h(E) = 1$, $h(F) = 3$, $h(G) = 1$, $h(H) = 2$, $h(J) = 0$.

V matici K stačí přičíst 2. řádek ke 3. řádku (úprava (3)) – vznikne matice $\begin{pmatrix} 7 & 2 & 3 \\ 0 & -5 & -2 \\ 0 & 0 & 1 \end{pmatrix}$. Tato matice je už odstupňovaná, má tři nenulové řádky – je tedy $h(K) = 3$.

V matici L je 3. řádek čtyřnásobkem 2. řádku (neboli 2. řádek je $\frac{1}{4}$-násobkem 3. řádku), a je tedy možné jeden z nich vynechat (úprava (4)). Vynecháme-li např. 3. řádek, vznikne odstupňovaná matice $\begin{pmatrix} 8 & -1 & 3 \\ 0 & 0 & 8 \end{pmatrix}$, její hodnost, a tedy i hodnost matice L je 2. Je tedy $h(L) = 2$.

V matici M můžeme např. vydělit 2. řádek čtyřmi a 3. řádek třemi – vzniknou dva stejné řádky a je tedy možné jeden z nich vynechat (úpravy (2) a (4)): $\begin{pmatrix} 5 & 2 \\ 0 & 1 \end{pmatrix}$, $h(M) = 2$.

Matici N převedeme na odstupňovanou pouhou záměnou pořadí řádků (úprava (1)):

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad h(N) = 1.$$

Poznámka (hodnost dvouřádkové matice)

Hodnost matice, která má pouze dva řádky, určíme snadno: je-li jeden řádek násobkem druhého, pak ho lze vynechat (úprava (4)), a hodnost matice je tedy $h = 1$. Pokud není jeden řádek násobkem druhého, je $h = 2$. Hodnost matice, která má větší počet řádků, zpravidla na první pohled nepoznáme; to, že např. v matici o třech řádcích není žádný z řádků násobkem jiného, neznamená, že $h = 3$.

5.3 Hodnost transponované matice. Elementární sloupcové úpravy

Pravidlo 5.1 *(hodnost navzájem transponovaných matic)*

Hodnost navzájem transponovaných matic je stejná:

$$h(A) = h(A^T).$$

[12]Pojem lineární kombinace vektorů viz např. v [1].

Protože transponovaná matice vznikne z původní matice záměnou řádků za sloupce, je zřejmé, že je možné libovolnou z úprav (1) – (4) provádět také na sloupcích matice; dokonce je při převodu matice na odstupňovanou možné jakkoliv střídat úpravy řádkové se sloupcovými. Této možnosti však nebudeme využívat.

Pojem 5.5 (elementární sloupcové úpravy)

Úpravy (1), (2), (3) (viz Pojem 5.3) prováděné na sloupcích místo na řádcích matice se nazývají elementární sloupcové úpravy.

Poznámka (platnost pro sloupce)

Postřehy uvedené pro řádky tedy platí i pro sloupce matice – např. je-li v matici, která má pouze dva sloupce jeden sloupec násobkem druhého, pak je $h = 1$. Není-li jeden sloupec násobkem druhého, je $h = 2$.

Z rovnosti $h(A) = h(A^T)$ dále vyplývá, že např. hodnost matice A typu 4×3 nemůže být 4, ale nanejvýš 3; matice A má totiž 3 sloupce, matice A^T má tedy 3 řádky – její hodnost (a tedy i hodnost matice A) je tudíž nanejvýš 3. Z této úvahy vyplývá následující tvrzení:

Pravidlo 5.2 *(vztah hodnosti s počtem řádků a sloupců)*

Pro hodnost h matice typu $m \times n$ platí: $h \leq min\{m, n\}$.

Příklad 5.3

Matice $\begin{pmatrix} 1 & -2 & 3 & 4 & -5 \\ 9 & 0 & 7 & -2 & 1 \\ 5 & 4 & -2 & 3 & 8 \end{pmatrix}$ je typu 3×5. Její hodnost je nanejvýš 3.

Matice $\begin{pmatrix} 5 & -2 & -2 \\ 3 & 4 & 7 \\ -2 & 9 & 1 \\ 1 & 8 & 0 \end{pmatrix}$ je typu 4×3. Její hodnost je nanejvýš 3.

Matice $\begin{pmatrix} 1 & 2 \\ 5 & 10 \\ 4 & 8 \end{pmatrix}$ je typu 3×2. Její hodnost je nanejvýš 2. Protože však lze hodnost matice se dvěma sloupci poznat na první pohled, vidíme, že hodnost je jedna (druhý sloupec je dvojnásobkem prvního).

5.4 Výpočet hodnosti matice převodem matice na odstupňovanou

Než popíšeme podstatu převodu matice na odstupňovanou, uvědomíme si, jakými elementárními úpravami lze vynulovat daný prvek v matici. Prvek v i-tém řádku a j-tém sloupci budeme popisovat jako "prvek na místě (resp. na pozici) (i, j)" (označení a_{ij} by bylo zavádějící, protože se prvky matice během úprav mění).

Příklad 5.4

V matici $\begin{pmatrix} 2 & -5 \\ -3 & 4 \\ 8 & 7 \end{pmatrix}$ vynulujeme postupně prvky na pozici $(1, 0)$ (tzn. získáme nulu místo čísla -3) a na pozici $(2, 0)$ (tzn. získáme nulu místo čísla 8). Chceme tedy vynulovat celý nultý sloupec kromě prvku na pozici $(0, 0)$. To uděláme tak, že postupně přičteme vhodné násobky nultého řádku ke každému z následujících řádků, vynásobenému vhodným číslem. Pro názornost budeme vedle řádku uvádět číslo, kterým tento řádek násobíme, a šipkou naznačíme, ke kterému řádku násobek přičteme. Mezi upravovanými maticemi píšeme symbol $\sim$.

Nejprve vynulujeme prvek na pozici $(1, 0)$ (tzn. číslo -3), a to pomocí prvku na pozici $(0, 0)$ (tzn. čísla 2). K tomu stačí vynásobit tato čísla "jedno druhým", přičemž u jednoho změníme znaménko – např. $3 \cdot 2 + 2 \cdot (-3) = 0$. Stejnou úpravu musíme udělat s celými řádky, ve kterých tyto dva prvky leží (provádíme elementární řádkové úpravy), proto přičteme trojnásobek nultého řádku k dvojnásobku prvního řádku (tak jak je v následujícím výpočtu naznačeno vpravo od matice).

Abychom získali nulu na místě prvku $(2, 0)$, stačí přičíst minus čtyřnásobek nultého řádku ke druhému – není třeba násobit příslušnými čísly oba řádky, jak tomu bylo v případě nulování prvku na pozici $(1, 0)$.

$$\begin{pmatrix} 2 & -5 \\ -3 & 4 \\ 8 & 7 \end{pmatrix} \begin{matrix} \\ \overset{3}{\downarrow} \\ 2 \end{matrix} \begin{matrix} \\ \overset{-4}{\vert} \\ \leftarrow \end{matrix} \sim \begin{pmatrix} 2 & -5 \\ 0 & -7 \\ 0 & 27 \end{pmatrix}.$$

Poznámka (nulování prvku)

Jak jsme viděli v předchozím příkladu při nulování prvku na pozici $(2,0)$, není pro získání nuly vždy nutné násobit vhodnými čísly oba řádky. Při "ručním" počítání se tak výpočty zjednoduší, při programování se ovšem budou i v této situaci násobit oba řádky (v tomto případě např. nultý řádek -8 a druhý 2).

Nyní již popíšeme podstatu převodu matice na odstupňovanou.

1) Je-li na pozici $(0,0)$ nenulové číslo, tzn. prvek na pozici $(0,0)$ je pivot v 0. řádku budoucí odstupňované matice, získáme nuly v celém sloupci pod tímto pivotem (postupným přičítáním příslušných násobků nultého řádku k násobkům řádků následujících – tak, jak tomu bylo v předchozím příkladu).

2) Je-li na pozici $(1,1)$ nenulové číslo, tzn. prvek na pozici $(1,1)$ je pivot v 1. řádku, získáme nuly v celém sloupci pod tímto pivotem (podobně jako v předchozím kroku). Všimněme si, že při tomto postupu se nuly získané v 0. sloupci zachovají.

3) Je-li dále vždy prvek na pozici (i,i) nenulový, tzn. každý diagonální prvek je pivot, nulujeme vždy sloupec pod tímto pivotem.

Výše slovně popsaný postup naznačíme pomocí matic. Symbol $*$ bude označovat nenulový prvek a symbol x prvek, u kterého není podstatné, je-li nenulový nebo ne (prvky se při úpravách budou měnit, zde jen rozlišujeme nulovost, resp. nenulovost).

$$\begin{pmatrix} * & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & 0 & * & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & 0 & * & x \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & 0 & * & x \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Je-li při výše uvedeném postupu příslušný diagonální prvek nulový, tzn. není pivotem, pak záleží na tom, zda je některý z prvků pod ním nenulový, či zda jsou pod ním všechny prvky nulové. V prvním případě je potřeba zaměnit řádek s nulovým diagonálním prvkem za řádek, ve kterém je onen nenulový prvek – ten se stane pivotem a dále lze použít postup popsaný výše:

$$\begin{pmatrix} * & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & x & x \\ 0 & x & x & x \\ 0 & * & x & x \\ 0 & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & * & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \sim \dots$$

Je-li pod nulovým diagonálním prvkem na pozici (i,i) celý sloupec nulový, a je-li prvek vpravo od prvku (i,i) (tzn. prvek na pozici $(i,i+1)$) nenulový, je tento prvek pivotem daného řádku – pak opět stačí pomocí příslušných násobků vynulovat celý sloupec pod ním:

$$\begin{pmatrix} * & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & * & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & * & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & x \end{pmatrix} \sim \dots$$

V případě, že je i prvek $(i,i+1)$ nulový, pak opět záleží na tom, zda je pod ním nějaký nenulový prvek či ne – podle toho postupujeme analogicky situacím popsaným výše – např. :

$$\begin{pmatrix} * & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & * & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & * & x \\ 0 & 0 & 0 & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \sim \begin{pmatrix} * & x & x & x \\ 0 & 0 & * & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & x \end{pmatrix} \sim \dots$$

Poznámka (úpravy při "ručním" počítání)

Při "ručním" počítání je výhodné mít na pozici pivota nejlépe jedničku – toho lze v některých případech snadno dosáhnout záměnou pořadí řádků. Jedničku na pozici pivota lze vždy získat vydělením řádku

s pivotem hodnotou pivota, pak ale vydělením ostatních prvků v řádku nemusí vzniknout celá čísla, což je pro "ruční" počítání nepříjemné (tuto úpravu proto při "ručním" počítání nedoporučujme).

Pokud je některý řádek dělitelný nějakým číslem, je pro usnadnění počítání vhodné řádek tímto číslem vydělit. Úpravy usnadňující "ruční" počítání se ale při programování nepoužívají, proto je nebudeme ani v následujících příkladech zdůrazňovat a procvičovat.

P ř í k l a d 5.5

Převedeme matici

$$\begin{pmatrix} 2 & -2 & 3 & 0 \\ -3 & 3 & -4 & 3 \\ 5 & -4 & 3 & 5 \\ 2 & 1 & -12 & 10 \end{pmatrix}.$$

na odstupňovanou a zjistíme tak její hodnost.

Matici budeme upravovat výše uvedeným postupem – pomocí prvku na místě $(0,0)$ chceme dostat nuly pod ním. Abychom získali nulu na místě prvku $(1,0)$, přičteme trojnásobek nultého řádku k dvojnásobku prvního řádku, tak jak je naznačeno vpravo od matice v následujícím výpočtu. Obdobně – abychom získali nuly na místě $(2,0)$, přičteme minus pětinásobek prvního řádku k dvojnásobku třetího. Abychom získali nuly na místě $(3,0)$, stačí přičíst minus jedna násobek nultého řádku ke třetímu řádku:

$$\begin{pmatrix} 2 & -2 & 3 & 0 \\ -3 & 3 & -4 & 3 \\ 5 & -4 & 3 & 5 \\ 2 & 1 & -12 & 10 \end{pmatrix} \begin{matrix} \\ 2 \\ 2 \\ \end{matrix} \begin{matrix} 3 & -5 & -1 \\ \downarrow & & \\ \leftarrow & \downarrow & \\ \leftarrow & & \downarrow \end{matrix} \sim \begin{pmatrix} 2 & -2 & 3 & 0 \\ 0 & 0 & 1 & 6 \\ 0 & 2 & -9 & 10 \\ 0 & 3 & -15 & 10 \end{pmatrix}.$$

Abychom získali nuly na patřičných místech v 1. sloupci, je nutné zaměnit pořadí řádků tak, aby prvek na místě $(1,1)$ byl nenulový. Zaměníme tedy 1. a např. 2. řádek. Dále pokračujeme tak, jak je naznačeno:

$$\begin{pmatrix} 2 & -2 & 3 & 0 \\ 0 & 2 & -9 & 10 \\ 0 & 0 & 1 & 6 \\ 0 & 3 & -15 & 10 \end{pmatrix} \begin{matrix} \\ \\ \\ 2 \leftarrow \end{matrix} \begin{matrix} \\ \\ \\ -3 \downarrow \end{matrix} \sim \begin{pmatrix} 2 & -2 & 3 & 0 \\ 0 & 2 & -9 & 10 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & -3 & -10 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \end{matrix} \begin{matrix} \\ \\ \\ 3 \downarrow \end{matrix} \sim \begin{pmatrix} 2 & -2 & 3 & 0 \\ 0 & 2 & -9 & 10 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 8 \end{pmatrix}.$$

Matice je odstupňovaná, má 4 nenulové řádky – hodnost dané matice je tedy $h = 4$.

P ř í k l a d 5.6

Vypočítáme hodnost matice

$$\begin{pmatrix} -2 & 3 & 1 & 4 & -3 \\ 2 & -3 & -5 & -1 & 3 \\ -4 & 6 & 6 & 5 & -7 \\ 6 & -9 & -11 & -6 & 9 \end{pmatrix}.$$

Upravujeme nám již známým způsobem. Při "ručním" počítání využijeme toho, že v nultém sloupci jsou pod prvkem na pozici $(0,0)$ všechny prvky jeho celočíselnými násobky; při počítačovém zpracování bychom ale této skutečnosti nevyužili (jak již bylo řečeno v Poznámce za Příkladem 5.4).

$$\begin{pmatrix} -2 & 3 & 1 & 4 & -3 \\ 2 & -3 & -5 & -1 & 3 \\ -4 & 6 & 6 & 5 & -7 \\ 6 & -9 & -11 & -6 & 9 \end{pmatrix} \begin{matrix} 1 & -2 & 3 \\ \downarrow & & \\ \leftarrow & \downarrow & \\ \leftarrow & & \downarrow \end{matrix} \sim \begin{pmatrix} -2 & 3 & 1 & 4 & -3 \\ 0 & 0 & -4 & 3 & 0 \\ 0 & 0 & 4 & -3 & -1 \\ 0 & 0 & -8 & 6 & 0 \end{pmatrix}.$$

Vidíme, že nejen prvek na místě $(1,1)$, ale i celý 1. sloupec pod ním je nulový. Budeme se tedy snažit získat nuly pod prvkem na místě $(1,2)$ (při "ručním" počítání bychom ještě mohli vynechat buď 1. nebo 3. řádek, protože 3. řádek je dvojnásobkem 1.):

$$\begin{pmatrix} -2 & 3 & 1 & 4 & -3 \\ 0 & 0 & -4 & 3 & 0 \\ 0 & 0 & 4 & -3 & -1 \\ 0 & 0 & -8 & 6 & 0 \end{pmatrix} \begin{matrix} 1 & -2 \\ \downarrow & \\ \leftarrow & \downarrow \end{matrix} \sim \begin{pmatrix} -2 & 3 & 1 & 4 & -3 \\ 0 & 0 & -4 & 3 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matice je odstupňovaná, jsou v ní tři nenulové řádky – hodnost matice je $h = 3$.

5.5 Úlohy k procvičení

$\boxed{A}$ **Teoretická část**

Posuďte pravdivost následujících tvrzení:

a) Každá trojúhelníková matice je odstupňovaná.

b) Každá čtvercová odstupňovaná matice je trojúhelníková.

c) Hodnost libovolné matice je rovna počtu nenulových řádků v této matici.

d) Jestliže pro matici typu $m \times n$ platí $m < n$, pak $h = m$.

e) Není-li v matici typu 3×3 žádný z řádků násobkem jiného řádku, pak je $h = 3$.

f) Jsou-li v matici aspoň dva řádky takové, že jeden není násobkem druhého, pak je hodnost této matice $h \geq 2$.

g) Počet nenulových řádků v libovolné matici je roven počtu jejích nenulových sloupců.

h) Pro libovolnou matici typu 4×3 platí: $h \leq 3$.

i) Hodnost jednotkové matice typu $n \times n$ je n.

j) Hodnost trojúhelníkové matice je rovna počtu nenulových řádků v této matici.

k) Navzájem transponované matice mají stejnou hodnost.

l) Hodnost libovolné čtvercové matice řádu 4 je větší než hodnost libovolné matice řádu 3.

m) Vynásobením libovolného řádku matice libovolným reálným číslem obdržíme matici, která má stejnou hodnost jako původní matice.

n) Přičtením libovolného násobku libovolného řádku matice k jinému řádku této matice obdržíme matici, která má stejnou hodnost jako původní matice.

o) Není-li v matici typu 3×3 žádný z řádků součtem zbylých dvou, je $h = 3$.

p) Je-li v matici typu 3×4 jeden z řádků součtem zbylých dvou, je $h < 3$.

q) Hodnost čtvercové matice typu $n \times n$ je rovna n, právě když v jejím odstupňovaném tvaru není na diagonále žádný prvek nulový.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N (např. $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ je trojúhelníková, ale ne odstupňovaná)

b) P

c) N (např. hodnost matice $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ není 2)

d) N (např. pro matici $A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ je $m = 2$, $n = 3$, ale $h = 1$)

e) N (např. $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 4 & 4 & 4 \end{pmatrix}$, $h = 2$)

f) P

g) N (např. $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$))

h) P

i) P

j) N (např. $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$, $h = 2$)

k) P

l) N (hodnost matice 4×4 může být např. 2, a hodnost matice 3×3 např. 3)

m) N (platí pouze pro násobení nenulovým číslem)

n) P

o) N (např. $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$)

p) P

q) P

B Praktická část

1 Určete, které z uvedených matic jsou odstupňované; u těchto matic určete jejich hodnost.

a) $\begin{pmatrix} 2 & -3 & 1 \\ 0 & -3 & 4 \\ 0 & 0 & 3 \end{pmatrix}$ b) $\begin{pmatrix} 3 & -3 & 7 \\ 0 & 0 & -5 \\ 0 & 0 & 4 \end{pmatrix}$ c) $\begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \end{pmatrix}$ d) $\begin{pmatrix} 5 & 1 & 7 \\ 0 & 4 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & -2 \end{pmatrix}$

e) $\begin{pmatrix} -4 & 1 & 0 & 8 \\ 0 & 0 & 9 & -3 \\ 0 & 0 & 0 & 2 \end{pmatrix}$

2 Bez úprav matice určete její hodnost (všímejte si i tvaru sloupců).

a) $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ b) $\begin{pmatrix} -1 & 2 & 4 \\ -2 & 4 & 8 \end{pmatrix}$ c) $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ d) $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

e) $\begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{pmatrix}$ f) $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$ g) $\begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{pmatrix}$ h) $\begin{pmatrix} 5 & 3 & 1 \\ 0 & 0 & 0 \\ -2 & 4 & 3 \end{pmatrix}$

i) $\begin{pmatrix} -2 & 3 & -5 \\ -4 & 6 & -10 \\ 1 & 1 & 1 \end{pmatrix}$ j) $\begin{pmatrix} 1 & 2 \\ 3 & -4 \\ 5 & 7 \end{pmatrix}$ k) $\begin{pmatrix} 2 & -1 \\ 1 & -2 \\ -4 & 2 \end{pmatrix}$ l) $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3 Vypočítejte hodnost matice.

a) $\begin{pmatrix} 1 & -2 & 2 & 2 \\ 2 & -3 & 7 & 6 \\ -3 & 9 & 3 & 2 \\ 1 & -1 & 5 & 5 \end{pmatrix}$ b) $\begin{pmatrix} -3 & -3 & 1 & 1 \\ 5 & 6 & -6 & -2 \\ 1 & 0 & 4 & 0 \\ 2 & 3 & -5 & -1 \\ -1 & -3 & 9 & 1 \end{pmatrix}$ c) $\begin{pmatrix} 1 & -2 & -1 & -2 & -3 \\ 3 & -6 & 1 & 6 & 5 \\ -2 & 4 & 0 & 2 & 1 \end{pmatrix}$ d) $\begin{pmatrix} -2 & 1 \\ 8 & -4 \\ -6 & 3 \\ 12 & -6 \end{pmatrix}$

e) $\begin{pmatrix} -1 & 3 & 0 & 2 \\ 2 & -5 & 4 & -3 \\ 3 & -7 & 5 & -4 \\ 1 & 0 & 10 & 2 \end{pmatrix}$ f) $\begin{pmatrix} 1 & 3 & 2 \\ 5 & 1 & 3 \\ 7 & 1 & 4 \end{pmatrix}$ g) $\begin{pmatrix} 1 & 2 & 3 & 3 & 5 \\ 3 & 0 & 1 & -5 & -7 \\ 3 & 3 & 5 & 2 & 4 \\ 5 & 4 & 7 & 1 & 2 \end{pmatrix}$ h) $\begin{pmatrix} 4 & 2 & 7 \\ 3 & 1 & 5 \\ 5 & 3 & 9 \\ 2 & 0 & 3 \\ -2 & 8 & 1 \end{pmatrix}$

i) $\begin{pmatrix} 1 & 2 & -1 \\ 4 & 6 & -1 \\ 5 & 10 & -1 \end{pmatrix}$ j) $\begin{pmatrix} 3 & -3 & 6 & 3 \\ -2 & 4 & -2 & 2 \\ 4 & -1 & 11 & 10 \\ 2 & -5 & 1 & -4 \end{pmatrix}$

Výsledky B

1 a) odstupňovaná, $h = 3$, b) není odstupňovaná, c) odstupňovaná, $h = 1$, d) není odstupňovaná, e) odstupňovaná, $h = 3$

2 a) $h = 1$, b) $h = 1$, c) $h = 2$, d) $h = 0$, e) $h = 2$, f) $h = 1$, g) $h = 1$, h) $h = 2$, i) $h = 2$, j) $h = 2$, k) $h = 2$, l) $h = 3$

3 a) $h = 3$, b) $h = 2$, c) $h = 3$, d) $h = 1$, e) $h = 4$, f) $h = 2$, g) $h = 3$, h) $h = 2$, i) $h = 3$, j) $h = 2$

6. HODNOST MATICE Z POHLEDU INFORMATIKA

Hodnost matice (viz Pojem 5.3) lze vypočítat převedením matice na odstupňovanou matici (viz Pojem 5.1) za použití řádkových úprav: (1) prohození dvou řádků; (2) vynásobení řádku nenulovou konstantou; (3) přičtení násobku řádku k jinému řádku; (4) vynechání nulového řádku. Úpravy (1), (2) a (3) se nazývají elementární řádkové úpravy (viz Pojem 5.4).

Převod matice na odstupňovanou matici budeme zkráceně nazývat odstupňování matice. Hodnost odstupňované matice je rovna počtu nenulových řádků této matice. Pokud na matici použijeme posloupnost úprav (1) až (4), má upravená matice stejnou hodnost jako matice původní. Hodnost původní matice je rovna hodnosti matice odstupňované.

Vynechání nulového řádku nebudeme implementovat. Při ručním počítání nám tato úprava ušetří opisování nul. Při počítačovém zpracování nám nulové řádky nevadí, pokud nulových řádků není příliš mnoho vzhledem k celkovému počtu řádků matice. Při počítačovém zpracování vynechávání nulového řádku dokonce vede k prodloužení běhu programu z důvodu nutnosti alokace nové matice a překopírování hodnot z původní matice do matice s vynechaným nulovým řádkem.

6.1 Elementární řádkové úpravy

Při implementaci elementárních řádkových úprav můžeme buďto modifikovat výchozí matici, nebo vytvořit novou modifikovanou matici. Při převodu matice na odstupňovaný tvar budeme provádět dlouhou sérii elementárních řádkových úprav, přičemž matice vzniklé aplikací jednotlivých úprav nemají žádné další využití. Z tohoto důvodu jsme se rozhodli modifikovat výchozí matici. Pokud si přejeme výchozí matici zachovat, můžeme pomocí funkce `kopie_matice` (viz Zdrojový kód 4.8) vytvořit její kopii před začátkem aplikace elementárních řádkových úprav.

Ve Zdrojovém kódu 6.1 je pomocí funkce `prohod_radky_matice` implementováno prohození dvou řádků matice. Funkce v matici A prohodí dva řádky `r1` a `r2`. Na řádku `04` probíhá kontrola, zda řádky `r1` a `r2` se nacházejí v matici. Kontrolujeme, zda počet řádků matice je vyšší než pořadová čísla prohazovaných řádků. Rovněž bychom mohli kontrolovat, zda zadaná čísla řádků nejsou záporná. My jsme se rozhodli z důvodu větší přehlednosti zdrojového kódu a omezené pravděpodobnosti výskytu této chyby čísla řádků na zápornost netestovat.

V cyklu (řádek `05`) procházíme jednotlivé sloupce matice. V daném sloupci `j` prohazujeme hodnotu nacházející se na řádku `r1` s hodnotou nacházející se na řádku `r2`. Prohození hodnot probíhá na řádcích `07` až `09` za použití pomocné proměnné `tmp`. Do proměnné `tmp` je nejprve uložena hodnota prvku matice s indexem `(r1, j)`. Následně je do prvku s indexem `(r1, j)` uložena hodnota prvku s indexem `(r2, j)`. Nakonec je do prvku s indexem `(r2, j)` uložena hodnota proměnné `tmp`, která uchovává původní hodnotu prvku s indexem `(r1, j)`.

Zdrojový kód 6.1 (prohození řádků matice)

```
00 void prohod_radky_matice(double ** A, int m, int n, int r1, int r2)
01 {
02   int j;
03   double tmp;
04   if ((r1 >= m) || (r2>= m)) chyba("Radek se nenachazi v matici.");
05   for (j=0; j<n; j++)
06   {
07     tmp = A[r1][j];
08     A[r1][j] = A[r2][j];
09     A[r2][j] = tmp;
10   }
11 }
```

Vynásobení řádku `r1` nenulovou konstantou `k` se implementuje velmi snadno. Jako základ vezmeme Zdrojový kód 6.1. V těle cyklu místo prohazování hodnot dvou řádků budeme násobit hodnoty jednoho řádku konstantou `k`, obdobně jako na řádku `06` ve Zdrojovém kódu 2.10, ve kterém je implementována funkce `násobeni_vektoru_konstantou`. V modifikovaném Zdrojovém kódu 6.1 obsahuje tělo cyklu jediný řádek `A[r1][j] *= k;` (bez využití zkráceného operátoru `*=` bude mít tvar `A[r1][j] = A[r1][j] * k;`).

Přičtení `k` - násobku řádku `r1` k řádku `r2` se implementuje také snadno. Opět využijeme Zdrojový kód 6.1, ve kterém vyměníme tělo cyklu. Inspirujeme se funkcí `soucet_vektoru` implementovanou ve Zdrojovém kódu 2.11 – řádkem `07`. V modifikovaném Zdrojovém kódu 6.1 obsahuje tělo cyklu jediný řádek `A[r2][j] += k * A[r1][j];` (bez využití zkráceného operátoru `+=` bude řádek mít tvar `A[r2][j] = A[r2][j] + k * A[r1][j];`).

Násobení řádku konstantou a přičtení násobku řádku k jinému řádku v rámci této knihy využíváme zejména, pokud chceme v prvku nacházejícím se ve sloupci pod pivotem (viz Pojem 5.2) získat hodnotu `0`. Z důvodu chyb vznikajících při počítačovém zpracování reálných čísel (viz Poznámka za Zdrojovým kódem 2.9) je výhodnější nemít samostatné funkce pro tyto elementární řádkové úpravy. Je výhodnější provádět tyto elementární řádkové úpravy v rámci komplexnější funkce, ve které danému prvku přiřadíme hodnotu `0`. V situaci, kdy potřebujeme za daný prvek získat hodnotu `0`, by získání nenulové hodnoty (lišící se od nuly v posledním reprezentovatelném desetinném místě) způsobilo zásadní chybu, měnící smysl výpočtu, jehož částí tyto elementární řádkové úpravy byly[13].

6.2 Odstupňování matice

Odstupňování matice je natolik složitý algoritmus, že jsme ho nejprve popsali slovy a až poté ho budeme popisovat jako okomentované zdrojové kódy. Algoritmus je rozdělen do tří částí (A), (B) a (C). Část (A) popisuje modelovou situaci, část (B) řeší situaci nulového prvku na pozici pivota, část (C) řeší situaci nulového prvku na pozici pivota v kombinaci s nulovým sloupcem pod tímto prvkem.

Algoritmus 6.1 (odstupňování matice)

Část (A): Při odstupňování matice se nejprve podíváme na prvek matice s indexem (0,0) – tento prvek se nazývá prvek na pozici pivota. Pokud je prvek na pozici pivota nenulový, stává se pivotem (viz Pojem 5.2). Pomocí elementárních řádkových úprav postupně vynulujeme všechny prvky nacházející se ve sloupci pod pivotem. Proces vynulování prvku nacházejícího se ve sloupci pod pivotem se nazývá odstupňování řádku (od řádku s pivotem). Odstupňování řádku můžeme provést pomocí přičtení specifického násobku řádku s pivotem k (jinému) specifickému násobku řádku obsahujícího nulovaný prvek. Protože pivot je nenulový, vždy najdeme vhodné násobky řádků, abychom získali na daném řádku ve sloupci pod pivotem nulu. Postupně se prvkem na pozici pivota stávají další prvky na diagonále (viz Pojem 3.6). V případě jejich nenulovosti se z nich stávají pivoti. Pro každého pivota vynulujeme sloupec, který se pod ním nachází. Cyklus končí ve chvíli, kdy by se index prvku na pozici pivota měl ocitnout mimo matici. Matice je po skončení cyklu v odstupňovaném tvaru.

Část (B): V části algoritmu (A) jsme si ukázali idealizovanou situaci, kdy prvek na pozici pivota je nenulový, a stává se okamžitě pivotem. Pokud je prvek na pozici pivota nulový, najdeme ve sloupci pod ním (první) nenulový prvek. Pomocí elementární řádkové úpravy prohodíme řádek obsahující prvek na pozici pivota s řádkem, na kterém se nachází tento nalezený nenulový prvek. Nyní je na pozici pivota nenulový prvek, tento prvek se stává pivotem a pokračujeme postupem uvedeným v části algoritmu (A).

Část (C): Může nastat situace, že nulový je nejen prvek na pozici pivota, ale i celý sloupec pod ním (viz Příklad 5.6). V tomto případě se prvkem na pozici pivota stává prvek napravo od aktuálního prvku na pozice pivota. Pozor, nedochází k výměně prvků (ani sloupců), posouvá se pouze pozice pivota. Pro nový prvek na pozici pivota otestujeme, zda je nenulový, případně zda podle postupu uvedeného v části (B) může být za nenulový prvek vyměněn. V kladném případě můžeme pokračovat částí (A). V případě, že prvek na nové pozici pivota je nulový a nulový je i celý sloupec pod tímto prvkem, prvkem na pozici pivota se stává prvek napravo od něj. Tento postup lze opakovat (v případě dalších nulových prvků na pozici pivota a sloupců pod těmito prvky) až do posledního sloupce matice.

Ve Zdrojovém kódu 6.2 vidíme funkci `preusporadat_matici_pro_odstupnovani`, která řeší část (B) algoritmu 6.1. Vstupem funkce je matice `A` a index prvku na pozici pivota (`m_pivot`, `n_pivot`). Cyklus (řádek `04`) prochází jednotlivé řádky matice nacházející se na úrovni prvku na pozici pivota nebo dole pod tímto prvkem. V těle cyklu v podmínce na řádku `05` testujeme, zda na procházeném řádku `i` ve sloupci daném prvkem na pozici pivota `n_pivot` je nenulový prvek. Pokud je podmínka splněna, našli jsme (první) nenulový prvek ve sloupci pod pozicí pivota. Na řádku `07` provedeme prohození řádku `i` (obsahujícího tento nenulový prvek) s řádkem prvku na pozici pivota `m_pivot`, pokud se nejedná o shodné

[13] Podrobněji vysvětleno v [20].

řádky (`i != m_pivot`). Na pozici pivota je nyní nenulový prvek, který se stává pivotem. Celá funkce je ukončena (řádek 08).

Jakmile je v těle cyklu v podmínce na řádku 05 nalezen nenulový prvek, tak na závěr této iterace cyklu celá funkce skončí (řádek 08).

V případě, že prvek na pozici pivota je nenulový, stává se pivotem a není třeba prohazovat řádek se sebou samým, což je zajištěno nesplněním podmínky (`i != m_pivot`) na řádku 07.

Pokud ve sloupci pod prvkem na pozici pivota jsou samé nulové prvky, proběhne celý cyklus, aniž dojde k prohození řádků. Situace bude muset být řešena v nadřazené funkci dle části (C) algoritmu 6.1.

Zdrojový kód 6.2 (přeuspořádání matice pro odstupňování)

```
01 void preusporadat_matici_pro_odstupnovani(double ** A, int m, int n, int m_pivot,
int n_pivot)
02 {
03   int i;
04   for (i=m_pivot; i<m; i++)
05     if (A[i][n_pivot])
06     {
07       if (i != m_pivot) prohod_radky_matice(A,m,n,m_pivot,i);
08       return;
09     }
10 }
```

Ve Zdrojovém kódu 6.3 vidíme funkci `odstupnuj_radek_matice`, která je využita k odstupňování řádku `r` matice `A` od řádku obsahujícího pivota `m_pivot`. Pivot je prvek matice s indexem `(m_pivot, n_pivot)`. Tato problematika je v části (A) algoritmu 6.1 zmíněna jen velmi obecně, nyní se na ni podíváme podrobněji.

Řádky 04 až 07 řeší chybové stavy, pokud se řádek nebo pivot nenacházejí v matici, pokoušíme se odstupňovat řádek sám od sebe, nebo pivot je nulový. Na řádku 08 je vyřešena speciální situace, kdy daný řádek obsahuje ve sloupci daném pivotem `n_pivot` nulový prvek. Tento řádek je již odstupňovaný, funkce končí.

Zdrojový kód 6.3 (odstupňování jednoho řádku matice)

```
01 void odstupnuj_radek_matice(double ** A, int m, int n, int r, int m_pivot, int n_pivot)
02 {
03   int j;
04   if (r >= m) chyba("Radek se nenachazi v matici.");
05   if ((m_pivot >= m) || (n_pivot >= n)) chyba("Pivot není v matici.");
06   if (r == m_pivot) chyba("Nelze odstupnovat radek sam od sebe.");
07   if (!A[m_pivot][n_pivot]) chyba("Pivot nesmi byt nulovy.");
08   if (!A[r][n_pivot]) return;
09   for (j=n_pivot+1; j<n; j++)
10     A[r][j] = A[r][j]*A[m_pivot][n_pivot] - A[m_pivot][j]*A[r][n_pivot];
11   A[r][n_pivot]=0;
12 }
```

Abychom odstupňovali řádek `r` matice `A` od řádku `m_pivot`, použijeme dvě elementární řádkové úpravy. Nejprve řádek `r` vynásobíme hodnotou pivota `A[m_pivot][n_pivot]`. Následně od tohoto vynásobeného řádku `r` odečteme řádek `m_pivot` vynásobený hodnotou prvku nacházejícího se na řádku `r` ve sloupci daném pivotem `A[r][n_pivot]`.

Cyklus na řádku 09 prochází jednotlivé sloupce matice `A` od sloupce nacházejícího se napravo od pivota `n_pivot+1`. Vlevo od pivota jsou nulové hodnoty, těmi se nemusíme zabývat. V těle cyklu (řádek 10) prvkům matice v odstupňovávaném řádku `r` a aktuálně zpracovávaném sloupci `j` přiřazujeme novou hodnotu, která vznikne vynásobením aktuálně zpracovávaného prvku hodnotou `A[m_pivot][n_pivot]` a odečtením `A[r][n_pivot]` násobku prvku v odpovídajícím sloupci řádku `m_pivot`.

Sloupec obsahující pivota je řešen samostatně na řádku `11`. Pokud bychom vzorec uvedený na řádku `10` aplikovali i na prvek nacházející se ve sloupci pod pivotem, měli bychom z matematického hlediska získat hodnotu `0`. Vzhledem k zaokrouhlovacím chybám vznikajícím při počítačovém zpracování reálných čísel by tomu tak být nemuselo. Z tohoto důvodu provádíme přiřazení nuly explicitně.

Ve Zdrojovém kódu 6.4 vidíme funkci `odstupnuj_matici`, která odstupňuje zadanou matici `A`. Hlavní cyklus (řádek `04`) prochází matici po diagonále. V každém kroku cyklu nejprve pomocí vnořeného cyklu (řádek `06`) hledáme nenulového pivota. Následně po skončení tohoto vnořeného cyklu v druhém vnořeném cyklu (řádek `11`) odstupňujeme řádky nacházející se pod pivotem od řádku obsahujícího pivota. Jeden krok hlavního cyklu odpovídá jednomu přechodu mezi dvěma maticemi uvedenými v kapitole 5.4 v rámci popisu postupu odstupňování matice.

Nejprve ve vnořeném cyklu (řádek `06`) procházíme matici po sloupcích za účelem získání nenulového prvku na pozici pivota. Pomocí funkce `preusporadat_matici_pro_odstupnovani` se pokusíme nalézt nenulový prvek v aktuálním sloupci `j` (řádek `08`). Pokud jsme získali nenulový prvek na pozici pivota, vnořený cyklus končí (řádek `09`). V opačném případě pokračujeme v hledání nenulového prvku v dalším sloupci dle části (C) algoritmu 6.1.

Na řádku `11` pomocí vnořeného cyklu procházíme matici po řádcích. Začínáme na řádku nacházejícím se pod řádkem obsahujícím pivota `k+1` a pokračujeme směrem dolů. Jednotlivé řádky matice odstupňujeme od řádku obsahujícího pivota (řádek `12`) pomocí funkce `odstupnuj_radek_matice`.

