

POČÍTAČOVÁ CVIČENÍ ŠKOLA MATEMATICKÉHO MODELOVÁNÍ 2014

Petr Beremlijski, Rajko Ćosić, Marie Sadowská, Matyáš Theuer



evropský
sociální
fond v ČR



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

POČÍTAČOVÁ CVIČENÍ

ŠKOLA MATEMATICKÉHO MODELOVÁNÍ

Petr Beremlijski, Rajko Čosić, Marie Sadowská, Matyáš Theuer



Katedra
aplikované matematiky

Katedra aplikované matematiky
Fakulta elektrotechniky a informatiky
VŠB - Technická univerzita Ostrava
2014

Poděkování

Autoři textu děkují Doc. RNDr. Jiřímu Bouchalovi, Ph.D. za pozorné přečtení a řadu cenných připomínek k tomuto textu.

Tato akce byla podpořena z prostředků projektu Matematika s radostí (registrační číslo CZ.1.07/1.1.00/26.0042). Jeho webová stránka je <http://msr.vsb.cz>.



evropský
sociální
fond v ČR



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Abychom se mohli věnovat numerickému řešení matematických úloh, potřebujeme vhodné prostředí, které nám to umožní. A tak jako fyzici či chemikové mají své laboratoře, mají i numeričtí matematici svou Maticovou laboratoř¹ - Matlab. Podrobně se tomuto pracovnímu prostředí a jeho příkazům věnuje přiložený Matlabovský slabikář [2] nebo také úvod textu [1]. My si v tomto textu uvedeme pouze stručný přehled matlabovských proměnných a příkazů, které budeme potřebovat.

Prostředí

- *help, demos, intro, who, whos, clear, size, length*

Proměnné

- Skaláry
- Vektory
- Matice

Příkazy

- Skalární funkce - *sin, cos, tan, cot, exp, log, abs, sqrt, round*
- Vektorové funkce a generování vektorů - *max, min, sort*
- Maticové funkce a generování matic - *det, rand, ones, zeros, eye*
- Skalární operace - *+, -, *, /, ^*
- Maticové a vektorové operace - *+, -, *, ' (transponování), \ (A \setminus v = x \Leftrightarrow Ax = v)*
Operace „po prvcích“ - *.*, .^, ./*
- 2D grafika (vykreslení grafů funkcí jedné proměnné) - *plot, hold on, hold off, figure*
- 3D grafika (vykreslení grafů funkcí dvou proměnných) - *meshgrid, mesh, contour, hold on, hold off, figure*
- Řídící příkazy - *if* (podmíněný příkaz), *for, while* (příkazy cyklu se známým počtem opakování a podmínkou na začátku)
- Relace a logické operace - *<, >, <=, >=, ==, ~=, &, |, ~*
- Skripty a funkce - *function*

¹MATrix LABoratory

Vše si nyní vyzkoušíme při řešení následujících úloh.

Úkol 1.1 Kolik členů harmonické řady² musíme nejméně sečíst, aby tento částečný součet řady měl hodnotu alespoň 10 (15, 20)? ▲

Úkol 1.2 Zkuste odhadnout s využitím Matlabu součet řady

$$\sum_{n=1}^{\infty} \left(\frac{1}{n} \cdot \frac{1}{n+1} \right).$$

▲

Úkol 1.3 Sestrojte grafy následujících funkcí:

- $f(x) := x^2$
- $f(x) := \sqrt{1 - x^2}$
- $f(x) := x^2 \sin\left(\frac{1}{x^2}\right)$
- $f(x) := |x|$

▲

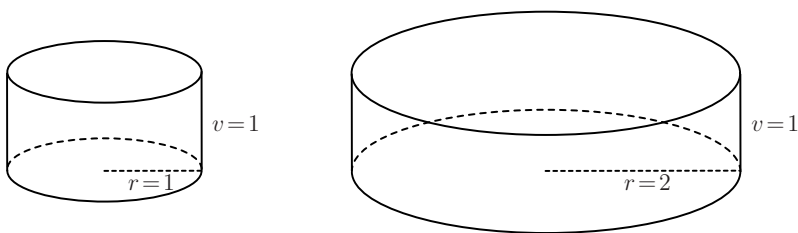
Úkol 1.4 Sestrojte grafy následujících funkcí:

- $f(x, y) := x^2 + y^2$
- $f(x, y) := \sqrt{x^2 + y^2}$
- $f(x, y) := (x^2 + y^2) \sin\left(\frac{1}{x^2 + y^2}\right)$
- $f(x, y) := \sqrt{|xy|}$

▲

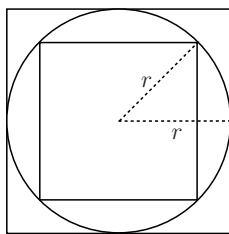
²Řadou (reálných čísel) rozumíme výraz $a_1 + a_2 + \dots + a_n + \dots =: \sum_{n=1}^{\infty} a_n$, kde pro každé $n \in \mathbb{N}$ je $a_n \in \mathbb{R}$. Harmonickou nazýváme řadu $\sum_{n=1}^{\infty} \frac{1}{n}$.