Zdrojový kód 6.4 (odstupňování matice)

```
01 void odstupnuj_matici(double ** A, int m, int n)
02 {
03   int i, j, k;
04   for (k=0; (k<n) && (k<m); k++)
05   {
06     for (j=k; j<n; j++)
07     {
08       preusporadat_matici_pro_odstupnovani(A,m,n,k,j);
09       if (A[k][j]) break;
10     }
11     for (i=k+1; i<m; i++)
12       odstupnuj_radek_matice(A,m,n,i,k,j);
13   }
14 }
```

Poznámka (optimalizace zdrojového kódu)

Ve Zdrojovém kódu 6.4 v cyklu na řádku `06` výraz `k-j` určuje, kolik sloupců bylo přeskočeno, protože obsahovaly pouze nulové prvky ve sloupci pod nulovým prvkem na pozici pivota. Pokud bychom tento rozdíl uložili do pomocné proměnné `tmp`, mohli bychom tuto proměnnou využít při příštím vyvolání tohoto cyklu (pro další diagonální prvek). Cyklus by začínal od hodnoty `k+tmp` místo od hodnoty `k`. U běžných matic nenastává přeskakování sloupců příliš často, proto jsme se rozhodli tuto optimalizaci neprovádět. Rovněž by se snížila srozumitelnost zdrojového kódu.

Poznámka (ruční počítání vs. program)

V rámci této poznámky budeme předpokládat, že máme již nalezeného nenulového pivota. Při ručním počítání se snažíme, aby pivot měl hodnotu jedna (viz Poznámka před Příkladem 5.5). Toho lze někdy docílit prohozením řádku obsahujícího pivota s řádkem, který obsahuje prvek s hodnotou jedna ve sloupci odpovídajícím sloupci pivota. Pokud se toto prohození řádků nenabízí, pak můžeme vydělit řádek obsahující pivota hodnotou pivota (s respektováním doporučení uvedeného v poznámce).

Při počítačovém zpracování jednička na pozici pivota nepřináší znatelné zrychlení výpočtu programu. Pokud jedničku na pozici pivota získáme prohozením dvou řádků, snížíme nepřesnost plynoucí z počítačového zpracování reálných čísel – méně násobení znamená méně příležitostí k zaokrouhlovací chybě. Ve specializovaném numerickém software se tato optimalizace pravděpodobně používá, v našem případě by se jednalo o zbytečné snížení srozumitelnosti zdrojového kódu. Pokud bychom jedničku na místě pivota získali vydělením řádku obsahujícího pivota hodnotou pivota, naopak nepřesnost plynoucí z počítačového

zpracování reálných čísel zvýšíme – dělení způsobuje závažnější nepřesnosti než násobení.

Při ručním počítání si můžeme všimnout, že jeden řádek je násobkem druhého řádku. Dle Poznámky o vynechání řádku uvedené za Pojmem 5.4 lze jeden z těchto řádků vynechat. Při počítačovém zpracování by hledání těchto dvojic řádků vedlo spíše ke zpomalení výpočtu programu než k jeho zrychlení.

6.3 Hodnost odstupňované matice

Při převodu matice na odstupňovanou matici pomocí elementárních řádkových úprav nedochází ke změně hodnosti matice. Původní matice i odstupňovaná matice mají stejnou hodnost (viz Pojem 5.3). Pro zjištění hodnosti matice nám postačí mít dvě funkce. Funkce `odstupnuj_matici` (viz Zdrojový kód 6.4) převede matici na odstupňovanou matici. Funkce `hodnost_odstupnovane_matice` (bude uvedena ve Zdrojovém kódu 6.7) zjistí hodnost této odstupňované matice.

Ve Zdrojovém kódu 6.5 je implementována funkce `pocet_nul_z_leva`, která v zadaném řádku `r` matice `A` vypočítá počet nulových prvků nalevo od prvního nenulového prvku. Na řádku `02` deklarujeme a zároveň inicializujeme proměnnou `nuly` na hodnotu `0`. Tato proměnná udává aktuální počet nul. V cyklu (řádek `04`) procházíme jednotlivé sloupce matice směrem zleva. Pokud je na řádku `r` v aktuálně procházeném sloupci nulový prvek, zvýšíme hodnotu proměnné `nuly` (řádek `05`). Pokud je tento prvek nenulový, funkce končí a vrací návratovou hodnotou `nuly` (řádek `07`). Pokud proběhne celý cyklus, aniž by byl násilně ukončen, je celý řádek nulový – návratovou hodnotou funkce je hodnota `nuly` (řádek `09`).

Zdrojový kód 6.5 (počet nul zleva)

```
00 int pocet_nul_z_leva(double ** A, int m, int n, int r)
01 {
02   int j, nuly = 0;
03   if (r >= m) chyba("Radek se nenachazi v matici.");
04   for (j=0; j<n; j++)
05   {
06     if (!A[r][j]) nuly++;
07     else return nuly;
08   }
09   return nuly;
10 }
```

Ve Zdrojovém kódu 6.6 je implementována funkce `matice_je_odstupnovana`, která testuje, zda zadaná matice `A` je odstupňovaná. Proměnná `nuly` udává počet nul na aktuálně zpracovávaném řádku matice. Proměnná `nuly_predchozi` udává počet nul na předchozím řádku matice. V cyklu (řádek `04`) procházíme jednotlivé řádky matice. Nejprve (řádek `06`) do proměnné `nuly_predchozi` uložíme hodnotu nuly, která udává počet nul na předchozím řádku. Na řádku `07` do proměnné `nuly` uložíme počet nul, které se na aktuálním řádku matice nacházejí zleva před prvním nenulovým prvkem. Pokud (řádek `08`) počet nul na předchozím řádku byl vyšší nebo roven počtu nul na aktuálním řádku, funkce končí a vrací návratovou hodnotu `0` – matice není odstupňovaná. Pokud celý cyklus proběhne v pořádku, funkce vrací návratovou hodnotu `1` – matice je odstupňovaná.

Zdrojový kód 6.6 (odstupňovaná matice)

```
00 int matice_je_odstupnovana(double ** A, int m, int n)
01 {
02   int i;
03   int nuly = -1, nuly_predchozi;
04   for (i=0; i<m; i++)
05   {
06     nuly_predchozi = nuly;
07     nuly = pocet_nul_z_leva(A,m,n,i);
08      if ((nuly_predchozi >= nuly) && (nuly != n)) return 0;
09   }
10   return 1;
11 }
```

Poznámka (první iterace cyklu)

V první iteraci cyklu je (řádek 06) do proměnné `nuly_predchozi` uložena hodnota `-1`, kterou jsme nastavili při inicializaci proměnné `nuly` (řádek 03). Podmínka (řádek 08) testující, zda počet nul na předchozím řádku je vyšší nebo roven počtu nul na aktuálním řádku, nebude v první iteraci cyklu nikdy splněna. Speciální hodnota proměnné, která (ne)projde testem podmínky v první iteraci cyklu, bývá v programování používána často.

Ve Zdrojovém kódu 6.7 je implementována funkce `hodnost_odstupnovane_matice`, která pro zadanou odstupňovanou matici `A` zjistí její hodnost. Hodnost matice bude uložená v proměnné `hodnost` (řádek 02) inicializované na hodnotu `0`. V cyklu (řádek 03) procházíme jednotlivé řádky matice. Pro každý řádek zjišťujeme počet nul zleva před prvním nenulovým prvkem. Podmínka na řádku 06 testuje, zda není řádek nulový. Každý nenulový řádek zvyšuje hodnost matice uloženou v proměnné `hodnost`. Po zpracování všech řádků matice je hodnota proměnné `hodnost` návratovou hodnotou funkce (řádek 08).

Zdrojový kód 6.7 (hodnost odstupňované matice)

```
00 int hodnost_odstupnovane_matice(double ** A, int m, int n)
01 {
02   int i, nuly, hodnost = 0;
03   for (i=0; i<m; i++)
04   {
05     nuly = pocet_nul_z_leva(A,m,n,i);
06     if  (nuly != n) hodnost++;
07   }
08   return hodnost;
09 }
```

Poznámka (test vstupního parametru)

Ve zdrojovém kódu netestujeme, zda zadaná matice `A` je skutečně odstupňovaná. Rozhodli jsme se důvěřovat nadřazené funkci, že vstup bude korektní. Pokud bychom chtěli testování odstupňovanosti matice `A` provádět, můžeme využít funkci `matice_je_odstupnovana` (viz Zdrojový kód 6.6).

6.4 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 6.

2 Implementujte funkce pro jednotlivé elementární řádkové úpravy (viz Pojem 5.4) a pro jednotlivé elementární sloupcové úpravy (viz Pojem 5.5).

3 Na příkladech náhodných matic demonstrujte tvrzení, že hodnost matice je rovna hodnosti transponované matice (viz Pravidlo 5.1).

4 Implementujte optimalizaci Zdrojového kódu 6.4 popsanou v Poznámce pod tímto zdrojovým kódem.

5 Do Zdrojového kódu 6.4 doplňte postupy pro ruční počítání uvedené v Poznámce pod tímto zdrojovým kódem. Na příkladu náhodných matic porovnejte přesnost a rychlost původního zdrojového kódu a modifikovaného zdrojového kódu.

7. SOUSTAVY LINEÁRNÍCH ROVNIC

7.1 Soustava lineárních rovnic – základní pojmy

Pojem 7.1 (soustava lineárních rovnic, pravé strany, řešení soustavy)

Systém

$$\begin{array}{rcrcrcrcl} a_{00}x_0 & + & a_{01}x_1 & + & \ldots & + & a_{0,n-1}x_{n-1} & = & b_0 \\ a_{10}x_0 & + & a_{11}x_1 & + & \ldots & + & a_{1,n-1}x_{n-1} & = & b_1 \\ & & & & & & & \vdots & \\ a_{m-1,0}x_0 & + & a_{m-1,1}x_1 & + & \ldots & + & a_{m-1,n-1}x_{n-1} & = & b_{m-1}, \end{array}$$

kde $a_{00}, ..., a_{m-1,n-1}$ a $b_0, ..., b_{m-1}$ jsou daná reálná čísla, se nazývá soustava m lineárních rovnic o n neznámých $x_0, ..., x_{n-1}$. Čísla $b_0, ..., b_{m-1}$ se nazývají pravé strany. Vektor, jehož všechny složky po dosazení za neznámé vyhovují dané soustavě lineárních rovnic, se nazývá řešení soustavy.

Pojem 7.2 (homogenní a nehomogenní soustava lineárních rovnic)

Jsou-li v soustavě lineárních rovnic všechny pravé strany rovny nule, tzn. $b_0 = ... = b_{m-1} = 0$, pak se soustava nazývá homogenní. Je-li aspoň jedno $b_i \neq 0$, soustava se nazývá nehomogenní.

Soustavy lineárních rovnic (dále pro stručnost většinou jen soustavy) lze popsat a řešit užitím matic.

Pojem 7.3 (matice soustavy)

Matice typu $m \times n$ sestávající z koeficientů u jednotlivých neznámých, tj. matice

$$A = \begin{pmatrix} a_{00} & a_{01} & \ldots & a_{0,n-1} \\ a_{10} & a_{11} & \ldots & a_{1,n-1} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m-1,0} & a_{m-1,1} & \ldots & a_{m-1,n-1} \end{pmatrix},$$

se nazývá matice soustavy.

Pojem 7.4 (rozšířená matice soustavy)

Matice, která vznikne z matice soustavy přidáním dalšího sloupce tvořeného hodnotami $b_0, ..., b_{m-1}$ (pravými stranami), se nazývá rozšířená matice soustavy; pro přehlednost oddělujeme sloupec pravých stran svislou čarou):

$$A' = \left(\begin{array}{cccc|c} a_{00} & a_{01} & \ldots & a_{0,n-1} & b_0 \\ a_{10} & a_{11} & \ldots & a_{1,n-1} & b_1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m-1,0} & a_{m-1,1} & \ldots & a_{m-1,n-1} & b_{m-1} \end{array}\right).$$

Příklad 7.1

Soustava

$$\begin{array}{rcrcl} -\ 3x_0 & + & 2x_1 & = & 0 \\ 4x_0 & - & x_1 & = & 5 \end{array}$$

je soustava dvou rovnic o dvou neznámých x_0, x_1, pravé strany jsou 0 a 5. Vektor $\bar{x} = (2, 3)$ (není v zadání příkladu vidět) je řešením této soustavy, neboť jeho složky vyhovují dané soustavě – dosazením složek do obou rovnic dostaneme: $-3 \cdot 2 + 2 \cdot 3 = 0,\ 4 \cdot 2 - 1 \cdot 3 = 5$.

Tuto soustavu lze reprezentovat rozšířenou maticí soustavy

$$\left(\begin{array}{cc|c} -3 & 2 & 0 \\ 4 & -1 & 5 \end{array}\right),$$

přičemž matice soustavy je

$$\begin{pmatrix} -3 & 2 \\ 4 & -1 \end{pmatrix}.$$

7.2 Řešitelnost soustavy

Obecně mohou nastat tři případy: soustava nemá řešení, soustava má právě jedno řešení, soustava má nekonečně mnoho řešení. V případě, že má soustava nekonečně mnoho řešení, jsou některé neznámé volitelné (tzn. můžeme za ně dosadit libovolné číslo), ostatní neznámé jsou pak jejich volbou jednoznačně určeny.

Pojem 7.5 (partikulární řešení, obecné řešení)

Dosadíme-li za volitelné neznámé konkrétní hodnoty, dostaneme tzv. partikulární řešení.

Dosadíme-li za volitelné neznámé parametry, dostaneme tzv. obecné řešení (popisující množinu všech řešení).

Poznámka (partikulární řešení dosazením nul)

V případě, že chceme získat partikulární řešení nehomogenní soustavy, budeme v naší knize pro jednoduchost vždy za volitelné neznámé dosazovat nuly.

Platí důležité tvrzení (Frobeniova věta), které nám umožní rozeznat pomocí hodnosti matice, zda daná soustava má řešení. Podle dalšího tvrzení lze navíc ještě určit počet řešení:

Pravidlo 7.1 *(Frobeniova věta – řešitelnost soustavy)*

Soustava má řešení, právě když hodnost matice soustavy (ozn. h) je rovna hodnosti rozšířené matice soustavy (ozn. h').

Pojem 7.6 (Frobeniova podmínka)

Rovnost $h = h'$ se nazývá Frobeniova podmínka.

Poznámka (vztah h a h')

V případě, že není splněna Frobeniova podmínka, tzn. $h \neq h'$, je vždy $h < h'$, přičemž h' je vždy o jednu větší než h.

Pravidlo 7.2 *(počet řešení soustavy)*

Je-li soustava řešitelná, pak:

má právě jedno řešení, právě když je hodnost matice soustavy rovna počtu neznámých;

má nekonečně mnoho řešení, právě když je hodnost matice menší než počet neznámých

(v tomto případě je $n - h$ neznámých volitelných).

Mohou tedy nastat následující případy (pro soustavu o n neznámých) – shrnutí:

Pravidlo 7.3 *(řešitelnost a počet řešení soustavy)*
$h < h'$ *... soustava nemá řešení*
$h = h'$ *... soustava má řešení*
$h = h' = n$ *... právě jedno řešení*
$h = h' < n$ *... nekonečně mnoho řešení,*
$n - h =$ *počet volitelných neznámých*
(případ $h > h'$ nastat nemůže).

Poznámka (počet řešení)

Je-li v soustavě víc neznámých než rovnic, pak soustava jistě nemá právě jedno řešení (buď řešení nemá nebo jich má nekonečně mnoho). Matice takové soustavy je totiž obdélníková typu $m \times n$, kde $m < n$; hodnost takové matice nemůže být rovna n (viz Pravidlo 5.2), a tudíž nemůže nastat případ $h = h' = n$.

Pro ostatní soustavy (tzn. pro soustavy se čtvercovou maticí nebo s obdélníkovou maticí $m \times n$, kde $m > n$) může nastat kterýkoliv z případů – soustava má právě jedno řešení, soustava má nekonečně mnoho řešení, soustava nemá řešení.

Příklad 7.2

Soustavu

$$\begin{array}{rcrcrcl} 3x_0 & - & 2x_1 & + & x_2 & = & -7 \\ & & x_1 & - & 2x_2 & = & -1 \\ & & & & 4x_2 & = & 8 \end{array}$$

lze reprezentovat rozšířenou maticí soustavy

$$\left(\begin{array}{ccc|c} 3 & -2 & 1 & -7 \\ 0 & 1 & -2 & -1 \\ 0 & 0 & 4 & 8 \end{array}\right).$$

Matice soustavy je odstupňovaná – je vidět, že $h = 3$ (pro určení h' stačí "odmyslet" si svislou čáru), $h' = 3$. Je splněno $h = h'$, soustava tedy má řešení. Protože je navíc $h = h' = n$ (počet neznámých, tj. počet sloupců matice soustavy), má soustava jediné řešení.

Příklad 7.3

Soustava reprezentovaná maticí

$$\left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & 0 & 0 & 4 \end{array}\right)$$

nemá řešení, neboť je $h = 1$ a $h' = 2$, není tedy splněna Frobeniova podmínka.

Příklad 7.4

Pro soustavu reprezentovaná maticí

$$\left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & 5 & 2 & 4 \end{array}\right)$$

je $h = 2$ a $h' = 2$, tedy je splněna Frobeniova podmínka $h = h'$ – soustava má řešení. Počet neznámých je $n = 3$, je tedy $h = h' < n$, a proto má soustava nekonečně řešení. Budou přitom 2 neznámé volitelné, neboť $n - h = 3 - 1 = 2$.

Víme, že při řešení soustav lineárních rovnic je možno zaměnit pořadí rovnic, vynásobit libovolnou rovnici nenulovým číslem, přičíst rovnici k jiné rovnici, či v některých případech vynechat některou z rovnic; množina všech řešení dané soustavy se přitom nezmění.

Pojem 7.7 (ekvivalentní soustavy)

Soustavy, které mají stejnou množinu řešení, se nazývají ekvivalentní.

Výše uvedeným úpravám rovnic odpovídají úpravy na řádcích rozšířené matice soustavy, které nemění její hodnost – viz (1) – (4) v Pojmu 3.3 (to jsou ty úpravy, které používáme při převodu matice na odstupňovanou). Ze sloupcových úprav je možno provést pouze záměnu pořadí sloupců matice soustavy; to odpovídá záměně pořadí neznámých (tuto záměnu je potřeba při výpočtu registrovat). Nebude-li to nutné, nebudeme tuto úpravu používat.

Pravidlo 7.4 *(úpravy rozšířené matice soustavy)*

Úpravami (1) – (4) na řádcích rozšířené matice soustavy, příp. záměnou pořadí sloupců v matici soustavy obdržíme matice, které reprezentují soustavy ekvivalentní s původní soustavou.

7.3 Gaussova eliminační metoda

Gaussova metoda pro řešení soustav lineárních rovnic je jednou z tzv. eliminačních metod (při nichž se eliminují – nulují – příslušné prvky v matici). Gaussova metoda spočívá v tom, že matici soustavy převedeme úpravami (1) – (4) na odstupňovanou a z tohoto tvaru pak neznámé dopočítáváme od posledního řádku směrem nahoru: z poslední rovnice (představované posledním řádkem rozšířené matice v odstupňovaném tvaru) určíme poslední neznámou, dosazením do předposlední rovnice určíme předposlední neznámou atd. Princip podrobněji vysvětlíme na následujících příkladech.

P ř í k l a d 7.5

Vyřešíme soustavu

$$\begin{array}{rrrrrrr} - & x_0 & + & 2x_1 & - & 3x_2 & = & 7 \\ & 2x_0 & - & x_1 & + & 8x_2 & = & -14 \\ - & 3x_0 & - & 3x_1 & - & 16x_2 & = & 23. \end{array}$$

Soustavu přepíšeme do matice. To, zda je soustava řešitelná, z tohoto tvaru nepoznáme. Převedeme proto matici soustavy povolenými úpravami na řádcích rozšířené matice soustavy (jsou naznačeny vpravo od matice) na odstupňovanou – získáme tak matici, reprezentující soustavu ekvivalentní s původní soustavou. Zdůrazňujeme, že úpravy provádíme vždy na celém řádku rozšířené matice – včetně čísla za čarou (reprezentujícího pravou stranu rovnice):

$$\left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 2 & -1 & 8 & -14 \\ -3 & -3 & -16 & 23 \end{array}\right) \begin{array}{l} \\ \leftarrow 2 \\ \\ \end{array} \begin{array}{l} \\ \\ \leftarrow -3 \end{array} \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 0 & 3 & 2 & 0 \\ 0 & -9 & -7 & 2 \end{array}\right) \begin{array}{l} \\ \\ \leftarrow 3 \end{array} \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{array}\right)$$

Vidíme, že $h = h' = 3 = n$, soustava má tedy právě jedno řešení. Poslední řádek upravené rozšířené matice soustavy představuje rovnici

$$-x_2 = 2,$$

odtud

$$x_2 = -2.$$

Předposlední řádek představuje rovnici

$$3x_1 + 2x_2 = 0.$$

Za x_2 už ovšem můžeme dosadit vypočtenou hodnotu $x_2 = -2$:

$$3x_1 + 2 \cdot (-2) = 0,$$

odtud

$$3x_1 - 4 = 0,$$

a tedy

$$x_1 = \frac{4}{3}\,.$$

Podobně do nulté rovnice představované nultým řádkem rozšířené matice soustavy, tj. do rovnice

$$-x_0 + 2x_1 - 3x_2 = 7,$$

můžeme rovnou dosadit vypočtené hodnoty $x_2 = -2$, $x_1 = \frac{4}{3}$:

$$-x_0 + 2 \cdot \frac{4}{3} - 3 \cdot (-2) = 7,$$

odtud postupně

$$-x_0 + \frac{8}{3} + 6 = 7 \quad \Longrightarrow \quad -x_0 = 7 - \frac{8}{3} - 6 \quad \Longrightarrow \quad -x_0 = -\frac{5}{3} \quad \Longrightarrow \quad x_0 = \frac{5}{3}.$$

Soustava má tedy řešení $x_0 = \frac{5}{3}$, $x_1 = \frac{4}{3}$, $x_2 = -2$, což můžeme zapsat pomocí aritmetického vektoru jako uspořádanou trojici

$$\bar{x} = (x_0, x_1, x_2) = \left(\frac{5}{3}, \frac{4}{3}, -2\right).$$

P ř í k l a d 7.6

Řešme soustavu reprezentovanou maticí

$$\left(\begin{array}{cccc|c} 2 & -1 & 3 & 4 & 2 \\ 3 & -2 & 4 & 5 & -1 \\ 5 & -2 & 8 & 11 & 7 \end{array}\right).$$

Soustavu přepíšeme do matice a povolenými úpravami převedeme matici soustavy na odstupňovanou:

$$\left(\begin{array}{cccc|c} 2 & -1 & 3 & 4 & 2 \\ 3 & -2 & 4 & 5 & -1 \\ 5 & -2 & 8 & 11 & 7 \end{array}\right) \begin{matrix} \\ 2\leftarrow \\ 2\leftarrow \end{matrix} \begin{matrix} \\ -3 \\ \end{matrix} \begin{matrix} \\ \\ -5 \end{matrix} \sim \left(\begin{array}{cccc|c} 2 & -1 & 3 & 4 & 2 \\ 0 & -1 & -1 & -2 & -8 \\ 0 & 1 & 1 & 2 & 4 \end{array}\right) \begin{matrix} \\ \\ \leftarrow \end{matrix} \sim \left(\begin{array}{cccc|c} 2 & -1 & 3 & 4 & 2 \\ 0 & -1 & -1 & -2 & -8 \\ 0 & 0 & 0 & 0 & -4 \end{array}\right).$$

Protože je $h = 2$, $h' = 3$, nemá soustava řešení. Poslední řádek upravené matice totiž představuje rovnici $0 \cdot x_0 + 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 = -4$, což nemůže být splněno pro žádné x_0, x_1, x_2, x_3. Toho, že soustava nemá řešení, jsme si mohli všimnout už v předešlé matici: poslední dva řádky představují (po vynásobení jednoho z řádků číslem –1) rovnice, kde neznámé jsou na levé straně svázány stejnými vztahy, hodnoty na pravých stranách jsou však různé. Takové rovnice nemohou být současně splněny. Kdykoliv takováto situace při úpravách matice nastane, je možno výpočet ukončit (a nepřevádět dál matici na odstupňovanou) – soustava jistě nemá řešení.

P ř í k l a d 7.7

Nalezneme partikulární řešení soustavy

$$\begin{aligned} -x_0 + x_1 + x_2 + x_3 &= 2 \\ x_2 + x_3 &= 3. \end{aligned}$$

Tato soustava je reprezentována maticí

$$\left(\begin{array}{cccc|c} -1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right),$$

která již je odstupňovaná. Vidíme, že hodnost matice i matice rozšířené je $h = h' = 2$, je tedy splněna Frobeniova podmínka – soustava má řešení. Počet neznámých je $n = 4$, je tedy $h < n$ – soustava má nekonečně mnoho řešení, přičemž je $n - h = 2$ – dvě neznámé jsou volitelné.

Chceme-li nalézt nějaké partikulární řešení, dosazujeme za volitelné neznámé nějaké konkrétní hodnoty. V Poznámce za Pojmem 7.5 jsme se zmínili, že v této knize budeme pro jednoduchost za volitelné neznámé dosazovat nuly.

Za které dvě neznámé můžeme nuly (příp. jiné hodnoty) dosadit? Chceme postupovat při výpočtu neznámých Gaussovou metodou od posledního řádku směrem nahoru, je proto vhodné volit hodnoty za poslední neznámé (je-li to možné). Zvolíme-li např. za neznámou x_3 nulu, tzn. $x_3 = 0$, a dosadíme-li do rovnice reprezentované posledním řádkem odstupňované matice, tzn. do rovnice $x_2 + x_3 = 3$, dostaneme $x_2 + 0 = 3$ neboli $x_2 = 3$. Vidíme, že neznámá x_2 je volbou neznámé x_3 už jednoznačně určena – neznámé x_2 a x_3 nejsou volitelné současně.

Můžeme ale k neznámé $x_3 = 0$ volit jako druhou neznámou např. x_1; opět zde pro jednoduchost zvolíme $x_1 = 0$. Z poslední rovnice už máme $x_2 = 3$. Dosadíme $x_3 = 0$, $x_2 = 3$ a $x_1 = 0$ do předchozí (tzn. první) rovnice a vypočítáme odsud zbylou neznámou x_0: $-x_0 + 0 + 3 + 0 = 2$, odtud $x_0 = 1$.

Volbou neznámých $x_3 = 0$, $x_1 = 0$ jsme dostali partikulární řešení

$$\bar{x} = (1, 0, 3, 0).$$

Mohli jsme ale k neznámé x_3 volit současně neznámou x_0, nebo jsme mohli volit dvojici x_2 a x_1 nebo dvojici x_0 a x_2 (x_0 a x_1 současně volit nelze). Problematikou volitelných neznámými se ještě budeme zabývat.

V naší knize nebude cílem vyčerpat *všechny* možnosti, jak volit neznámé – spokojíme se s určením jedné z nich. Obecně platí, že volitelné jsou ty neznámé, jimž v odstupňovaném tvaru matice soustavy odpovídají sloupce, které neobsahují pivota (neznamená to ale, že jiné neznámé volitelné nejsou). V Příkladu 7.7 jsou pivoty -1 a 1 obsažené v 1. a 3. sloupci, proto x_2 a x_4 mohou být volitelné.

V soustavě reprezentované následující maticí (nad sloupci matice jsou poznačeny odpovídající neznámé, symbol $*$ označuje nenulový prvek, symbol x označuje libovolný prvek)

$$\left(\begin{array}{ccccc|c} x_0 & x_1 & x_2 & x_3 & x_4 & \\ * & x & x & x & x & x \\ 0 & 0 & 0 & * & x & x \\ 0 & 0 & 0 & 0 & * & x \end{array}\right).$$

jsou pivoty obsaženy v 0., 3. a 4. sloupci – volitelné neznámé budou tedy x_1 a x_2.

Pravidlo 7.5 *(volitelné neznámé)*

Neznámé, jimž v odstupňovaném tvaru matice soustavy odpovídají sloupce, které neobsahují pivota, jsou volitelné.

Dosadíme-li za volitelné nikoliv konkrétní hodnoty, ale parametry (zbylé neznámé budou na těchto parametrech záviset), dostaneme obecné řešení. To popisuje množinu všech řešení dané soustavy – následným dosazením čísel za parametry do obecného řešení dostaneme jednotlivá partikulární řešení.

P ř í k l a d 7.8

Nalezneme obecné řešení soustavy z Příkladu 7.7, tzn. soustavy reprezentované maticí

$$\left(\begin{array}{cccc|c} -1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right).$$

Víme už, že volitelnými neznámými mohou být x_3 a x_1. Zvolíme za tyto neznámé parametry, např. $x_3 = t$, $x_1 = s$, a dosadíme (při Gaussově metodě dosazujeme odspoda směrem nahoru, začneme dosazením do poslední rovnice):

$$x_2 + t = 3 \qquad \text{odtud} \qquad x_2 = 3 - t.$$

Dosazením zvolených $x_1 = s$, $x_3 = t$ a vypočítaného $x_2 = 3 - t$ do předchozí rovnice dostaneme:

$$-x_0 + s + (3 - t) + t = 2 \qquad \text{odtud} \qquad x_0 = 1 + s.$$

Obecným řešením je vektor $\bar{x} = (1 + s, s, 3 - t, t)$, kde t, $s \in \mathbb{R}$.

Dosadíme-li do obecného řešení za t a s konkrétní hodnoty, dostaneme jedno partikulární řešení – např. dosazením nul za t i s dostaneme partikulární řešení z Příkladu 7.7: $\bar{x} = (1, 0, 3, 0)$.

Poznámka (úmluva: obecné řešení)

Obecným řešením se v této knize budeme zabývat pouze orientačně.

7.4 Jordanova eliminační metoda

Druhou eliminační metodou, kterou zde uvedeme, je Jordanova metoda. Ta – stejně jako Gaussova – využívá převodu matice soustavy na odstupňovanou, liší se ale od Gaussovy metody dalším postupem. Jordanova metoda totiž pokračuje dál v úpravách matice – s cílem získat nuly navíc ještě na určitých pozicích nad diagonálou. Přesněji – při Jordanově metodě se matice soustavy převede na tzv. redukovaný odstupňovaný tvar.

Pojem 7.8 (redukovaný odstupňovaný tvar matice)

Matice je v redukovaném odstupňovaném tvaru, je-li odstupňovaná, přičemž navíc

(1) všechny pivoty jsou rovny jedné

(2) ve sloupcích, ve kterých je obsažen pivot, jsou všechny prvky kromě pivotů rovny nule.

Např. matice (x značí libovolné číslo)

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

je v redukovaném odstupňovaném tvaru.

Poznámka (redukovaný odstupňovaný tvar)

Všimněme si, že odmyslíme-li si nulový řádek a sloupce obsahující čísla x, zbude jednotková matice.

Při převodu matice z odstupňovaného tvaru na redukovaný odstupňovaný tvar budeme v každém (nenulovém) řádku získávat jedničku na místě pivota tak, že celý řádek vydělíme hodnotou tohoto pivota.

Pojem 7.9 (normování pivota)

Proces tvoření jedničky na místě pivota (vydělením řádku hodnotou pivota) nazveme normování pivota.

Převod matice z odstupňovaného tvaru na redukovaný odstupňovaný tvar probíhá podobně jako převod matice na odstupňovanou, ale nuly získáváme místo postupu "odshora dolů" směrem "odzdola nahoru" (tím je zaručeno, že si nezkazíme nuly pod diagonálou). Navíc normujeme pivoty.

Postupujeme tak, že poslední řádek matice (není-li nulový) vydělíme jeho pivotem – tak, aby na jeho místě vznikla jednička. Pak pomocí pivota dostaneme nuly nad ním – tak, že přičítáme příslušné násobky tohoto řádku k řádkům předchozím (tzn. těm, které jsou v matici na vyšších pozicích). Pak se přesuneme o řádek výš a proces opakujeme: vydělíme řádek pivotem a přičítáním příslušných násobků tohoto řádku k řádkům předchozím získáme nuly v celém sloupci nad pivotem atd. (ve sloupcích neobsahujících pivota nuly netvoříme). Při tomto postupu se zachovají nuly pod diagonálou – matice zůstane odstupňovaná.

Ilustrujeme tento postup na naznačených úpravách následující matice z odstupňovaného tvaru na redukovaný odstupňovaný tvar. Při znázornění platí to, co bylo řečeno v kapitole 5.4 – symbol $*$ značí nenulové číslo, symbol x značí jakékoliv číslo; prvky matice se během výpočtu mění, znázorňujeme pouze nulovost resp. nenulovost prvků:

$$\begin{pmatrix} * & x & x & x & x & x \\ 0 & 0 & * & x & x & x \\ 0 & 0 & 0 & * & x & x \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix} \sim \begin{pmatrix} * & x & x & x & x & x \\ 0 & 0 & * & x & x & x \\ 0 & 0 & 0 & * & x & x \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} * & x & x & x & x & 0 \\ 0 & 0 & * & x & x & 0 \\ 0 & 0 & 0 & * & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} * & x & x & x & x & 0 \\ 0 & 0 & * & x & x & 0 \\ 0 & 0 & 0 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim$$

$$\sim \begin{pmatrix} * & x & x & 0 & x & 0 \\ 0 & 0 & * & 0 & x & 0 \\ 0 & 0 & 0 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} * & x & x & 0 & x & 0 \\ 0 & 0 & 1 & 0 & x & 0 \\ 0 & 0 & 0 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} * & x & 0 & 0 & x & 0 \\ 0 & 0 & 1 & 0 & x & 0 \\ 0 & 0 & 0 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & x & 0 & 0 & x & 0 \\ 0 & 0 & 1 & 0 & x & 0 \\ 0 & 0 & 0 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Matice je v redukovaném odstupňovaném tvaru.

Poznámka (neměnit pořadí)

Při úpravách "zdola nahoru" už nemůžeme měnit pořadí řádků, neboť by se porušil odstupňovaný tvar matice.

Poznámka (jednoznačnost redukovaného odstupňovaného tvaru)

Zatímco odstupňovaný tvar matice není jednoznačný (různými úpravami téže matice lze dospět k různým odstupňovaným tvarům), redukovaný odstupňovaný tvar matice jednoznačný je.

Poznámka (jednotková matice jako redukovaný odstupňovaný tvar)

Upravujeme-li na redukovaný odstupňovaný tvar čtvercovou matici řádu n, která má hodnost n, je výsledkem jednotková matice.

Poznámka (volitelné neznámé)

Vyskytují-li se v matici soustavy, která je v redukovaném odstupňovaném tvaru, sloupce neobsahující pivota, jsou to sloupce odpovídající volitelným neznámým (pokud soustava má řešení) – viz Pravidlo 7.5.

Obsahují-li v matici soustavy, která je v redukovaném odstupňovaném tvaru, všechny sloupce pivota, pak soustava má právě jedno řešení (pokud má řešení). Tato matice je pak jednotková (případně navíc s nulovými řádky pod jednotkovou maticí).

Příklad 7.9

Vyřešíme Jordanovou metodou soustavu z Příkladu 7.5:

$$\begin{array}{rcrcrcr} - & x_0 & + & 2x_1 & - & 3x_2 & = & 7 \\ & 2x_0 & - & x_1 & + & 8x_2 & = & -14 \\ - & 3x_0 & - & 3x_1 & - & 16x_2 & = & 23. \end{array}$$

Počáteční postup je stejný jako u Gaussovy metody, z odstupňovaného tvaru matice soustavy určíme hodnosti matice soustavy a matice rozšířené. Jak víme z Příkladu 7.5, je v tomto případě $h = h' = n$ – soustava má právě jedno řešení.

Dále pokračujeme výše popsaným postupem. Pivotem v posledním řádku je číslo -1. V prvním kroku normujeme tohoto pivota, tzn. vydělíme (nebo vynásobíme – v tomto případě je to jedno) poslední řádek číslem -1. Pak přičtením -2-násobku posledního řádku k předposlednímu řádku a přičtením 3-násobku posledního řádku k nultému řádku (raději připomínáme, že v této knize číslujeme řádky od nuly) získáme nuly ve sloupci nad pivotem. Obdobně postupujeme dál – tak, jak je naznačeno vedle matic; pivotem v 1. řádku je číslo 3 – dělíme proto 1. řádek 3 (neboli násobíme 1/3):

$$\left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 2 & -1 & 8 & -14 \\ -3 & -3 & -16 & 23 \end{array}\right) \sim \cdots \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{array}\right)_{-1} \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 7 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 1 & -2 \end{array}\right)^{\begin{array}{l}\leftarrow\\ \leftarrow\\ -2 \quad 3\end{array}} \sim$$

$$\sim \left(\begin{array}{ccc|c} -1 & 2 & 0 & 1 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array}\right)_{1/3} \sim \left(\begin{array}{ccc|c} -1 & 2 & 0 & 1 \\ 0 & 1 & 0 & 4/3 \\ 0 & 0 & 1 & -2 \end{array}\right)^{\leftarrow}_{-2} \sim \left(\begin{array}{ccc|c} -1 & 0 & 0 & -5/3 \\ 0 & 1 & 0 & 4/3 \\ 0 & 0 & 1 & -2 \end{array}\right)_{-1} \sim$$

$$\sim \left(\begin{array}{ccc|c} 1 & 0 & 0 & 5/3 \\ 0 & 1 & 0 & 4/3 \\ 0 & 0 & 1 & -2 \end{array}\right).$$

Matice soustavy je v redukovaném odstupňovaném tvaru (spec. je jednotková). Je zřejmé, že nultý řádek představuje rovnici $x_0 = \frac{5}{3}$, první $x_1 = \frac{4}{3}$ a druhý $x_2 = -2$. Všimněme si – jsou to hodnoty za čarou v rozšířené matici soustavy.

Řešením dané soustavy je vektor $\bar{x} = \left(\frac{5}{3}, \frac{4}{3}, -2\right)$.

Poznámka (normování pivotů)

Poznamenejme, že při "ručním" počítání je vhodnější (abychom se vyhnuli počítání se zlomky) během výpočtu nenormovat pivoty, ale násobit vhodnými čísly jak řádek s pivotem, tak řádek, ke kterému řádek s pivotem přičítáme – tak, jak jsme činili při převodu matice na odstupňovaný tvar. Pivoty ve všech řádcích se znormují až nakonec.

Např. v Příkladu 7.9 bychom v matici

$$\left(\begin{array}{ccc|c} -1 & 2 & 0 & 1 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & -1 & 2 \end{array}\right)$$

nedělili prostřední řádek třemi, ale postupovali bychom tak, jak je naznačeno vedle matic:

$$\left(\begin{array}{ccc|c} -1 & 2 & 0 & 1 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & -1 & 2 \end{array}\right)^{3\leftarrow}_{-2} \sim \left(\begin{array}{ccc|c} -3 & 0 & 0 & -5 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & -1 & 2 \end{array}\right).$$

Pak už stačí znormovat pivoty vydělením prvního řádku číslem -3 a druhého číslem 3:

$$\sim \left(\begin{array}{ccc|c} 1 & 0 & 0 & 5/3 \\ 0 & 1 & 0 & 4/3 \\ 0 & 0 & 1 & -2 \end{array}\right).$$

P ř í k l a d 7.10

Nalezneme Jordanovou metodou partikulární řešení soustavy z Příkladu 7.7:

$$\begin{array}{rcrcrcl} -x_0 & + & x_1 & + & x_2 & + & x_3 = 2 \\ & & & & x_2 & + & x_3 = 3. \end{array}$$

Matice soustavy

$$\left(\begin{array}{cccc|c} -1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right),$$

je již odstupňovaná, je $h = h' = 2$, $n = 4$. Soustava má nekonečně mnoho řešení, přičemž dvě neznámé jsou volitelné, neboť je $n - h = 2$.

Soustavu budeme řešit Jordanovou metodou – matici soustavy převedeme na redukovaný odstupňovaný tvar. V posledním řádku je pivotem číslo 1, stačí proto vynulovat sloupec (který je tvořen pouze prvkem na pozici $(0, 2)$) nad tímto pivotem (shodou okolností se vynuluje i prvek na pozici $(0, 3)$, o což nijak neusilujeme). Pak znormujeme pivota v 0. řádku vydělením (resp. vynásobením) číslem -1:

$$\left(\begin{array}{cccc|c} -1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right)\begin{array}{l}\\ \leftarrow -1\end{array} \sim \left(\begin{array}{cccc|c} -1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right)\begin{array}{l}\\ \cdot(-1)\end{array} \sim \left(\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 3 \end{array}\right).$$

Podle Pravidla 7.5 víme, že můžeme volit hodnoty např. za x_3 a x_1. Stejně jako v Příkladu 7.7 dosadíme za volitelné neznámé nuly: $x_3 = 0$, $x_1 = 0$. Nultý řádek matice představuje rovnici $x_0 - x_1 = 1$. Dosazením hodnoty $x_1 = 0$ do této rovnice získáme $x_0 = 1$. První řádek matice představuje rovnici $x_2 + x_3 = 3$. Dosazením hodnoty $x_3 = 0$ do této rovnice získáme $x_2 = 1$. Řešením (partikulárním) soustavy je vektor

$$\bar{x} = (1, 0, 3, 0).$$

Poznámka (více soustav se stejnou maticí)

Někdy je potřeba řešit více soustav se stejnou maticí soustavy – lišících se jen pravými stranami. V takovém případě bychom při řešení každé z těchto soustav opakovaně upravovali tutéž matici soustavy na odstupňovanou. To lze obejít tak, že místo jednoho sloupce pravých stran napíšeme za svislou čáru vedle sebe sloupce pravých stran všech soustav, a při převodu matice soustavy na odstupňovanou provádíme úpravy na řádcích takto vzniklé "dvojmatice". Předpokládejme, že matice soustavy je čtvercová řádu n a má hodnost $h = n$, pak bude mít každá ze soustav právě jedno řešení (v jiných případech by byla situace složitější – pro některou z pravých stran by soustava nemusela mít řešení, pro jinou pravou stranu by mohla mít řešení nekonečně mnoho). Použijeme-li Jordanovu metodu, pak řešení jednotlivých soustav získáme z výsledné "dvojmatice" jako jednotlivé sloupce matice vpravo od svislé čáry (vlevo bude jednotková matice) – první sloupec bude odpovídat řešení první soustavy atd.

P ř í k l a d 7.11

Vyřešíme Jordanovou metodou současně následující soustavy (lišící se jen pravými stranami).

$$\begin{array}{rcrcrcl} 2x_0 & - & x_1 & - & 3x_2 & = & 1 \\ 3x_0 & + & 3x_1 & + & 4x_2 & = & -2 \\ 4x_0 & - & 5x_1 & - & 11x_2 & = & 3, \end{array}$$

$$\begin{array}{rcrcrcl} 2x_0 & - & x_1 & - & 3x_2 & = & -11 \\ 3x_0 & + & 3x_1 & + & 4x_2 & = & 26 \\ 4x_0 & - & 5x_1 & - & 11x_2 & = & -47, \end{array}$$

$$\begin{array}{rcrcrcl} 2x_0 & - & x_1 & - & 3x_2 & = & 5 \\ 3x_0 & + & 3x_1 & + & 4x_2 & = & 7 \\ 4x_0 & - & 5x_1 & - & 11x_2 & = & 11. \end{array}$$

Pravé strany všech soustav zapíšeme jako sloupce za svislou čáru a upravujeme řádky celé "dvojmatice" – tak, jak je vedle matic naznačeno:

$$\left(\begin{array}{ccc|ccc} 2 & -1 & -3 & 1 & -11 & 5 \\ 3 & 3 & 4 & -2 & 26 & 7 \\ 4 & -5 & -11 & 3 & -47 & 11 \end{array}\right)\begin{array}{l}\\ \cdot 2 \leftarrow \\ \\ \end{array}\begin{array}{c}\cdot(-3)\\ \\ \leftarrow -2\end{array} \sim \left(\begin{array}{ccc|ccc} 2 & -1 & -3 & 1 & -11 & 5 \\ 0 & 9 & 17 & -7 & 85 & -1 \\ 0 & -3 & -5 & 1 & -25 & 1 \end{array}\right)\begin{array}{l}\\ \\ \cdot 3 \leftarrow\end{array} \sim$$

$$\sim \left(\begin{array}{ccc|ccc} 2 & -1 & -3 & 1 & -11 & 5 \\ 0 & 9 & 17 & -7 & 85 & -1 \\ 0 & 0 & 2 & -4 & 10 & 2 \end{array}\right)\begin{array}{l}\\ \\ \cdot 1/2\end{array} \sim \left(\begin{array}{ccc|ccc} 2 & -1 & -3 & 1 & -11 & 5 \\ 0 & 9 & 17 & -7 & 85 & -1 \\ 0 & 0 & 1 & -2 & 5 & 1 \end{array}\right)\begin{array}{l}\\ \\ \leftarrow -17\end{array}\begin{array}{c}\\ \\ \leftarrow 3\end{array} \sim$$

$$\sim \left(\begin{array}{ccc|ccc} 2 & -1 & 0 & -5 & 4 & 8 \\ 0 & 9 & 0 & 27 & 0 & -18 \\ 0 & 0 & 1 & -2 & 5 & 1 \end{array}\right)_{1/9} \sim \left(\begin{array}{ccc|ccc} 2 & -1 & 0 & -5 & 4 & 8 \\ 0 & 1 & 0 & 3 & 0 & -2 \\ 0 & 0 & 1 & -2 & 5 & 1 \end{array}\right) \sim$$

$$\sim \left(\begin{array}{ccc|ccc} 2 & 0 & 0 & -2 & 4 & 6 \\ 0 & 1 & 0 & 3 & 0 & -2 \\ 0 & 0 & 1 & -2 & 5 & 1 \end{array}\right)_{1/2} \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & 2 & 3 \\ 0 & 1 & 0 & 3 & 0 & -2 \\ 0 & 0 & 1 & -2 & 5 & 1 \end{array}\right).$$

Řešením první soustavy je vektor $(-1, 3, -2)$ (1. sloupec za svislou čarou), řešením druhé soustavy je vektor $(2, 0, 5)$ (2. sloupec za svislou čarou), řešením třetí soustavy je vektor $(3, -2, 1)$ (3. sloupec za svislou čarou).

7.5 Homogenní soustava

Homogenní soustava

$$\begin{array}{rcrcrcrcl} a_{00}x_0 & + & a_{01}x_1 & + & \ldots & + & a_{0,n-1}x_{n-1} & = & 0 \\ a_{10}x_0 & + & a_{11}x_1 & + & \ldots & + & a_{1,n-1}x_{n-1} & = & 0 \\ & & & & & & & \vdots & \\ a_{m-1,0}x_0 & + & a_{m-1,1}x_1 & + & \ldots & + & a_{m-1,n-1}x_{n-1} & = & 0 \end{array}$$

je speciálním případem soustavy lineárních rovnic – platí tedy pro ni to, co bylo řečeno pro soustavy rovnic obecně (včetně uvedených eliminačních metod řešení). Má však navíc některé speciální vlastnosti. Je například zřejmé, že dosadíme-li do homogenní soustavy za všechny neznámé nuly, budou rovnice splněny. Homogenní soustava tedy má vždy za řešení nulový vektor.

Pojem 7.10 (triviální a netriviální řešení)

Řešení soustavy, kde jsou všechny neznámé rovny nule (tedy nulový vektor), se nazývá triviální řešení. Každé jiné řešení (tzn. takové, že aspoň jedna neznámá je různá od nuly) se nazývá netriviální.

Každá homogenní soustava má tedy přinejmenším triviální řešení (některé homogenní soustavy mají triviální i netriviální řešení), neboli nulový vektor je vždy řešením homogenní soustavy (zatímco nehomogenní nikdy ne – dosazením nul za všechny neznámé nemůže na pravé straně vyjít nenulové číslo). To znamená, že homogenní soustava má vždy řešení.

Pravidlo 7.6 *(řešitelnost homogenní soustavy)*

Homogenní soustava je vždy řešitelná; řešením (resp. jedním z řešení) je vždy nulový vektor.

Pro počet řešení platí totéž, co pro nehomogenní soustavu. Navíc z předchozího vyplývá, že má-li homogenní soustava jediné řešení, je tímto řešením nulový vektor (triviální řešení).

Pravidlo 7.7 *(počet řešení homogenní soustavy, triviální a netriviální řešení)*

Pro homogenní soustavu o n neznámých platí:

Je-li $h = n$, soustava má právě jedno řešení – pouze triviální,

je-li $h < n$, soustava má nekonečně mnoho řešení – má tedy i netriviální řešení,

$n - h =$ *počet volitelných neznámých.*

Poznámka (triviální a netriviální řešení)

Zjistíme-li tedy, že homogenní soustava má právě jedno řešení, nemusíme soustavu dál řešit – řešením je nulový vektor. Neboli – víme-li, že hodnost matice typu $n \times n$ je n, má homogenní soustava s touto maticí pouze triviální řešení.

Homogenní soustava má netriviální řešení, právě když má nekonečně mnoho řešení.

Poznámka (neměnné pravé strany)

Při úpravách rozšířené matice homogenní soustavy na odstupňovanou se pravé strany (tj. nuly) nemění, neboť na pravých stranách pouze přičítáme násobky nul k jiným nulám (případně měníme pořadí).

Příklad 7.12

Vyřešíme soustavu

$$\begin{array}{rcrcrcl} - & x_0 & + & 2x_1 & - & 3x_2 & = 0 \\ & 2x_0 & - & x_1 & + & 8x_2 & = 0 \\ - & 3x_0 & - & 3x_1 & - & 16x_2 & = 0. \end{array}$$

Soustavu přepíšeme do matice a povolenými úpravami na řádcích rozšířené matice soustavy převedeme matici soustavy na odstupňovanou:

$$\left(\begin{array}{ccc|c} -1 & 2 & -3 & 0 \\ 2 & -1 & 8 & 0 \\ -3 & -3 & -16 & 0 \end{array}\right) \begin{array}{l} \\ \leftarrow 2 \\ \leftarrow -3 \end{array} \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 0 \\ 0 & 3 & 2 & 0 \\ 0 & -9 & -7 & 0 \end{array}\right) \begin{array}{l} \\ \\ \leftarrow 3 \end{array} \sim \left(\begin{array}{ccc|c} -1 & 2 & -3 & 0 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & -1 & 0 \end{array}\right).$$

Protože je $h = 3 = n$ ($h = h'$ samozřejmě), má soustava právě jedno řešení. Má-li homogenní soustava jediné řešení, je to jistě řešení triviální. Nemusíme proto soustavu dál řešit a rovnou víme, že $\bar{x} = (0, 0, 0)$ (řešení bychom ovšem snadno získali Gaussovou metodou).

Triviální řešení není pro nás příliš zajímavé – má ho každá homogenní soustava. Homogenní soustava má netriviální řešení pouze v případě, že má řešení nekonečně mnoho. Pokud bychom chtěli získat nějaké netriviální partikulární řešení, nemohli bychom za všechny volitelné neznámé dosadit nuly (tak, jak jsme činili u soustavy nehomogenní) – zbylé neznámé by totiž také vyšly nulové, a obdrželi bychom tak pouze triviální řešení. Budeme proto volit neznámé následovně: za jednu z volitelných neznámých (z praktických důvodů nejlépe za tu, která odpovídá co nejpravějšímu sloupci v odstupňované matici soustavy) zvolíme jedničku, za ostatní neznámé zvolíme nuly.

Příklad 7.13

Nalezneme netriviální partikulární řešení soustavy

$$\begin{array}{rcrcrcl} -x_0 & + & x_1 & + & x_2 & + & x_3 & = 0 \\ & & & & x_2 & + & x_3 & = 0 \end{array}$$

(v Příkladu 7.7 byla řešena příslušná nehomogenní soustava). Tato soustava je reprezentována maticí

$$\left(\begin{array}{cccc|c} -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right),$$

která již je odstupňovaná. Hodnost matice je $h = 2$, počet neznámých je $n = 4$, je tedy $h < n$ – soustava má nekonečně mnoho řešení, přičemž je $n - h = 2$ – dvě neznámé jsou volitelné.

Dál můžeme postupovat Gaussovou nebo Jordanovou metodou – vybereme si např. Gaussovu. Víme, že neznámé x_3 a x_1 jsou volitelné (viz Pravidlo 7.5) – zvolíme $x_3 = 1$ a $x_1 = 0$. Dosazením $x_3 = 1$ do 1. rovnice představované 1. řádkem rozšířené matice dostaneme:

$$x_2 + 1 = 0, \qquad \text{odtud} \qquad x_2 = -1.$$

Dosazením $x_1 = 0$, $x_2 = -1$, $x_3 = 1$ do 0. rovnice představované 0. řádkem rozšířené matice dostaneme:

$$-x_0 + 0 + (-1) + 1 = 0, \qquad \text{odtud} \qquad x_0 = 0.$$

Dostali jsme netriviální partikulární řešení

$$\bar{x} = (0, 0, -1, 1).$$

Poznámka (obecné řešení homogenní soustavy)

Pokud bychom chtěli získat obecné řešení homogenní soustavy, postupovali bychom stejně jako v případě soustavy nehomogenní. Dosazením nul za všechny parametry v obecném řešení bychom dostali triviální řešení, jakákoliv jiná volba dosazených hodnot by vedla k řešení netriviálnímu.

7.6 Úlohy k procvičení

$\boxed{A}$ **Teoretická část**

Posuďte pravdivost následujících tvrzení (soustava = soustava lineárních rovnic, h = hodnost matice soustavy, h' = hodnost rozšířené matice soustavy, n = počet neznámých):

a) Je-li počet rovnic větší než počet neznámých, nemá soustava řešení.

b) Soustava má právě jedno řešení, právě když počet rovnic je roven počtu neznámých.

c) Soustava má řešení, právě když počet rovnic je menší nebo roven počtu neznámých.

d) Řešitelnost soustavy lze posoudit na základě znalosti počtu rovnic a počtu neznámých.

e) Soustava má nekonečně mnoho řešení, právě když $h < h'$.

f) Soustava má řešení, právě když $h = h'$.

g) Pro libovolnou soustavu platí: $h \leq h'$.

h) Existuje soustava, pro kterou je $h > h'$.

i) Soustava nemá řešení, právě když $h < h'$.

j) Existuje soustava, pro kterou je $h = 2$, $h' = 4$.

k) Soustava má nekonečně mnoho řešení, právě když $h = h' < n$.

l) Má-li soustava nekonečně mnoho řešení, pak počet volitelných neznámých je dán rozdílem: počet neznámých – počet rovnic.

m) Řešitelná nehomogenní soustava má vždy netriviální řešení.

n) Každá homogenní soustava má právě jedno řešení.

o) Každá homogenní soustava má řešení.

p) Má-li homogenní soustava právě jedno řešení, je tímto řešením nulový vektor.

q) Ekvivalentní soustavy jsou takové, které mají aspoň jedno společné řešení.

r) V rozšířené matici soustavy lze provádět jakékoliv řádkové či sloupcové úpravy neměnící hodnost matice soustavy; množina všech řešení soustavy odpovídající upravené matici je stejná jako množina všech řešení původní soustavy.

Výsledky $\boxed{A}$

$\boxed{A}$ P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N (soustava může mít řešení)

b) N (počet rovnic není směrodatný – v soustavě může být např. jedna rovnice násobkem jiné)

c) N (řešitelnost soustavy na počtu rovnic nezávisí)

d) N (počet rovnic nemusí odpovídat hodnosti matice)

e) N (v případě $h < h'$ soustava nemá řešení)

f) P (Frobeniova věta)

g) P

h) N

i) P

j) N (je-li $h < h'$, je určitě $h' = h + 1$)

k) P

l) N (počet neznámých je dán rozdílem $n - h$; hodnost matice nemusí odpovídat počtu rovnic)

m) P (nehomogenní soustava nemůže mít triviální řešení – po dosazení nul za všechny neznámé by výsledky ve všech rovnicích (tj. pravé strany rovnic) byly rovny nule)

n) N (může mít i nekonečně mnoho)

o) P

p) P

q) N (ekvivalentní soustavy mají stejnou celou množinu řešení)

r) N (pouze řádkové úpravy a záměnu pořadí sloupců v matici soustavy (s odpovídající záměnou pořadí neznámých))

B Praktická část

1 Řešte následující soustavy. V případě, že soustava má nekonečně mnoho řešení, nalezněte jedno partikulární řešení dosazením nul za volitelné neznámé. Zvídaví čtenáři mohou nalézt i obecné řešení.

a) $\begin{aligned} -x_0 + x_1 + x_2 &= -2 \\ 3x_0 - 3x_1 + 2x_2 &= 16 \\ 6x_0 - 5x_1 + 2x_2 &= 27, \end{aligned}$

b) $\begin{aligned} 2x_0 + 2x_2 &= 2 \\ -x_0 - x_1 &= -1 \\ 2x_0 - x_1 + 3x_2 &= 1, \end{aligned}$

c) $\begin{aligned} x_0 - x_1 &= 5 \\ -x_0 - 5x_1 &= 1 \\ x_0 + 2x_1 &= 2, \end{aligned}$

d) $\begin{aligned} x_0 + 3x_1 - x_2 &= 5 \\ 7x_0 + x_1 + 3x_2 &= 10 \\ x_0 - x_1 + x_2 &= 0, \end{aligned}$

e) $\begin{aligned} x_0 - x_1 + x_2 &= 3 \\ 2x_0 - 2x_1 - x_2 + 3x_3 &= 3 \\ 3x_0 - 3x_1 - 2x_2 + 5x_3 &= 4, \end{aligned}$

f) $\begin{aligned} -2x_0 + x_1 - x_2 &= -1 \\ x_0 + x_2 &= 1 \\ x_0 - x_1 &= 0, \end{aligned}$

g) $\begin{aligned} x_0 + x_1 + 3x_2 - x_3 &= 2 \\ -x_0 + x_1 + x_2 + x_3 + 2x_4 &= 4 \\ 3x_0 + 2x_2 + x_3 &= 0, \end{aligned}$

h) $\begin{aligned} -x_0 + 2x_1 - 2x_2 &= 0 \\ -x_0 - x_2 &= 0 \\ 3x_0 + 3x_1 &= 1, \end{aligned}$

i) $\begin{aligned} 2x_0 + x_1 - 3x_3 - x_4 &= 2 \\ x_0 + x_2 &= 1 \end{aligned}$

2 Řešte soustavy reprezentované následujícími maticemi. V případě, že soustava má nekonečně mnoho řešení, nalezněte jedno partikulární řešení dosazením nul za volitelné neznámé. Zvídaví čtenáři mohou nalézt i obecné řešení.

a) $\left(\begin{array}{cccc|c} 1 & 1 & 2 & 1 & 1 \\ -1 & 1 & -1 & 1 & 0 \\ 1 & 0 & 1 & 0 & -1 \\ 1 & 1 & 0 & 1 & 2 \end{array}\right)$ b) $\left(\begin{array}{cccc|c} 1 & 2 & 3 & 4 & 1 \\ 2 & 1 & 4 & 3 & 2 \\ 3 & 4 & 1 & 2 & 3 \\ 4 & 3 & 2 & 1 & 4 \end{array}\right)$ c) $\left(\begin{array}{ccc|c} 2 & 3 & 1 & 5 \\ 2 & 0 & -1 & -2 \\ 1 & -1 & -2 & -5 \end{array}\right)$

d) $\left(\begin{array}{cc|c} -5 & -1 & 1 \\ -1 & 1 & 5 \\ 2 & 1 & 2 \end{array}\right)$ e) $\left(\begin{array}{cccc|c} 1 & -3 & 4 & -8 & 2 \\ 2 & -6 & 1 & -2 & -3 \\ 3 & -9 & 5 & -10 & -1 \end{array}\right)$ f) $\left(\begin{array}{ccc|c} -1 & 1 & 2 & 3 \\ 2 & 1 & -1 & -3 \\ 2 & 0 & -2 & 1 \end{array}\right)$

g) $\left(\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 4 & 1 & -3 & -9 \\ 4 & 3 & -3 & -5 \end{array}\right)$ h) $\left(\begin{array}{cc|c} 2 & 5 & -1 \\ 4 & 1 & 7 \\ 4 & 6 & 2 \end{array}\right)$ i) $\left(\begin{array}{ccc|c} 1 & 0 & 1 & 2 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{array}\right)$

3 Doplňte k rovnici $x_1 + 2x_2 = 3$ ještě jednu rovnici tak, aby vzniklá soustava

a) neměla řešení,

b) měla právě jedno řešení,

c) měla nekonečně mnoho řešení.

4 Řešte následující soustavy. V případě, že soustava má nekonečně mnoho řešení, nalezněte jedno partikulární řešení dosazením jedničky za jednu a nul za zbylé volitelné neznámé. Zvídaví čtenáři mohou nalézt i obecné řešení.

a) $\begin{aligned} -x_0 + 2x_1 + x_2 &= 0 \\ x_0 + 2x_1 + 3x_2 &= 0 \\ 3x_0 + x_1 + 4x_2 &= 0, \end{aligned}$

b) $\begin{aligned} 5x_0 + 3x_1 - 5x_3 &= 0 \\ 3x_0 - x_1 + 2x_2 + x_3 &= 0 \\ 2x_0 - 3x_1 + 3x_2 + 5x_3 &= 0, \end{aligned}$

c) $\begin{aligned} x_0 + x_1 + 3x_2 + x_3 &= 0 \\ 3x_0 + 2x_1 + 8x_2 + 4x_3 &= 0 \\ 3x_0 - 2x_1 + 4x_2 + 8x_3 &= 0, \end{aligned}$

d) $\begin{aligned} x_0 + 2x_1 - 3x_2 &= 0 \\ 3x_0 + 2x_1 - x_2 &= 0, \end{aligned}$

e) $\begin{aligned} 2x_0 - 6x_1 + 4x_2 &= 0 \\ -3x_0 + 9x_1 - 6x_2 &= 0, \end{aligned}$

f) $\begin{aligned} 2x_0 + x_1 + x_2 &= 0 \\ 3x_0 + 4x_1 + x_2 &= 0 \\ -x_0 + 2x_1 - 3x_2 &= 0, \end{aligned}$

g) $2x_0 + 6x_1 - 2x_2 + 4x_3 + 8x_4 = 0$

Výsledky $\boxed{B}$

$\boxed{1}$ a) $\bar{x} = (3, -1, 2)$, b) nemá řešení, c) $\bar{x} = (4, -1)$,

d) partikulární řešení: např. $\bar{x} = (\frac{5}{4}, \frac{5}{4}, 0)$, obecné řešení: $\bar{x} = (\frac{5}{4} - \frac{1}{2}t, \frac{5}{4} + \frac{1}{2}t, t)$

e) partikulární řešení: např. $\bar{x} = (2, 0, 1, 0)$, obecné řešení: $\bar{x} = (2 + s - t, s, 1 + t, t)$,

f) partikulární řešení: např. $\bar{x} = (1, 1, 0)$, obecné řešení: $\bar{x} = (1 - t, 1 - t, t)$,

g) partikulární řešení: např. $\bar{x} = (2, 9, -3, 0, 0)$,
obecné řešení: $\bar{x} = (2 - 3s - 2t, 9 - 8s - 7t, -3 + 4s + 3t, s, t)$,

h) $\bar{x} = (\frac{2}{3}, -\frac{1}{3}, -\frac{2}{3})$,

i) partikulární řešení: např. $\bar{x} = (1, 2, 0, 0, 0)$, obecné řešení: $\bar{x} = (1 - u, 2u + 3s + t, u, s, t)$

$\boxed{2}$ a) nemá řešení, b) partikulární řešení: např. $\bar{x} = (1, 0, 0, 0)$, obecné řešení: $\bar{x} = (1 + t, -t, -t, t)$,

c) $\bar{x} = (0, 1, 2)$, d) $\bar{x} = (-1, 4)$,

e) partikulární řešení: např. $\bar{x} = (-2, 0, 1, 0)$, obecné řešení: $\bar{x} = (-2 + 3s, s, 1 + 2t, t)$,

f) nemá řešení, g) $\bar{x} = (1, 2, 5)$, h) $\bar{x} = (2, -1)$, i) $\bar{x} = (1, -1, 1)$

$\boxed{3}$ a) např. $2x_1 + 4x_2 = 5$, b) např. $2x_1 + 3x_2 = 4$, c) např. $2x_1 + 4x_2 = 6$

$\boxed{4}$ a) partikulární řešení: např. $\bar{x} = (-1, -1, 1)$, obecné řešení: $\bar{x} = (-t, -t, t)$,

b) partikulární řešení: např. $\bar{x} = (-\frac{3}{7}, \frac{5}{7}, 1, 0)$, obecné řešení: $\bar{x} = (-\frac{3}{7}t, \frac{5}{7}t, t, 0)$,

c) partikulární řešení: např. $\bar{x} = (-2, 1, 0, 1)$, obecné řešení: $\bar{x} = (-2s - 2t, -s + t, s, t)$,

d) partikulární řešení: např. $\bar{x} = (-1, 2, 1)$, obecné řešení: $\bar{x} = (-t, 2t, t)$,

e) partikulární řešení: např. $\bar{x} = (-2, 0, 1)$, obecné řešení: $\bar{x} = (3s - 2t, s, t)$,

f) $\bar{x} = (0, 0, 0)$,

g) partikulární řešení: např. $\bar{x} = (-4, 0, 0, 0, 1)$, obecné řešení: $\bar{x} = (-3v + u - 2s - 4t, v, u, s, t)$

8. SOUSTAVY ROVNIC Z POHLEDU INFORMATIKA

Homogenní soustavu lineárních rovnic (viz Pojem 7.2) lze zapsat pomocí matice soustavy (viz Pojem 7.3). Nehomogenní soustavu lineárních rovnic (viz Pojem 7.2) lze zapsat pomocí rozšířené matice soustavy (viz Pojem 7.4). Rozšířená matice soustavy vznikne z matice soustavy přidáním sloupce pravých stran. Matici soustavy budeme pro potřeby této kapitoly nazývat nerozšířenou maticí soustavy, abychom ji přehledně odlišili od rozšířené matice soustavy.

Zdrojové kódy funkcí z kapitol 4 a 6 nenabízí možnost, jak v rozšířené matici soustavy rozlišovat sloupec pravých stran a matici soustavy. Zdrojové kódy by šly upravit, aby toto rozlišování možné bylo, ale pro funkce by to nemělo žádný přínos. V příslušných funkcích by byla rozšířená matice soustavy nahrazena dvojicí tvořenou nerozšířenou maticí soustavy a vektorem pravých stran. Práce s posledním sloupcem matice, která se nyní vykonává v rámci cyklu, by musela být nahrazena samostatným příkazem, který by následoval po skončení tohoto cyklu, a pracoval by se sloupcem pravých stran. Upravené zdrojové kódy funkcí by byly delší a méně srozumitelné.

Z pohledu informatika není nutné mít ve zdrojovém kódu pro sloupec pravých stran zavedené zvláštní označení. K sloupci pravých stran přistupujeme jako k posledními sloupci rozšířené matice soustavy.

8.1 Řešitelnost soustavy

Ve Zdrojovém kódu 8.1 je implementována funkce, která určuje, zda nehomogenní soustava lineárních rovnic reprezentovaná rozšířenou maticí soustavy `A` je či není řešitelná. Vstupem funkce je matice `A`, která musí být odstupňovaná. Soustavu s odstupňovanou maticí zde budeme nazývat odstupňovaná soustava. Využíváme Frobeniovu větu (viz Pravidlo 7.1). Nejprve zjistíme hodnost rozšířené matice soustavy (řádek `03`) a uložíme ji do proměnné `h2`. Následně zjistíme hodnost nerozšířené matice soustavy (řádek `04`) a uložíme ji do proměnné `h1`. Výsledkem funkce je hodnota získaná porovnáním hodností `h2` a `h1`. Pokud se hodnosti rovnají, soustava je řešitelná. Pokud se hodnosti nerovnají, soustava není řešitelná.

Zdrojový kód 8.1 (řešitelnost odstupňované nehomogenní soustavy)

```
00 int odstupnovana_soustava_je_resitelna(double ** A, int m, int n)
01 {
02   int h1, h2;
03   h2 = hodnost_odstupnovane_matice(A,m,n);
04   h1 = hodnost_odstupnovane_matice(A,m,n-1);
05   return h1 == h2;
06 }
```

Poznámka (převod rozšířené matice soustavy na matici soustavy)

Nerozšířenou matici soustavy získáme z rozšířené matice vynecháním posledního sloupce. Funkci zjišťující hodnost matice (řádek `04`) předáváme jako její parametr matici `A`. Této funkci tvrdíme, že matice `A` má o jeden sloupec méně, než kolik má sloupců ve skutečnosti (třetí parametr funkce je `n-1`). Funkce zjišťující hodnost matice nebude o tomto posledním sloupci vědět a vypočítá hodnost nerozšířené matice soustavy, přestože `A` je rozšířenou maticí soustavy.

Poznámka (počet řešení soustavy)

Pokud je hodnost rozšířené matice soustavy rovna hodnosti nerozšířené matice soustavy a tato hodnost je menší než počet sloupců nerozšířené matice soustavy (viz Pravidlo 7.3), má nehomogenní soustava lineárních rovnic reprezentovaná touto rozšířenou maticí soustavy nekonečně mnoho řešení. Zatímco v matematice bývá snaha popsat obecné řešení (viz Pojem 7.5), v informatice nám obvykle postačuje nalezení jednoho partikulárního řešení (viz Pojem 7.5).

8.2 Gaussova eliminační metoda

Gaussova eliminační metoda (viz kapitola 7.3) je jedním ze způsobů řešení soustavy lineárních rovnic. Tato metoda převede matici na odstupňovaný tvar a následně směrem odspoda nahoru prochází jednotlivé řádky matice. V každém řádku jsou dosazeny hodnoty neznámých vypočtených z předchozích řádků matice. Na daném řádku zůstává jediná neznámá, které bude v tomto kroku vypočtena.

Zdrojový kód 8.2 (Gaussova eliminační metoda)

```
00 double * Gaussova_eliminacni_metoda(double ** A, int m, int n)
01 {
02   int i, j, hodnost, pozice;
03   double * vysledek;
04   odstupnuj_matici(A,m,n);
05   if (!odstupnovana_soustava_je_resitelna(A,m,n)) return 0;
06   vysledek = nulovy_vektor(n-1);
07   hodnost = hodnost_odstupnovane_matice(A,m,n);
08   for (i = hodnost-1; i >= 0; i--)
09   {
10     pozice = pocet_nul_z_leva(A,m,n,i);
11     for (j = pozice+1; j < n; j++)
12       vysledek[pozice] += A[i][j]*vysledek[j];
13     vysledek[pozice] = (A[i][n-1] - vysledek[pozice]) / A[i][pozice];
14   }
15   return vysledek;
16 }
```

Ve Zdrojovém kódu 8.2 je implementována funkce `Gaussova_eliminacni_metoda`, která na zadanou rozšířenou matici soustavy `A` reprezentující nehomogenní soustavu lineárních rovnic provede Gaussovu eliminační metodu. Funkce vrací jako návratovou hodnotu vektor, který je řešením této soustavy. Pokud má soustava nekonečně mnoho řešení, výsledkem je jedno partikulární řešení. Pokud soustava nemá řešení, výsledkem je nulový ukazatel `0`.

Nejprve musíme matici odstupňovat (řádek `04`). Této problematice byla věnována celá kapitola 6.2. Následně zjistíme, zda je soustava řešitelná (řádek `05`). Pokud soustava není řešitelná, funkce končí a vrací nulový ukazatel `0`.

Vytvoříme si vektor `vysledek`, do kterého budeme postupně ukládat řešení soustavy. Vektor vytváříme jako nulový vektor (řádek `06`). Pokud soustava bude mít nekonečně mnoho řešení, volitelné neznámé budou mít hodnotu `0` – tato hodnota je ve výsledném vektoru přednastavena. U ostatních (nevolitelných) neznámých bude jejich hodnota uložena do vektoru `vysledek` v průběhu výpočtu. Hodnoty jednotlivých neznámých budeme vypočítávat odzadu (řádky `08` až `13`). Nejprve vypočítáme poslední z nevolitelných neznámých.

Na řádku `07` si do proměnné `hodnost` uložíme hodnost odstupňované matice `A`. Zjišťování hodnosti odstupňované matice probíhá už v rámci funkce určující řešitelnost soustavy. Nabízí se možná optimalizace zdrojového kódu. Místo volání funkce na řádku `07` bychom mezi řádky `06` a `08` vložili celé její tělo. Poté bychom hodnost odstupňované matice `A` nemuseli počítat dvakrát. Drobné zrychlení, které tato optimalizace přináší, nestojí za snížení čitelnosti zdrojového kódu, které by bylo touto optimalizací způsobeno.

V cyklu (řádek `08`) procházíme jednotlivé řádky matice (netradičně) směrem zdola nahoru. Cyklus začíná odzdola na prvním nenulovém řádku matice – řádku s pořadovým číslem `hodnost-1`. Případné nulové řádky se nacházejí pod tímto řádkem (vyplývá ze způsobu výpočtu hodnoty `hodnost`) a nejsou v rámci cyklu zpracovávány.

Nyní se podíváme na tělo cyklu (řádky `10` až `13`). Pro každý zpracovávaný řádek `i` uložíme do proměnné `pozice` počet nul zleva, které se nacházejí před prvním nenulovým prvkem tohoto řádku (řádek `10`). Proměnná `pozice` udává pro první nenulový prvek na zpracovávaném řádku `i` pořadové číslo sloupce, ve kterém se tento prvek nachází. Proměnná `pozice` určuje také index neznámé, jejíž hodnotu budeme v této iteraci cyklu počítat. Neznámé, které mají vyšší index a nebyly doposud vypočteny v předchozích iteracích cyklu, mají hodnotu `0` – jsou to volitelné neznámé.

Vnořený cyklus (řádek `11`) prochází sloupce matice následující po prvním nenulovém prvku na aktuálně zpracovávaném řádku (vnějším cyklem). Zároveň tento cyklus prochází i neznámé, jejichž hodnota je již určena. Z matematického hlediska budeme nyní dosazovat hodnoty již vypočtených neznámých do aktuálně řešené rovnice (tj. aktuálního řádku). Hodnota prvku v aktuálně zpracovávaném sloupci (a aktuálně zpracovávaném řádku) `A[i][j]` je vynásobena hodnotou proměnné `vysledek[j]`, jejíž index

odpovídá pořadovému číslu aktuálně zpracovávaného sloupce `j`. Výsledek tohoto součinu je přičten k hodnotě aktuálně vypočítávané neznámé `vysledek[pozice]` (řádek `12`). Aktuálně vypočítávaná neznámá měla před zahájením vnořeného cyklu hodnotu `0`, proto je její využití pro postupné přičítání hodnot korektní. Z hlediska přehlednosti zdrojového kódu by stálo za zvážení zavedení nové proměnné `tmp`, která by se na řádku `12` používala pouze k tomuto účelu a před začátkem vnořeného cyklu by byla nulována. Vnořený cyklus končí.

Na řádku `13` vypočítáme hodnotu neznámé s indexem pozice. Z matematického hlediska řešíme rovnici $ax + s = b$, kde `s` je součet součinů získaný na řádku `12`. Od posledního prvku matice na aktuálně zpracovávaném řádku `A[i][n-1]` (jedná se o prvek ze sloupce pravých stran) odečteme součet součinů získaný na řádku `12`. Výsledek rozdílu vydělíme hodnotou prvního nenulového prvku na aktuálním řádku `A[i][pozice]`. Získali jsme hodnotu aktuálně vypočítávané neznámé `vysledek[pozice]`.

Ve Zdrojovém kódu 8.3 je implementována funkce `vypocti_pravou_stranu`. Tato funkce do zadané nerozšířené matice soustavy `A` dosadí hodnoty neznámých zadaných ve formě vektoru `nezname`. Výsledkem funkce je vektor `prava_strana` reprezentující vypočtený sloupec pravých stran.

V cyklu (řádek `05`) procházíme jednotlivé řádky nerozšířené matice. Pro aktuálně zpracovávaný řádek `i` vypočteme skalární součin tohoto řádku a vektoru `nezname` (řádek `08`). Výsledek skalárního součinu je uložen do prvku vektoru `prava_strana` s indexem `i`. Pro potřeby výpočtu skalárního součinu musíme z matice `A` získat kopii příslušného řádku `i` (řádek `07`) a kopii tohoto řádku na konci cyklu smazat (řádek `09`).

Tuto funkci lze využít pro ověření správnosti výpočtu řešení nehomogenní soustavy lineárních rovnic, provedeného například pomocí funkce `Gaussova_eliminacni_metoda`. Funkci `vypocti_pravou_stranu` zavoláme se dvěma parametry. Prvním parametrem funkce je nerozšířená matice soustavy a druhým parametrem funkce je vypočtené řešení nehomogenní soustavy.