V tomto cvičení se budeme věnovat odhadům Ludolfova čísla π . Nejprve hledejme obsah kruhu o poloměru r (představme si na chvíli, že neznáme příslušný vzorec a nic nevíme o číslu π). Prvním nápadem by mohlo být zhotovení válcových nádob s různými poloměry podstav (např. s poloměry 1 a 2 jednotky) a jednotkovou výškou, viz obr. 1. Objem vody, který se do takových nádob vejde, je roven obsahu podstavy válce, tj. ob-



Obrázek 1: Válcové nádoby

sahu kruhu s poloměrem r . Rychle si všimneme, že pokud zvětšíme poloměr podstavy dvakrát, zvětší se objem čtyřikrát, dalšími experimenty můžeme odvodit, že obsah kruhu je přímo úměrný druhé mocnině poloměru. Také zjistíme, že druhou mocninu poloměru kruhu musíme vynásobit vždy stejnou konstantou, abychom dostali obsah daného kruhu. Tuto konstantu označíme π . Existuje mnoho možností, jak tuto konstantu odhadnout. Snad nejjednodušším způsobem, jak stanovit interval, ve kterém leží π , je vepsat do kruhu o poloměru r čtverec a stejnému kruhu opsat jiný čtverec, viz obr. 2. Délka strany většího



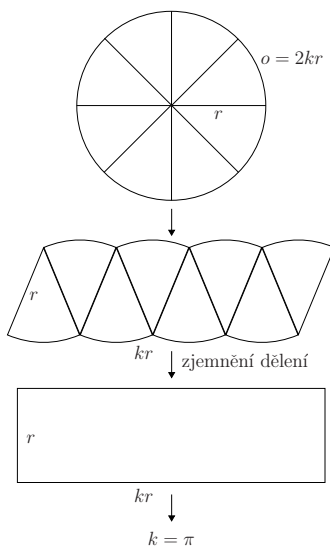
Obrázek 2: Odhad intervalu obsahující π

čtverce je $2r$ a z Pythagorovy věty dále zjistíme, že délka strany menšího čtverce je $\sqrt{2}r$, tudíž obsah většího resp. menšího čtverce je $4r^2$ resp. $2r^2$. Jelikož jsme si už „odvodili“, že obsah kruhu o poloměru r je dán vzorcem πr^2 , pouhým porovnáním obsahů čtverců a kruhu zjistíme, že $\pi \in (2, 4)$. Mnohem rozumnější výsledek získáme, pokud budeme danému kruhu vepisovat n -úhelníky a počítat jejich obsahy. Tuto metodu nazýváme vyčerpávací (exhaustní) a pravděpodobně první ji použil Eudoxos³.

Než se budeme věnovat odhadům čísla π , podívejme se krátce na výpočet obvodu kruhu. Jistě víme, že obvod kruhu je přímo úměrný dvojnásobku jeho poloměru. Otázkou je, jaká

³Eudoxos (410 nebo 408 př. n. l. – 355 nebo 347 př. n. l.) – řecký astronom, matematik a fyzik, student Platóna

je příslušná konstanta úměrnosti k . Na obrázku 3 provedeme přeuspořádání kruhu na útvar, který se pro zjemňující se dělení kruhu blíží obdélníku. Porovnáním obsahu kruhu (πr^2) a obsahu vzniklého obdélníku (kr^2) lze odvodit, že konstanta k je opět rovna π .



Obrázek 3: Stanovení vzorce pro obvod kruhu

Nyní si ukážeme několik způsobů, jak nalézt přibližnou hodnotu čísla π .

Buffonova metoda

Důsledkem řešení tzv. Buffonova⁴ problému s jehlou je aproximace čísla π . Úloha spočívá v opakovaném házení jehly o délce ℓ na rovinu, na které máme vyznačenu síť rovnoběžek se vzdáleností 2ℓ . Jestliže jehlu hodíme n -krát a x -krát nám během těchto pokusů po dopadu zkrátí některou z rovnoběžek, pak číslo

$$\frac{n}{x}$$

aproximuje číslo π .

V roce 1975 Perlman a Wichura⁵ publikovali následující výsledek týkající se přesnosti Buffonovy metody. S pravděpodobností 95 procent nemá chyba aproximace hodnotu větší než $5/\sqrt{n}$. Tzn. například pro 10^4 pokusů nám s pravděpodobností 95 procent chyba nepřekročí hodnotu 0,05.

Úkol 2.1 Implementujte Buffonovu metodu a použijte ji k aproximaci čísla π . Porovnejte vaši aproximaci se „skutečnou“ hodnotou čísla π a určete chybu aproximace. ▲

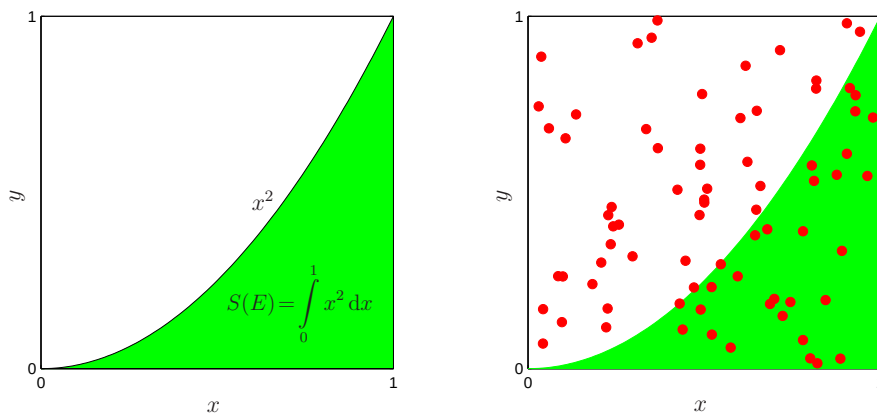
⁴G. L. Buffon (1707–1788) – francouzský přírodovědec

⁵M. Perlman a M. Wichura – američtí matematikové

Metoda Monte Carlo

Monte Carlo je třída výpočetních algoritmů založená na provádění náhodných experimentů. Této metody se často používá pro simulaci fyzikálních a matematických systémů. Výsledkem provedení velkého množství experimentů je obvykle pravděpodobnost určitého jevu. Na základě získané pravděpodobnosti a známých vztahů pak spočítáme potřebné výsledky. Protože metoda vyžaduje generování velkého souboru náhodných dat, je vhodné pro její implementaci použití počítače. Metod Monte Carlo se používá v případě, kdy je příliš pracné nebo nemožné nalézt přesný výsledek jiným způsobem. Její výhodou je jednoduchá implementace, nevýhodou relativně malá přesnost. Metoda byla vytvořena skupinou fyziků pracujících na projektu jaderné pumy v Los Alamos, jméno metody bylo navrženo v roce 1940 von Neumannem⁶.

V matematice se Monte Carlo používá zejména pro výpočet určitých integrálů (zejména vícenásobných určitých integrálů), které je obtížné či nemožné vyčíslit analyticky nebo jinou vhodnou numerickou metodou. Např. obsah plochy E ohraničené grafem funkce $y = x^2$, osou x a přímkou $x = 1$ (tj. $S(E) = \int_0^1 x^2 dx$, viz obr. 4) je možné metodou Monte Carlo aproximovat následujícím způsobem. Nechť náš program generuje náhodně dvojice čísel $[x, y]$, přičemž každé z čísel x a y je vybráno nezávisle z intervalu $\langle 0, 1 \rangle$. Tuto dvojici budeme chápat jako souřadnice bodu, který je náhodně zvolen ve čtverci $\langle 0, 1 \rangle \times \langle 0, 1 \rangle$. Pravděpodobnost toho, že bod leží uvnitř zadaného čtverce, je 1. Pravděpodobnost toho, že bod leží uvnitř podmnožiny E zadaného čtverce, je rovna obsahu plochy E , tj. $S(E)$. Takže obsah plochy, která je podmnožinou zvoleného čtverce, můžeme odhadnout jako pravděpodobnost, že náhodně zvolený bod z daného čtverce leží v této podmnožině.



Obrázek 4: Plocha E a Monte Carlo

Úkol 2.2 Implementujte metodu Monte Carlo pro přibližný výpočet $\int_0^1 x^2 dx$. Získané výsledky porovnejte s analytickým vyčíslením integrálu. ▲

⁶John von Neumann (1903–1957) – významný maďarský matematik

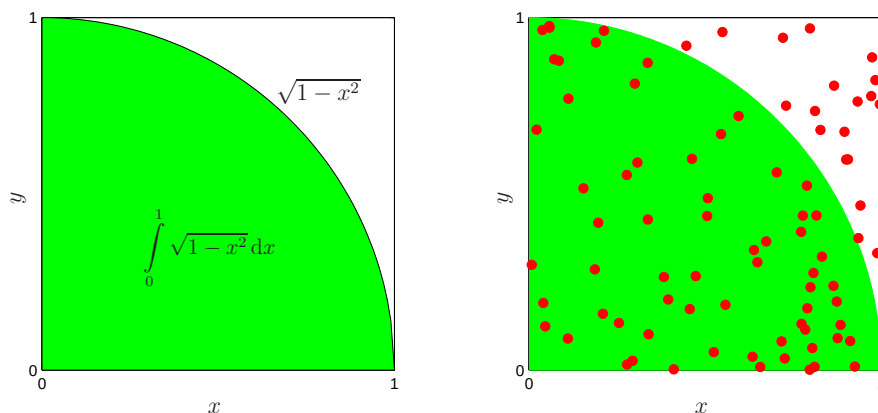
Pro odhad přesnosti metody Monte Carlo platí následující tvrzení. S pravděpodobností 95 procent nemá chyba aproximace hodnotu větší než $1/\sqrt{n}$. Tzn. například pro 10^4 pokusů nám s pravděpodobností 95 procent chyba nepřekročí hodnotu 0,01.

Chceme-li využít metodu Monte Carlo k aproximaci čísla π , vypočteme přibližně touto metodou (například) integrál

$$\int_0^1 \sqrt{1-x^2} dx.$$

Snadno si uvědomíme (viz obr. 5), že tímto způsobem získáme aproximaci hodnoty $\pi/4$.

Úkol 2.3 Implementujte metodu Monte Carlo pro přibližný výpočet integrálu $\int_0^1 \sqrt{1-x^2} dx$ a použijte ji k aproximaci čísla π . Porovnejte vaši aproximaci se „skutečnou“ hodnotou čísla π a určete chybu aproximace. ▲



Obrázek 5: π a Monte Carlo

Připomeňme ještě, že výše uvedené použití metody Monte Carlo je pouze ilustrativní; takto jednoduchý integrál z úkolu 2.3 lze velmi efektivně spočítat vhodnou numerickou kvadraturou.

Aproximace π pomocí číselných řad

Poslední metodou nalezení aproximace čísla π , kterou si v tomto přehledu ukážeme, je využití číselných řad. K této aproximaci použijeme tzv. Gregoryho⁷ řadu, která je rozvojem funkce $\arctg x$:

$$\arctg x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots =: \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^{2n-1}}{2n-1}.$$

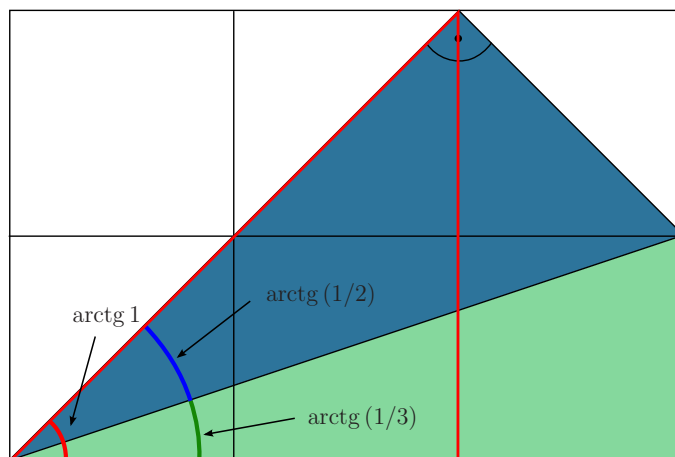
Tato řada má konečný součet pro $x \in \langle -1, 1 \rangle$ (říkáme, že řada konverguje), navíc platí, že čím více je $|x|$ menší než 1, tím méně členů řady potřebujeme použít k nahrazení $\arctg x$ s „uspokojivou“ přesností. První možností, jak aproximovat π , je tudíž nahradit $\arctg 1$ částečným součtem Gregoryho řady, jelikož

$$\arctg 1 = \pi/4.$$

Aproximaci s rychlejší konvergencí získáme, pokud použijeme rovnost

$$\arctg 1 = \arctg (1/2) + \arctg (1/3), \quad (1)$$

viz obr. 6.