Zdrojový kód 8.3 (výpočet pravé strany)

```
00 double * vypocti_pravou_stranu(double ** A, int m, int n, double * nezname)
01 {
02   int i;
03   double *radek, * prava_strana;
04   prava_strana = vytvor_vektor(m);
05   for (i=0; i<m; i++)
06   {
07     radek = radek_matice(A,m,n,i);
08     prava_strana[i] = skalarni_soucin_vektoru(nezname, n, radek, n);
09     smaz_vektor(radek,n);
10   }
11   return prava_strana;
12 }
```

8.3 Jordanova eliminační metoda

Jordanova eliminační metoda (viz kapitola 7.4) je jedním ze způsobů řešení soustavy lineárních rovnic. Obdobně jako Gaussova eliminační metoda nejprve převede matici na odstupňovaný tvar. Na rozdíl od Gaussovy eliminační metody ale pokračuje v převodu až na redukovaný odstupňovaný tvar (viz Pojem 7.8). Vektor pravých stran takto redukované matice obsahuje hodnoty neznámých.

Ve Zdrojovém kódu 8.4 je implementována funkce `Jordanova_eliminacni_metoda`, která na zadanou rozšířenou matici soustavy `A` reprezentující nehomogenní soustavu lineárních rovnic provede Jordanovu eliminační metodu. Funkce vrací jako návratovou hodnotu vektor, který je řešením této soustavy. Pokud má soustava nekonečně mnoho řešení, výsledkem je jedno partikulární řešení. Pokud soustava nemá řešení, výsledkem je nulový ukazatel `0`.

Řádky `01` až `10` jsou shodné s odpovídajícími řádky u Gaussovy eliminační metody (viz Zdrojový kód 8.2). Bližší podrobnosti k těmto řádkům jsou popsány v kapitole 8.2. Nejprve provedeme odstupňování matice (řádek `04`), zjištění řešitelnosti soustavy (řádek `05`), inicializaci vektoru `vysledek` nulovými hodnotami (řádek `06`) a zjištění hodnosti matice (řádek `07`).

V cyklu (řádek 08) procházíme nenulové řádky matice směrem zdola nahoru. Proměnná `pozice` (řádek 10) udává pořadové číslo sloupce, který obsahuje první prvek s nenulovou hodnotu na tomto řádku. Tento prvek nazveme pivot. Proměnná `pozice` rovněž udává index neznámé, jejíž hodnotu budeme v této iteraci cyklu počítat.

Hodnoty nacházející se na aktuálním řádku za pivotem jsou buďto nulové nebo na jejich hodnotě nezáleží, protože odpovídají volitelným neznámým, jejichž hodnoty volíme nulové. Splnění tohoto tvrzení zajišťuje vnořený cyklus (řádek 13), který si vysvětlíme o odstavec později. Důsledkem tohoto tvrzení můžeme vypočítat hodnotu aktuální neznámé `vysledek[pozice]` vydělením hodnoty prvku nacházejícího se na aktuálním řádku ve sloupci pravých stran `A[i][n-1]` hodnotou pivota `A[i][pozice]` (řádek 11). Tuto nově vypočítanou hodnotu rovněž uložíme do prvku nacházejícího se na aktuálním řádku ve sloupci pravých stran `A[i][n-1]`. Pivota nastavíme na hodnotu `1` (řádek 12).

Vnořený cyklus (řádek 13) prochází matici od řádku `i` zpracovávaného ve vnějším cyklu směrem zdola nahoru. Cílem vnořeného cyklu je vynulovat sloupec nad pivotem pozice přičítáním příslušných násobků řádků s pivotem `i` k aktuálně zpracovávanému řádku `k`. Z řádku `i` obsahujícího pivota nás v této fázi výpočtu zajímá pouze hodnota pravé strany `A[i][n-1]`, pivot má hodnotu `1` a ostatní hodnoty jsou buďto nulové nebo na jejich hodnotě nezáleží, protože odpovídají volitelným neznámým. Prvek nacházející se na aktuálním řádku ve sloupci pravých stran `A[k][n-1]` je zmenšen o součin hodnoty nulovaného prvku `A[k][pozice]` a hodnoty pravé strany řádku s pivotem `A[i][n-1]` (řádek 15). Nakonec vynulujeme prvek na aktuálním řádku ve sloupci nad pivotem `[k][pozice]` (řádek 16).

Koncem vnořeného cyklu (řádek 17) končí i vnější cyklus (řádek 18). V každé iteraci vnějšího cyklu byl (za pomocí vnitřního cyklu) vynulován (směrem od pivota nahoru) jeden sloupec matice. Matice je nyní v redukovaně odstupňovaném tvaru. Hodnoty neznámých jsou uloženy ve vektoru `vysledek`, který je vrácen jako návratová hodnota funkce (řádek 22).

Řádky 19 až 21 provádějí pomocí dvou vnořených cyklů nulování prvků ve sloupcích odpovídajících volitelným neznámým, za které jsme zvolili hodnotu `0`. Na hodnotách těchto prvků nezáleží, provádíme pouze kosmetickou úpravu, která není nutná, a lze ji ze zdrojového kódu vynechat.

Zdrojový kód 8.4 (Jordanova eliminační metoda)

```
00 double * Jordanova_eliminacni_metoda(double ** A, int m, int n)
01 {
02   int i, j, k, hodnost, pozice;
03   double * vysledek;
04   odstupnuj_matici(A,m,n);
05   if (!odstupnovana_soustava_je_resitelna(A,m,n)) return 0;
06   vysledek = nulovy_vektor(n-1);
07   hodnost = hodnost_odstupnovane_matice(A,m,n);
08   for (i = hodnost-1; i >= 0; i--)
09   {
10     pozice = pocet_nul_z_leva(A,m,n,i);
11     A[i][n-1] = vysledek[pozice] = A[i][n-1] / A[i][pozice];
12     A[i][pozice] = 1;
13     for (k = i-1; k >= 0; k--)
14     {
15       A[k][n-1] -= A[k][pozice] * A[i][n-1];
16       A[k][pozice] = 0;
17     }
18   }
19   for (i = 0; i<m; i++)
20     for (j = 0; j<n-1; j++)
21       if (!vysledek[j]) A[i][j]=0;
22   return vysledek;
23 }
```

8.4 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 8.

2 Vytvořte náhodnou rozšířenou matici soustavy reprezentující nehomogenní soustavu lineárních rovnic. Udělejte kopii této matice. Soustavu vyřešte nejprve Gaussovou eliminační metodou (použijte původní matici). Následně soustavu vyřešte Jordanovou eliminační metodou (použijte kopii matice). Porovnejte dosažené výsledky. Proveďte zkoušku správnosti výpočtu.

3 Úkol 2 opakujte pro matice různých typů.

4 Upravte implementace eliminačních metod (Gaussova, Jordanova) tak, aby umožňovaly nastavit volitelným neznámým jinou hodnotu než `0`. Pro jednoduchost všem volitelným neznámým nastavte shodnou hodnotu.

5 Implementujte funkce (`Gauss_Homogenni`, `Jordan_Homogenni`), které vypočtou jedno z netriviálních řešení (viz Pojem 7.10) homogenní soustavy lineárních rovnic.

6 Funkce z úkolů 4 a 5 otestujte způsobem popsaným v úkolech 2 a 3.

9. INVERZNÍ MATICE. MATICOVÉ ROVNICE

9.1 Regulární a singulární matice. Inverzní matice

V kapitole 3 jsme zavedli operace s maticemi – sčítání a odčítání matic, násobení matic reálnými čísly a násobení matic. Dělení matic však definováno není. Porovnejme tuto situaci s dělením reálných čísel: pro $a, b \in \mathbb{R}$, $b \neq 0$, lze místo podílu $\frac{a}{b}$ psát součin ab^{-1} (nebo také $b^{-1}a$). Lze se tedy bez dělení obejít, budeme-li místo dělení číslem uvažovat násobení číslem k němu inverzním. Inverzní číslo k danému číslu je takové číslo, které v součinu s původním číslem dává jedničku; existuje ke každému nenulovému číslu. Inverzní číslo k číslu $a \neq 0$ je tedy číslo a^{-1}, pro které je $a \cdot a^{-1} = 1$.

Je možné podobným způsobem nahradit dělení i u matic? Lze k dané matici definovat inverzní matici způsobem analogickým k reálným číslům? Ukazuje se, že ano (roli jedničky bude u matic hrát jednotková matice), ale na rozdíl od reálných čísel, kde inverzní číslo existuje ke každému nenulovému číslu, k existenci inverzní matice nenulovost původní matice nestačí. Matice bude muset být tzv. regulární.

Pojem 9.1 (regulární a singulární matice)

Čtvercová matice se nazývá regulární, má-li maximální možnou hodnost, tzn. hodnost rovnou řádu matice. Čtvercová matice řádu n je tedy regulární, je-li $h = n$.

Čtvercová matice se nazývá singulární, nemá-li maximální možnou hodnost, tzn. má-li hodnost menší než je její řád. Čtvercová matice řádu n je tedy singulární, je-li $h < n$.

Pojem regulární matice se definuje pouze pro čtvercové matice. Každá čtvercová matice je buď regulární nebo singulární.

Příklad 9.1

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ regulární ($h = 2$), $\begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix}$ singulární (jeden řádek je násobkem druhého – $h = 1$),

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 0 & 0 \end{pmatrix}$ singulární ($h = 2$), $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ ani regulární ani singulární (není čtvercová),

I (jednotková matice libovolného řádu) – regulární,

O (nulová matice libovolného řádu) – singulární,

$\begin{pmatrix} 2 & -2 & 3 & 0 \\ -3 & 3 & -4 & 3 \\ 5 & -4 & 3 & 5 \\ 2 & 1 & -12 & 10 \end{pmatrix}$ regulární (výpočtem hodnosti – viz Příklad 5.5 z kapitoly 5: $h = 4$)

Protože má regulární matice maximální možnou hodnost, dostaneme po jejím odstupňování čtvercovou matici, která má všechny řádky nenulové, a je tudíž dále převeditelná na matici jednotkovou (jako u Jordanovy metody řešení soustav – viz kapitola 7.4).

Pravidlo 9.1 *(kritérium regularity)*

Matice A je regulární, právě když ji lze elementárními úpravami převést na matici jednotkovou.

Pravidlo 9.2 *(součin regulárních matic)*

Součin regulárních matic je regulární matice.

Poznámka (součet regulárních matic)

Pro součet podobné tvrzení neplatí – součet regulárních matic může být matice singulární, např.

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ je singulární matice.

Pravidlo 9.3 *(zachování hodnosti vynásobením regulární maticí)*

Vynásobením libovolné matice A regulární maticí příslušného řádu se nezmění hodnost matice A.

Pojem 9.2 (inverzní matice)

Nechť A je čtvercová matice. Inverzní maticí k matici A nazveme matici A^{-1}, pro kterou platí

$$A \cdot A^{-1} = I.$$

Pravidlo 9.4 *(existence inverzní matice)*

A^{-1} existuje, právě když A je regulární. Ke každé regulární matici existuje právě jedna inverzní matice.

Poznámka (singulární a obdélníková matice)

K singulárním maticím inverzní matice neexistuje. K maticím obdélníkovým se inverzní matice nedefinuje.

Pravidlo 9.5 *(regularita inverzní matice)*

Je-li A regulární, pak A^{-1} je také regulární.

Pravidlo 9.6 *(vlastnosti inverzní matice)*

Jsou-li A, B regulární matice stejného řádu, $c \in \mathbb{R}$, $c \neq 0$, pak

(1) $(A^{-1})^{-1} = A$,

(2) $AA^{-1} = A^{-1}A = I$,

(3) $(AB)^{-1} = B^{-1}A^{-1}$,

(4) $(cA)^{-1} = \frac{1}{c}A^{-1}$,

(5) $(A^T)^{-1} = (A^{-1})^T$.

Poznámka (navzájem inverzní matice)

Bod (1) v předchozím tvrzení znamená, že matice A je inverzní k matici A^{-1}. Matice A a A^{-1} jsou tedy navzájem inverzní.

Podle bodu (2) je zřejmé, že při násobení navzájem inverzních matic nezáleží na pořadí.

9.2 Výpočet inverzní matice

Chceme-li vypočítat matici inverzní ke čtvercové matici A, napíšeme vpravo od matice A jednotkovou matici stejného řádu; od matice A ji oddělíme svislou čarou (podobně jako při řešení soustav[14]). Elementárními řádkovými úpravami vzniklé "dvojmatice" převedeme matici A na redukovaný odstupňovaný tvar, spec. na jednotkovou matici (víme už z kapitoly 7.4, že při převodu čtvercové matice řádu n o hodnosti n na redukovaný odstupňovaný tvar vznikne jednotková matice) – tak jako při řešení soustav Jordanovou metodou. Se sloupci matice žádné úpravy neprovádíme. Pokud je matice A regulární, vznikne na místě původně jednotkové matice matice inverzní k matici A (pokud je A singulární, pozná se to při převodu matice na odstupňovanou). Postup hledání inverzní matice lze schematicky zapsat:

$$(A|I) \sim \ldots \sim (I|A^{-1}).$$

P ř í k l a d 9.2

Vypočítáme inverzní matici k matici

$$A = \begin{pmatrix} 1 & 2 & -3 \\ -1 & -1 & 7 \\ 0 & -1 & -2 \end{pmatrix}.$$

Postupujeme, jak bylo popsáno výše a jak je naznačeno vedle matic (připomínáme, že úpravy provádíme vždy na celém řádku "dvojmatice" – nezapomínejme pracovat i s jednotkovou maticí):

$$\left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0 \\ -1 & -1 & 7 & 0 & 1 & 0 \\ 0 & -1 & -2 & 0 & 0 & 1 \end{array}\right) \hookleftarrow \sim \left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 1 & 0 \\ 0 & -1 & -2 & 0 & 0 & 1 \end{array}\right) \hookleftarrow \sim$$

[14] Jak uvedeme podrobněji v kapitole 9.5, skutečně se jedná o současné řešení více soustav s maticí A.

$$\sim\left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0\\ 0 & 1 & 4 & 1 & 1 & 0\\ 0 & 0 & 2 & 1 & 1 & 1\end{array}\right).$$

Matici A jsme odstupňovali – z tohoto tvaru poznáme, zda je regulární či singulární. V tomto případě je hodnost matice $h = 3$, matice je tedy regulární. Můžeme pokračovat ve výpočtech – stejně jako u řešení soustav Jordanovou metodou převádíme matici na levé straně na jednotkovou (a opět provádíme úpravy i na řádcích pravé matice):

$$\left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0\\ 0 & 1 & 4 & 1 & 1 & 0\\ 0 & 0 & 2 & 1 & 1 & 1\end{array}\right)_{1/2} \sim \left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0\\ 0 & 1 & 4 & 1 & 1 & 0\\ 0 & 0 & 1 & 1/2 & 1/2 & 1/2\end{array}\right)\begin{array}{l} \leftarrow \\ \leftarrow \\ -4 \quad 3\end{array} \sim$$

$$\sim\left(\begin{array}{ccc|ccc} 1 & 2 & 0 & 5/2 & 3/2 & 3/2\\ 0 & 1 & 0 & -1 & -1 & -2\\ 0 & 0 & 1 & 1/2 & 1/2 & 1/2\end{array}\right)\begin{array}{l} \leftarrow \\ -2 \\ \end{array} \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{9}{2} & \frac{7}{2} & \frac{11}{2}\\ 0 & 1 & 0 & -1 & -1 & -2\\ 0 & 0 & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{array}\right).$$

Inverzní maticí k matici A je

$$A^{-1} = \begin{pmatrix} \frac{9}{2} & \frac{7}{2} & \frac{11}{2}\\ -1 & -1 & -2\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{pmatrix} = \frac{1}{2}\cdot\begin{pmatrix} 9 & 7 & 11\\ -2 & -2 & -4\\ 1 & 1 & 1\end{pmatrix}.$$

Pokud bychom chtěli udělat zkoušku, stačí ověřit, že $A \cdot A^{-1} = I$. Při násobení bývá (při "ručním" počítání) příjemnější počítat s celočíselnými maticemi – využijeme proto tvaru s vytknutou $\frac{1}{2}$ (připomínáme, že při násobení matice číslem nezáleží na pořadí):

$$A\cdot A^{-1} = \begin{pmatrix} 1 & 2 & -3\\ -1 & -1 & 7\\ 0 & -1 & -2\end{pmatrix}\cdot\begin{pmatrix} 9 & 7 & 11\\ -2 & -2 & -4\\ 1 & 1 & 1\end{pmatrix}\cdot\frac{1}{2} = \begin{pmatrix} 2 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 2\end{pmatrix}\cdot\frac{1}{2} = \begin{pmatrix} 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} = I.$$

Poznámka (odstupňovaný tvar singulární matice)

Pokud by matice, ke které se snažíme počítat inverzní, byla singulární, poznali bychom to snadno ve chvíli, kdy je matice převedená na odstupňovanou: přinejmenším poslední řádek matice by byl nulový. V takovém případě nemá smysl pokračovat ve výpočtech (a ani by to nebylo možné– diagonální prvek v posledním řádku, jehož pomocí bychom chtěli vynulovat poslední sloupec levé matice, je nulový). Inverzní matice neexistuje.

Poznámka (úpravy při "ručním" počítání)

Jak jsme již uvedli v Poznámce za Příkladem 7.9 v kapitole 7.4, při "ručním" počítání může být počítání se zlomky značně nepříjemné. Při převodu matice na jednotkovou se tomu lze vyhnout tak, jak jsme popsali ve zmíněné Poznámce – že nebudeme v průběhu výpočtu normovat pivoty. Odstupňovanou matici z Příkladu 9.2 bychom při "ručním" počítání upravovali následovně:

$$\left(\begin{array}{ccc|ccc} 1 & 2 & -3 & 1 & 0 & 0\\ 0 & 1 & 4 & 1 & 1 & 0\\ 0 & 0 & 2 & 1 & 1 & 1\end{array}\right)\begin{array}{l} 2\leftarrow \\ \leftarrow \\ -2 \quad 3\end{array} \sim \left(\begin{array}{ccc|ccc} 2 & 4 & 0 & 5 & 3 & 3\\ 0 & 1 & 0 & -1 & -1 & -2\\ 0 & 0 & 2 & 1 & 1 & 1\end{array}\right)\begin{array}{l} \leftarrow \\ -4 \\ \end{array} \sim$$

$$\sim\left(\begin{array}{ccc|ccc} 2 & 0 & 0 & 9 & 7 & 11\\ 0 & 1 & 0 & -1 & -1 & -2\\ 0 & 0 & 2 & 1 & 1 & 1\end{array}\right)\begin{array}{l} 1/2 \\ \\ 1/2\end{array} \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{9}{2} & \frac{7}{2} & \frac{11}{2}\\ 0 & 1 & 0 & -1 & -1 & -2\\ 0 & 0 & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{array}\right).$$

Pro jednodušší výklad budeme nadále v této knize pro "ruční" počítání používat tento příjemnější postup. Při počítačovém zpracování dodržíme postup s průběžným normováním pivotů.

9.3 Maticové rovnice

Maticovou rovnicí budeme rozumět rovnici, kde neznámou je matice. Maticové rovnice lze upravovat podobně jako rovnice číselné, ne vždy ale lze neznámou matici z rovnice vyjádřit. Při úpravách maticových rovnic je třeba dbát toho, že násobení matic není komutativní. Maticové rovnice lze řešit rozepsáním po jednotlivých prvcích matic, což vede na soustavy rovnic (tento postup zde nebudeme uvádět). Některé maticové rovnice lze řešit pomocí inverzní matice – tento způsob si ukážeme.

V této knize se omezíme pouze na základní maticové rovnice $AX = B$ a $XA = B$ (vzhledem k nekomutativitě násobení matic se jedná o dvě různé rovnice!), kde A je regulární matice.

Uvažujme nejprve maticovou rovnici $AX = B$, kde A, B a X jsou matice vhodných typů. Budeme-li chtít celou rovnici vynásobit nějakou maticí, budeme muset vzhledem k nekomutativitě násobení rozlišovat následující úpravy:

Pojem 9.3 (násobení rovnice maticí zprava a zleva)

Násobit rovnici maticí (vhodného typu) zprava znamená, že matice, kterou je rovnice násobena, stojí na obou stranách rovnice vpravo od násobených výrazů.

Násobit rovnici maticí (vhodného typu) zleva znamená, že matice, kterou je rovnice násobena, stojí na obou stranách rovnice vlevo od násobených výrazů.

Násobení rovnice $AX = B$ maticí C zprava tedy vypadá:

$$\begin{aligned} AX &= B \qquad / \cdot C \text{ zprava} \\ AXC &= BC, \end{aligned}$$

násobení rovnice $AX = B$ maticí C zleva:

$$\begin{aligned} AX &= B \qquad / \cdot C \text{ zleva} \\ CAX &= CB. \end{aligned}$$

Nelze násobit rovnici následujícím způsobem:

$$\begin{aligned} AX &= B \qquad / \cdot C \\ CAX &= BC \end{aligned}$$

nebo

$$\begin{aligned} AX &= B \qquad / \cdot C \\ AXC &= CB. \end{aligned}$$

nebo

$$\begin{aligned} AX &= B \qquad / \cdot C \\ ACX &= CB \end{aligned}$$

apod.

Vraťme se k maticové rovnici $AX = B$, kde $A_{n\times n}$, $B_{n\times p}$ jsou dané matice a X je neznámá matice, kterou chceme z rovnice vyjádřit. Představíme-li si obdobnou číselnou rovnici $ax = b$, vidíme, že k výpočtu neznámé x bychom potřebovali rovnici vydělit číslem a (za předpokladu, že je nenulové). Dělení matic není definováno, ale z úvodu víme, že lze v případě, že je příslušná matice regulární, nahradit násobením inverzní maticí. Budeme tedy chtít rovnici vynásobit maticí A^{-1} – k tomu je potřeba, aby matice A byla regulární.

V případě rovnice $AX = B$ vynásobíme rovnici maticí A^{-1} zleva – tak, aby matice A a A^{-1} byly "těsně vedle sebe":

$$\begin{aligned} AX &= B \qquad / \cdot A^{-1} \text{ zleva} \\ A^{-1}AX &= A^{-1}B \\ IX &= A^{-1}B \\ X &= A^{-1}B. \end{aligned}$$

Kdybychom násobili maticí A^{-1} zprava, dostali bychom $AXA^{-1} = BA^{-1}$ a nemohli bychom využít toho, že součin navzájem inverzních matic je jednotková matice.

Pravidlo 9.7 *(řešení rovnice $AX = B$)*

Je-li A regulární, má rovnice $AX = B$ právě jedno řešení $X = A^{-1}B$ pro libovolnou matici B vhodného typu.

Poznámka (rovnice $AX = B$ se singulární maticí)

Zdůrazňujeme, že je-li A singulární, neznamená to, že rovnice $AX = B$ nemá řešení! V takovém případě rovnice $AX = B$ buď řešení nemá, nebo jich má nekonečně mnoho. Je ale potřeba postupovat jiným způsobem (výpočet rozepsáním po jednotlivých prvcích), který zde neuvádíme.

Příklad 9.3

Vyřešíme maticovou rovnici $AX = B$, kde

$$A = \begin{pmatrix} -3 & 1 \\ 1 & -2 \end{pmatrix}, \qquad B = \begin{pmatrix} -5 & 10 \\ -20 & 15 \end{pmatrix}.$$

Víme, že za předpokladu, že matice A je regulární, je řešením dané rovnice matice $X = A^{-1}B$. Nejprve vypočítáme matici A^{-1} (během výpočtu zjistíme, zda A je regulární):

$$\left(\begin{array}{cc|cc} -3 & 1 & 1 & 0 \\ 1 & -2 & 0 & 1 \end{array}\right)_{3\leftarrow} \sim \left(\begin{array}{cc|cc} -3 & 1 & 1 & 0 \\ 0 & -5 & 1 & 3 \end{array}\right)^{5\leftarrow} \sim \left(\begin{array}{cc|cc} -15 & 0 & 6 & 3 \\ 0 & -5 & 1 & 3 \end{array}\right) \sim \left(\begin{array}{cc|cc} 1 & 0 & -\frac{2}{5} & -\frac{1}{5} \\ 0 & 1 & -\frac{1}{5} & -\frac{3}{5} \end{array}\right),$$

$$A^{-1} = \begin{pmatrix} -\frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{3}{5} \end{pmatrix} = -\frac{1}{5} \cdot \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}.$$

Nyní stačí dosadit za matice A^{-1} a B do vztahu $X = A^{-1}B$:

$$X = A^{-1}B = -\frac{1}{5} \cdot \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} -5 & 10 \\ -20 & 15 \end{pmatrix} = -\frac{1}{5} \cdot \begin{pmatrix} -30 & 35 \\ -65 & 55 \end{pmatrix} = \begin{pmatrix} 6 & -7 \\ 13 & -11 \end{pmatrix}.$$

Maticová rovnice $XA = B$, kde $A_{n\times n}$, $B_{p\times n}$ jsou dané matice a X je neznámá matice, se řeší analogicky – změna je pouze v tom, že při řešení násobíme rovnici maticí A^{-1} nikoliv zleva, ale zprava (opět je potřeba, aby matice A byla regulární):

$$\begin{aligned} XA &= B \qquad / \cdot A^{-1} \text{ zprava} \\ XAA^{-1} &= BA^{-1} \\ XI &= BA^{-1} \\ X &= BA^{-1}. \end{aligned}$$

Pravidlo 9.8 *(řešení rovnice $XA = B$)*

Je-li A regulární, má rovnice $XA = B$ právě jedno řešení $X = BA^{-1}$ pro libovolnou matici B vhodného typu.

Jako ukázku využití výše vyloženého uvedeme způsob šifrování textu pomocí matic.

Příklad 9.4

Uvažujme následující způsob šifrování textu: každému písmenu abecedy (bez háčků a čárek) přiřadíme číslo podle tabulky:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
8	7	5	13	9	16	18	22	4	23	11	3	21	1	6	15	12	19	2	14	17	20	25	24	10	26

Text, který chceme zašifrovat, zapíšeme (v číselné podobě) po řádcích do čtvercové matice; pokud by byl počet písmen textu menší než počet prvků matice, doplníme zbylá místa nulami. Zformujeme např. text "BÍLÁ KOČKA" (jakožto "BILA KOCKA") do matice A:

$$A = \begin{pmatrix} 7 & 4 & 3 \\ 8 & 11 & 6 \\ 5 & 11 & 8 \end{pmatrix}.$$

Dále musí být dána čtvercová matice příslušného řádu, která – jak uvidíme později – musí být regulární. Nechť je to např. matice

$$C = \begin{pmatrix} 2 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Text zašifrujeme vynásobením maticí C zleva:

$$Z = CA = \begin{pmatrix} 19 & 19 & 14 \\ 12 & 15 & 11 \\ 8 & 11 & 6 \end{pmatrix}.$$

Pro dešifrování textu (tzn. vyjádření matice A z maticové rovnice $Z = CA$) je třeba matici Z vynásobit maticí C^{-1} zleva (výpočet C^{-1} ponecháváme jako cvičení):

$$C^{-1}Z = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 19 & 19 & 14 \\ 12 & 15 & 11 \\ 8 & 11 & 6 \end{pmatrix} = A.$$

Osoba, která má text dešifrovat, tedy musí kromě výše uvedené tabulky znát matici C^{-1} a vědět, že pro dešifrování má matici Z vynásobit zleva.

Protože násobení matic není komutativní, je třeba důsledně dodržovat pořadí násobených matic. Kdybychom vynásobili matice C^{-1} a Z v opačném pořadí, dostali bychom

$$ZC^{-1} = \begin{pmatrix} 19 & 19 & 14 \\ 12 & 15 & 11 \\ 8 & 11 & 6 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 5 & 9 & 19 \\ 1 & 10 & 15 \\ 2 & 4 & 11 \end{pmatrix}.$$

Po zpětném přiřazení příslušných písmen bychom obdrželi text "CERNY PSIK" neboli "ČERNÝ PSÍK"!

Poznámka (ilustrativní příklad)

Na předchozím příkladu jsme dobře viděli, jaký problém může způsobit, nehlídáme-li pořadí v součinu matic. Poznamenejme však, že tento příklad byl za tímto účelem uměle zkonstruován – tabulka i matice C byly sestrojeny tak, aby při vynásobení matic v opačném pořadí vyšlo sdělení "opačného" významu. To by pro praktické použití nebylo možné.

V praxi by bylo možné pro přiřazení písmeno-číslo použít tabulku ASCII[15], která je veřejně známá. Matici, jejímž vynásobením je text zašifrován (resp. matici k ní inverzní) už smí znát jen osoba "pověřená".

Poznámka (úpravy maticových rovnic)

Pro zvídavé čtenáře se zde nad zamýšlený rámec této knihy pouze stručně zmíníme o úpravách, pomocí nichž lze řešit i další maticové rovnice. Potřebné úpravy (odpovídající příslušným úpravám v číselných rovnicích) se provádějí s ohledem na to, že násobení matic není komutativní. To se týká i vytýkání: zatímco u čísel je jedno, jestli vytkneme před nebo za závorku, u matic to musíme rozlišovat – je rozdíl, zda vytkneme matici před závorku (doleva) nebo za závorku (doprava). Vytýkáme na tu stranu, na které stojí vytýkaná matice v součinech (v součinu matice s číslem na pořadí nezáleží) – např. (matice předpokládáme vhodných typů):

$AX - 2BX = (A - 2B)X$, $\quad 3XC + 5XD = X(3C + 5D)$, $\quad 2EX + XF$ nelze vytknout X.

Pro ilustraci uvedeme bez podrobnějšího komentáře postup, jak z maticové rovnice $AX + 3C = 2BX$ vyjádříme matici X (porovnejte s postupem řešení číselné rovnice $ax + 3c = 2bx$):

[15] ASCII je standardizováno v RFC 20 – viz [8].

$$AX + 3C = 2BX$$
$$AX - 2BX = -3C$$
$$(A - 2B)X = -3C$$

a vynásobením maticí $(A-2B)^{-1}$zleva (je-li matice $A-2B$ regulární):

$$(A-2B)^{-1}(A-2B)X = (A-2B)^{-1}(-3C)$$
$$X = -3(A-2B)^{-1}C.$$

Poznámka (efektivnější postup řešení maticových rovnic)

V kapitole 9.5 uvedeme efektivnější postup, jak lze vyřešit maticové rovnice $AX = B$ a $XA = B$.

9.4 Maticový zápis soustavy lineárních rovnic. Řešení soustavy užitím inverzní matice

Jak uvidíme v následujícím příkladu, lze každou soustavu lineárních rovnic považovat za speciální případ maticové rovnice $AX = B$, kde matice B i X jsou tvořeny jediným sloupcem – tyto matice pak můžeme považovat za (sloupcové) vektory.

P ř í k l a d 9.5

Soustavu

$$\begin{array}{rcrcrcl} 4x_0 & - & 3x_1 & + & x_2 & = & 1 \\ 2x_0 & + & x_1 & - & 3x_2 & = & 5 \end{array}$$

lze psát jako maticovou rovnici:

$$\begin{pmatrix} 4 & -3 & 1 \\ 2 & 1 & -3 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$$

(ověřte si vynásobením matic na levé straně a následným porovnáním odpovídajících si prvků matic na obou stranách rovnice).

Pravidlo 9.9 *(maticový zápis soustavy lineárních rovnic)*

Každou soustavu lineárních rovnic lze zapsat jako maticovou rovnici

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

kde $A_{m\times n}$ *je matice soustavy,*

$\bar{x} = \begin{pmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_{n-1} \end{pmatrix}$ *(matice* $n \times 1$*, tj. sloupcový vektor) je vektor neznámých a*

$\bar{b} = \begin{pmatrix} b_0 \\ b_1 \\ . \\ . \\ . \\ x_{m-1} \end{pmatrix}$ *(matice* $m \times 1$*, tj. sloupcový vektor) je vektor pravých stran soustavy.*

Homogenní soustavu lze zapsat jako rovnici $A\bar{x} = \bar{o}$.

Pojem 9.4 (pravá strana soustavy)

Vektor $\bar{b}$ (vektor pravých stran) často stručněji nazýváme pravá strana.

Poznámka (úmluva: řádkový a sloupcový vektor)

Je-li vektor $\bar{x}$ řádkový, pak sloupcový vektor by měl být značen $\bar{x}^T$ (a naopak). Jak jsme se už zmínili v 1. kapitole, nebudeme to pro jednodušší zápis důsledně dodržovat – pokud nemůže dojít k omylu,

nebudeme rozlišovat značení řádkových a sloupcových vektorů. V maticovém zápisu soustavy uvažujeme vektory zásadně *sloupcové*.

Soustavu $A\bar{x} = \bar{b}$ lze tedy řešit jako maticovou rovnici a platí, co bylo řečeno v kapitole 9.3.

Pravidlo 9.10 *(řešení soustavy užitím inverzní matice)*

Je-li matice soustavy regulární, má soustava $A\bar{x} = \bar{b}$ pro každou pravou stranu $\bar{b}$ právě jedno řešení

$$\bar{x} = A^{-1}\bar{b}.$$

Poznámka (soustavy se singulární a obdélníkovou maticí)

Je-li matice soustavy singulární, daná soustava buď nemá řešení nebo jich má nekonečně mnoho, ale nelze je nalézt užitím inverzní matice.

Není-li matice A čtvercová, nedá se obecně o řešitelnosti (nehomogenní) soustavy nic říci, rozhodně však v tomto případě nelze soustavu řešit užitím inverzní matice.

Poznámka (řešení homogenní soustavy užitím inverzní matice)

Pro homogenní soustavu je řešení užitím inverzní matice naprosto nevhodné:

Homogenní soustava s regulární maticí má podle předchozího právě jedno řešení. Tímto jediným řešením je triviální řešení. V tomto případě ale nemá smysl získávat triviální řešení pomocí inverzní matice.

Homogenní soustava se singulární maticí má nekonečně mnoho řešení, kromě triviálního i netriviální. Nelze je však získat pomocí inverzní matice, stejně jako v případě, že je matice soustavy obdélníková.

Pravidlo 9.11 *(soustava s regulární a singulární maticí)*

Je-li A čtvercová matice, pak platí:

soustava $A\bar{x} = \bar{b}$ má právě jedno řešení $\iff$ matice A je regulární.

Speciálně pro homogenní soustavu se čtvercovou maticí A tedy platí:

homogenní soustava $A\bar{x} = \bar{o}$ má pouze triviální řešení $\iff$ matice A je regulární

homogenní soustava $A\bar{x} = \bar{o}$ má i netriviální řešení $\iff$ matice A je singulární

P ř í k l a d 9.6

Užitím inverzní matice vyřešíme soustavu

$$\begin{array}{rcrcrcr} x_0 & + & 2x_1 & - & 3x_2 & = & -4 \\ -x_0 & - & x_1 & + & 7x_2 & = & 5 \\ & - & x_1 & - & 2x_2 & = & 1. \end{array}$$

Matice soustavy je matice A z Příkladu 9.2, tam je také vypočítána matice A^{-1}. Protože A je regulární, má daná soustava právě jedno řešení dané vztahem

$$\bar{x} = A^{-1}\bar{b} = \begin{pmatrix} 1 & 2 & -3 \\ -1 & -1 & 7 \\ 0 & -1 & -2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} -4 \\ 5 \\ 1 \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} 9 & 7 & 11 \\ -2 & -2 & -4 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} -4 \\ 5 \\ 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 10 \\ -6 \\ 2 \end{pmatrix} = \begin{pmatrix} 5 \\ -3 \\ 1 \end{pmatrix}.$$

Je tedy $x_0 = 5$, $x_1 = -3$, $x_2 = 1$.

Poznámka (řešení jediné soustavy užitím inverzní matice)

Řešení jediné soustavy užitím inverzní matice není efektivní – je příliš pracné. Metoda je lépe použitelná v případech, kdy je třeba řešit více soustav se stejnou maticí soustavy, ale různými pravými stranami (pro všechny soustavy se vypočítá jedna matice A^{-1}, a řešení každé soustav se získá jako součin této matice s příslušným vektorem pravých stran). I v tomto případě je ale méně pracný způsob, který jsme uvedli v kapitole 7.4. Zápis soustavy rovnic $A\bar{x} = \bar{b}$ je však velmi praktický a hojně používaný při formulaci různých tvrzení o soustavách rovnic.

9.5 Efektivnější způsob řešení maticových rovnic $AX = B$ a $XA = B$

Uvažujme opět maticovou rovnici $AX = B$, kde $A_{n\times n}$ a $B_{n\times p}$, jsou dané matice, přičemž A je regulární. Víme, že v takovém případě má tato rovnice právě jedno řešení, které lze získat pomocí inverzní matice. Zde uvedeme efektivnější způsob řešení této rovnice.

Lze si rozmyslet, že maticová rovnice $A_{n\times n}X_{n\times p} = B_{n\times p}$ vlastně popisuje problém řešení několika (a to p) soustav se stejnou maticí A. Víme z kapitoly 9.4, že každou soustavu můžeme psát ve tvaru $A\bar{x} = \bar{b}$. Maticová rovnice $AX = B$ pak popisuje systém takovýchto soustav – jednotlivé sloupce matice B popisují pravé strany jednotlivých soustav, jim přísluší odpovídající vektory neznámých – sloupce matice X. Přesněji: označíme-li sloupce matice X jako $\bar{x}_{(0)}, \ldots, \bar{x}_{(p-1)}$ a sloupce matice B jako $\bar{b}_{(0)}, \ldots, \bar{b}_{(p-1)}$, pak pro každé $i = 0, 1, \ldots, p-1$ popisuje rovnice $A\bar{x}_{(i)} = \bar{b}_{(i)}$ jednu z p soustav.

V kapitole 7.4 (viz Příklad 7.11) jsme ukázali, jak lze systém více soustav se stejnou maticí soustavy řešit užitím Jordanovy metody – vedle matice soustavy napíšeme (odděleny svislou čarou) sloupce všech pravých stran, v případě maticové rovnice $AX = B$ tedy matici B. Pak řádkovými úpravami takto vzniklé "dvojmatice" převedeme matici soustavy na jednotkovou matici, sloupce pravé matice budou řešeními jednotlivých soustav. To znamená, že vpravo dostaneme hledanou matici X.

Tento proces lze schematicky zapsat:

$$(A|B) \sim \ldots \sim (I|X).$$

Pravidlo 9.12 *(postup řešení rovnice $AX = B$)*

Maticovou rovnici $AX = B$, kde A je regulární matice a B matice vhodného typu, lze řešit podle následujícího schematu:

$$(A|B) \sim \ldots \sim (I|X).$$

Poznámka (maticová rovnice $AX = I$)

Proces hledání inverzní matice k matici A – tak, jak byl popsán v kapitole 9.2 – odpovídá řešení maticové rovnice $AX = I$ (podle definice inverzní matice $AA^{-1} = I$ je zřejmé, že $X = A^{-1}$) způsobem uvedeným v Pravidle 9.12:

$$(A|I) \sim \ldots \sim (I|X).$$

Při hledání inverzní matice k matici A řádu n tedy vlastně současně řešíme n soustav s maticí A a pravými stranami rovnými sloupcům jednotkové matice.