Obrázek 6: Ilustrace k odvození vztahu (1)

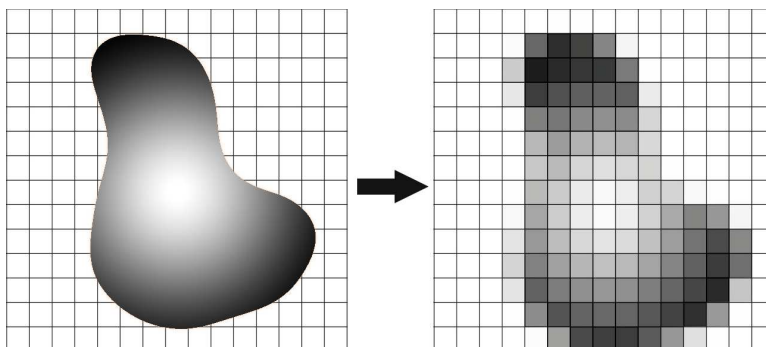
Úkol 2.4 Implementujte metodu, která využije Gregoryho řadu k nalezení aproximace čísla π . ▲

⁷James Gregory (1638 – 1675) – skotský matematik a astronom

V dobách analogových fotoaparátů se vyfocený obraz v každém okamžiku uchovával skutečně jako obraz. Ať už na negativu filmu, nebo po vyvolání na fotografickém papíře či diapozitivu. Dnes je tomu jinak. Při zmáčknutí spouště digitálního fotoaparátu se aktuální scéna před objektivem zachytí pomocí snímacího čipu a uloží jako soubor čísel na paměťovou kartu. I kdybychom tuto kartu rozebrali, obrázky na ní neuvidíme. K jejich zobrazení potřebujeme opět nějaké digitální zařízení, které umí obraz uložený v jedničkách a nulách převést do viditelné formy.

Matice obrazu

Pro jednoduchost se nejprve zabýváme černobílými obrázky - přesněji obrázky ve stupních šedé. V okamžiku exponování dojde k zachycení snímání scény na čip, který je tvořen soustavou miniaturních fotodiod uspořádaných do řádků a sloupců. V závislosti na osvětlení příslušné části čipu každá z diod vyprodukuje určitý náboj, který je změřen, a jeho hodnota je zaznamenána ve formě čísla. V případě JPEG obrázku se jedná o celé číslo od 0 do 255,⁸ přičemž hodnota 0 odpovídá černé a hodnota 255 bílé. Proces rozdělení spojitého obrazu na diskrétní hodnoty v jednotlivých oddělených bodech se nazývá kvantování a vzorkování.



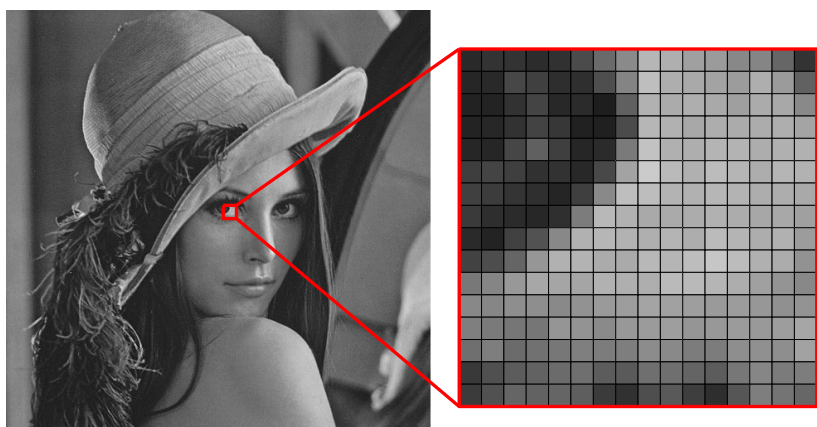
Obrázek 7: Příklad kvantování a vzorkování

Pomineme-li ve skutečnosti binární uložení obrazu, můžeme si jeho digitální podobu představit jako obecně obdélníkovou tabulku čísel, ve které každá hodnota reprezentuje jas určité malé plošky (pixelu) v zachyceném obraze. Tuto tabulku budeme označovat pojmem matice obrazu:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \text{ kde } a_{i,j} \in \{0, 1, 2, \dots, 255\}.$$

Příklad části matice obrazu můžeme vidět na obr. 8.

⁸Tyto hodnoty odpovídají osmibitové reprezentaci čísla.

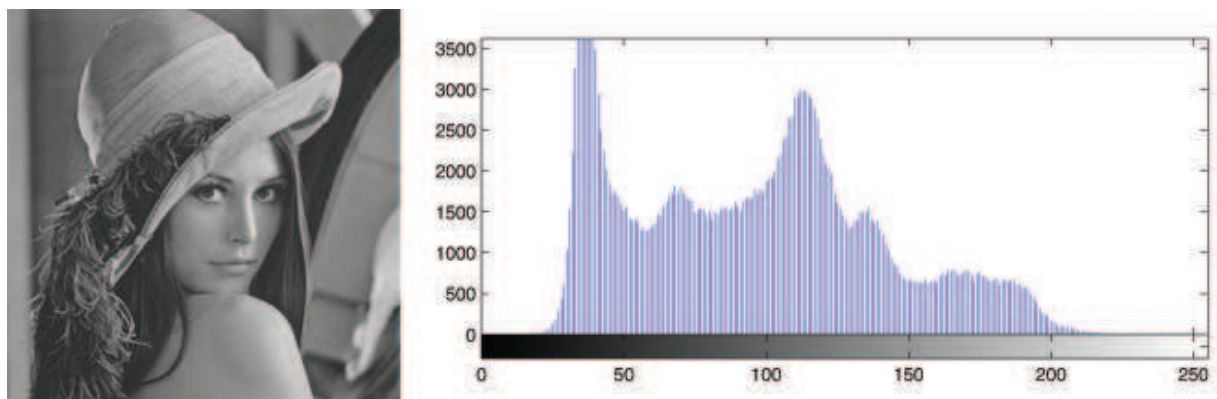


41	51	51	44	49	75	106	139	182	175	160	155	137	134	102	51
38	41	71	64	46	47	78	133	190	174	169	163	156	174	155	97
34	35	50	68	40	43	30	96	177	171	174	169	159	171	168	137
35	39	50	67	55	33	32	75	182	177	168	173	171	172	174	164
38	38	70	96	63	38	43	122	194	176	174	180	179	173	176	177
62	68	63	49	50	40	72	163	203	186	176	188	181	173	181	178
64	60	52	47	37	63	138	190	189	179	177	182	179	173	173	169
58	46	43	39	47	124	184	177	169	170	172	177	172	167	168	160
42	35	65	82	136	177	182	178	175	180	173	187	180	173	177	156
66	77	101	150	177	179	171	170	184	181	189	198	188	176	172	146
120	133	152	167	165	165	177	175	174	186	185	181	173	160	148	136
137	134	142	142	137	146	154	157	164	161	158	154	156	151	145	147
127	122	118	118	149	148	138	152	155	158	149	152	155	153	157	166
111	126	107	113	119	127	124	126	140	137	124	124	143	144	142	162
57	80	92	98	107	111	92	90	111	111	100	92	119	134	123	125
59	81	101	98	101	94	59	49	78	89	62	46	86	126	116	103

Obrázek 8: Část matice obrazu

Histogram

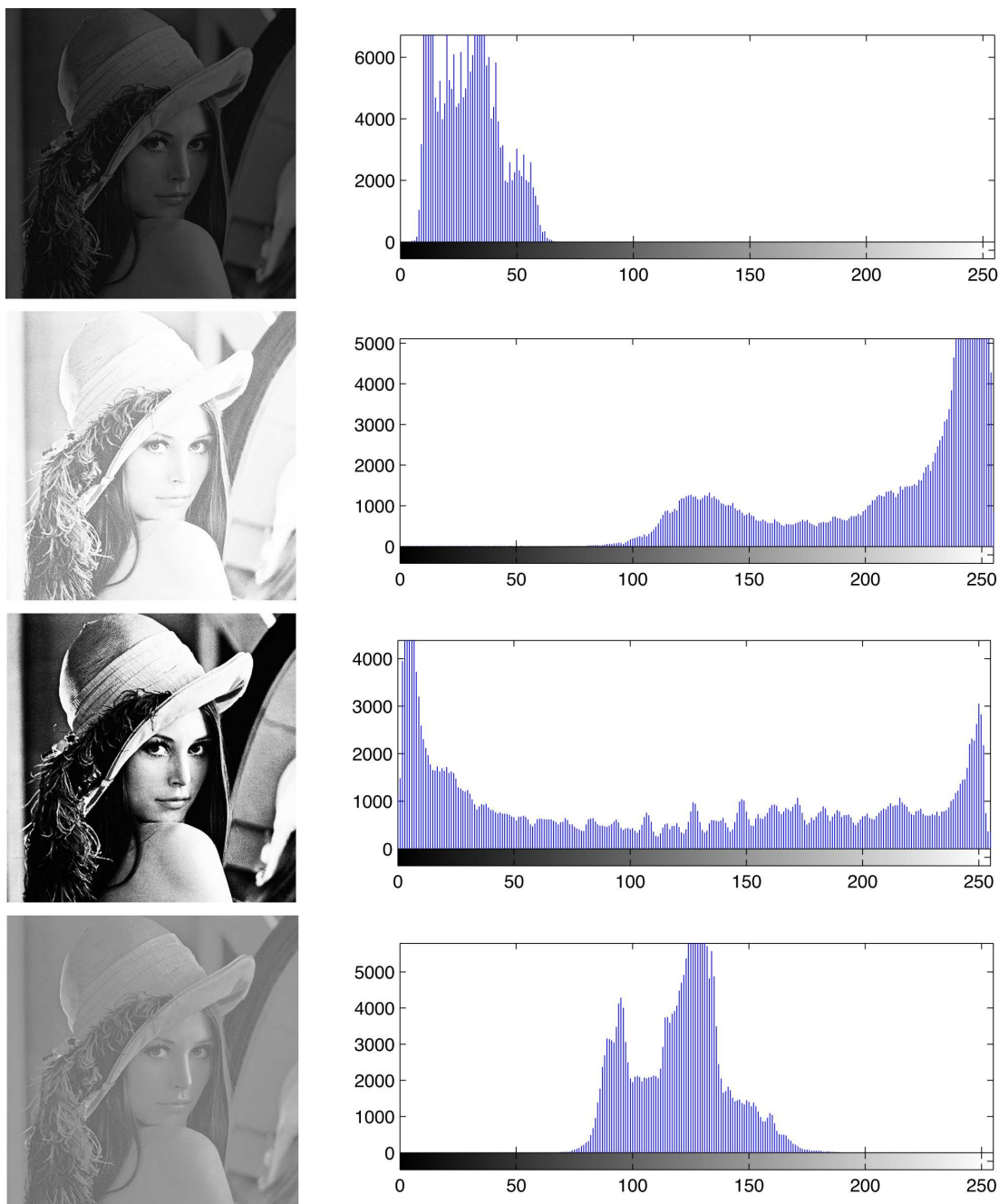
Máme-li obrázek ve stupních šedé (číslích od 0 do 255), může nás zajímat, kolikrát se v obrázku jednotlivé stupně (čísla) vyskytují. Graf četnosti výskytu jednotlivých hodnot v obrázku se nazývá histogram.