Poznámka (upozornění)

Podotýkáme, že matice B nemusí být čtvercová – počet jejích sloupců odpovídá počtu soustav, které maticovou rovnicí $AX = B$ současně řešíme.

P ř í k l a d 9.7

Vyřešíme maticovou rovnici $AX = B$ pro matice $A = \begin{pmatrix} 1 & -1 & 2 \\ 2 & -3 & 3 \\ -1 & 1 & -1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 5 & 2 \\ -4 & -2 & 1 \end{pmatrix}$

(podle předchozího víme, že vlastně budeme současně řešit 3 (počet sloupců matice B) soustavy s maticí A a pravými stranami tvořenými sloupci matice B).

Napíšeme matici soustavy, svislou čarou oddělíme matici B. Matici A převedeme na jednotkovou:

$$\left(\begin{array}{ccc|ccc} 1 & -1 & 2 & 2 & 3 & 0 \\ 2 & -3 & 3 & 5 & 5 & 2 \\ -1 & 1 & -1 & -4 & -2 & 1 \end{array}\right) \begin{matrix} \\ {\scriptstyle -2} \\ \\ \end{matrix} \sim \left(\begin{array}{ccc|ccc} 1 & -1 & 2 & 2 & 3 & 0 \\ 0 & -1 & -1 & 1 & -1 & 2 \\ 0 & 0 & 1 & -2 & 1 & 1 \end{array}\right) \begin{matrix} \\ \\ {\scriptstyle -2} \end{matrix} \sim$$

$$\sim \left(\begin{array}{ccc|ccc} 1 & -1 & 0 & 6 & 1 & -2 \\ 0 & -1 & 0 & -1 & 0 & 3 \\ 0 & 0 & 1 & -2 & 1 & 1 \end{array}\right) \begin{matrix} \\ {\scriptstyle -1} \\ \\ \end{matrix} \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 7 & 1 & -5 \\ 0 & 1 & 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -2 & 1 & 1 \end{array}\right)$$

(jednotlivé sloupce matice vpravo od čáry jsou řešeními příslušných soustav – celkově tvoří hledanou matici X). Je tedy

$$X = \begin{pmatrix} 7 & 1 & -5 \\ 1 & 0 & -3 \\ -2 & 1 & 1 \end{pmatrix}.$$

Podívejme se nyní na maticovou rovnici $XA = B$, kde $A_{n\times n}$, $B_{p\times n}$. Matici A opět budeme předpokládat regulární. Rozdíl od maticové rovnice $AX = B$ je v tom, že matice X stojí v součinu s maticí A na opačné straně. Takovou maticovou rovnici už jako systém více soustav interpretovat nelze. Pomůžeme si ale jinak.

Platí-li rovnost matic $XA = B$, musí tato rovnost platit i tehdy, když matice na obou stranách rovnosti transponujeme:

$$(XA)^T = B^T.$$

Odtud s využitím vztahu pro transpozici součinu: $(XA)^T = A^T X^T$ (viz kapitola 3.3, Pravidlo 3.2, bod (7)) pak dostáváme

$$A^T X^T = B^T.$$

Neznámá matice X^T stojí nyní v součinu vpravo od matice A^T – tuto rovnici už lze řešit způsobem uvedeným výše – stejně, jako maticovou rovnici $AX = B$.

Stačí tedy zadané matice A, B transponovat, získat matici X^T vyřešením rovnice $A^T X^T = B^T$ pomocí řádkových úprav "dvojmatice" $(A^T|B^T)$ a nakonec dostat matici X transponováním matice X^T (neboť $(X^T)^T = X$).

Pravidlo 9.13 *(postup řešení rovnice $XA = B$)*

Maticovou rovnici $XA = B$, kde A je regulární matice a B matice vhodného typu, lze řešit podle následujícího schematu:

$$(A^T|B^T) \sim \ldots \sim (I|X^T), \qquad X = (X^T)^T.$$

P ř í k l a d 9.8

Vyřešíme maticovou rovnici $XA = B$ pro matice $A = \begin{pmatrix} 1 & -1 & 2 \\ 2 & -3 & 3 \\ -1 & 1 & -1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 5 & 2 \\ -4 & -2 & 1 \end{pmatrix}$.

Řešíme nejprve rovnici $A^T X^T = B^T$. Napíšeme matici A^T, vedle matici B^T, a řádkovými úpravami převedeme matici A^T na jednotkovou:

$$\left(\begin{array}{ccc|ccc} 1 & 2 & -1 & 2 & 5 & -4 \\ -1 & -3 & 1 & 3 & 5 & -2 \\ 2 & 3 & -1 & 0 & 2 & 1 \end{array}\right) \overset{-2}{\sim} \left(\begin{array}{ccc|ccc} 1 & 2 & -1 & 2 & 5 & -4 \\ 0 & -1 & 0 & 5 & 10 & -6 \\ 0 & -1 & 1 & -4 & -8 & 9 \end{array}\right) \overset{-1}{\sim}$$

$$\sim \left(\begin{array}{ccc|ccc} 1 & 2 & -1 & 2 & 5 & -4 \\ 0 & -1 & 0 & 5 & 10 & -6 \\ 0 & 0 & 1 & -9 & -18 & 15 \end{array}\right) \sim \left(\begin{array}{ccc|ccc} 1 & 2 & 0 & -7 & -13 & 11 \\ 0 & -1 & 0 & 5 & 10 & -6 \\ 0 & 0 & 1 & -9 & -18 & 15 \end{array}\right) \overset{2}{\sim}$$

$$\sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 3 & 7 & -1 \\ 0 & 1 & 0 & -5 & -10 & 6 \\ 0 & 0 & 1 & -9 & -18 & 15 \end{array}\right).$$

Obdrželi jsme matici $X^T = \begin{pmatrix} 3 & 7 & -1 \\ -5 & -10 & 6 \\ -9 & -18 & 15 \end{pmatrix}$, a tedy řešením zadané rovnice je matice

$$X = \begin{pmatrix} 3 & -5 & -9 \\ 7 & -10 & -18 \\ -1 & 6 & 15 \end{pmatrix}.$$

Poznámka (řešení rovnice $XA = B$ sloupcovými úpravami)

Maticovou rovnici $XA = B$ lze ovšem také řešit přímo – bez transponování matic. V tomto případě je však nutné místo řádkových úprav provádět úpravy sloupcové (je pak vhodnější psát matice A a B nikoliv vedle sebe, ale pod sebe). Takový postup řešení pak můžeme schematicky zapsat (pozor – nejedná se o zlomky!):

$$\left(\frac{A}{B}\right) \sim \ldots \sim \left(\frac{I}{X}\right).$$

Protože se v této knize sloupcovým úpravám pokud možno vyhýbáme, nebudeme se tímto postupem dále zabývat.

9.6 Úlohy k procvičení

$\boxed{A}$ **Teoretická část**

Posuďte pravdivost následujících tvrzení (pokud se v následujících úlohách vyskytuje součin matic, předpokládáme, že matice jsou vhodných typů):

a) Je-li matice singulární, pak některý z jejích řádků je násobkem jiného řádku.

b) Je-li matice regulární, pak žádný z jejích řádků není násobkem jiného řádku.

c) Je-li matice řádu 3 singulární, pak je její hodnost rovna 2.

d) Čtvercová matice, která má maximální možnou hodnost, je regulární.

e) Každá matice je buď regulární nebo singulární.

f) Součin regulárních matic je regulární matice.

g) Součet regulárních matic je regulární matice.

h) Ke každé nenulové matici existuje matice inverzní.

i) Ke každé regulární matici existuje právě jedna matice inverzní.

j) K singulární matici může existovat víc inverzních matic.

k) Matice A^{-1} je inverzní k matici A, platí-li $A \cdot I = A^{-1}$.

l) Matice X je inverzní k matici A, platí-li $A \cdot X = I$.

m) Součin vzájemně inverzních matic nezáleží na jejich pořadí.

n) Pro regulární matice A, B platí: $(AB)^{-1} = A^{-1} \cdot B^{-1}$.

o) Pro regulární matice A, B platí: $(AB)^{-1} = B^{-1} \cdot A^{-1}$.

p) Pro regulární matice A, B platí: $(A + B)^{-1} = B^{-1} + A^{-1}$.

q) Pro regulární matici A platí: $\left((A^{-1})^{-1}\right)^{-1} = A^{-1}$.

r) Pro jednotkovou matici I platí: $I^{-1} = I$.

s) Maticová rovnice $AX = B$ se čtvercovou maticí A má právě jedno řešení.

t) Maticová rovnice $AX = B$ s regulární maticí A má právě jedno řešení.

u) Maticová rovnice $AX = O$ (O – nulová matice) s regulární maticí A má právě jedno řešení $X = O$.

v) Maticová rovnice $AX = I$ s regulární maticí A má právě jedno řešení $X = I$.

w) Maticová rovnice $AX = I$ s regulární maticí A má právě jedno řešení $X = A^{-1}$.

x) Je-li A singulární matice, pak rovnice $AX = B$ nemá řešení.

y) Je-li A regulární matice, pak soustava $A\bar{x} = \bar{b}$ má právě jedno řešení.

z) Je-li A singulární matice, pak soustava $A\bar{x} = \bar{b}$ má nekonečně mnoho řešení.

aa) Je-li A singulární matice, pak soustava $A\bar{x} = \bar{o}$ má nekonečně mnoho řešení.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N (některý řádek může být např. součtem jiných řádků)

b) P
c) N (hodnost může být rovna i jedné)
d) P
e) N (platí pouze pro čtvercovou matici)
f) P
g) N (viz Poznámka za Příkladem 1)
h) N (matice musí být regulární)
i) P
j) N (k singulární matici neexistuje inverzní)
k) N ($A \cdot I = A$)
l) N (platí pouze pro čtvercovou matici A)
m) P
n) N (opačné pořadí v součinu inverzních matic)
o) P
p) N (tak jako analogický vztah neplatí pro čísla)
q) P
r) P
s) N (pouze, když A je navíc regulární)
t) P
u) P
v) N (stačí ověřit dosazením)
w) P
x) N (rovnice může mít i nekonečně mnoho řešení)
y) P
z) N (soustava nemusí mít řešení)
aa) P

B Praktická část

1 Zjistěte, které z následujících matic jsou regulární a které singulární. K regulárním vypočítejte inverzní a proveďte zkoušku podle definice inverzní matice.

a) $\begin{pmatrix} 3 & -1 \\ 1 & 2 \end{pmatrix}$ b) $\begin{pmatrix} 4 & 1 \\ -8 & -2 \end{pmatrix}$ c) $\begin{pmatrix} 5 & -1 & 2 \\ 1 & -2 & 3 \end{pmatrix}$ d) $\begin{pmatrix} -3 & 1 \\ 6 & -2 \\ -9 & 3 \end{pmatrix}$

e) $\begin{pmatrix} 0 & 1 & 3 \\ 1 & 0 & -1 \\ -3 & -1 & 2 \end{pmatrix}$ f) $\begin{pmatrix} 2 & 1 & 3 \\ -1 & 2 & 1 \\ 3 & -1 & 2 \end{pmatrix}$ g) $\begin{pmatrix} 1 & 1 & 0 & 1 \\ 3 & 0 & 2 & 5 \\ 2 & 5 & -1 & 0 \\ 0 & 3 & -1 & -1 \end{pmatrix}$ h) $A = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 1 \\ 5 & 4 & 2 \end{pmatrix}$

2 Vypočítejte A^{-1}, B^{-1}, AB, $(AB)^{-1}$, $B^{-1}A^{-1}$ pro $A = \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 1 \\ -3 & 0 \end{pmatrix}$.

3 Proveďte zkoušku správnosti řešení Příkladu 9.3 (dosazením výsledné matice X do rovnice).

4 Řešte maticovou rovnici z Příkladu 9.3 způsobem popsaným v kapitole 9.5.

5 Řešte maticovou rovnici $XA = B$ pro matice A, B z Příkladu 9.3 pomocí inverzní matice.

6 Řešte maticovou rovnici $XA = B$ pro matice A, B z Příkladu 9.3 způsobem popsaným v kapitole 9.5.

7 Řešte maticovou rovnici $AX = B$ pro dané matice A, B, je-li matice A regulární.

a) $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$ b) $A = \begin{pmatrix} 2 & 1 \\ 3 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

c) $A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ d) $A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, B = \begin{pmatrix} 3 & 5 \\ 6 & 10 \end{pmatrix}$

8 Řešte maticovou rovnici $XA = B$ pro matice A, B z úlohy 5, je-li matice A regulární.

9 Řešte následující soustavy pomocí inverzní matice. Využijte případně výsledků úloh 1 a), e), g).

a)
$$\begin{aligned} 3x_0 - x_1 &= 21 \\ x_0 + 2x_1 &= 14, \end{aligned}$$

b)
$$\begin{aligned} x_1 + 3x_2 &= -2 \\ x_0 - x_2 &= 1 \\ -3x_0 - x_1 + 2x_2 &= 0, \end{aligned}$$

c)
$$\begin{aligned} x_0 + x_1 + x_3 &= 1 \\ 3x_0 + 2x_2 + 5x_3 &= -3 \\ 2x_0 + 5x_1 - x_2 &= 4 \\ 3x_1 - x_2 - x_3 &= -3 \end{aligned}$$

10 Pro zvídavé čtenáře: Z maticové rovnice nejdříve vyjádřete matici X, do výsledku pak dosaďte dané matice a vypočítejte.

a) $AX - A = BX$, kde $A = \begin{pmatrix} 2 & 1 \\ 4 & 2 \end{pmatrix}$, $B = \begin{pmatrix} 4 & 2 \\ 1 & 2 \end{pmatrix}$

b) $XA + XB = 2X + B$, kde $A = \begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 2 \\ 3 & -1 \end{pmatrix}$

c) $3AX - 2X = 10I$, kde $A = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix}$

d) $XA - 2XB = C$, kde $A = \begin{pmatrix} 3 & 2 \\ 4 & -2 \end{pmatrix}$, $B = \begin{pmatrix} 1 & -3 \\ 2 & 1 \end{pmatrix}$, $C = \begin{pmatrix} 5 & 4 \\ -2 & 8 \end{pmatrix}$

Výsledky B

1 a) regulární, $A^{-1} = \frac{1}{7}\begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix}$, b) singulární, c) ani regulární ani singulární,

d) ani regulární ani singulární, e) regulární, $A^{-1} = \frac{1}{2}\begin{pmatrix} 1 & 5 & 1 \\ -1 & -9 & -3 \\ 1 & 3 & 1 \end{pmatrix}$, f) singulární,

g) regulární, $A^{-1} = \frac{1}{3}\begin{pmatrix} 0 & -1 & 3 & -5 \\ -3 & 1 & 0 & 2 \\ -15 & 3 & 3 & 0 \\ 6 & 0 & -3 & 3 \end{pmatrix}$, h) regulární, $A^{-1} = \begin{pmatrix} -2 & 4 & -1 \\ 1 & -3 & 1 \\ 3 & -4 & 1 \end{pmatrix}$

2 $A^{-1} = \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix}, B^{-1} = \begin{pmatrix} 0 & -\frac{1}{3} \\ 1 & \frac{2}{3} \end{pmatrix}$, $AB = \begin{pmatrix} 8 & 1 \\ 13 & 2 \end{pmatrix}, (AB)^{-1} = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{13}{3} & \frac{8}{3} \end{pmatrix} = B^{-1}A^{-1}$

3 $AX = \begin{pmatrix} -3 & 1 \\ 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} 6 & -7 \\ 13 & -11 \end{pmatrix} = \begin{pmatrix} -5 & 10 \\ -20 & 15 \end{pmatrix} = B.$

4 $X = \begin{pmatrix} 6 & -7 \\ 13 & -11 \end{pmatrix}$

5 $X = \begin{pmatrix} 0 & -5 \\ 5 & -5 \end{pmatrix}$

6 $X = \begin{pmatrix} 0 & -5 \\ 5 & -5 \end{pmatrix}$

7 a) $X = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & 1 \end{pmatrix}$, b) $X = \begin{pmatrix} 2 & 2 \\ -3 & -2 \end{pmatrix}$,

c) matice A není regulární (lze ukázat, že rovnice nemá řešení),

d) matice A není regulární (čtenáři si mohou dosazením do rovnice ověřit, že rovnici vyhovuje libovolná matice tvaru $X = \begin{pmatrix} 3-2s & 5-2t \\ s & t \end{pmatrix}, t, s \in \mathbb{R}$)

8 a) $X = \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix}$,

b) $X = \begin{pmatrix} 1-2s & s \\ 2-2t & t \end{pmatrix}, t, s \in \mathbb{R}$,

c) matice A není regulární (lze ukázat, že rovnice nemá řešení),

d) matice A není regulární (lze ukázat, že rovnice nemá řešení)

9 a) $x_0 = 8$, $x_1 = 3$, b) $x_0 = \frac{3}{2}$, $x_1 = -\frac{7}{2}$, $x_2 = \frac{1}{2}$, c) $x_0 = 10$, $x_1 = -4$, $x_2 = -4$, $x_3 = -5$

10 a) $X = (A-B)^{-1}A = \begin{pmatrix} \frac{4}{3} & \frac{2}{3} \\ -\frac{14}{3} & -\frac{7}{3} \end{pmatrix}$, b) $X = B(A+B-2I)^{-1} = \begin{pmatrix} 2 & \frac{1}{5} \\ -1 & \frac{3}{5} \end{pmatrix}$,

c) $X = 10(3A-2I)^{-1} = \begin{pmatrix} -2 & -3 \\ 0 & -5 \end{pmatrix}$, d) $X = C(A-2B)^{-1} = \begin{pmatrix} 5 & 9 \\ -2 & -6 \end{pmatrix}$

10. INVERZNÍ MATICE Z POHLEDU INFORMATIKA

10.1 Inverzní matice

Pro čtvercovou matici A, která je regulární (viz Pojem 9.1), existuje dle Pravidla 9.4 inverzní matice A^{-1} (viz Pojem 9.2). Výpočet inverzní matice budeme provádět modifikovanou verzí Jordanovy eliminační metody (viz kapitola 9.2). V průběhu výpočtu budeme potřebovat dvě pomocné funkce: první funkce bude sloužit ke konkatenaci dvou matic, druhou funkci použijeme na získání podmatice (viz Pojem 10.1) ze zadané matice.

Zdrojový kód 10.1 (regulární matice)

```
00 int matice_je_regularni(double ** A, int m, int n)
01 {
02   double ** B;
03   int hodnost;
04   if (n!=m) chyba("Matice musi byt ctvercova.");
05   B = kopie_matice(A,m,n);
06   odstupnuj_matici(B,m,n);
07   hodnost = hodnost_odstupnovane_matice(B,m,n);
08   smaz_matici(B,m,n);
09   return hodnost == m;
10 }
```

Ve Zdrojovém kódu 10.1 je implementována funkce `matice_je_regularni`, která zjišťuje, zda zadaná matice `A` je regulární. Pojem regulární matice je definován pouze pro čtvercové matice. Pokud matice `A` není čtvercová, je vyvolána chyba (řádek `04`). Potřebujeme zjistit hodnost matice `A` tak, abychom při tomto zjišťování hodnosti matici `A` nezměnili. Do proměnné `B` vytvoříme kopii matice `A` (řádek `05`). Okopírovanou matici `B` odstupňujeme (řádek `06`) a zjistíme její hodnost (řádek `07`). Následně matici `B` smažeme (řádek `08`). Výsledkem funkce je porovnání hodnosti matice `hodnost` a řádu matice `n` (řádek `09`). Matice `A` je regulární, pokud se tyto dvě hodnoty rovnají.

Zdrojový kód 10.2 (horizontální konkatenace matic)

```
00 double ** horizontalni_konkatenace_matic(double ** A, int m_A, int n_A, double ** B,
int m_B, int n_B)
01 {
02   double ** C;
03   int i, j;
04   if (m_A != m_B) chyba("Matice museji mit shodne pocty radku.");
05   C = vytvor_matici(m_A,n_A+n_B);
06   for (i=0; i<m_A; i++)
07     for (j=0; j<n_A+n_B; j++)
08       C[i][j] = (j<n_A) ? A[i][j] : B[i][j-n_A];
09   return C;
10 }
```

Při výpočtu inverzní matice A^{-1} pomocí modifikované verze Jordanovy eliminační metody musíme nejprve k původní matici A přidat zprava jednotkovou matici I, získáme matici $(A|I)$. Ve Zdrojovém kódu 10.2 je implementována funkce `horizontalni_konkatenace_matic`, která se k tomuto účelu dá využít. Funkce k první zadané matici `A` připojí zprava druhou zadanou matici `B`. Spojením matic `A` a `B` vznikne matice `C`. Matice `A` a `B` musejí mít shodný počet řádků, jinak je vyvolána chyba (řádek `04`). Vytvoříme prostor pro matici `C` (řádek `05`). Počet sloupců matice `C` je dán součtem počtu sloupců matic `A` a `B`, počet řádků matice `C` je roven počtu řádků matic `A` i `B`. Ve dvou zanořených cyklech (řádky `06` a `07`) procházíme jednotlivé prvky matice `C` a kopírujeme do nich hodnoty z matic `A` a `B` (řádek `08`). Podle aktuálně zpracovávaného sloupce `j` matice `C` rozhodneme, zda do prvku matice `C` zkopírujeme prvek matice `A` nebo `B`. Prvky matice `C`, které odpovídají prvkům matice `A`, mají s těmi prvky i shodný index (tj. `C[i][j]=A[i][j]`). Prvky matice `C`, které odpovídají prvkům matice `B`, mají index posunutý o počet

sloupců `n_A` matice `A` (tj. `C[i][j]=B[i][j-n_A]`). Funkce vrací jako návratovou hodnotu matici `C` (řádek 09).

Poznámka (ternární operátor)

Ve Zdrojovém kódu 10.2 na řádku 08 je použit ternární operátor `? :`. V některých programovacích jazycích tento operátor neexistuje a musí být nahrazen ekvivalentním zápisem. Příkaz tvořený ternárním operátorem `x = podmínka ? hodnota1 : hodnota 2;` lze rozepsat jako podmíněný příkaz `if (podmínka) x = hodnota1; else x = hodnota2;`.

Pojem 10.1 (podmatice)

Matice B je podmaticí matice A, pokud existují taková čísla x a y, že hodnota každého prvku matice B s indexem (i, j) je rovna hodnotě prvku matice A s indexem $(i + x, j + y)$. Neformálně: Matice B je podmaticí matice A, pokud B tvoří souvislou část matice A.

Na závěr výpočtu inverzní matice A^{-1} pomocí modifikované verze Jordanovy eliminační metody budeme muset z matice $(I|A^{-1})$ získat její pravou část A^{-1}. Ve Zdrojovém kódu 10.3 je implementována funkce `podmatice`, která se k tomuto účelu dá využít. Funkce ze zadané matice `A` (například z matice $(I|A^{-1})$) vytvoří podmatici `B` (například podmatici A^{-1}) určenou indexy levého horního rohu (`m_od`, `n_od`) a pravého dolního rohu (`m_do`, `n_do`).

Pokud se indexy levého horního rohu a pravého dolního rohu požadované podmatice `B` nenacházejí v matici `A` (řádek 04), je vyvolána chyba (řádek 05). Požadovaná podmatice `B` musí mít alespoň jeden sloupec a jeden řádek (řádek 06), jinak je vyvolána chyba (řádek 07). Vytvoříme prostor pro matici `B` (řádek 08). Ve dvou zanořených cyklech (řádky 09 a 10) procházíme jednotlivé prvky matice `B` a kopírujeme do nich odpovídající hodnoty (dle Pojmu 10.1) z matice `A` (řádek 11). Funkce vrací jako návratovou hodnotu matici `B` (řádek 12).

Zdrojový kód 10.3 (podmatice)

```
00 double ** podmatice(double ** A, int m_A, int n_A, int m_od, int n_od, int m_do, int n_do)
01 {
02   double ** B;
03   int i, j;
04   if ((m_od > m_A) || (n_od > n_A) || (m_do > m_A) || (n_do > n_A))
05     chyba("Podmatici nelze vytvorit.");
06   if ((m_od >= m_do) || (n_od >= n_do))
07     chyba("Podmatici nelze vytvorit.");
08   B = vytvor_matici(m_do-m_od,n_do-n_od);
09   for (i=m_od; i<m_do; i++)
10     for (j=n_od; j<n_do; j++)
11        B[i-m_od][j-n_od] = A[i][j];
12   return B;
13 }
```

Nyní máme k dispozici všechny pomocné funkce, které budeme potřebovat při samotném výpočtu inverzní matice A^{-1} pomocí modifikované verze Jordanovy eliminační metody. Ve Zdrojovém kódu 10.4 je implementována funkce `inverzni_matice`, která pro vstupní čtvercovou matici `A` vypočítá matici k ní inverzní. Nejprve vytvoříme jednotkovou matici `I` (řádek 04). Pokud matice `A` nebyla čtvercová, dojde při vytváření jednotkové matice k chybě. Z tohoto důvodu nemusíme v rámci naší funkce testovat, zda je matice čtvercová. Matice `AI` vznikne přidáním matice `I` zprava k matici `A` (řádek 05). Odstupňujeme matici `AI` (řádek 06) a zjistíme její hodnost (řádek 07). Pokud se hodnost matice `AI` liší od počtu řádků této matice, nebyla původní matice `A` regulární, a proto je vyvolána chyba (řádek 08).

V cyklu (řádek 09) procházíme jednotlivé řádky matici `AI` směrem zdola nahoru. V těle cyklu budeme provádět převod na redukovaný odstupňovaný tvar (viz Pojem 7.8). Oproti obecnému postupu uvedeném ve Zdrojovém kódu 8.4 dochází ke dvěma změnám. Na pravé straně upravujeme celou matici místo jednoho vektoru, což postup komplikuje. Původní matice `A` je regulární, což může vést ke zjednodušení postupu (viz úkol č. 7 v kapitole 10.1).

V každé iteraci cyklu nejprve zjistíme sloupec, ve kterém se nachází pivot (řádek 11). Z předchozích

iterací cyklu máme zajištěno, že napravo na řádku s pivotem v části matice `AI` odpovídající matici `A` jsou pouze nulové prvky. Pomocí vnořeného cyklu (řádek 12) procházejícího matici po sloupcích vydělíme hodnotou pivota všechny prvky na řádku s tímto pivotem v části matice `AI` odpovídající matici `I` (řádek 13). Do pivota dosadíme hodnotu 1 (řádek 14).

Z hlavního cyklu spustíme opět vnořený cyklus (řádek 15) procházející matici po řádcích od pivota směrem nahoru. Tento cyklus má za úkol vynulovat prvky ve sloupci nad pivotem pomocí odečtení vhodného násobku řádku s pivotem od řádku, jehož prvek nulujeme. Pomocí třetího vnořeného cyklu (řádek 17) procházejícího matici po sloupcích modifikujeme prvky v části matice `AI` odpovídající matici `I` (řádek 18). Po skončení třetího zanořeného cyklu je do prvku nad pivotem v příslušném řádku dosazena hodnota 0 (řádek 19).

Po skončení hlavního cyklu (řádek 21) se v části matice `AI` odpovídající matici `A` nachází jednotková matice a v části matice `AI` odpovídající matici `I` se nachází inverzní matice k původní matici `A`. Inverzní matici `A_inv` k původní matici `A` získáme jako podmatici matice `AI` (řádek 22). Musíme smazat pomocné matice `I` a `AI` (řádky 23 a 24). Návratovou hodnotou funkce je matice `A_inv` (řádek 25).

Zdrojový kód 10.4 (inverzní matice)

```
00 double ** inverzni_matice(double ** A, int m, int n)
01   {
02   double ** I, ** AI, ** A_inv;
03   int i,j, k, pozice, hodnost;
04   I = jednotkova_matice(m,n);
05   AI = horizontalni_konkatenace_matic(A,m,n,I,m,n);
06   odstupnuj_matici(AI,m,2*n);
07   hodnost = hodnost_odstupnovane_matice(AI,m,2*n);
08   if (hodnost != m) chyba("Matice musi byt regularni.");
09   for (i = hodnost-1; i >= 0; i--)
10   {
11     pozice = pocet_nul_z_leva(AI,m,n,i);
12     for (j=n; j<2*n; j++)
13       AI[i][j] = AI[i][j] / AI[i][pozice];
14     AI[i][pozice] = 1;
15     for (k = i-1; k >= 0; k--)
16     {
17       for (j=n; j<2*n; j++)
18         AI[k][j] -= AI[k][pozice] * AI[i][j];
19       AI[k][pozice] = 0;
20     }
21   }
22   A_inv = podmatice(AI,m,2*n,0,n,m,2*n);
23   smaz_matici(I,m,n);
24   smaz_matici(AI,m,2*n);
25   return A_inv;
26 }
```

10.2 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 10.

2 Implementujte funkci zjišťující, zda matice je singulární (viz Pojem 9.1).

3 Na příkladech náhodných matic demonstrujte tvrzení, že součin regulárních matic je regulární matice (viz Pravidlo 9.2).

4 Na příkladech náhodných matic demonstrujte tvrzení, že vynásobením matice regulární maticí příslušného řádu se nezmění její hodnost (viz Pravidlo 9.3).

5 Na příkladech náhodných matic demonstrujte tvrzení, že inverzní matice k regulární matici je také

regulární (viz Pravidlo 9.5).

6 Na příkladech náhodných matic demonstrujte vlastnosti inverzních matic (viz Pravidlo 9.6).

7 Zjednodušte výpočet inverzní matice (Zdrojový kód 10.4) s využitím znalosti, že matice A je regulární.

8 Implementujte funkce na řešení maticových rovnic $AX = B$ a $XA = B$ podle základních postupů uvedených v kapitole 9.3. Na příkladech náhodných matic vyzkoušejte správnost implementovaných funkcí dosazením vypočtených výsledků do původních rovnic. Matice A musí být regulární.

9 Implementujte funkce z úkolu 8 za použití efektivnějších postupů uvedených v kapitole 9.5. Vyzkoušejte správnost těchto funkcí obdobně jako v úkolu 8.

10 Napište program, který bude šifrovat soubory dle postupu uvedeného v Příkladu 9.4. Pro převod znaků souborů na prvky matice A doporučujeme použit jednobytovou znakovou sadu. Matice C je šifrovací klíč, který musí být utajen. Pro testování zvolte matici C řádu 3×3.

11. DETERMINANTY

11.1 Determinant – definice

Determinant čtvercové matice A (ozn. $\det A$) je číslo, přiřazené této matici určitým způsobem; pro matice, které nejsou čtvercové, se determinant nedefinuje. Obvykle uváděná definice determinantu[16] je poněkud složitější, využívá dalších pojmů a většinou nebývá potřeba determinant dle této definice počítat. Uvedeme proto jinou formu definice determinantu – definujeme determinant rekurentně.

Pojem 11.1 (determinant)

Determinant matice A typu 1×1, tj. $A = (a_{00})$ je $\det A = a_{00}$.

Determinant matice $A = (a_{ij})$ typu $n \times n$ $(n \geq 2)$ je číslo:

$$\det A = (-1)^{0+0} a_{00} \det A_{00} + (-1)^{0+1} a_{01} \det A_{01} + ... + (-1)^{0+(n-1)} a_{0,n-1} \det A_{0,n-1},$$

kde $\det A_{0j}$ je determinant matice, která vznikne vynecháním nultého řádku a j-tého sloupce matice A.

Determinant matice budeme značit svislými čarami namísto závorek.

Poznámka (vyznačení nul)

Součty v exponentech $0 + 0$, $0 + 1$ ... jsme napsali s vyznačením nul, aby byla lépe vidět souvislost těchto exponentů s indexy příslušného prvku v matici. Navíc se nám později tento tvar zápisu bude hodit při zobecnění na jiný než nultý řádek (viz Pravidlo 11.3).

Poznámka (sumační symbolika)

Součet $(-1)^{0+0} a_{00} \det A_{00} + (-1)^{0+1} a_{01} \det A_{01} + ... + (-1)^{0+(n-1)} a_{0,n-1} \det A_{0,n-1}$ lze stručněji zapsat s užitím sumační symboliky[17]:

$$\det A = \sum_{j=0}^{n-1} (-1)^{0+j} a_{0j} \det A_{0j}.$$

P ř í k l a d 11.1

Podle předchozí definice vypočítáme determinant matice $A = \begin{pmatrix} 2 & -3 \\ 4 & 5 \end{pmatrix}$.

Vynecháme-li 0. řádek a 0. sloupec matice A, obdržíme matici $A_{00} = (5)$, vynecháme-li 0. řádek a 1. sloupec matice A, obdržíme matici $A_{01} = (4)$ – obě tyto matice jsou v tomto případě typu 1×1:

$$\begin{vmatrix} 2 & -3 \\ 4 & 5 \end{vmatrix} = (-1)^{0+0} \cdot (2) \cdot \det(5) + (-1)^{0+1} \cdot (-3) \cdot \det(4) = 2 \cdot 5 + (-1) \cdot (-3) \cdot 4 = 22.$$

11.2 Výpočet determinantu matice řádu 2 a 3

Determinanty matic až do řádu 3 není nutné počítat podle definice, ale podle následujících pravidel.

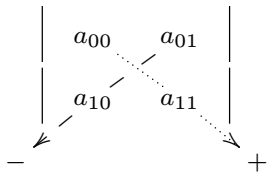
Pravidlo 11.1 *(výpočet determinantu řádu 2)*

Determinant matice řádu 2 lze počítat dle vyznačeného schématu (počítáme součiny prvků spojených šipkou a těmto součinům přidělíme uvedené znaménko):

---

[16] Viz např. v [6].

[17] Symbol $\sum_{i=0}^{n-1}$ (suma od nuly do $n-1$) je symbolem pro součet, kde index i nabývá postupně hodnot od nuly do $n-1$: $\sum_{i=0}^{n-1} a_i = a_0 + a_1 + ... + a_{n-1}$.



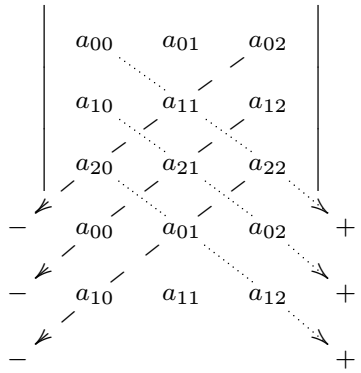
$$= a_{00}a_{11} - a_{01}a_{10}.$$

P ř í k l a d 11.2

$$\begin{vmatrix} 2 & -3 \\ 4 & 5 \end{vmatrix} = 2 \cdot 5 - (-3) \cdot 4 = 22.$$

Pravidlo 11.2 *(výpočet determinantu řádu 3 – Sarrusovo pravidlo)*

Determinant matice řádu 3 lze počítat následovně: pod matici opíšeme její první dva řádky a počítáme dle vyznačeného schématu obdobně jako pro determinant řádu 2:



$$= a_{00}a_{11}a_{22} + a_{10}a_{21}a_{02} + a_{20}a_{01}a_{12} - \\ - a_{02}a_{11}a_{20} - a_{12}a_{21}a_{00} - a_{22}a_{01}a_{10}.$$

Tento způsob výpočtu determinantu se nazývá Sarrusovo pravidlo.

Poznámka (připsání sloupců)

Místo připsání prvních dvou řádků pod matici lze připsat první dva sloupce doprava od matice a postupovat obdobně. Při získání určité zběhlosti je možno počítat bez jakéhokoliv připisování řádků či sloupců.

Poznámka (Sarrusovo pravidlo jen pro matice 3×3)

Důrazně upozorňujeme, že Sarrusovo pravidlo je určeno výhradně pro výpočet determinantů matic řádu 3 – nelze ani v žádné analogické podobě aplikovat na determinanty matic řádu většího než 3.

P ř í k l a d 11.3

$$\begin{vmatrix} 2 & -1 & 4 \\ 3 & 0 & -2 \\ 1 & -4 & -3 \end{vmatrix} \begin{matrix} 2 & -1 & 4 \\ 3 & 0 & -2 \end{matrix} = 2 \cdot 0 \cdot (-3) + 3 \cdot (-4) \cdot 4 + 1 \cdot (-1) \cdot (-2) - \\ -4 \cdot 0 \cdot 1 - (-2) \cdot (-4) \cdot 2 - (-3) \cdot (-1) \cdot 3 = -71.$$

11.3 Výpočet determinantu rozvojem podle řádku nebo sloupce

Zatímco Sarrusovo pravidlo lze použít pouze pro výpočet determinantů matic řádu 3, způsob výpočtu, který uvedeme v této kapitole, je použitelný pro výpočet determinantů matic jakéhokoliv řádu (většího než 1). Pro matice řádu 2 a 3 je ovšem nejrychlejší způsob uvedený v Pravidlech 11.1 a 11.2.

Platí, že v definici determinantu můžeme místo prvního řádku vzít libovolný jiný řádek a výsledek bude stejný:

Pravidlo 11.3 *(rozvoj determinantu podle i-tého řádku matice)*

$$\det A = (-1)^{i+0} a_{i0} \det A_{i0} + (-1)^{i+1} a_{i1} \det A_{i1} + ... + (-1)^{i+(n-1)} a_{i,n-1} \det A_{i,n-1}$$

$$= \sum_{j=0}^{n-1} (-1)^{i+j} a_{ij} \det A_{ij},$$

kde $\det A_{ij}$ *je determinant matice, která vznikne vynecháním i-tého řádku a j-tého sloupce matice A. Předchozí výraz se nazývá rozvoj determinantu podle i-tého řádku matice A.*

Lze dokonce místo řádku matice vzít libovolný sloupec matice a obdobně provést rozvoj podle tohoto sloupce:

Pravidlo 11.4 *(rozvoj determinantu podle j-tého sloupce matice)*

$$\det A = (-1)^{0+j} a_{0j} \det A_{0j} + (-1)^{1+j} a_{1j} \det A_{1j} + ... + (-1)^{(n-1)+j} a_{n-1,j} \det A_{n-1,j} =$$

$$= \sum_{i=0}^{n-1} (-1)^{i+j} a_{ij} \det A_{ij}$$

– *rozvoj determinantu podle j-tého sloupce matice A.*

Pojem 11.2 (algebraický doplněk (kofaktor))

Výraz $(-1)^{i+j} \det A_{ij}$ se nazývá algebraický doplněk (kofaktor) prvku a_{ij}.