Obrázek 9: Obrázek a jeho histogram

Přestože obrázek není svým histogramem jednoznačně definován, můžeme z histogramu

o původním obrázku hodně vyčíst. Tmavé obrázky mají v histogramu velké hodnoty nahromaděné v levé části grafu. Naopak světlé obrázky mají v histogramu velké hodnoty v pravé části grafu. Histogram obrázků s nízkým kontrastem vypadá jako jeden relativně úzký „kopce“, zatímco obrázky s vysokým kontrastem mají většinou dva „kopce“, každý na opačné straně grafu.



Obrázek 10: Příklad různých obrázků a jejich histogramů

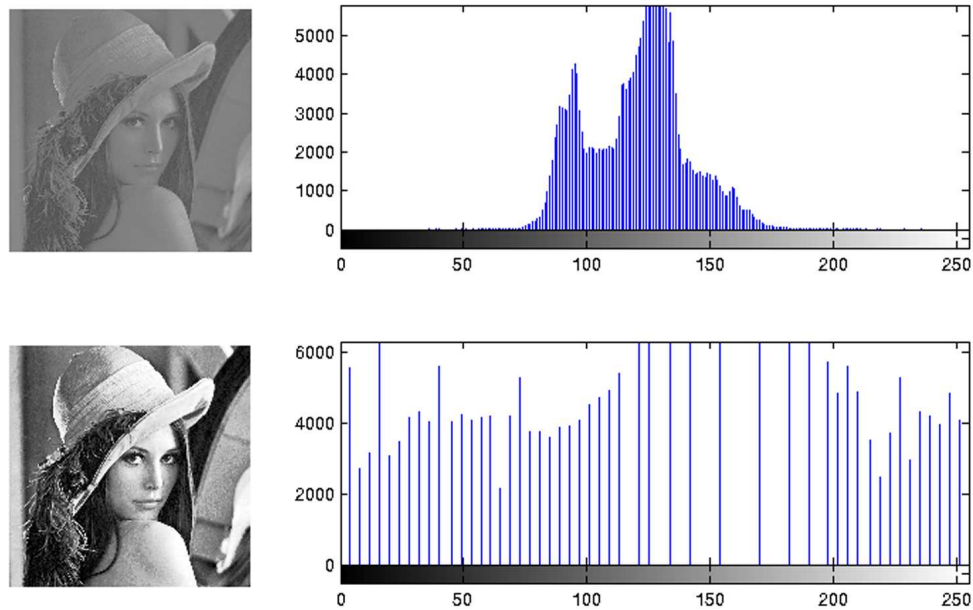
Jednoduché úpravy

Jak jsme si řekli, obrázky jsou v počítači uloženy jako matice čísel, takže místo manipulace s barevnými ploškami nám stačí provádět operace s čísly. Nyní si v jednoduchosti popíšeme některé základní operace s maticemi obrázků. Předpokládejme, že máme obrázek ve stupních šedé, který má 512×512 pixelů. Matice takového obrazu, kterou označíme I_{orig} , má 512 řádků a 512 sloupců. Dohromady je to 262144 čísel, takže je jasné, že nebudeme počítat s každou hodnotou ručně, ale nastavíme nějaké pravidlo, které počítači řekne, co má s jednotlivými hodnotami udělat.

- Ořez obrázku provedeme jednoduše tak, že vybereme jen omezený rozsah řádků a sloupců původní matice. Například pro výřez pravé horní čtvrtiny obrázku I_{orig} vezmeme pouze 1. až 256. řádek a 257. až 512. sloupec. Dostaneme tam matici I_{crop} , která má 256 řádků a 256 sloupců.
- Zesvětlení obrázku provedeme například tak, že k hodnotě každého pixelu v matici I_{orig} přičteme nějaké kladné číslo, například 100. Touto operací se však může stát, že se některé hodnoty ocitnou mimo rozsah 0 až 255. Opravíme to například tak, že všechny hodnoty, které vyjdou větší než 255, zmenšíme právě na 255. Ve všech bodech, kde došlo k takovém ořezu hodnot ovšem ztrácíme obrazovou informaci.
- Ztmavení obrázku můžeme provést obdobně jako zesvětlení, ale musíme zkontrolovat, zda některé hodnoty nevyšly záporné. Pokud ano, nastavíme je na 0. Další možností je vynásobit všechny hodnoty matice obrazu nějakým číslem z intervalu $(0, 1)$. Zde nehrozí, že bychom se dostali mimo rozsah 0 až 255, ale může se stát, že výsledkem nebudou jen celá čísla, a tak je potřeba každou výslednou hodnotu zaokrouhlit na nejbližší celé číslo.
- Negativ obrázku je obraz, který má obrácenou reprezentaci čísel. V klasickém obrázku odpovídá 0 černé a 255 bílé hodnoty mezi těmito čísly odpovídají různým stupňům šedé od tmavé po světlou. V negativním zobrazení je to naopak. Abychom nemuseli přeprogramovat zobrazovací zařízení k obrácení stupnice, můžeme obrátit stupnici přímo v našem obrazu. Hodnoty 0 změníme na 255 a naopak. Všechny hodnoty tedy vypočítáme tak, že původní hodnotu odečteme od čísla 255 a dostaneme právě negativ původního obrázku, který již zobrazíme klasicky.
- Prahování je jednoduchá operace, kde zvolíme $t \in (0, 255)$ (jednoduchý práh). Jednotlivé prvky matice I_{prah} nastavíme na 0, pokud příslušná hodnota matice I_{orig} je menší než t , a na 255, pokud je příslušná hodnota matice I_{orig} je větší nebo rovna t .
- Vyrovnání histogramu je metoda, která slouží k automatickému upravení kontrastu tak, aby byla četnost jednotlivých hodnot rozdělena přibližně rovnoměrně. Tato metoda je automatická a nezohledňuje, o jaký obrázek se jedná, takže výsledek může působit poněkud uměle.



Obrázek 11: Negativ obrázku (vlevo), obrázek po prahování s parametrem $t = 50$ (uprostřed) a $t = 150$ (vpravo)



Obrázek 12: Příklad vyrovnání histogramu obrázku s nízkým kontrastem

Barevné obrázky

Pro reprezentaci barevného obrazu se používají matice rovnou tři - jedna pro každou barevnou složku RGB⁹. Výsledný obraz pak vznikne složením těchto tří obrazů. Hodnota každého pixelu obrazu se tedy skládá ze tří čísel, a tak se tato ploška zobrazuje v jedné z 256^3 možných barev.

Pro jiný formát souboru může matice obrazu obsahovat i jiná čísla než celá čísla od 0 do 255. Můžeme například použít reálná čísla z intervalu $(0, 1)$, kdy řekneme, že černá je 0

⁹RGB je systém rozložení barevného obrazu na červený (R), zelený (G) a modrý (B) kanál.

a bílá odpovídá hodnotě 1. Pokud uložíme hodnoty každého pixelu ve formátu 14-bitových čísel, dostaneme až 16384^3 barev. Běžné zobrazovací zařízení však neumí takový rozsah barev zobrazit, a proto ve většině případů stačí ukládat obrázky do osmi bitů.



Obrázek 13: JPEG obrázek složený z RGB kanálů

Úkol 3.1 Načtěte svůj obrázek do Matlabu pomocí příkazu `imread`. Zobrazte tento obrázek pomocí funkce `imshow` a zobrazte jeho jednotlivé složky R, G a B. Převed'te tento obrázek do stupňů šedé pomocí funkce `rgb2gray` a uložte pomocí funkce `imwrite`. ▲

Úkol 3.2 Proveďte libovolný výřez vašeho černobílého obrázku obsahující 512×512 pixelů a uložte tento oříznutý obrázek. Zobrazte histogram tohoto obrázku pomocí funkce `imhist`. ▲

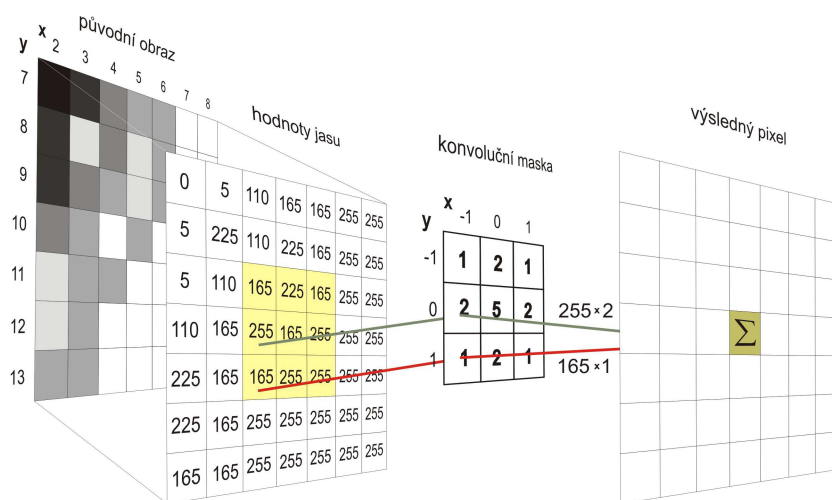
Úkol 3.3 Proveďte zesvětlení a ztmavení vašeho obrázku a vytvořte jeho negativ. Proveďte prahování s prahem 10, 100, 150, 200. ▲

Úkol 3.4 Proveďte zvýšení kontrastu vašeho obrázku pomocí vyrovnání histogramu. ▲

Na závěr poznamenejme, že mnohem více lze o problematice zpracování obrazu nalézt v textu [3].

CVIČENÍ 4: KONVOLUCE

V tomto cvičení si zkusíme aplikovat na obrázky diskretní konvoluci. Diskretní konvoluce je zobrazení, do kterého vstupuje obrázek a takzvaná konvoluční maska. Konvoluční maska specifikuje, co konkrétně konvoluce s obrázkem provede. Protože se v počítačích obrázky vyskytují nejčastěji ve formě matice, jejíž každá hodnota odpovídá jasu v daném pixelu, nepřekvapí nás, když i konvoluční maska bude ze stejného světa. Diskretní konvoluce tedy vezme matici obrázku a každému bodu přiřadí novou hodnotu podle následujícího schématu:



Obrázek 14: Schéma diskretní konvoluce

Jednoduše řečeno: položíme masku na obrázek tak, aby střed masky byl v bodě, kde konvoluci počítáme, přenásobíme každou hodnotou v masce příslušnou hodnotu obrázku a potom vše sečteme¹⁰.

Pomocí konvoluce jsme schopni odstranit z obrázků šum nebo zvýraznit hrany, což je hojně využíváno, chceme-li, aby počítač dokázal sám rozpoznat objekty na obrázku (třeba při zpracování dat z průmyslových kamer).

Čím je maska větší, tím větší okolí bodu nám může zasáhnout do výpočtu. Názorně je to vidět v následujícím příkladu, kdy byla nejdříve použita maska

$$\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

která vlastně novou hodnotu spočítá jako průměr hodnot v okolních bodech. Poté na stejný obrázek aplikujeme masku větší, ale opět s hodnotami, které zprůměrují hodnoty obrázku:

¹⁰Konvoluci vypočteme pomocí předpisu $(f * h)(x, y) = \sum_{r=-k}^k \sum_{s=-k}^k f(x+r, y+s) h(r, s)$, kde funkce f popisuje obrázek a funkce h masku.

$$\frac{1}{25} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$



Obrázek 15: Lena před aplikací konvoluce (vlevo), po konvoluci s maskou 3×3 (uprostřed) a po konvoluci s maskou 5×5 (vpravo)

Jak si můžeme všimnout, naše maska zprůměrováním hodnot na okolí obraz rozmazala, a to tím více, čím větší okolí bodu se do průměru započítalo. Kdyby byla maska stejně veliká jako obrázek, tak by celý výsledný obrázek byl jen v jedné barvě.