P ř í k l a d 11.4

Při "ručním" výpočtu determinantu rozvojem je výhodné vybrat si řádek (nebo sloupec), který obsahuje co nejvíc nul (v rozvoji pak bude méně sčítanců), a podle tohoto řádku (nebo sloupce) provést rozvoj. Následující determinant vypočítáme např. rozvojem podle 1. řádku (raději připomínáme, že při číslování řádků i sloupců začínáme nultým):

$$\det A = \begin{vmatrix} -3 & 2 & 1 & -2 \\ \mathbf{4} & \mathbf{0} & \mathbf{-3} & \mathbf{0} \\ 0 & -1 & 2 & 5 \\ -2 & 3 & 0 & 0 \end{vmatrix} =$$

$$= (-1)^{1+0} a_{10} \det A_{10} + (-1)^{1+1} a_{11} \det A_{11} + (-1)^{1+2} a_{12} \det A_{12} + (-1)^{1+3} a_{13} \det A_{13} =$$

$$= (-1)^1 \cdot \mathbf{4} \cdot \begin{vmatrix} 2 & 1 & -2 \\ -1 & 2 & 5 \\ 3 & 0 & 0 \end{vmatrix} + (-1)^2 \cdot \mathbf{0} \cdot \begin{vmatrix} -3 & 1 & -2 \\ 0 & 2 & 5 \\ -2 & 0 & 0 \end{vmatrix} +$$

$$+ (-1)^3 \cdot (\mathbf{-3}) \cdot \begin{vmatrix} -3 & 2 & -2 \\ 0 & -1 & 5 \\ -2 & 3 & 0 \end{vmatrix} + (-1)^4 \cdot \mathbf{0} \cdot \begin{vmatrix} -3 & 2 & 1 \\ 0 & -1 & 2 \\ -2 & 3 & 0 \end{vmatrix}.$$

Sčítance, kde $a_{ij} = 0$ samozřejmě nemusíme vůbec psát (proto volíme řádek či sloupec obsahující co nejvíc nul) – zde je uvádíme pouze pro zřetelnější výklad. Výrazy $(-1)^1$, $(-1)^2$ atd. vlastně předřazují znaménko součinu, který za nimi následuje: $(-1)^1 \cdot 4 \cdot ... = -4 \cdot ...$, podobně $(-1)^2 \cdot 0 \cdot ... = +0 \cdot ...$ atd. Vzniklé determinanty řádu 3 vypočítáme Sarrusovým pravidlem (nebo dalším rozvojem); řádky pod matice v následujícím výpočtu už neopisujeme, uvádíme pouze výsledky příslušných součinů:

$$\det A = -4(0 + 0 + 15 - (-12) - 0 - 0) + 3(0 + 0 - 20 - (-4) - (-45) - 0) = -21.$$

Poznámka (znaménka v rozvoji)

Znaménka $(-1)^{i+j}$ u prvků a_{ij} není třeba vypočítávat – střídají se pravidelně jako šachovnice (doporučujeme rozmyslet si), v levém horním rohu je vždy $+$:

$$A = \begin{pmatrix} + & - & + & - & \dots \\ - & + & - & + & \dots \\ + & - & + & - & \dots \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \end{pmatrix}.$$

11.4 Výpočet determinantu převodem matice na odstupňovanou

Uvedeme ještě další způsob, jak lze počítat determinanty jakéhokoliv řádu. Ten využívá následující vlastnosti:

Pravidlo 11.5 *(determinant trojúhelníkové matice)*

Determinant trojúhelníkové matice je roven součinu prvků na její diagonále.

Poznámka (determinant horní a dolní trojúhelníkové matice, odstupňované matice)

Připomínáme, že pojmem trojúhelníková matice v této knize běžně rozumíme *horní* trojúhelníkovou matici. Předchozí tvrzení však platí i pro *dolní* trojúhelníkovou matici (viz kapitola 3, Pojem 3.7). Protože každá čtvercová odstupňovaná matice je současně i trojúhelníková (viz kapitola 5.1), platí předchozí tvrzení i pro matici *odstupňovanou*.

P ř í k l a d 11.5

$$\begin{vmatrix} -2 & 3 & 1 \\ 0 & 5 & 7 \\ 0 & 0 & 4 \end{vmatrix} = -2 \cdot 5 \cdot 4 = -40, \quad \begin{vmatrix} 2 & 7 & 4 \\ 0 & 8 & 1 \\ 0 & 0 & 0 \end{vmatrix} = 2 \cdot 8 \cdot 0 = 0, \quad \begin{vmatrix} -7 & 0 & 0 \\ 3 & 2 & 0 \\ -2 & 4 & -1 \end{vmatrix} = -7 \cdot 2 \cdot (-1) = 14,$$

$$\det I = 1.$$

Tvrzení obsažené v Pravidle 11.5 nám umožní použít následující postup výpočtu determinantu: matici, jejíž determinant počítáme, převedeme na odstupňovanou (a tedy trojúhelníkovou) úpravami jako při výpočtu hodnosti matice. Není ovšem možné vynechávat řádky nebo sloupce a je třeba dbát toho, že některé úpravy mění determinant:

Pravidlo 11.6 *(změny determinantu při elementárních úpravách matice)*

(1) Záměnou pořadí dvou řádků matice se změní znaménko determinantu.

(2) Vynásobením nějakého řádku matice nenulovým číslem k se determinant k-krát vynásobí.

(3) Přičtením násobku řádku k jinému řádku se determinant nezmění.

Totéž platí pro sloupce matice.

Poznámka (význam změn determinantu)

Změny uvedené v Pravidle 11.6 pro výpočet determinantu konkrétně znamenají:

(1) Vyměníme-li v matici mezi sebou dva řádky, napíšeme při výpočtu před determinant znaménko minus (tím "kompenzujeme" změnu (1)).

(2) Vynásobíme-li v matici některý řádek číslem k, pak při výpočtu determinant vynásobíme číslem $\frac{1}{k}$ (tím "kompenzujeme" změnu (2)). Vynásobíme-li např. řádek číslem 5, pak determinant vynásobíme číslem $\frac{1}{5}$. Vydělíme-li řádek např. číslem 3 (což odpovídá násobení číslem $\frac{1}{3}$), pak determinant vynásobíme číslem 3 – vlastně číslo 3 z řádku vytkneme před determinant.

(3) Přičtení násobku řádku k jinému (nevynásobenému!) řádku determinant nezmění.

Poznámka (nulový determinant)

Vynechávat řádky matice při výpočtu determinantu nelze (matice musí být čtvercová). V případě, že nastane situace, ve které by při výpočtu hodnosti matice bylo možno řádek vynechat (v matici je např. nějaký řádek násobkem jiného), bude determinant roven nule (rozmyslíme si později – viz Pravidlo 11.7 a Poznámka za ním následující). Obsahuje-li tedy matice nulový řádek nebo řádky, z nichž je jeden násobkem jiného, je determinant matice roven nule (nemusíme dál počítat).

Vše, co bylo řečeno v předchozích Poznámkách, platí i pro sloupce.

Příklad 11.6

$$\begin{vmatrix} 2 & -1 & -4 \\ 4 & -2 & -8 \\ 3 & 5 & 7 \end{vmatrix} = 0, \text{ neboť první řádek je dvojnásobkem nultého.}$$

Poznámka (rovnost determinantů)

Protože determinant je číslo, které úpravami neměníme, píšeme mezi jednotlivými úpravami symbol $=$ (nikoliv $\sim$ jako u počítání hodnosti).

Příklad 11.7

Vypočítáme determinant převodem matice na odstupňovanou. Úpravy provádíme jako při výpočtu hodnosti, hlídáme ale případné změny determinantu. V prvním kroku vyměníme 0. a 1. řádek, znaménko determinantu se tedy změní (viz výše úprava (1)):

$$\det A = \begin{vmatrix} 0 & -5 & 3 & -5 \\ 1 & -3 & 3 & 1 \\ -3 & 7 & -7 & -5 \\ 2 & -3 & 0 & 4 \end{vmatrix} = - \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -5 & 3 & -5 \\ -3 & 7 & -7 & -5 \\ 2 & -3 & 0 & 4 \end{vmatrix}.$$

Dále postupujeme, jak je naznačeno vedle matice: trojnásobek 0. řádku přičteme ke 2. řádku a minus dvojnásobek 0. řádku přičteme ke 3. řádku – determinant se těmito úpravami nezmění (viz úprava (3)):

$$\begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -5 & 3 & -5 \\ -3 & 7 & -7 & -5 \\ 2 & -3 & 0 & 4 \end{vmatrix} \begin{matrix} \\ \\ {\scriptstyle 3} \\ \\ \end{matrix} \begin{matrix} \\ \\ {\scriptstyle -2} \\ \\ \end{matrix} = - \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -5 & 3 & -5 \\ 0 & -2 & 2 & -2 \\ 0 & 3 & -6 & 2 \end{vmatrix}.$$

Vidíme, že 2. řádek je dělitelný dvěma. Provedeme-li tuto úpravu, je to totéž, jako bychom násobili řádek $\frac{1}{2}$ – determinant se tedy touto úpravou také vynásobí $\frac{1}{2}$ (viz úprava (2)) – abychom neporušili rovnost, musíme tuto změnu "kompenzovat" vynásobením determinantu dvěma. Poté vyměníme 1. a 2. řádek, aby na diagonále byla –1 (pro snadnější "ruční" počítání); znaménko determinantu se tedy opět změní. Další naznačené úpravy (přičtení násobků 1. řádku) determinant nezmění:

$$-\begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -5 & 3 & -5 \\ 0 & -2 & 2 & -2 \\ 0 & 3 & -6 & 2 \end{vmatrix} = -2 \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -5 & 3 & -5 \\ 0 & -1 & 1 & -1 \\ 0 & 3 & -6 & 2 \end{vmatrix} = +2 \cdot \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & -5 & 3 & -5 \\ 0 & 3 & -6 & 2 \end{vmatrix} \begin{matrix} \\ \\ {\scriptstyle -5} \\ \\ \end{matrix} \begin{matrix} \\ \\ {\scriptstyle 3} \\ \\ \end{matrix} = 2 \cdot \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & -3 & -1 \end{vmatrix}.$$

Abychom dostali nulu na místo prvku a_{32}, přičteme minus trojnásobek 2. řádku k dvojnásobku 3. řádku, tak jak je naznačeno dále. Determinant se v tomto případě změní! Přičtení násobku 2. řádku sice determinant nemění, ale násobíme upravovaný 3. řádek (tj. řádek, *ke kterému* přičítáme) dvěma (viz úprava (2)), determinant se tedy také dvakrát zvětší – musíme ho tudíž dvěma vydělit:

$$2 \cdot \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & -3 & -1 \end{vmatrix} \begin{matrix} \\ \\ \\ {\scriptstyle -3} \\ {\scriptstyle 2} \end{matrix} = 2 \cdot \frac{1}{2} \cdot \begin{vmatrix} 1 & -3 & 3 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{vmatrix}.$$

Matice je trojúhelníková – její determinant je tedy roven součinu prvků na diagonále:

$$\det A = 2 \cdot \frac{1}{2} \cdot 1 \cdot (-1) \cdot (-2) \cdot (-2) = -4.$$

Poznámka (upozornění)

Je nutno opravdu pečlivě hlídat případné změny determinantu. Jsme zvyklí při výpočtu hodnosti matice, že je jedno, zda odčítáme např. 1. řádek od druhého nebo naopak, že můžeme libovolně měnit pořadí řádků, změnit znaménko u prvků v celém řádku apod. – při výpočtu determinantu se pak snadno můžeme dopustit chyby; např. (odečtení 1. řádku od druhého a naopak):

$$\begin{vmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ . & . & . & . \\ . & . & . & . \end{vmatrix} \overset{-1}{\leftarrow\!\lrcorner} = \begin{vmatrix} 1 & 2 & 4 & 6 \\ 0 & 3 & 3 & 3 \\ . & . & . & . \\ . & . & . & . \end{vmatrix},$$

ale

$$\begin{vmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ . & . & . & . \\ . & . & . & . \end{vmatrix} \overset{}{\lrcorner\!\rightarrow -1} = - \begin{vmatrix} 1 & 2 & 3 & 4 \\ 0 & -3 & -3 & -3 \\ . & . & . & . \\ . & . & . & . \end{vmatrix}.$$

Také je dobré si uvědomit rozdíl při počítání s maticemi a s determinanty – např. před matici lze vytknout číslo vydělením všech prvků matice:

$$\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix},$$

ale při výpočtu determinantu se vytýká z každého řádku zvlášť:

$$\begin{vmatrix} 2 & 4 \\ 6 & 8 \end{vmatrix} = 2 \cdot \begin{vmatrix} 1 & 2 \\ 6 & 8 \end{vmatrix} = 2 \cdot 2 \cdot \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = 4 \cdot \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}.$$

Poznámka (kombinace úprav a rozvoje)

Při výpočtu determinantů matic vyšších řádů bývá výhodné kombinovat úpravy matice s rozvojem – nejprve provést úpravy tak, abychom v nějakém řádku (či sloupci) obdrželi dostatečný počet nul, pak podle tohoto řádku (či sloupce) provést rozvoj.

11.5 Determinant regulární matice. Vlastnosti determinantu

Podívejme se nyní na velmi důležitý vztah regularity matice s hodnotou determinantu. Víme, že determinant trojúhelníkové matice je roven součinu prvků na diagonále. Regulární trojúhelníková matice má na diagonále všechny prvky nenulové, její determinant je tedy nenulový. Singulární trojúhelníková matice má aspoň jeden (poslední) řádek nulový – na diagonále je tedy nula a determinant této matice je nulový. Při převodu matice na odstupňovanou (trojúhelníkovou) se elementárními úpravami sice může měnit determinant, ale ne jeho nulovost či nenulovost. Je tedy determinant každé regulární matice nenulový a každé singulární matice nulový.

Pravidlo 11.7 *(determinant regulární a singulární matice)*

A je regulární $\iff \det A \neq 0$ *(a tedy A je singulární* $\iff \det A = 0$*).*

Poznámka (determinant singulární matice)

Pokud si při výpočtu determinantu všimneme, že je v matici jeden řádek násobkem jiného řádku, je matice singulární a determinant je tedy roven nule – není třeba dál počítat. Stejně tak pro sloupce matice.

Poznatky z této kapitoly a kapitoly 9 týkající se regularity matic shrneme do důležitého tvrzení:

Pravidlo 11.8 *(vztah regularity, determinantu, řešitelnosti soustavy)*

Je-li A čtvercová matice, pak platí

A je regulární $\iff \det A \neq 0 \iff A^{-1}$ *existuje* $\iff$ *soustava $A\bar{x} = \bar{b}$ má právě jedno řešení pro libovolnou pravou stranu $\bar{b}$* $\iff$ *homogenní soustava $A\bar{x} = \bar{o}$ má pouze triviální řešení*

neboli

A je singulární $\iff \det A = 0 \iff A^{-1}$ neexistuje $\iff$ soustava $A\bar{x} = \bar{b}$ má nekonečně mnoho řešení nebo nemá řešení $\iff$ homogenní soustava $A\bar{x} = \bar{o}$ má i netriviální řešení

Zdůrazňujeme, jsou kterékoliv dva z výroků ekvivalentní, takže např. platí:

$$\text{homogenní soustava } A\bar{x} = \bar{o} \text{ má netriviální řešení} \iff \det A = 0.$$

Znalosti těchto souvislostí mohou pak usnadnit řešení některých úloh.

P ř í k l a d 11.8

Zjistíme, zda následující soustava má netriviální řešení.

$$\begin{array}{rcrcrcl} 2x_0 & - & x_1 & + & 3x_2 & = & 0 \\ x_0 & + & 4x_1 & - & x_2 & = & 0 \\ -3x_0 & + & x_1 & - & 2x_2 & = & 0 \end{array}$$

Vypočítáme determinant matice soustavy:

$$\begin{vmatrix} 2 & -1 & 3 \\ 1 & 4 & -1 \\ -3 & 1 & -2 \end{vmatrix} = -16 + 3 - 3 + 36 + 2 - 2 = 20 \neq 0.$$

Protože je determinant matice soustavy nenulový, má soustava jediné řešení, a to triviální: $\bar{x} = (0, 0, 0)$. Netriviální řešení tedy nemá.

Poznamenejme ale ještě, že kdyby determinant vyšel nulový, věděli bychom sice, že soustava má netriviální řešení, k jeho nalezení bychom však determinantu nijak nemohli využít.

Uvedeme zde ještě pro informaci některé důležité vlastnosti determinantů:

Pravidlo 11.9 *(vlastnosti determinantů)*

Jsou-li A, B matice typu $n \times n$, pak platí

(1) $\det(k \cdot A) = k^n \cdot \det A$,

(2) $\det(A \cdot B) = \det A \cdot \det B$,

(3) $\det A = \det A^T$,

(4) je-li A regulární, pak $\det A^{-1} = \frac{1}{\det A}$.

Uvědomme si, že vlastnost (1) jsme si už rozmysleli dříve – je-li každý řádek matice dělitelný číslem k, vytýkáme při výpočtu determinantu toto číslo z každého řádku, což odpovídá vzorci uvedenému v (1).

11.6 Cramerovo pravidlo pro řešení soustav lineárních rovnic

V této kapitole uvedeme další metodu (Cramerovo pravidlo), jak řešit soustavy lineárních rovnic. Jak ale uvidíme dál, tuto metodu lze použít jen v případě, že soustava má jediné řešení (stejně jako metodu využívající inverzní matici) – metoda je tedy použitelná jen pro soustavy s regulární maticí. Metoda využívá determinantů a její výhodou je, že dokáže vypočítat jednotlivou neznámou ze soustavy, aniž by bylo potřeba počítat ostatní neznámé.

Pravidlo 11.10 *(Cramerovo pravidlo)*

Předpokládejme, že matice soustavy $A\bar{x} = \bar{b}$ je regulární. Označme A_i matici, která vznikne z matice A (tj. matice soustavy) nahrazením i-tého sloupce vektorem $\bar{b}$ (tj. sloupcem pravých stran). Pak lze i-tou neznámou x_i vypočítat podle vzorce:

$$x_i = \frac{\det A_i}{\det A}.$$

Poznámka (náhrada sloupce)

V matici soustavy tedy nahradíme sloupcem pravých stran ten sloupec, který odpovídá neznámé, kterou chceme vypočítat (dostaneme tak matici A_i). Neznámou pak získáme jako podíl determinantu této matice (čitatel) a matice soustavy (jmenovatel).

Poznámka (použitelnost Cramerova pravidla)

Všimněme si, že Cramerovo pravidlo je použitelné skutečně jen pro soustavy s regulární maticí, neboť jmenovatel výše uvedeného vzorce je nenulový právě jen pro regulární matici.

Pro homogenní soustavy není Cramerovo pravidlo vhodné: je-li matice soustavy regulární, má homogenní soustava pouze triviální řešení a to by bylo zbytečné získávat výpočtem determinantů.

Řešit celou soustavu Cramerovým pravidlem je pracné, avšak Cramerovo pravidlo umožňuje vypočítat třeba jen jednu neznámou ze soustavy:

P ř í k l a d 11.9

Cramerovým pravidlem vypočítáme x_1 ze soustavy

$$\begin{array}{rcrcrcl} 2x_0 & - & x_1 & & + & 3x_3 & = 0 \\ x_0 & & & - & 3x_2 & & = 1 \\ -x_0 & - & 2x_1 & & + & x_3 & = 3 \\ & & 3x_1 & + & 4x_2 & - & x_3 = 0. \end{array}$$

Nejprve vypočítáme determinant matice soustavy – kdyby byl nulový, nemělo by smysl počítat $\det A_1$ – soustava by nebyla řešitelná Cramerovým pravidlem (to ovšem nutně neznamená, že by soustava neměla řešení – mohla by jich mít nekonečně mnoho). Determinant vypočítáme např. rozvojem podle 2. sloupce:

$$\det A = \begin{vmatrix} 2 & -1 & 0 & 3 \\ 1 & 0 & -3 & 0 \\ -1 & -2 & 0 & 1 \\ 0 & 3 & 4 & -1 \end{vmatrix} = 3 \cdot \begin{vmatrix} 2 & -1 & 3 \\ -1 & -2 & 1 \\ 0 & 3 & -1 \end{vmatrix} - 4 \cdot \begin{vmatrix} 2 & -1 & 3 \\ 1 & 0 & 0 \\ -1 & -2 & 1 \end{vmatrix} = 3(4-9-6+1) - 4(-6+1) = -10.$$

Determinant matice A_1, která vznikne nahrazením 1. sloupce matice soustavy sloupcem pravých stran, vypočítáme např. také rozvojem podle 2. sloupce:

$$\det A_1 = \begin{vmatrix} 2 & 0 & 0 & 3 \\ 1 & 1 & -3 & 0 \\ -1 & 3 & 0 & 1 \\ 0 & 0 & 4 & -1 \end{vmatrix} = 3 \cdot \begin{vmatrix} 2 & 0 & 3 \\ -1 & 3 & 1 \\ 0 & 0 & -1 \end{vmatrix} - 4 \cdot \begin{vmatrix} 2 & 0 & 3 \\ 1 & 1 & 0 \\ -1 & 3 & 1 \end{vmatrix} = 3(-6) - 4(2+9+3) =$$

$= -18 - 56 = -74.$

Je tedy

$$x_1 = \frac{\det A_1}{\det A} = \frac{-74}{-10} = \frac{37}{5}.$$

Poznámka (výhody Cramerova pravidla)

Jak už jsme se zmínili, výhodou Cramerova pravidla je možnost vypočítat jedinou neznámou. Všimněme si navíc, že pokud jsou všechny prvky matice soustavy celočíselné, není potřeba při výpočtu neznámé Cramerovým pravidlem počítat se zlomky, i když neznámá není celočíselná (zlomek dostaneme až v závěru jako podíl vypočítaných celočíselných determinantů). To je výhoda oproti např. Gaussově metodě.

11.7 Užití determinantu pro výpočet inverzní matice

Determinantu lze využít i k výpočtu inverzní matice. Výpočet je ale poměrně pracný – uvedeme jej zde spíše informativně. V popisu metody užijeme pojem algebraický doplněk prvku a_{ij} (viz Pojem 11.2), tzn. výraz

$$(-1)^{i+j} \det A_{ij},$$

kde A_{ij} je matice, která vznikla z matice A vynecháním i-tého řádku a j-tého sloupce.

Pravidlo 11.11 *(výpočet A^{-1} pomocí determinantu)*

Je-li A regulární matice řádu n, pak

$$A^{-1} = \frac{1}{detA} \begin{pmatrix} b_{00} & b_{01} & \dots & b_{0,n-1} \\ b_{10} & b_{11} & \dots & b_{1,n-1} \\ \dots & \dots & \dots & \dots \\ b_{n-1,0} & b_{n-1,1} & \dots & b_{n-1,n-1} \end{pmatrix}^T,$$

kde b_{ij} je algebraický doplněk prvku a_{ij}.

Příklad 11.10

Užitím determinantu vypočítáme inverzní matici k matici (viz Příklad 9.2 v kapitole 9 – můžeme porovnat pracnost při "ručním" počítání):

$$A = \begin{pmatrix} 1 & 2 & -3 \\ -1 & -1 & 7 \\ 0 & -1 & -2 \end{pmatrix}.$$

Vypočítáme nejdřív determinant matice A:

$$\det A = \begin{vmatrix} 1 & 2 & -3 \\ -1 & -1 & 7 \\ 0 & -1 & -2 \end{vmatrix} = 2 - 3 - 4 + 7 = 2.$$

Dále vypočítáme algebraické doplňky b_{ij} všech prvků a_{ij}:

$$b_{00} = (-1)^{0+0} \begin{vmatrix} -1 & 7 \\ -1 & -2 \end{vmatrix} = 9 \quad b_{01} = (-1)^{0+1} \begin{vmatrix} -1 & 7 \\ 0 & -2 \end{vmatrix} = -2 \quad b_{02} = (-1)^{0+2} \begin{vmatrix} -1 & -1 \\ 0 & -1 \end{vmatrix} = 1$$

$$b_{10} = (-1)^{1+0} \begin{vmatrix} 2 & -3 \\ -1 & -2 \end{vmatrix} = 7 \qquad b_{11} = (-1)^{1+1} \begin{vmatrix} 1 & -3 \\ 0 & -2 \end{vmatrix} = -2 \qquad b_{12} = (-1)^{1+2} \begin{vmatrix} 1 & 2 \\ 0 & -1 \end{vmatrix} = 1$$

$$b_{20} = (-1)^{2+0} \begin{vmatrix} 2 & -3 \\ -1 & 7 \end{vmatrix} = 11 \quad b_{21} = (-1)^{2+1} \begin{vmatrix} 1 & -3 \\ -1 & 7 \end{vmatrix} = -4 \quad b_{22} = (-1)^{2+2} \begin{vmatrix} 1 & 2 \\ -1 & -1 \end{vmatrix} = 1.$$

Odtud

$$A^{-1} = \frac{1}{2} \begin{pmatrix} 9 & -2 & 1 \\ 7 & -2 & 1 \\ 11 & -4 & 1 \end{pmatrix}^T = \frac{1}{2} \begin{pmatrix} 9 & 7 & 11 \\ -2 & -2 & -4 \\ 1 & 1 & 1 \end{pmatrix}.$$

Poznámka (determinant, singularita a inverzní matice)

Připomínáme, že pokud by vyšel determinant matice A roven nule, znamenalo by to, že matice A je singulární, a že k ní tudíž neexistuje matice inverzní.

Poznámka (inverzní matice k matici řádu 2)

Výše uvedeným postupem lze rychle vypočítat inverzní matici řádu 2 – viz následující příklad.

Příklad 11.11

Užitím determinantu vypočítáme inverzní matici k matici

$$A = \begin{pmatrix} 5 & -2 \\ -3 & 4 \end{pmatrix}.$$

Vypočítáme

$$\det A = \begin{vmatrix} 5 & -2 \\ -3 & 4 \end{vmatrix} = 14,$$

$$b_{00} = (-1)^{0+0} \cdot 4 = 4, \quad b_{01} = (-1)^{0+1} \cdot (-3) = 3, \quad b_{10} = (-1)^{1+0} \cdot (-2) = 2, \quad b_{11} = (-1)^{1+1} \cdot 5 = 5,$$

odtud

$$A^{-1} = \frac{1}{14} \begin{pmatrix} 4 & 2 \\ 3 & 5 \end{pmatrix}.$$

Poznámka (tvar inverzní matice řádu 2)

Na základě předchozího příkladu si lze snadno rozmyslet, že k matici řádu 2 tvaru

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

je inverzní matice tvaru

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

11.8 Užití determinantů

Už víme, že determinanty lze využít pro zjištění regularity matice (a řešení úloh s tím spojených), pro řešení soustav Cramerovým pravidlem a pro výpočet inverzní matice. Determinanty mají však mnohem širší využití – v oblastech, které přesahují rámec této knihy (např. výpočet tzv. vlastních čísel matice[18], v oblasti matematické analýzy při výpočtu extrémů funkcí více proměnných[19] atd.).

Jako ukázku možného využití determinantů zde uvedeme příklad, který sice vyžaduje znalost základních pojmů z teorie grafů, které jsou však na potřebné úrovni snadno zvládnutelné. Pro stručné uvedení do problematiky zde tyto pojmy volně nastíníme[20].

Mnohé situace z praktického života (např. problém dopravního spojení mezi městy) lze popsat zjednodušeným modelem – tzv. *grafem*. Ten sestává z bodů, tzv. *vrcholů* neboli *uzlů* (reprezentujících např. města), a z úseček tyto body spojujících, tzv. *hran* (reprezentujících např. cesty mezi městy). Graf může být *neorientovaný* (viz obr. 11.1) – pokud není u hran dán směr, ze kterého vrcholu vycházejí a do kterého vedou (např. obousměrné silnice mezi městy), nebo *orientovaný* (viz obr. 11.2) – v případě, že směr dán je (např. jednosměrné silnice mezi městy). Vrcholy můžeme označit např. čísly, hrany mezi vrcholy pak dvojicemi příslušných čísel, přičemž tyto dvojice jsou uspořádané v případě orientovaného grafu a neuspořádané v případě neorientovaného grafu. Např. uspořádaná dvojice [0, 2] označuje hranu vedoucí z vrcholu 0 do vrcholu 2, zatímco dvojice {0, 2} označuje prostě hranu, která spojuje vrcholy 0 a 2; je tedy {0, 2} = {2, 0}.

V grafu může být obsažena tzv. *kružnice* neboli *cyklus*, tzn. posloupnost vrcholů a hran taková, že "spojuje přes další vrcholy vrchol sám se sebou" – např. trojúhelník o vrcholech 0, 1, 2 propojených hranami {0, 1}, {1, 2}, {2, 0}.

Souvislý graf, který neobsahuje žádnou kružnici, se nazývá *strom*. Je-li ve stromu vrchol, ze kterého postupně vycházejí ("větví se") všechny hrany (tzn. je dána orientace vedoucí od tohoto vrcholu dál) – tak jako od výchozích rodičů v rodokmenu – nazývá se tento vrchol *kořen stromu* a strom se pak nazývá *kořenový* nebo *zakořeněný*.

Libovolný podgraf daného grafu, který hranami spojuje všechny vrcholy daného grafu a který neobsahuje žádnou kružnici (tzn. je stromem), se nazývá *kostra grafu* (reprezentuje např. způsob, jak propojit všechna města, aby přitom mezi dvěma městy nebylo více cest). Např. v případě grafu o vrcholech 0, 1, 2 a hranách {0, 1}, {1, 2}, {2, 0} jsou kostrami podgrafy o vrcholech 0, 1, 2 a hranách {0, 1}, {1, 2}, resp. {0, 1}, {0, 2}, resp. {0, 2}, {1, 2}.

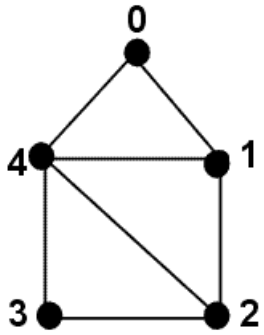
Jak dále uvidíme, lze počet koster grafu, resp. počet kořenových stromů grafu, určovat pomocí determinantů.

---

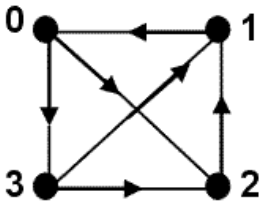
[18] Viz např. v [18].

[19] Viz např. v [1].

[20] Pro podrobnější seznámení odkazujeme např. na [17].



Obr. 11.1. Neorientovaný graf



Obr. 11.2. Orientovaný graf

a) Uvažujme neorientovaný graf o vrcholech 0, 1, 2, 3, 4 s množinou hran

$$E = \{\{0,1\}, \{0,4\}, \{1,2\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}\},$$

kde $\{i,j\}$ značí hranu spojující vrcholy i a j, přičemž $\{i,j\} = \{j,i\}$ (viz obr. 11.1). Představme si např., že graf představuje 5 měst a cesty mezi nimi. Naším úkolem bude nalézt všechny možnosti, jak se z libovolného města dostat do kteréhokoliv jiného právě jedním způsobem. Jedná se tedy o úlohu nalezení všech koster daného grafu.

Graf lze charakterizovat maticí A typu 5×5 definovanou následovně: diagonální prvky a_{ii} budou pro každé $i = 0, ..., 4$ rovny počtu hran obsahujících vrchol i, mimodiagonální prvky a_{ij} $(i \neq j)$ budou rovny -1, pokud je mezi vrcholy i, j hrana (tj. $\{i,j\} \in E$), a rovny 0 v opačném případě. Protože se jedná o neorientovaný graf (a tedy $\{i,j\} = \{j,i\}$), bude matice A symetrická:

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{pmatrix}.$$

Lze ukázat[21], že počet neidentických koster grafu v neorientovaném grafu je roven hodnotě algebraického doplňku libovolného prvku matice A.

V případě uvažovaného grafu zjistíme počet koster grafu výpočtem algebraického doplňku např. prvku a_{44}:

$$(-1)^8 \begin{vmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 2 \end{vmatrix} = 2 \begin{vmatrix} 3 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix} + \begin{vmatrix} -1 & 0 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 21.$$

V daném grafu lze najít 21 koster (viz obr. 11.3).

b) Uvažujme orientovaný graf o vrcholech 0, 1, 2, 3 s množinou hran

$$E = \{[0,2], [0,3], [1,0], [2,1], [3,1], [3,2]\},$$

kde $[i,j]$ značí hranu vedoucí z vrcholu i do vrcholu j (viz obr. 11.2).

Matice A, charakterizující graf, bude mít pro každé $i = 0, ..., 3$ diagonální prvky a_{ii} rovny počtu hran vstupujících do vrcholu i, a mimodiagonální prvky a_{ij} $(i \neq j)$ budou rovny -1 v případě, že z vrcholu i do vrcholu j vede hrana (tj. $[i,j] \in E$), a rovny 0 v opačném případě. V případě orientovaného grafu ($[i,j] \neq [j,i]$) nebude matice A symetrická:

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

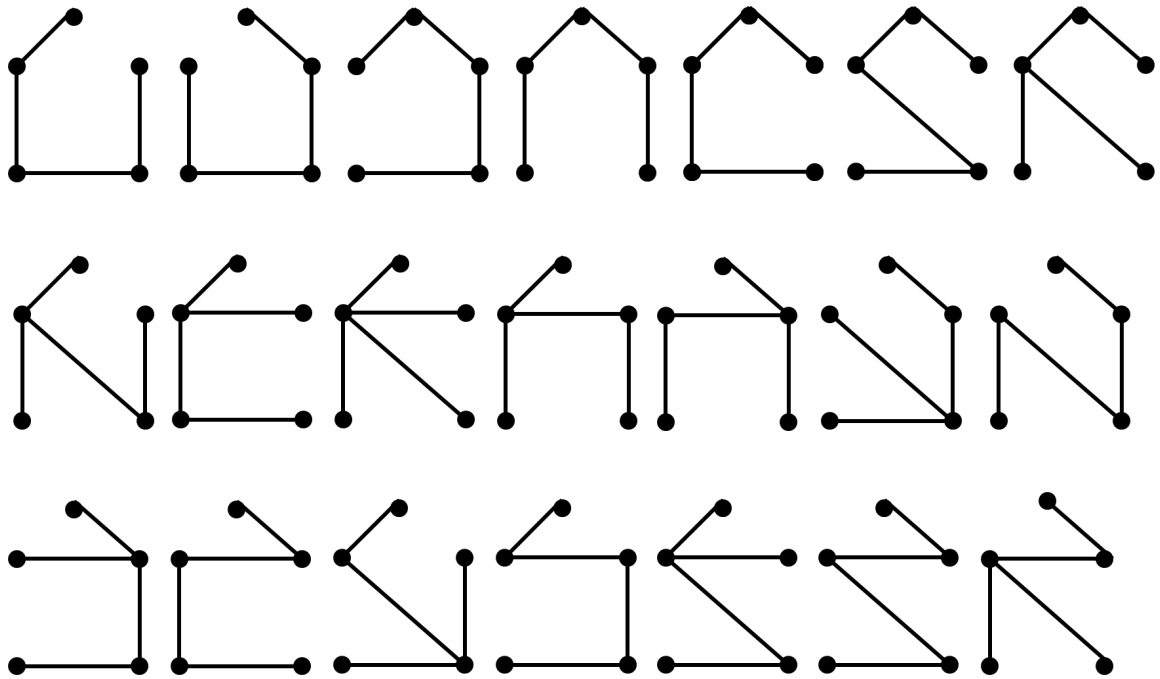
[21] Viz [12].

Lze ukázat[22], že počet neidentických kořenových stromů s kořenem ve vrcholu i je roven hodnotě algebraického doplňku prvku a_{ii}.

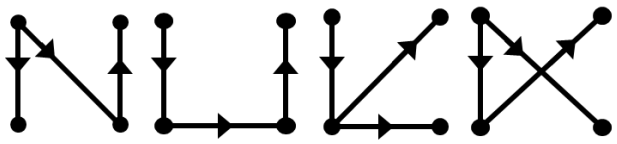
V případě uvažovaného grafu zjistíme např. počet stromů grafu s kořenem ve vrcholu 0 výpočtem algebraického doplňku prvku a_{00}:

$$(-1)^0 \begin{vmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ -1 & -1 & 1 \end{vmatrix} = 4.$$

V daném grafu lze najít 4 stromy s kořenem ve vrcholu 0 (viz obr. 11.4).



Obr. 11.3. Kostry grafu



Obr. 11.4. Stromy s kořenem ve vrcholu 0

[22] Viz [12].