V minulém cvičení jsme si ukázali, jak se v Matlabu pracuje s maticemi obrazu. Nyní si zkusíme naprogramovat diskrétní konvoluci podle následujícího algoritmu.

Algoritmus 1: Diskrétní konvoluce

```
obrazek = nacti_obrazek()
N,M = rozmery(obrazek)
maska = matice(n,n)
for i=1..N
    for j=1..M
        restrikce = obrazek((i-n/2):(i+n/2),(j-n/2):(j+n/2))
        novy_obrazek(i,j) = suma_prvku(maska .* restrikce)
    end
end
uloz_obrazek(novy_obrazek)
```

Abychom se vyhnuli přechívnání masky „do prázdna“ při počítání hodnot u krajních bodů, nebudeme konvoluci počítat v bodech, ve kterých by maska přechívala přes okraj obrázku. Vytvoříme si vlastně takový rámeček, na kterém výpočet konvoluce prostě vynecháme.

Úkol 4.1 Naprogramujte diskretní konvoluci podle návodu výše a na obrázcích Lena.png a bricks.jpg (můžete použít i své obrázky z minulého cvičení) otestujte následující konvoluční masky. Obrázky načítejte v odstínech šedi.

- $\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$... odstraní šum
- $\frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$... Gaussova maska
- $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$... Laplaceova maska
- $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$... zvýrazní hrany
- $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$... zvýrazní svislé hrany
- $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$... zvýrazní vodorovné hrany

U posledních čtyř masek přičtete ke každému bodu ještě hodnotu 128, ať obrázky nejsou moc tmavé. ▲

Pro ilustraci praktického využití diskretní konvoluce použijeme na zašuměný obrázek konvoluční masku

$$\frac{1}{25} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

viz obr. 16.

Jak je vidět, použili jsme stejnou masku, na které jsme si ukazovali rozmazání obrázku. Při odstraňování šumu pomocí diskretní konvoluce se rozmazání obrázku pravděpodobně nevyhneme. Čím větší šum potřebujeme z obrázku odfiltrovat, tím více budeme muset obrázek rozmazat.

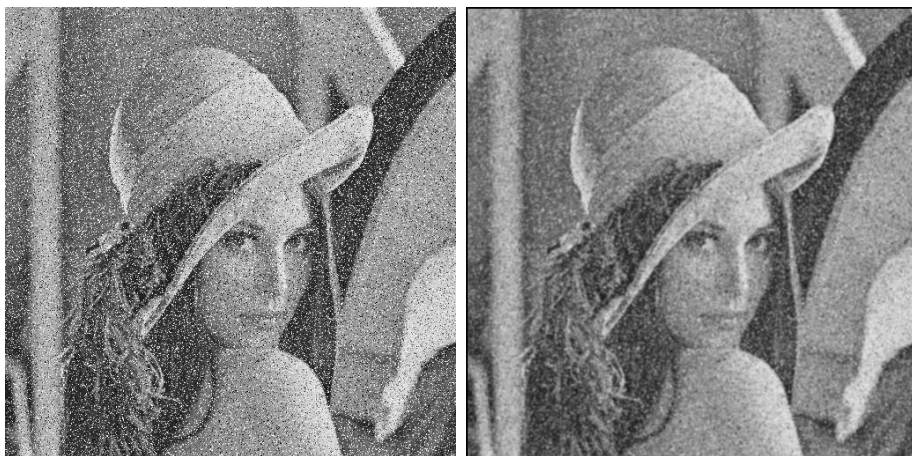


Obrázek 16: Lena před odstraněním šumu (vlevo) a Lena po odstranění šumu (vpravo)

Velmi podobný výsledek bychom obdrželi i při použití Gaussovy masky

$$\frac{1}{273} \cdot \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix},$$

viz obr. 17.



Obrázek 17: Lena před odstraněním šumu (vlevo) a Lena po odstranění šumu pomocí Gaussovy masky (vpravo)

Další informace o využití konvoluce pro zpracování obrazu může zvědavý čtenář získat v textu [3].

Reference

- [1] T. Kozubek, T. Brzobohatý, V. Hapla, M. Jarošová, A. Markopoulos: *Lineární algebra s Matlabem*. Text vytvořený při realizaci projektu „Matematika pro inženýry 21. století“, Vysoká škola báňská - Technická univerzita Ostrava (2012). (<http://mi21.vsb.cz/modul/linearni-algebra-s-matlabem>)
- [2] K. Sigmon: *Matlab Primer*. University of Florida (1993).
- [3] E. Sojka, J. Gaura, M. Krumnikl: *Matematické základy digitálního zpracování obrazu*. Text vytvořený při realizaci projektu „Matematika pro inženýry 21. století“, Vysoká škola báňská - Technická univerzita Ostrava (2012). (<http://mi21.vsb.cz/modul/matematicke-zaklady-digitalniho-zpracovani-obrazu>)

Definice A.1 Necht' jsou dány prvky $a_{1,1}, a_{1,2}, \dots, a_{m,n}$ z dané množiny $\mathcal{F}$. Matice typu (m, n) (stručně $m \times n$ matice) je obdélníková tabulka

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix},$$

kteřá má $m \cdot n$ prvků $a_{i,j}$ uspořádaných do m řádků r_i^A a n sloupců s_j^A , takže

$$A = \begin{bmatrix} r_1^A \\ \vdots \\ r_m^A \end{bmatrix} = [s_1^A, \dots, s_n^A],$$

$$r_i^A = [a_{i,1}, \dots, a_{i,n}], \quad s_j^A = \begin{bmatrix} a_{1,j} \\ \vdots \\ a_{m,j} \end{bmatrix}.$$

Stručně píšeme též $A = [a_{i,j}]$.