11.9 Úlohy k procvičení

$\boxed{A}$ Teoretická část

Posuďte pravdivost následujících tvrzení:

a) Determinant je definován pro libovolnou matici.

b) Determinant je definován pouze pro regulární matice.

c) Záměnou pořadí řádků v matici se změní znaménko jejího determinantu.

d) Je-li na diagonále čtvercové matice nula, je determinant této matice nulový.

e) Determinant trojúhelníkové matice je roven součinu prvků na diagonále této matice.

f) Zaměníme-li v matici řádky za sloupce (při zachování pořadí), determinant se nezmění.

g) Není-li ve čtvercové matici jeden sloupec násobkem jiného, je determinant matice nenulový.

h) Je-li ve čtvercové matici jeden sloupec násobkem jiného, je determinant matice nulový.

i) Pro čtvercovou matici platí: determinant je roven nule, právě když hodnost matice je menší než je řád matice.

j) Determinant regulární matice je kladný.

k) Pro čtvercovou matici platí: determinant je nenulový, právě když hodnost matice je rovna jejímu řádu.

l) Determinant čtvercové matice je nenulový, právě když k matici existuje inverzní.

m) Je-li $\det A = 0$, pak soustava $A\bar{x} = \bar{b}$ nemá řešení.

n) Je-li $\det A = 0$, pak soustava $A\bar{x} = \bar{o}$ má nekonečně mnoho řešení.

o) Je-li $\det A \neq 0$, pak soustava $A\bar{x} = \bar{b}$ má právě jedno řešení.

p) Je-li $\det A \neq 0$, pak soustava $A\bar{x} = \bar{o}$ má pouze triviální řešení.

q) Soustava se čtvercovou maticí je řešitelná Cramerovým pravidlem jen tehdy, má-li právě jedno řešení.

r) $\det I = 1$, $\det O = 0$.

s) Pro čtvercové matice A, B platí: $\det(A \cdot B) = \det A \cdot \det B$.

t) Pro čtvercové matice A, B platí: $\det(A + B) = \det A + \det B$.

u) $\det(kA) = k \det A$.

v) Jsou-li A, B regulární, pak $\det(A \cdot B) \neq 0$.

w) Je-li A regulární, pak $\det A \cdot \det A^{-1} = 1$.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N (pouze pro čtvercové matice)

b) N (je definován i pro singulární matice)

c) N (záleží na počtu záměn dvou řádků mezi sebou)

d) N

e) P

f) P

g) N (může být např. jeden sloupec součtem jiných)

h) P

i) P

j) N (nenulový – může být i záporný)

k) P

l) P

m) N (může mít i nekonečně mnoho řešení)

n) P
o) P
p) P
q) P
r) P
s) P
t) N
u) N
v) P
w) P

B Praktická část

1 Vypočítejte následující determinanty.

a) $\begin{vmatrix} -2 & 3 \\ 1 & -4 \end{vmatrix}$ b) $\begin{vmatrix} 6 & 0 \\ -1 & 2 \end{vmatrix}$ c) $\begin{vmatrix} 0 & -3 \\ -2 & 7 \end{vmatrix}$ d) $\begin{vmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{vmatrix}$

e) $\begin{vmatrix} 1 & -2 & 3 \\ 4 & 3 & 2 \\ -1 & 0 & -2 \end{vmatrix}$ f) $\begin{vmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{vmatrix}$ g) $\begin{vmatrix} 2 & -5 & 1 \\ 3 & 4 & -2 \\ -1 & 1 & 2 \end{vmatrix}$

2 Převodem matice na trojúhelníkovou vypočítejte následující determinanty.

a) $\begin{vmatrix} 0 & 1 & 2 & -3 \\ -1 & 2 & 0 & 3 \\ 2 & -3 & 4 & -7 \\ -3 & 6 & 2 & 7 \end{vmatrix}$ b) $\begin{vmatrix} 2 & 3 & 0 & -3 \\ 3 & 6 & -2 & -7 \\ -2 & -5 & 4 & 3 \\ 4 & 5 & 3 & -8 \end{vmatrix}$ c) $\begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$ d) $\begin{vmatrix} 4 & -3 & 4 \\ -3 & 4 & 3 \\ 4 & 3 & 4 \end{vmatrix}$

3 Bez výpočtu určete následující determinanty.

a) $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$ b) $\begin{vmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{vmatrix}$ c) $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{vmatrix}$

4 Řešte následující rovnice.

a) $\begin{vmatrix} x & 1 & x \\ 2 & x & 3 \\ x & 4 & x \end{vmatrix} = 6$ b) $\begin{vmatrix} 0 & x & 1 \\ 1 & -1 & 1 \\ 3 & 4 & x \end{vmatrix} = -3$ c) $\begin{vmatrix} x & 0 & x \\ -1 & x & 1 \\ x & 2 & x \end{vmatrix} = 4$ d) $\begin{vmatrix} -3 & -2 & x & -3 \\ 0 & 3 & -1 & 3 \\ 0 & 1 & x & 0 \\ 2 & 1 & 0 & 2 \end{vmatrix} = 9$

5 Nalezněte kořeny polynomu

$$\begin{vmatrix} x & x & x \\ 1 & x & x \\ 1 & 1 & x \end{vmatrix}.$$

6 Vypočítejte rozvojem podle zvoleného řádku nebo sloupce následující determinanty.

a) $\begin{vmatrix} -3 & 1 & 2 & 0 \\ 1 & -1 & 0 & 1 \\ 0 & 5 & 1 & -2 \\ 2 & 0 & -4 & 3 \end{vmatrix}$ b) $\begin{vmatrix} 4 & 3 & 1 & -1 \\ 1 & 2 & 0 & -3 \\ -2 & 0 & 5 & 2 \\ 3 & 1 & 0 & 2 \end{vmatrix}$ c) $\begin{vmatrix} 0 & -2 & 3 & 0 \\ 1 & 0 & 5 & -3 \\ 4 & 0 & 3 & -1 \\ 2 & 1 & 0 & 4 \end{vmatrix}$ d) $\begin{vmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix}$

7 Vypočítejte:

$$|1| + \begin{vmatrix} 1 & 3 \\ 3 & 3 \end{vmatrix} + \begin{vmatrix} 1 & 1 & 3 \\ 1 & 3 & 3 \\ 3 & 3 & 3 \end{vmatrix} + \begin{vmatrix} 1 & 1 & 1 & 3 \\ 1 & 1 & 3 & 3 \\ 1 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{vmatrix}.$$

8 Užitím determinantu rozhodněte, je-li matice regulární.

a) $\begin{pmatrix} 1 & 3 & 6 \\ -1 & 1 & 2 \\ 1 & 1 & 0 \end{pmatrix}$ b) $\begin{pmatrix} 2 & -2 & 0 & 3 \\ -1 & 0 & 1 & 0 \\ 2 & 3 & 1 & 0 \\ 0 & 1 & 2 & -1 \end{pmatrix}$ c) $\begin{pmatrix} -2 & -4 & -2 & 2 \\ 1 & 0 & 0 & 3 \\ 0 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{pmatrix}$

9 Užitím determinantu rozhodněte, zda k matici A existuje inverzní.

a) $A = \begin{pmatrix} 4 & -1 & 2 \\ 0 & 3 & -4 \\ -4 & -2 & 2 \end{pmatrix}$ b) $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 2 & 5 \\ 2 & 0 & 2 \end{pmatrix}$

10 Užitím determinantu rozhodněte, zda daná soustava má netriviální řešení.

a)
$$\begin{aligned} 3x_0 - x_1 + 2x_2 + x_3 &= 0 \\ 2x_0 + x_1 - 2x_3 &= 0 \\ x_0 - 3x_1 + 5x_2 &= 0 \\ 4x_0 - 3x_3 &= 0, \end{aligned}$$

b)
$$\begin{aligned} -x_0 + 4x_1 &= 0 \\ -2x_0 + x_1 - x_2 &= 0 \\ -3x_0 + 5x_1 - x_2 &= 0 \end{aligned}$$

11 Cramerovým pravidlem vypočítejte x_0 ze soustavy

$$\begin{aligned} x_0 - 2x_1 + 3x_2 + 4x_3 &= 3 \\ 2x_0 - 3x_1 + x_3 &= 0 \\ 3x_0 - 4x_2 - x_3 &= -2 \\ -2x_0 + x_1 + x_3 &= 4. \end{aligned}$$

12 Vyřešte soustavu Cramerovým pravidlem.

$$\begin{aligned} x_0 + 2x_1 &= -1 \\ 2x_0 + 4x_1 + 3x_2 &= 2 \\ x_0 + 3x_1 + x_2 &= 0. \end{aligned}$$

Výsledky B

A

B

1 a) 5, b) 12, c) -6, d) 1, e) -9, f) 0, g) 47

2 a) -8, b) 2, c) -3, d) -144

3 a) -1 (záměnou 2. a 3. řádku vznikne I, $\det I = 1$), b) 0 (matice je singulární), c) 1 (matice je dolní trojúhelníková)

4 a) $x = -2$, b) $x = -2 \vee x = 5$, c) $x = -1$, d) $x = -1$

5 $x = 0$, $x = 1$

6 a) 70, b) 24, c) -153, d) 1

7 7

8 a) $\det A = -8 \neq 0 \implies$ regulární, b) $\det A = 21 \neq 0 \implies$ regulární, c) $\det A = 0 \implies$ singulární

9 a) $\det A = 0 \implies A^{-1}$ neexistuje, b) $\det A = -12 \neq 0 \implies A^{-1}$ existuje

10 a) $\det A = -61 \neq 0 \implies$ pouze triviální řešení, b) $\det A = 0 \implies$ i netriviální řešení

11 $x_0 = \frac{-100}{72} = \frac{-25}{18}$

12 $x_0 = \frac{1}{-3} = -\frac{1}{3}$, $x_1 = \frac{1}{-3} = -\frac{1}{3}$, $x_2 = \frac{-4}{-3} = \frac{4}{3}$

12. DETERMINANTY Z POHLEDU INFORMATIKA

Pojem determinant matice (viz Pojem 11.1) je definován pouze pro čtvercové matice. Determinant matice můžeme počítat několika různými postupy. Výpočet determinantu matice dle definice je pro praktické (ale i počítačové) použití nevhodný, protože tento postup dosahuje exponenciální časové složitosti[23]. Pro matice trojciferného řádu by výpočet determinantu tímto postupem nebyl dokončen do konce životnosti počítače, na kterém by byl počítán.

V Pravidle 11.1 je popsán postup pro výpočet determinantů matic řádu 2. Sarrusovo pravidlo (viz Pravidlo 11.2) slouží k výpočtu determinantu matic řádu 3. Pro vyšší řády matic nelze ani jedno z těchto pravidel použít. Tato pravidla jsou vhodná pro ruční počítání, ale z hlediska počítačového výpočtu je jejich implementace zbytečná. Potřebujeme obecný postup, který vypočítá determinant matice obecného řádu.

V Pravidle 11.3 je popsán postup pro výpočet determinantu matice rozvojem dle vybraného řádku. Analogicky v Pravidle 11.4 je popsán postup pro výpočet determinantu matice rozvojem dle vybraného sloupce. Oba tyto postupy jsou zobecněním postupu uvedeného při definici pojmu determinant matice. Při ručním počítání jsou tyto postupy vhodné pro matice, které v nějakém sloupci či řádku mají jednu (nebo několik málo) nenulových hodnot. Pro obecné matice, které neobsahují nulové hodnoty, jsou tyto postupy nevhodné, protože dosahují exponenciální časové složitosti.

V Pravidle 11.4 je popsán postup pro výpočet determinantu matice převodem matice na matici odstupňovanou. Tento postup je prakticky použitelný pro matice libovolného řádu, bez dalších podmínek na nenulovost prvků matice.

12.1 Výpočet determinantu převodem matice na odstupňovanou

Ve Zdrojovém kódu 12.1 je implementována funkce `determinant_matice`, která počítá determinant matice `A` převodem na matici odstupňovanou. Hodnotu determinantu matice `A` budeme počítat průběžně v proměnné `det`, její počáteční hodnota je nastavena na `1` (řádek `03`).

Pokud matice `A` není čtvercová (řádek `05`), je vyvolána chyba. Potřebujeme zjistit determinant matice `A` tak, abychom při tomto zjišťování determinantu matici `A` nezměnili. Do proměnné `B` vytvoříme kopii matice `A` (řádek `06`). V cyklu (řádek `07`) procházíme matici `B` po sloupcích. V těle cyklu (řádky `08` až `22`) převedeme matici `B` na odstupňovaný tvar.

Na konci cyklu bude dle Pravidla 11.5 determinant odstupňované matice `B` roven součinu prvků na diagonále odstupňované matice `B`. Při odstupňování matice `B` budeme využívat elementární řádkové úpravy. Při těchto úpravách dochází dle Pravidla 11.6 ke změnám determinantu upravené matice oproti determinantu původní matice. Pro získání determinantu matice `A` musíme determinant matice `B` za každou provedenou elementární řádkovou úpravu modifikovat (změnit jeho znaménko nebo ho vydělit konstantou).

Determinant matice `A` budeme počítat průběžně s pomocí proměnné `det`. Kdykoliv budeme znát hodnotu diagonálního prvku odstupňované matice `B`, tak touto hodnotou proměnnou `det` vynásobíme. Kdykoliv při odstupňování matice `B` provedeme prohození dvou řádků matice, změníme hodnotu znaménka proměnné `det`. Kdykoliv při odstupňování matice `B` vynásobíme její řádek konstantou, hodnotu proměnné `det` touto konstantou vydělíme.

Při převodu na odstupňovaný tvar potřebujeme na pozici pivota (`B[j][j]`) nenulovou hodnotu (řádek `09`). Pokud byla na pozici pivota nulová hodnota, je nutno přeuspořádat řádky matice (řádek `11`). Při přeuspořádání řádků matice dojde k právě jednomu prohození dvou řádků a dle Pravidla 11.6 dojde ke změně znaménka determinantu (řádek `12`). Průběžně počítanou hodnotu determinantu `det` vynásobíme prvkem na pozici pivota, který je diagonálním prvkem (budoucí odstupňované) matice `B` (řádek `14`).

Na řádku `15` testujeme situaci, kdy na pozici pivota zůstala nula i po přeuspořádání matice. V tomto případě je matice `B` singulární a její determinant je `0` dle Pravidla 11.8. Hodnota proměnné `det` se stala nulovou při jejím násobení nulovým diagonálním prvkem (řádek `14`). V této situaci můžeme přerušit cyklus provádějící výpočet determinantu pomocí konstrukce `break`.

Ve vnořeném cyklu (řádek `16`) procházíme matici po řádcích. Pokud je prvek daného řádku ve sloupci daném pozicí pivota (`B[i][j]`) nulový, je řádek `i` již odstupňován a pokračujeme další iterací cyklu (řádek `18`). V opačném případě provedeme odstupňování řádku `i` matice `B` (řádek `20`). Při tomto odstupňování

[23] Více informací o časové složitosti lze nalézt v [15].

dojde k vynásobení řádku `i` hodnotou pivota (`B[j][j]`). Determinant matice `B` se po tomto kroku také vynásobí hodnotou pivota dle Pravidla 11.6. Protože zjišťujeme determinant matice `A`, musíme průběžně počítanou hodnotu determinantu `det` vydělit hodnotou pivota (řádek `19`). Toto vydělení musí proběhnout ještě před odstupňováním.

Po skončení vnějšího cyklu (řádek `22`) je matice `B` v odstupňovaném tvaru a v proměnné `det` je hodnota determinantu matice `A`. Smažeme matici `B` (řádek `23`). Funkce končí a vrací návratovou hodnotu (řádek `24`) – determinant matice `A`.

Zdrojový kód 12.1 (determinant matice)

```
00 double determinant_matice(double ** A, int m, int n)
01 {
02   double ** B;
03   double det = 1;
04   int i, j;
05   if (m != n) chyba("Matice musi byt ctvercova.");
06   B = kopie_matice(A,m,n);
07   for (j=0; j<n; j++) //
08   {
09     if (!B[j][j])
10     {
11       preusporadat_matici_pro_odstupnovani(B,m,n,j,j);
12       det *= -1;
13     }
14     det *= B[j][j];
15     if (!B[j][j]) break;
16     for (i=j+1; i<m; i++)
17     {
18       if (!B[i][j]) continue;
19       det /= B[j][j];
20       odstupnuj_radek_matice(B,m,n,i,j,j);
21     }
22   }
23   smaz_matici(B,m,n);
24   return det;
25 }
```

12.2 Cramerovo pravidlo

Cramerovo pravidlo (viz Pravidlo 11.10) je zajímavým příkladem využití determinantů. Pokud máme nehomogenní soustavu lineárních rovnic, která má právě jedno řešení (matice homogenní soustavy je regulární), můžeme pomocí Cramerova pravidla vypočítat hodnotu libovolné její neznámé. Pokud bychom chtěli znát hodnoty všech neznámých, je výhodnější použit některou z eliminačních metod (viz kapitola 8).

V kapitole 6.1 jsme se zabývali elementárními řádkovými úpravami (viz Pojem 5.4). Ve Zdrojovém kódu 6.1 jsme implementovali funkci na prohození dvou řádků matice. Definovali jsme si rovněž elementární sloupcové úpravy (viz Pojem 5.5), jejichž implementace byla úkolem 2 v kapitole 6.4.

Pro potřeby implementace Cramerova pravidla jsme ve Zdrojovém kódu 12.2 implementovali `funkci prohod_sloupce_matice` provádějící elementární sloupcovou úpravu prohození dvou sloupců `s1` a `s2` matice `A`. Pokud se některý ze sloupců `s1` nebo `s2` nenachází v matici `A`, je vyvolána chyba (řádek `04`). V cyklu procházíme jednotlivé řádky matice `A` (řádek `05`). V těle cyklu prohazujeme na daném řádku `i` hodnoty ve sloupcích `s1` a `s2` (řádky `07` až `09`).

Zdrojový kód 12.2 (prohození sloupců matice)

```
00 void prohod_sloupce_matice(double ** A, int m, int n, int s1, int s2)
01 {
02   int i;
03   double tmp;
04   if ((s1 >= n) || (s2>= n)) chyba("Sloupec se nenachazi v matici.");
05   for (i=0; i<m; i++)
06   {
07     tmp = A[i][s1];
08     A[i][s1] = A[i][s2];
09     A[i][s2] = tmp;
10   }
11 }
```

Ve Zdrojovém kódu 12.3 jsme implementovali funkci `cramerovo_pravidlo`, která vypočte hodnotu neznámé s pořadovým číslem `neznama` z nehomogenní soustavy lineárních rovnic reprezentované rozšířenou maticí soustavy `A`. Při tomto výpočtu budeme v matici `A` prohazovat sloupce, ale na konci funkce je vrátíme do původního stavu. Matice `A` zůstane nepoškozena. V průběhu výpočtu se objedeme bez pomocné matice.

Nejprve si deklarujeme proměnné, do kterých budeme ukládat hodnoty determinantů (řádek `02`). První proměnná `determinant_A` bude obsahovat determinant nerozšířené matice soustavy. Druhá proměnná `determinant_Ai` bude obsahovat determinant nerozšířené matice soustavy, ve které je sloupec odpovídající hledané neznámé nahrazen sloupcem pravých stran.

Nejprve vypočteme determinant nerozšířené matice soustavy (řádek `03`) – výsledek uložíme do proměnné `determinant_A`. Nerozšířenou matici soustavy získáme vynecháním posledního řádku z matice `A`. Pokud je vypočtený determinant nulový, je vyvolána chyba (řádek `04`).

V matici A prohodíme (řádek `05`) sloupec odpovídající pořadovému číslu vypočítávané neznámé se sloupcem pravých stran – posledním sloupcem matice. Vypočteme determinant upravené matice `A` s vynecháním jejího posledního sloupce – sloupec, který v původní matici `A` odpovídal pořadovému číslu vypočítávané neznámé (řádek `06`). Výsledek uložíme do proměnné `determinant_Ai`. Opět prohodíme poslední sloupec matice a sloupec odpovídající vypočítávané neznámé (řádek `07`). Tímto prohozením jsme vrátili matici `A` do původního stavu, ve kterém vstupovala do funkce.

Hodnota vypočítávané neznámé se získá jako podíl obou determinantů (řádek `08`). V čitateli zlomku je determinant nerozšířené matice soustavy s prohozeným sloupcem. Ve jmenovateli zlomku je determinant původní nerozšířené matice soustavy. Funkce končí a vrací tento podíl jako svojí návratovou hodnotu.

Zdrojový kód 12.3 (Cramerovo pravidlo)

```
00 double cramerovo_pravidlo(double ** A, int m, int n, int neznama)
01 {
02   double determinant_A, deteraminant_Ai;
03   determinant_A = determinant_matice(A,m,n-1);
04   if (!determinant_A) chyba("Matice musi byt regularni.");
05   prohod_sloupce_matice(A,m,n,neznama,n-1);
06   deteraminant_Ai = determinant_matice(A,m,n-1);
07   prohod_sloupce_matice(A,m,n,n-1,neznama);
08   return deteraminant_Ai/determinant_A;
09 }
```

12.3 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 12.

2 Implementujte postup výpočtu determinantu matic řádu 2 dle Pravidla 11.1 a výpočet matic řádu 3 dle Pravidla 11.2 (Sarrusovo pravidlo).

3 Implementujte postupy výpočtu determinantu rozvojem dle vybraného řádku (viz Pravidlo 11.3) a dle

vybraného sloupce (viz Pravidlo 11.4).

4 Na příkladech náhodných matic vyzkoušejte, že všechny postupy pro výpočet determinantu (implementované v rámci úkolů 1, 2 a 3) dávají shodné výsledky. Na vhodně zvoleném řádu matic sledujte časovou náročnost jednotlivých postupů.

5 Na příkladech náhodných matic demonstrujte vlastnosti determinantu (viz Pravidlo 11.9).

6 Na příkladu náhodné matice řádu 4×4 (matici upravte, aby obsahovala cca 20% nul) aplikujte postup na zjištění počet koster grafu (viz kapitola 11.8). Pro kontrolu si jednotlivé kostry načrtněte.

13. DEFINITNOST MATICE

Tzv. definitnost matice úzce souvisí s pojmem kvadratická forma[24], který přesahuje zamýšlený rámec této knihy. Pro záměr této knihy postačí klasifikace matic (bez použití pojmu kvadratická forma) tak, jak uvedeme níže.

Definitnost matice hraje důležitou roli např. při určování lokálních extrémů funkcí více proměnných[25], a je tak hezkou ukázkou propojenosti dvou matematických disciplin – lineární algebry a matematické analýzy.

13.1 Součin $\bar{x}^T A\bar{x}$

Pojmy, které uvedeme dále, jsou definovány pro symetrické matice; připomínáme, že čtvercová matice $A = (a_{ij})$ je symetrická, platí-li $a_{ij} = a_{ji}$ pro všechna i, j, tzn. platí-li $A = A^T$ (viz Pojem 3.5).

Poznámka (řádkové a sloupcové vektory)

V celé 13. kapitole budeme pracovat současně s řádkovými i sloupcovými vektory z prostoru $\mathbb{R}_n$, je třeba proto důsledně odlišovat jejich značení (na rozdíl od jiných kapitol, kde nemohlo dojít k omylu). Symbolem $\bar{x}$ apod. budeme značit *sloupcový* vektor, zatímco symbolem $\bar{x}^T$ apod. vektor *řádkový* ($\bar{x}^T$ značí vektor transponovaný k vektoru $\bar{x}$ – viz Pojem 3.4 a Poznámka za ním následující).

V dalším budeme pracovat se součinem $\bar{x}^T A\bar{x}$, kde A je daná symetrická matice řádu n a $\bar{x} \in \mathbb{R}_n$ je n-složkový vektor. Uvědomme se, že výsledkem tohoto součinu je *číslo*, neboť násobíme matice typů $1 \times n$, $n \times n$ a $n \times 1$ (viz násobení matic – Pojem 3.13). Nejprve si na příkladu rozmyslíme, jak tento součin vypadá rozepsaný po složkách vektoru $\bar{x}$.

P ř í k l a d 13.1

Rozepíšeme po složkách součin $\bar{x}^T A\bar{x}$ pro matici

$$A = \begin{pmatrix} -4 & 2 & -3 \\ 2 & 1 & 5 \\ -3 & 5 & -2 \end{pmatrix}.$$

Označíme-li $\bar{x}^T = (x_0, x_1, x_2)$, dostáváme

$$\bar{x}^T A\bar{x} = (x_0, x_1, x_2) \begin{pmatrix} -4 & 2 & -3 \\ 2 & 1 & 5 \\ -3 & 5 & -2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} =$$

$$= (\ -4x_0 + 2x_1 - 3x_2,\ \ 2x_0 + x_1 + 5x_2,\ \ -3x_0 + 5x_1 - 2x_2\ ) \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} =$$

$$= (-4x_0 + 2x_1 - 3x_2)x_0 + (2x_0 + x_1 + 5x_2)x_1 + (-3x_0 + 5x_1 - 2x_2)x_2 =$$

$$= -4x_0x_0 + 2x_1x_0 - 3x_2x_0 + 2x_0x_1 + x_1x_1 + 5x_2x_1 - 3x_0x_2 + 5x_1x_2 - 2x_2x_2 =$$

$$= -4{x_0}^2 + {x_1}^2 - 2{x_2}^2 + 4x_0x_1 - 6x_0x_2 + 10x_1x_2.$$

Je tedy :

$$\bar{x}^T A\bar{x} = -4{x_0}^2 + {x_1}^2 - 2{x_2}^2 + 4x_0x_1 - 6x_0x_2 + 10x_1x_2.$$

Rozmyslíme si, že k tomuto výsledku lze dospět jednodušším způsobem. Prohlédněme si dobře výsledek a všimněme si souvislosti mezi koeficienty u jednotlivých součinů x_ix_j a prvky matice A. Vidíme, že koeficienty u druhých mocnin ${x_0}^2$, ${x_1}^2$, ${x_2}^2$ odpovídají diagonálním prvkům matice (v příslušném pořadí) a koeficienty u ostatních součinů jsou rovny vždy dvojnásobku prvku ležícího nad diagonálou na příslušné pozici:

koeficient u součinu x_0x_1 je roven dvojnásobku prvku a_{01},

[24] Viz např. v [3].
[25] Viz např. v [1].

koeficient u součinu x_0x_2 je roven dvojnásobku prvku a_{02},
koeficient u součinu x_1x_2 je roven dvojnásobku prvku a_{12}.

Toto platí obecně pro jakoukoliv symetrickou matici A: ze symetrie matice vyplývá, že ke každému součinu x_ix_j (pro $i \neq j$) se dále vyskytne součin x_jx_i se stejným koeficientem a_{ij} (neboť $a_{ij} = a_{ji}$). Tyto součiny lze k sobě sečíst – koeficient u součinu x_ix_j bude tedy $2a_{ij}$:

$$\bar{x}^T A\bar{x} = (x_0, x_2 \dots, x_{n-1}) \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,n-1} \\ a_{10} & a_{11} & \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} =$$

$$= a_{00}x_0x_0 + a_{01}x_0x_1 + \dots + a_{10}x_1x_0 + a_{11}x_1x_1 + \dots + a_{n-1,n-1}x_{n-1}x_{n-1} =$$

$$= a_{00}{x_0}^2 + a_{11}{x_1}^2 + \dots + a_{n-1,n-1}{x_{n-1}}^2 + 2a_{01}x_0x_1 + 2a_{02}x_0x_2 + \dots + 2a_{n-2,n-1}x_{n-2}x_{n-1}.$$

P ř í k l a d 13.2

Napíšeme součin $\bar{x}^T A\bar{x}$ přímo podle prvků matice A:

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & 4 \\ 0 & 4 & 5 \end{pmatrix}.$$

Diagonální prvky budou koeficienty u druhých mocnin ${x_i}^2$, prvky nad diagonálou brané $2\times$ budou koeficienty u příslušných součinů x_ix_j $(i \neq j)$:

$$\bar{x}^T A\bar{x} = 2{x_0}^2 + 3{x_1}^2 + 5{x_2}^2 - 2x_0x_1 + 8x_1x_2.$$

P ř í k l a d 13.3

Ze součinu $\bar{x}^T A\bar{x}$ určíme symetrickou matici A:

$$\bar{x}^T A\bar{x} = 3{x_0}^2 - 2{x_1}^2 + 4{x_3}^2 - 6x_0x_1 + 2x_0x_2 + 4x_0x_3 - 8x_1x_3 + 10x_2x_3.$$

Na diagonále budou koeficienty u druhých mocnin, mimo diagonálu (symetricky) na příslušných pozicích poloviny koeficientů u ostatních součinů:

$$A = \begin{pmatrix} 3 & -3 & 1 & 2 \\ -3 & -2 & 0 & -4 \\ 1 & 0 & 0 & 5 \\ 2 & -4 & 5 & 4 \end{pmatrix}.$$

Poznámka (součin $\bar{x}^T D\bar{x}$ pro diagonální matici)

Uvědomme si speciální tvar součinu $\bar{x}^T D\bar{x}$, kde matice D je diagonální – složky vektoru $\bar{x}$ se v něm vyskytují jen v 2. mocninách:

$$D = \begin{pmatrix} d_{00} & 0 & 0 & 0 & \dots & 0 \\ 0 & d_{11} & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & d_{n-1,n-1} \end{pmatrix} \iff \bar{x}^T D\bar{x} = d_{00}{x_0}^2 + d_{11}{x_1}^2 + \dots + d_{n-1,n-1}{x_{n-1}}^2.$$

P ř í k l a d 13.4

Dosaďme do výsledku Příkladu 13.1 postupně složky vektorů $\bar{u} = (1, 0, -1)^T$, $\bar{v} = (1, 1, 1)^T$,
$\bar{w} = (1, -2, 0)^T$, $\bar{o} = (0, 0, 0)^T$:
$\bar{u}^T A\bar{u} = -4 \cdot 1^2 + 0^2 - 2 \cdot (-1)^2 + 4 \cdot 1 \cdot 0 - 6 \cdot 1 \cdot (-1) + 10 \cdot 0 \cdot (-1) = 0,$

$\bar{v}^T A\bar{v} = -4 \cdot 1^2 + 1^2 - 2 \cdot 1^2 + 4 \cdot 1 \cdot 1 - 6 \cdot 1 \cdot 1 + 10 \cdot 1 \cdot 1 = 3,$

$\bar{w}^T A\bar{w} = -4 \cdot 1^2 + (-2)^2 - 2 \cdot 0^2 + 4 \cdot 1 \cdot (-2) - 6 \cdot 1 \cdot 0 + 10 \cdot (-2) \cdot 0 = -8,$

$\bar{o}^T A\bar{o} = -4 \cdot 0^2 + 0^2 - 2 \cdot 0^2 + 4 \cdot 0 \cdot 0 - 6 \cdot 0 \cdot 0 + 10 \cdot 0 \cdot 0 = 0.$

Vidíme, že součin $\bar{x}^T A\bar{x}$ nabývá pro různé vektory kladných, záporných i nulových hodnot. Všimněme si, že nulové hodnoty bylo nabyto jak na nulovém vektoru (vektor $\bar{o}$), tak na nenulovém vektoru (vektor $\bar{u}$).

Je zřejmé, že pro nulový vektor je součin $\bar{o}^T A\bar{o}$ roven nule pro jakoukoliv matici A:

$$\bar{o}^T A\bar{o} = 0.$$

Platí pro libovolnou symetrickou matici A, že součin $\bar{x}^T A\bar{x}$ může pro různé vektory nabývat jak kladných, tak i záporných hodnot? Snadno si rozmyslíme, že ne. Podle toho, jakých hodnot součin může nabývat, definujeme tzv. definitnost matice.

13.2 Matice pozitivně/negativně (semi)definitní, indefinitní

Ukážeme si nejprve pro jednoduchost na případu diagonálních matic, jakých hodnot (kladných, záporných, nulových) může nabývat součin $\bar{x}^T A\bar{x}$.

P ř í k l a d 13.5

Určíme, jakých hodnot (kladných, záporných, nulových) může nabývat součin $\bar{x}^T A\bar{x}$ pro následující matice. Protože víme, že nulovému vektoru je vždy přiřazena nula, budeme se zabývat jen nenulovými vektory.

a)

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{pmatrix} \Longrightarrow \bar{x}^T A\bar{x} = 4{x_0}^2 + 3{x_1}^2 + 5{x_2}^2$$

nabývá kladných hodnot pro jakýkoliv nenulový vektor (součet druhých mocnin nemůže být záporný, roven nule je jen pro $x_0 = x_1 = x_2 = 0$).

b)

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \Longrightarrow \bar{x}^T A\bar{x} = {x_0}^2 + {x_1}^2$$

nabývá nezáporných hodnot pro jakýkoliv vektor – pro vektory, pro něž je $x_0 = x_1 = 0$ (x_2 libovolné) nabývá nuly (např. pro vektor $(0, 0, 3)$), pro ostatní nenulové vektory nabývá kladných hodnot.

c)

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \Longrightarrow \bar{x}^T A\bar{x} = {x_0}^2 - {x_1}^2$$

může nabývat i kladných hodnot – je-li ${x_0}^2 > {x_1}^2$ (např. $(-2, 1)$), i záporných hodnot – je-li ${x_0}^2 < {x_1}^2$ (např. $(1, 3)$), a také nuly – je-li ${x_0}^2 = {x_1}^2$ (např. $(2, -2)$).

Vidíme že existují matice, pro něž součin $\bar{x}^T A\bar{x}$ nabývá pouze kladných hodnot (kromě nulového vektoru – na něm je hodnota vždy nulová), nebo pouze nezáporných hodnot. Snadno si lze rozmyslet, že obdobně existují i matice, pro které součin $\bar{x}^T A\bar{x}$ nabývá (pro nenulové vektory) pouze záporných hodnot (stačí v Příkladu 13.5 a) vynásobit matici číslem -1) či pouze nekladných hodnot (stačí v Příkladu 13.5 b) vynásobit matici číslem -1). Podle toho rozlišujeme 5 typů symetrických matic:

Pojem 13.1 (matice pozitivně (semi)definitní, negativně (semi)definitní, indefinitní)

Symetrická matice A se nazývá

(1) pozitivně definitní, je-li $\bar{x}^T A\bar{x} > 0$ pro všechna $\bar{x} \neq \bar{o}$

(2) negativně definitní, je-li $\bar{x}^T A\bar{x} < 0$ pro všechna $\bar{x} \neq \bar{o}$

(3) pozitivně semidefinitní, je-li $\bar{x}^T A\bar{x} \geq 0$ pro všechna $\bar{x}$

(4) negativně semidefinitní, je-li $\bar{x}^T A\bar{x} \leq 0$ pro všechna $\bar{x}$

(5) indefinitní, existují-li vektory $\bar{x}$ a $\bar{y}$, pro něž je $\bar{x}^T A\bar{x} > 0$ a $\bar{y}^T A\bar{y} < 0$.

Poznámka (nulová matice)

Nulová matice je podle této definice současně pozitivně i negativně semidefinitní.

P ř í k l a d 13.6

Matice z Příkladu 13.5 jsou: a) pozitivně definitní, b) pozitivně semidefinitní, c) indefinitní.

Poznámka (násobek matice)

Protože je $\bar{x}^T(kA)\bar{x} = k(\bar{x}^T A\bar{x})$, platí:

Je-li matice A pozitivně (semi)definitní, pak je matice kA pro $k > 0$ také pozitivně (semi)definitní a pro $k < 0$ je negativně (semi)definitní. Obdobně pro A negativně (semi)definitní.

Speciálně je-li matice A pozitivně (semi)definitní, pak je matice $-A$ negativně (semi)definitní, a je-li matice A negativně (semi)definitní, pak je matice $-A$ pozitivně (semi)definitní.

Uvědomme si, že není souvislost mezi znaménkem všech prvků matice a její definitností. Např. matice

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

má všechny prvky kladné, ale není pozitivně definitní, ale indefinitní. Je totiž $\bar{x}^T A\bar{x} = {x_0}^2 + {x_1}^2 + 4x_0x_1$ a tento součin je kladný např. pro vektor $(1, 1)$ a záporný např. pro $(1, -1)$.

Naproti tomu matice

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

je pozitivně definitní (přenecháváme čtenářům jako cvičení po nastudování následujících kapitol), ale nemá všechny prvky kladné.

Platí ale následující tvrzení:

Pravidlo 13.1 *(diagonála (semi)definitní matice)*

Pozitivně definitní matice má všechny diagonální prvky kladné. Negativně definitní matice má všechny diagonální prvky záporné. Pozitivně semidefinitní matice má všechny diagonální prvky nezáporné. Negativně semidefinitní matice má všechny diagonální prvky nekladné.

Toto tvrzení neplatí obráceně – z kladnosti diagonálních prvků nevyplývá, že matice je pozitivně definitní, viz např. výše uvedená indefinitní matice $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$.

Pravidlo 13.2 *(regularita definitní matice)*

Pozitivně (resp. negativně) definitní matice je regulární.

Obrácené tvrzení neplatí – z regularity matice nevyplývá její pozitivní (resp. negativní) definitnost, viz např. opět matice $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$, která je regulární, ale není ani pozitivně ani negativně definitní. Tvrzení neplatí pro semidefinitní matice.

Zjistit, jakého typu matice je, nebývá obecně snadné. Matice z Příkladu 13.1 je evidentně indefinitní (součin $\bar{x}^T A\bar{x}$ nabývá jak kladných, tak záporných hodnot; že nabývá i nuly není podstatné). I kdybychom např. zjistili, že na všech námi zvolených několika vektorech nabývá součin $\bar{x}^T A\bar{x}$ kladných hodnot, samozřejmě to neznamená, že je matice pozitivně definitní (museli bychom to ukázat pro *všechny* nenulové vektory z $\mathbb{R}_n$). Dva ze způsobů, jak zjistit typ matice, uvedeme v následujících kapitolách.

13.3 Určení definitnosti převedením matice na diagonální matici

Každou symetrickou matici lze symetrickými úpravami (tzn. úpravami, kde po každé řádkové úpravě následuje stejná úprava sloupcová) převést na matici diagonální. Podle ní už snadno poznáme typ matice (doporučujeme rozmyslet si na příkladech z předchozí kapitoly).

Pravidlo 13.3 *(definitnost matice pomocí diagonálního tvaru)*

Nechť D je diagonální matice vzniklá symetrickými úpravami ze symetrické matice A. Pak platí:

(1) A je pozitivně definitní $\iff$ všechny diagonální prvky matice D jsou kladné

(2) A je negativně definitní $\iff$ všechny diagonální prvky matice D jsou záporné

(3) A je pozitivně semidefinitní $\iff$ všechny diagonální prvky matice D jsou buď kladné nebo nulové

(4) A je negativně semidefinitní $\iff$ všechny diagonální prvky matice D jsou buď záporné nebo nulové

(5) A je indefinitní $\iff$ na diagonále matice D jsou kladné i záporné prvky.

Pokud jsou na diagonále matice D jak kladné, tak záporné prvky, pak nezáleží na tom, jsou-li kromě nich na diagonále i nuly – matice je v každém případě indefinitní.

P ř í k l a d 13.7

Ověříme předchozí tvrzení na maticích z Příkladu 13.5 (tyto matice jsou přímo diagonální a jejich definitnost už známe):

a)
$$A = D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$
všechny diagonální prvky jsou kladné, matice je pozitivně definitní

b)
$$A = D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
všechny diagonální prvky jsou kladné nebo nulové, matice je pozitivně semidefinitní

c)
$$A = D = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
na diagonále jsou čísla kladná i záporná, matice je indefinitní.

P ř í k l a d 13.8

Určíme typ matice
$$A = \begin{pmatrix} 1 & -3 & 4 \\ -3 & 9 & -10 \\ 4 & -10 & 19 \end{pmatrix}.$$

Převedeme matici symetrickými úpravami na diagonální a podle Pravidla 13.3 určíme její typ. Matici převedeme známým postupem na odstupňovanou, po každé řádkové úpravě však musí následovat odpovídající úprava sloupcová. Doporučujeme poznamenávat si úpravy, které provádíme s řádky – tytéž pak musíme udělat se sloupci (po každé dvojici symetrických úprav musí být matice symetrická – doporučujeme průběžně kontrolovat):

$$\begin{pmatrix} 1 & -3 & 4 \\ -3 & 9 & -10 \\ 4 & -10 & 19 \end{pmatrix} \begin{matrix} \\ \overset{3}{\leftarrow} \\ \\ \end{matrix} \begin{matrix} \\ \overset{-4}{\downarrow} \\ \\ \end{matrix} \sim \begin{pmatrix} 1 & -3 & 4 \\ 0 & 0 & 2 \\ 0 & 2 & 3 \end{pmatrix}.$$

Nyní tedy musí následovat odpovídající sloupcová úprava: 0. sloupec vynásobit 3 a přičíst k 1. sloupci, pak vynásobit 0. sloupec –4 a přičíst ke 2. sloupci:

$$\begin{matrix} -4 \\ 3 \\ \\ \end{matrix} \begin{matrix} \\ \downarrow \\ \downarrow \\ \end{matrix} \begin{pmatrix} 1 & -3 & 4 \\ 0 & 0 & 2 \\ 0 & 2 & 3 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 3 \end{pmatrix}.$$

Pro další úpravu potřebujeme na diagonále nenulové číslo – vyměníme proto 1. a 2. řádek, totéž pak musíme udělat se sloupci:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 3 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 0 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 0 \end{pmatrix}.$$

Nyní přičteme minus dvojnásobek 1. řádku k trojnásobku 2. řádku, následně pak totéž se sloupci:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 0 \end{pmatrix} \begin{matrix} \\ \overset{-2}{\sim} \\ 3 \end{matrix} \begin{matrix} \\ \\ \leftarrow \end{matrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 0 & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -12 \end{pmatrix}.$$

Matice je diagonální, na diagonále jsou jak kladná, tak záporná čísla – matice je indefinitní.

Prohlédněme si znovu pozorně výsledky úprav, které jsme prováděli v předchozím příkladu. Všimněme si, že pokud pouze přičítáme násobky řádku k jinému (*nevynásobenému*) řádku – tak, aby na příslušných místech vznikly nuly, pak odpovídajícími sloupcovými úpravami pouze dostaneme nuly v řádku odpovídajícím "vynulovanému" sloupci, ostatní prvky se nezmění – při práci se sloupci totiž už na příslušných místech přičítáme jen násobky nul (viz počáteční dvojice úprav matice v předchozím příkladu):

$$\begin{pmatrix} 1 & -3 & 4 \\ -3 & 9 & -10 \\ 4 & -10 & 19 \end{pmatrix} \overset{\substack{3 \\ -4}}{\sim} \begin{pmatrix} 1 & -3 & 4 \\ \mathbf{0} & 0 & 2 \\ \mathbf{0} & 2 & 3 \end{pmatrix} \overset{\substack{-4 \\ 3}}{\sim} \begin{pmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0 & 2 \\ \mathbf{0} & 2 & 3 \end{pmatrix}.$$

V takovém případě tedy nemusíme se sloupci skutečně pracovat – stačí formálně doplnit symetricky nuly do příslušného řádku a ostatní prvky nechat beze změny.

Pokud ovšem násobíme i řádek, *ke kterému* přičítáme jiný řádek (viz přičtení minus dvojnásobku 1. řádku k trojnásobku 2. řádku), změní se při následných sloupcových úpravách i ostatní prvky – úpravy je pak nutno skutečně provádět.

V Příkladu 13.8 jsme během výpočtu potřebovali získat nenulové číslo na diagonále – povedlo se nám to výměnou řádků (a následnou odpovídající výměnou sloupců). Ne vždy ale výměna řádků v takové situaci pomůže – viz Příklad 13.9.

P ř í k l a d 13.9

Převedeme matici A symetrickými úpravami na diagonální matici:

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

Abychom převedli matici na diagonální matici, je potřeba získat na pozici $(0, 0)$ nenulové číslo. Záměna řádků (a následně pak sloupců) nám ale v tomto případě nepomůže:

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

Místo výměny řádků přičteme k 0. řádku 1. řádek a následně provedeme tutéž úpravu se sloupci:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}.$$

Pokračujeme dál v úpravách – 0. řádek přičteme k minus dvojnásobku 1. řádku, totéž pak se sloupci:

$$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}.$$

Matice je indefinitní.

Diagonální tvar matice není jednoznačný, platí ale důležité tvrzení:

Pravidlo 13.4 *(počet nul, kladných a záporných prvků)*

Je-li D diagonální matice vzniklá symetrickými elementárními úpravami symetrické matice A, pak počet kladných prvků, počet záporných prvků a počet nul na diagonále matice D je jednoznačně určen.

13.4 Určení definitnosti matice pomocí Sylvestrova kritéria

Tento postup využívá determinantů. Pro symetrickou matici

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots & a_{0,n-1} \\ a_{01} & a_{11} & a_{12} & \dots & a_{1,n-1} \\ a_{02} & a_{12} & a_{22} & \dots & a_{2,n-1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{0,n-1} & a_{1,n-1} & a_{2,n-1} & \dots & a_{n-1,n-1} \end{pmatrix}$$

označme $D_1 = a_{00}$, $D_2 = \begin{vmatrix} a_{00} & a_{01} \\ a_{01} & a_{11} \end{vmatrix}$, $D_3 = \begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{01} & a_{11} & a_{12} \\ a_{02} & a_{12} & a_{22} \end{vmatrix}$ atd.,

tzn. determinanty matic "zvětšujících se z levého horního rohu" až po determinant celé matice:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots & a_{0,n-1} \\ a_{01} & a_{11} & a_{12} & \dots & a_{1,n-1} \\ a_{02} & a_{12} & a_{22} & \dots & a_{2,n-1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{0,n-1} & a_{1,n-1} & a_{2,n-1} & \dots & a_{n-1,n-1} \end{pmatrix}$$

Pravidlo 13.5 *(Sylvestrovo kritérium)*

Nechť $A = (a_{ij})_{n\times n}$ je symetrická matice. Označme D_1, D_2, ... , D_n jako v předchozím. Pak platí:

(1) matice A je pozitivně definitní $\iff$ $D_1 > 0$, $D_2 > 0$, $D_3 > 0$, ... , $D_n > 0$ (tzn. všechny determinanty D_i jsou kladné)

(2) matice A je negativně definitní $\iff$ $D_1 < 0$, $D_2 > 0$, $D_3 < 0$, ... (tzn. determinanty střídají znaménko, přičemž první je záporný, neboli všechny determinanty s lichým indexem jsou záporné a všechny determinanty se sudým indexem jsou kladné)

(3) je-li některý z determinantů se sudým indexem záporný nebo existují-li dva determinanty s lichým indexem, z nichž jeden je kladný a jeden záporný, je matice A indefinitní.

Poznámka (situace nerozhodnutelné Sylvestrovým kritériem)

Podle Sylvestrova kritéria lze spolehlivě určit všechny pozitivně a negativně definitní matice. Nelze však určit semidefinitní matice a z indefinitních jen některé – bod (3) je totiž pouze implikace. Problém může nastat v případě, že je některý z determinantů D_i roven nule. Je-li některý z determinantů roven nule a všechny ostatní mají znaménka jako v (1) nebo (2), nelze podle Sylvestrova kritéria o typu matice rozhodnout (lze jen vyloučit, že matice je pozitivně nebo negativně definitní – pro ty jsou všechny determinanty nenulové).

Výslovně zdůrazňujeme, že jsou-li např. všechny determinanty D_i nezáporné, neznamená to, že je matice pozitivně semidefinitní – viz následující příklad.

P ř í k l a d 13.10

Pro matici

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

vypočítáme determinanty D_i:

$D_1 = 0$, $D_2 = \begin{vmatrix} 0 & 0 \\ 0 & -1 \end{vmatrix} = 0$, $D_3 = \begin{vmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -1 \end{vmatrix} = 1 > 0$. To neznamená, že matice je pozitivně semidefinitní (podle Sylvestrova kritéria nelze rozhodnout). Matice je indefinitní, neboť součin $\bar{x}^T A\bar{x} = -{x_1}^2 - {x_2}^2 + 2x_0x_2$ je např. pro vektor $(1, 0, -1)$ záporný a např. pro vektor $(1, 0, 1)$ je kladný.

P ř í k l a d 13.11

Určíme typ matice z Příkladu 13.8, tj. matice

$$A = \begin{pmatrix} 1 & -3 & 4 \\ -3 & 9 & -10 \\ 4 & -10 & 19 \end{pmatrix}.$$

Je $D_1 = 1 > 0$, $D_2 = \begin{vmatrix} 1 & -3 \\ -3 & 9 \end{vmatrix} = 0$ – to může být problém. Pokud ale vyjde D_3 záporný, pak je podle (3) matice indefinitní (neboť D_1 je kladný a spolu s D_3 bude tvořit dvojici determinantů

s lichými indexy a různými znaménky). Pokud by ale D_3 vyšel kladný nebo roven nule, nedalo by se podle Sylvestrova kritéria rozhodnout.

Vypočítáme tedy

$$D_3 = \begin{vmatrix} 1 & -3 & 4 \\ -3 & 9 & -10 \\ 4 & -10 & 19 \end{vmatrix} = 171 + 120 + 120 - 144 - 171 - 100 = -4 < 0.$$

Protože je $D_1 > 0$ a $D_3 < 0$, je matice indefinitní.

Poznámka (znaménka determinantů)

Poznamenejme, že nás zajímá jen znaménko determinantu, ne jeho přesná hodnota (kromě nuly) – často lze znaménko odhadnout v průběhu výpočtu a není třeba počítat přesný výsledek.

Poznámka (ukončení výpočtu)

Ne vždy je nutné počítat všechny determinanty $D_1, \dots, D_n$ – je-li např. $D_2 < 0$, netřeba pokračovat ve výpočtu – matice je indefinitní podle (3).

P ř í k l a d 13.12

Určíme typ matice

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

Je $D_1 = 1 > 0$, $D_2 = \begin{vmatrix} 1 & 3 \\ 3 & -2 \end{vmatrix} = -11 < 0$. Protože je $D_2 < 0$, není třeba počítat D_3 – matice je podle (3) indefinitní.

P ř í k l a d 13.13

Určíme typ matice

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

Je $D_1 = 1 > 0$, $D_2 = \begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix} = 1 > 0$, $D_3 = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{vmatrix} = 4 + 1 + 1 - 2 - 2 - 1 = 1 > 0$. Podle prvních tří determinantů to vypadá, že matice je pozitivně definitní. Abychom to potvrdili, je nutné vypočítat i determinant D_4 (pokud by byl záporný, byla by matice indefinitní, pokud by byl nulový, nedalo by se podle Sylvestrova kritéria rozhodnout). Pro počítání determinantu D_4 řádu 4 matici nejdřív upravíme (pro výpočet rozvojem je lepší mít v některém řádku či sloupci co nejvíc nul). Nuly v prvním sloupci dostaneme např. postupným odečtením 0. řádku od všech ostatních (determinant se touto úpravou nezmění):

$$D_4 = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

Po úpravě vznikla přímo trojúhelníková matice – determinant je roven součinu diagonálních prvků (nemusíme počítat rozvojem – jinak bychom provedli rozvoj podle 0. sloupce). Je tedy $D_4 = 1 > 0$.

Všechny determinanty D_1, D_2, D_3, D_4 jsou kladné – matice je pozitivně definitní.

Zejména pro matice do řádu tři bývá posuzování typu matice pomocí Sylvestrova kritéria rychlejší než pomocí převodu matice na diagonální matici. Nevýhodou je to, že pomocí Sylvestrova kritéria nelze rozhodnout ve všech případech.

Jak jsme se zmínili v úvodu, používá se zjištění typu matice např. při hledání lokálních extrémů funkcí více proměnných; při tomto procesu je důležité zjistit, zda určitá matice je pozitivně (resp. negativně) definitní, a to lze Sylvestrovým kritériem určit vždy. Navíc se v aplikacích poměrně často vyskytují funkce

dvou (resp. tří) proměnných, pro něž je potřeba posuzovat typ matice řádu dvě (resp. tři) – Sylvestrovo kritérium je pak užitečným pomocníkem.

13.5 Úlohy k procvičení

$\boxed{A}$ **Teoretická část**

Posuďte pravdivost následujících tvrzení:

a) Součin $\bar{x}^T A\bar{x}$ přiřazuje nulovému vektoru nulový vektor.

b) Součin $\bar{x}^T A\bar{x}$ přiřazuje pro pozitivně definitní matici A každému vektoru $\bar{x}$ z $\mathbb{R}_n$ vektor, jehož složky jsou kladná čísla.

c) Součin $\bar{x}^T A\bar{x}$ přiřazuje pro pozitivně definitní matici A každému vektoru $\bar{x}$ z $\mathbb{R}_n$ kladné číslo.

d) Negativně definitní matice má všechny prvky záporné.

e) Pozitivně definitní matice nemůže obsahovat nulu.

f) Matice, která má na diagonále všechna čísla nezáporná, je pozitivně semidefinitní.

g) Matice, která je pozitivně semidefinitní, nemůže mít na diagonále záporné číslo.

h) Regulární matice je pozitivně nebo negativně definitní.

i) Negativně definitní matice je regulární.

j) Je-li v diagonálním tvaru matice na diagonále nula, je matice pozitivně nebo negativně semidefinitní.

k) Jsou-li pro matici řádu n všechny determinanty D_1, …, D_n (viz Sylvestrovo kritérium) kladné, je tato matice pozitivně definitní.

l) Jsou-li pro matici řádu n všechny determinanty D_1, …, D_n (viz Sylvestrovo kritérium) záporné, je tato mtice negativně definitní.

m) Jsou-li pro matici řádu n všechny determinanty D_1, …, D_n (viz Sylvestrovo kritérium) nezáporné, je tato matice pozitivně semidefinitní.

n) Determinant pozitivně definitní matice je kladný.

o) Determinant negativně definitní matice je záporný.

Výsledky $\boxed{A}$

P = pravdivé tvrzení, N = nepravdivé tvrzení

a) N (ne nulový vektor, ale nulu)

b) N (výsledkem není vektor, ale číslo)

c) N (nulovému vektoru přiřazuje nulu)

d) N (např. matice $\begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix}$ je negativně definitní)

e) N (ale nemůže obsahovat nulu na diagonále)

f) N (platí obráceně: pozitivně semidefinitní matice má na diagonále všechna čísla nezáporná)

g) P

h) N (např. matice $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ je indefinitní)

i) P

j) N (matice může být i indefinitní, jsou li na diagonále kromě nuly kladná i záporná čísla)

k) P

l) N (negativně definitní je v případě, že je $D_1 < 0$, $D_2 > 0$, $D_3 < 0$ atd.)

m) N (může být i indefinitní – viz Příklad 13.10)

n) P (podle Sylvestrova kritéria jsou všechny determinanty D_i kladné, tedy i determinant celé matice)

o) N (podle Sylvestrova kritéria jsou determinanty D_i sudých řádů kladné a lichých záporné – záleží tedy na řádu celé matice)

$\boxed{B}$ Praktická část

$\boxed{1}$ Rozepište součin $\bar{x}^T A\bar{x}$ pro symetrickou matici A:

a) $\begin{pmatrix} -2 & 3 & 1 \\ 3 & 4 & 2 \\ 1 & 2 & -1 \end{pmatrix}$ b) $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ c) $\begin{pmatrix} 1 & -1 & 1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix}$ d) $\begin{pmatrix} 1 & -2 & 1 & -3 \\ -2 & 3 & 0 & 2 \\ 1 & 0 & 4 & -1 \\ -3 & 2 & -1 & 5 \end{pmatrix}$

$\boxed{2}$ Ze součinu $\bar{x}^T A\bar{x}$ ($\bar{x} \in \mathbb{R}_4$) určete symetrickou matici A:

a) $\bar{x}^T A\bar{x} = -4{x_0}^2 - 7{x_1}^2 - 7{x_2}^2 - 7{x_3}^2 + 4x_0x_1 - 4x_0x_2 - 4x_0x_3 - 2x_1x_2 - 2x_1x_3 + 2x_2x_3$

b) $\bar{x}^T A\bar{x} = -6{x_0}^2 - 6{x_1}^2 - 9{x_2}^2 - 9{x_3}^2 - 8x_0x_1 + 4x_0x_2 - 4x_0x_3 - 4x_1x_2 + 4x_1x_3 - 2x_2x_3$

c) $\bar{x}^T A\bar{x} = {x_0}^2 + {x_1}^2 - 4{x_2}^2 - 5{x_3}^2 - 2x_0x_1 + 6x_0x_2 - 2x_0x_3 + 4x_1x_2 - 6x_1x_3 + 2x_2x_3$

d) $\bar{x}^T A\bar{x} = 4{x_0}^2 + 3{x_1}^2 + 2{x_2}^2 - 2{x_3}^2 + 2x_0x_1 - 4x_0x_2 - 2x_1x_2$

e) $\bar{x}^T A\bar{x} = 3{x_0}^2 + 4{x_1}^2 + 4{x_2}^2 + 5{x_3}^2 - 2x_0x_1 + 4x_0x_2 + 2x_1x_2$

$\boxed{3}$ Určete typy matic

a) $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$, b) $\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, c) $\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, d) $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

e) $\begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$

$\boxed{4}$ Určete typy matic z úlohy $\boxed{2}$.

$\boxed{5}$ Určete typ matice a) převodem na diagonální matici, b) užitím Sylvestrova kritéria:

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

Výsledky $\boxed{B}$

$\boxed{1}$ a) $\bar{x}^T A\bar{x} = -2{x_0}^2 + 4{x_1}^2 - {x_2}^2 + 6x_0x_1 + 2x_0x_2 + 4x_1x_2$

b) matice není čtvercová (ani nelze provést součin $\bar{x}^T A\bar{x}$)

c) matice není symetrická

d) $\bar{x}^T A\bar{x} = {x_0}^2 + 3{x_1}^2 + 4{x_2}^2 + 5{x_3}^2 - 4x_0x_1 + 2x_0x_2 - 6x_0x_3 + 4x_1x_3 - 2x_2x_3$

$\boxed{2}$

a) $\begin{pmatrix} -4 & 2 & -2 & -2 \\ 2 & -7 & -1 & -1 \\ -2 & -1 & -7 & 1 \\ -2 & -1 & 1 & -7 \end{pmatrix}$, b) $\begin{pmatrix} -6 & -4 & 2 & -2 \\ -4 & -6 & -2 & 2 \\ 2 & -2 & -9 & -1 \\ -2 & 2 & -1 & -9 \end{pmatrix}$, c) $\begin{pmatrix} 1 & -1 & 3 & -1 \\ -1 & 1 & 2 & -3 \\ 3 & 2 & -4 & 1 \\ -1 & -3 & 1 & -5 \end{pmatrix}$,

d) $\begin{pmatrix} 4 & 1 & -2 & 0 \\ 1 & 3 & -1 & 0 \\ -2 & -1 & 2 & 0 \\ 0 & 0 & 0 & -2 \end{pmatrix}$, e) $\begin{pmatrix} 3 & -1 & 2 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 1 & 4 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$

$\boxed{3}$ Matice jsou diagonální.

a) pozitivně def., b) indefinitní, c) negativně semidefinitní, d) indefinitní, e) negativně def.

$\boxed{4}$ a) negativně definitní (Sylvestrovo kritérium: $D_1 < 0$, $D_2 > 0$, $D_3 < 0$, $D_4 > 0$),

b) negativně semidefinitní ($D_1 < 0$, $D_2 > 0$, $D_3 < 0$, $D_4 = 0 \implies$ nelze rozhodnout podle Sylvestrova kritéria, nutno převodem matice na diagonální),

c) indefinitní (Sylvestrovo kritérium: $D_1 > 0$, $D_2 = 0$, $D_3 < 0$),

d) indefinitní (Sylvestrovo kritérium: $D_1 > 0$, $D_2 > 0$, $D_3 > 0$, $D_4 < 0$),

e) pozitivně definitní (Sylvestrovo kritérium: $D_1 > 0$, $D_2 > 0$, $D_3 > 0$, $D_4 > 0$)

$\boxed{5}$ a) např. $d_{00} = 1$, $d_{11} = -1$, $d_{22} = 0$, b) $D_1 > 0$, $D_2 < 0$ (D_3 netřeba počítat) – indefinitní

14. DEFINITNOST MATICE Z POHLEDU INFORMATIKA

V Pojmu 13.1 jsme symetrické matice rozdělili dle jejich definitnosti na pět skupin: indefinitní, pozitivně definitní, pozitivně semidefinitní, negativně definitní a negativně semidefinitní. Každá symetrická matice patří do právě jedné skupiny. Výjimku tvoří pouze nulová matice, která je jak pozitivně semidefinitní tak zároveň negativně semidefinitní.

Dělení matic do skupin dle definitnosti je pro informatika ukázkovým příkladem pro definování výčtového datového typu. Výčtový datový typ se používá pro proměnné, které mohou nabývat relativně malého počtu hodnot a tyto hodnoty mají smysluplný význam v přirozeném jazyce. Využití výčtového datového typu zpřehledňuje zdrojový kód.

Ve Zdrojovém kódu 14.1 je definován výčtový datový typ `Definitnost`, který budeme využívat při určování definitnosti matice. Oproti typům matic uvedeným v Pojmu 13.1 zde máme navíc speciální typ matic `nulova` (řádek `02`). Tím je vyřešena situace, že nulové matice jsou jak pozitivně, tak i negativně semidefinitní (viz Poznámka o nulové matici za Pojmem 13.1).

Zdrojový kód 14.1 (definitnost)

```
00 enum Definitnost
01 {
02   nulova,
03   indefinitni,
04   pozitivne_semidefinitni,
05   pozitivne_definitni,
06   negativne_semidefinitni,
07   negativne_definitni
08 };
```

V kapitole 13 jsme si představili dva způsoby určování definitnosti matice. Prvním způsobem je určení definitnosti matice pomocí převodu matice na diagonální matici (viz kapitola 13.3). Tento způsob jsme implementovali v kapitole 14.1. Druhým způsobem je Sylvestrovo kritérium (viz kapitola 13.4), které nedokáže určit semidefinitnost (viz Poznámka za Pravidlem 13.5). Sylvestrovo kritérium je snadné na implementaci, stačí využít existující funkci pro výpočet determinantů. Implementace Sylvestrova kritéria je jedním z úkolů k naprogramování v kapitole 14.3.

14.1 Určení definitnosti převedením matice na diagonální matici

V kapitole 13.3 je řada příkladů demonstrujících problematiku určování definitnosti matice převedením matice na diagonální matici a následným využitím Pravidla 13.3. Nyní si celý proces formalizujeme do podoby funkce `urci_definitnost_matice` (viz Zdrojový kód 14.4). Budeme potřebovat dvě pomocné funkce, které budou matici upravovat při zachování její symetričnosti. První funkce (viz Zdrojový kód 14.2) přeuspořádá matici pro odstupňování tak, aby na pozici pivota byl nenulový prvek a zůstala zachována symetričnost matice. Druhá funkce (viz Zdrojový kód 14.4) provede odstupňování jednoho řádku a jednoho sloupce matice při zachování symetričnosti matice.

Ve Zdrojovém kódu 14.2 je implementována funkce `preusporadej_sym_matici`, která se pokusí v symetrické matici `A` přesunout na pozici danou pivotem prvek s nenulovou hodnotou tak, aby byla zachována symetričnost matice. Při hledání nenulového prvku na pozici pivota bez zachování symetričnosti matice nám postačilo (viz Zdrojový kód 6.3) nalézt řádek s nenulovou hodnotou ve sloupci na pozici pivota a tento řádek prohodit s řádkem obsahujícím pivota. Pro zachování symetričnosti matice bychom museli rovněž prohodit i sloupec odpovídající prohazovanému řádku se sloupcem obsahujícím pivota. Tento postup nevede vždy k cíli. Na místo pivota se dostane diagonální prvek z prohazovaného řádku a sloupce. V případě, že tento prvek je nulový, náš postup selhal. Musíme hledat nenulový prvek na diagonále.

V prvním cyklu (řádek `03`) procházejícím matici po diagonále se pokusíme nalézt první nenulový prvek `A[i][i]` na diagonále (řádek `04`). Pokud prvním nenulovým diagonálním prvkem není pivot (podmínka na řádku `06` je splněna), prohodíme řádek matice obsahující daný nenulový diagonální prvek s řádkem matice obsahujícím pivota (řádek `08`). Následně prohodíme sloupec matice s tímto nenulovým diagonálním prvkem se sloupcem matice s pivotem (řádek `09`). Pokud podmínka na řádku `06` nebyla splněna, byl prvním nenulovým diagonálním prvkem samotný pivot a nebylo třeba matici nijak přeuspořádávat. Nyní je v obou

případech (bez ohledu na podmínku na řádku 06) na pozici pivota nenulový prvek, funkce splnila svůj úkol a končí (řádek 11).

V cyklu na řádku 03 mohla nastat situace, že všechny diagonální prvky byly nulové – podmínka na řádku 04 nebyla nikdy splněna. Tato situace (viz Příklad 13.9) nejde vyřešit prohazováním řádků a sloupců. Pokud první cyklus skončil, aniž by byla ukončena celá funkce, nastala právě tato situace. Musíme využit druhý cyklus.

Pomocí druhého cyklu (řádek 13) budeme procházet matici po řádcích a hledat první nenulový prvek `A[i][pivot]` ve sloupci pod pivotem (řádek 14). Tento nalezený řádek nebudeme s řádkem obsahujícím pivota prohazovat, ale budeme tento nalezený řádek k řádku s pivotem přičítat (řádek 18). Rovněž budeme sloupec odpovídající tomuto řádku přičítat ke sloupci s pivotem (řádek 19). Přičítání řádku i sloupce provedeme pomocí jednoho vnořeného cyklu (řádek 16). Pro větší přehlednost bychom mohli použít cykly dva, jeden pro řádky, druhý pro sloupce. Po provedení těchto dvou přičtení je na pozici pivota nenulový prvek, funkce splnila svůj úkol a končí (řádek 21).

Pokud se ani druhému cyklu (řádek 13) nepodařilo nalézt nenulový prvek, funkce končí, aniž by provedla nějaké přeuspořádání. Na pozici pivota zůstal nulový prvek. Nulový je celý sloupec s pivotem.

Zdrojový kód 14.2 (přeuspořádání symetrické matice pro odstupňování)

```
00 void preusporadej_sym_matici (double ** A, int m, int n, int pivot)
01 {
02   int i,j;
03   for (i=pivot; i<m; i++)
04     if (A[i][i])
05     {
06       if (i != pivot)
07       {
08         prohod_radky_matice(A,m,n,pivot,i);
09         prohod_sloupce_matice(A,m,n,pivot,i);
10       }
11       return;
12     }
13   for (i=pivot; i<m; i++)
14     if (A[i][pivot])
15     {
16       for (j=pivot; j<n; j++)
17       {
18         A[pivot][j] += A[i][j];
19         A[j][pivot] += A[j][i];
20       }
21       return;
22     }
23 }
```

Ve Zdrojovém kódu 14.3 je implementována funkce `odstupnuj_radek_sloupec_matice`, která v symetrické matici `A` nejprve od řádku `n` odstupňuje řádek `r_s` a následně od sloupce `n` odstupňuje sloupec `r_s`. Symetričnost matrice `A` zůstane zachována. Řádky 03 až 07 kontrolují různé chybové stavy vstupních dat. Tyto řádky jsou podrobně vysvětleny v komentářích ke Zdrojovému kódu 6.3. Předpokládaná symetričnost matice se nekontroluje. Pokud je řádek již odstupňovaný, funkce končí (řádek 08).

V prvním cyklu (řádek 09) procházejícím matici po sloupcích odstupňujeme řádek `r_s` od řádku pivot (řádek 10). Vynulujeme prvek nacházející se ve sloupci pod pivotem (řádek 11). Tyto řádky jsou analogické k řádkům Zdrojového kódu 6.3 – lze opět využít komentář k tomu Zdrojovému kódu. Oproti Zdrojovému kódu 6.3 zde došlo ke zjednodušení – pivot se nachází na diagonále.

V druhém cyklu (řádek 12) procházejícím matici po řádcích odstupňujeme sloupec `r_s` od sloupce pivot (řádek 13). Vynulujeme prvek nacházející se v řádku za pivotem (řádek 14). Celý postup je symetrický vzhledem k prvnímu cyklu. Sloupce z prvního cyklu odpovídají řádkům z druhého cyklu a řádky z prvního cyklu odpovídají sloupcům z druhého cyklu.

Zdrojový kód 14.3 (odstupňování řádku a sloupce matice)

```
00 void odstupnuj_radek_sloupec_matice(double ** A, int m, int n, int r_s, int pivot)
01 {
02   int i,j;
03   if (r_s >= m) chyba("Radek se nenachazi v matici.");
04   if ((pivot >= m) || (pivot >= n))
05     chyba("Pivot se nenachazi v matici.");
06   if (r_s == pivot) chyba("Nelze odstupnovat radek sam od sebe.");
07   if (!A[pivot][pivot]) chyba("Pivot nesmi byt nulovy.");
08   if (!A[r_s][pivot]) return;
09   for (j=pivot+1; j<n; j++)
10     A[r_s][j] = A[r_s][j]*A[pivot][pivot] - A[pivot][j]*A[r_s][pivot];
11   A[r_s][pivot]=0;
12   for (i=pivot+1; i<n; i++)
13     A[i][r_s] = A[i][r_s]*A[pivot][pivot] - A[i][pivot]*A[pivot][r_s];
14   A[pivot][r_s]=0;
15 }
```

Ve Zdrojovém kódu 14.4 je implementována funkce `urci_definitnost_matice`, která pro symetrickou matici `A` určí její definitnost. Návratová hodnota funkce je výčtového datového typu `Definitnost` (viz Zdrojový kód 14.1). Nejprve otestujeme, zda matice `A` je symetrická (řádek `04`). Pokud matice `A` není symetrická, je vyvolána chyba (řádek `05`). Místo matice `A` budeme pracovat s její kopií – maticí `B` (řádek `06`). Matici `B` převedeme na diagonální tvar, matice `A` zůstane nezměněna.

V prvním vnějším cyklu (řádek `07`) procházíme matici po diagonále. Nejprve matici přeuspořádáme pro odstupňování, tak aby zůstala zachována její symetričnost (řádek `09`). Pokud je po tomto přeuspořádání na pozici pivota nulový prvek, je nulový celý sloupec s pivotem, a pokračujeme další iterací cyklu (řádek `10`). Ve vnitřním cyklu (řádek `11`) procházíme matici po řádcích. Jednotlivé řádky odstupňujeme od řádku s pivotem (řádek `12`).

Po skončení prvního vnějšího cyklu je matice `B` nejen odstupňovaná, ale vzhledem k zachovávání její symetričnosti v jednotlivých krocích je matice `B` diagonální. Dle Pravidla 13.3 můžeme určit definitnost matice – pomocí druhého vnějšího cyklu.

Druhý vnější cyklus (řádek `14`) prochází matici po diagonále a testuje hodnotu diagonálního prvku `B[i][i]` na nulovost (řádek `16`), zápornost (řádek `17`) a kladnost (řádek `18`). Na začátku funkce (řádek `03`) jsme si deklarovali a inicializovali tři proměnné, které nyní (řádky `16` až `18`) využijeme. Tyto proměnné nabývají hodnot `0` (nepravda) a `1` (pravda). Proměnná `semi` dává odpověď na otázku, zda se na diagonále někde vyskytla nula. Proměnná `pozitivne` dává odpověď na otázku, zda všechny doposud zpracované diagonální prvky byly nezáporné. Proměnná `negativne` dává odpověď na otázku, zda všechny doposud zpracované diagonální prvky byly nekladné.

Matici `B` již nepotřebujeme, smažeme ji (řádek `20`). Na řádcích `21` až `28` z hodnot proměnných `semi`, `pozitivne` a `negativne` stanovíme definitnosti matice. Pokud všechny prvky na diagonále matice byly nekladné (řádek `21`), můžou nastat tři situace (řádky `22` až `24`). Všechny prvky na diagonále matice byly nezáporné (řádek `22`), matice obsahovala samé nuly, matice je nulová. Diagonála matice obsahovala alespoň jednu nulu (řádek `23`), matice je negativně semidefinitní. Diagonála matice neobsahovala žádnou nulu (řádek `24`), matice je negativně definitní. Pokud nebyla splněna podmínka na řádku `21`, otestujeme, zda všechny prvky na diagonále matice byly nezáporné (řádek `25`). Pokud je tato podmínka splněna, je matice dle hodnoty proměnné `semi` buď pozitivně semidefinitní (řádek `26`) nebo pozitivně definitní (řádek `27`). Pokud nejsou splněny podmínky na řádcích `21` a `25`, obsahuje diagonála matice jak kladné, tak záporné prvky, matice je indefinitní (řádek `28`).

Zdrojový kód 14.4 (určení definitnosti matice)

```
00 enum Definitnost urci_definitnost_matice(double ** A, int m, int n)
01 {
02   double ** B;
03   int i, j, semi = 0, pozitivne = 1, negativne = 1;
04   if (!matice_je_symetricka(A,m,n))
05     chyba("Matice musi byt symetricka.");
06   B = kopie_matice(A,m,n);
07   for (j=0; j<n; j++)
08   {
09     preusporadej_sym_matici(B,m,n,j);
10     if (!B[j][j]) continue;
11     for (i=j+1; i<m; i++)
12       odstupnuj_radek_sloupec_matice(B,m,n,i,j);
13   }
14   for (i=0;i<m;i++)
15   {
16     if (!B[i][i]) semi = 1;
17     if (B[i][i]>0) negativne = 0;
18     if (B[i][i]<0) pozitivne = 0;
19   }
20   smaz_matici(B,m,n);
21   if (negativne)
22     if (pozitivne) return nulova;
23     else if (semi) return negativne_semidefinitni;
24     else return negativne_definitni;
25   else if (pozitivne)
26     if (semi) return pozitivne_semidefinitni;
27     else return pozitivne_definitni;
28   else return indefinitni;
29 }
```

Poznámka (přehlednost zdrojových kódů)

Ve Zdrojovém kódu 14.4 je několik situací, kdy jsme z důvodu přehlednosti zdrojového kódu použili na první pohled neefektivní programátorskou konstrukci. Podmínka na řádku `17` by mohla být v `else` větvi předchozí podmínky. Podmínka na řádku `18` by nemusela být uvedena explicitně, postačilo by použít `else` větev předchozí podmínky. Jednotlivé řádky těla druhého vnějšího cyklu (řádky `14` až `19`) by mohly být součástí prvního vnějšího cyklu (řádky `07` až `13`).

14.2 Úkoly k naprogramování

1 Ve svém oblíbeném programovacím jazyku implementujte všechny funkce uvedené v kapitole 14 – určení definitnosti matice pomocí převodu na diagonální matice.

2 Implementujte postup určení definitnosti matice dle Sylvestrova kritéria (viz kapitola 13.4). Vhodným způsobem modifikujte výčtový typ `Definitnost`.

3 Na příkladech náhodných symetrických matic otestujte funkce implementované v bodech 1 a 2. Porovnejte rychlost výpočtu v závislosti na řádu matice u metod výpočtu definitnosti matice pomocí převodu na diagonální matici a Sylvestrova kritéria.

4 Pomocí obou implementovaných metod určete definitnost matic z kapitoly 13.5 úkolů 3 a 4. Zkontrolujte správnost výsledků.

LITERATURA

[1] BATÍKOVÁ, B. a kol.: *Matematika pro 4MM101 s aplikačními příklady*, Oeconomica, Praha, 2006, ISBN 80-245-1097-9.

[2] BATÍKOVÁ, B. a kol.: *Učebnice matematiky pro ekonomické fakulty*, Oeconomica, 2009, ISBN 978-80-245-1539-7.

[3] BEČVÁŘ, J.: *Lineární algebra*, MatfyzPress, 2000, ISBN 80-85863-61-8.

[4] BICAN, L.: *Lineární algebra a geometrie*, Academia, 2009, ISBN 978-80-200-1707-9.

[5] BLYTH, T. S., ROBERTSON, E. F.: *Basic Linear Algebra*, Springer Verlag London, 2002, ISBN 978-1-85233-662-2.

[6] BUDINSKÝ, P., HAVLÍČEK, I.: *Matematika pro vysoké školy ekonomického a technického zaměření*, Praha: VŠFS, 2005, ISBN 80-86754-45-6.

[7] BUDINSKÝ, P., HAVLÍČEK, I.: *Sbírka příkladů z matematiky pro vysoké školy ekonomického a technického zaměření*, Praha: VŠFS, 2005, ISBN 80-86754-52-9.

[8] CERF, V.: *RFC 20: ASCII format for Network Interchange*, 1969.

[9] COUFAL, J., ULRYCHOVÁ, E.: *Šifrování a dešifrování užitím regulárních matic.* In: Mundus Symbolicus, Praha: VŠE, 1996, pp. 15–19. ISSN 1210–809X.

[10] DVORSKÝ, J., KRÁTKÝ, M.: *Multi-dimensional Sparse Matrix Storage.* In: V. Snášel, J. Pokorný, K. Richta (Eds.): Proceedings of the Dateso 2004 Workshop, pp. 152161, ISBN 80-248-0457-3.

[11] FRIEDBERG, S. H., INSEL, A. J., SPENCE, L. E.: *Linear Algebra*, Prentice Hall, 2003, ISBN 0-13-008451-4.

[12] GRIMALDI, R. P., LOPEZ, R. J. : *The Matrix-Tree Theorem.* In: Linear Algebra Gems – Assets for Undergraduate Mathematics, ed. by Carlson, D., C. R. Johnson, D. C. Lay, A. D. Porter, Mathematical Association of America, 2002, ISBN 0-88385-170-9.

[13] HENZLER, J. a kol.: *Matematika pro ekonomy*, Oeconomica, 2007, ISBN 978-80-245-1284-6.

[14] IEEE Computer Society: *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2008, ISBN 978-0-7381-5753-5.

[15] KOUBKOVÁ, A., PAVELKA, J.: *Úvod do teoretické informatiky*, Matfyzpress, 3. vydání, 2003, ISBN 978-80-86732039.

[16] LÁNSKÝ, J.: Programování v jazyce C. Distanční studijní opora, Praha: Vysoká škola finanční a správní, 2010.

[17] MATOUŠEK, J., NEŠETŘIL, J.: *Kapitoly z diskrétní matematiky*, Karolinum, 2010, ISBN 978-80-246-1740-4.

[18] OLŠÁK, P.: *Úvod do algebry, zejména lineární*, FEL ČVUT, Praha, 2007, ISBN 978-80-01-03775-1.

[19] POOLE, D.: *Linear Algebra: A Modern Introduction*, Brooks/Cole, 2003, ISBN 0-534-34174-8.

[20] TŮMA, J.: *Gaussova eliminace.* In: Lineární algebra I., pp. 12-31, 2003.
Přístupné z http://www.karlin.mff.cuni.cz/ tuma/2003/NNLinalg2.pdf.

[21] ULRYCHOVÁ, E.: *Matrices in Simple Economic Tasks.* In: Aplimat 2011 [CD-ROM], Bratislava: Faculty of Mechanical Engineering, 2011, pp. 1399–1406. ISBN 978–80–89313–51–8.

[22] ULRYCHOVÁ, E.: *Simple economic applications of matrices.* In: Acta academica karviniensia, 13(3), 2013, pp. 190–199, ISSN 1212–415X.

REJSTŘÍK

V rejstříku jsou v odkazech do informatických částí knihy čísla stran odlišena typem písma – je použit stejný font písma jako pro zdrojové kódy.