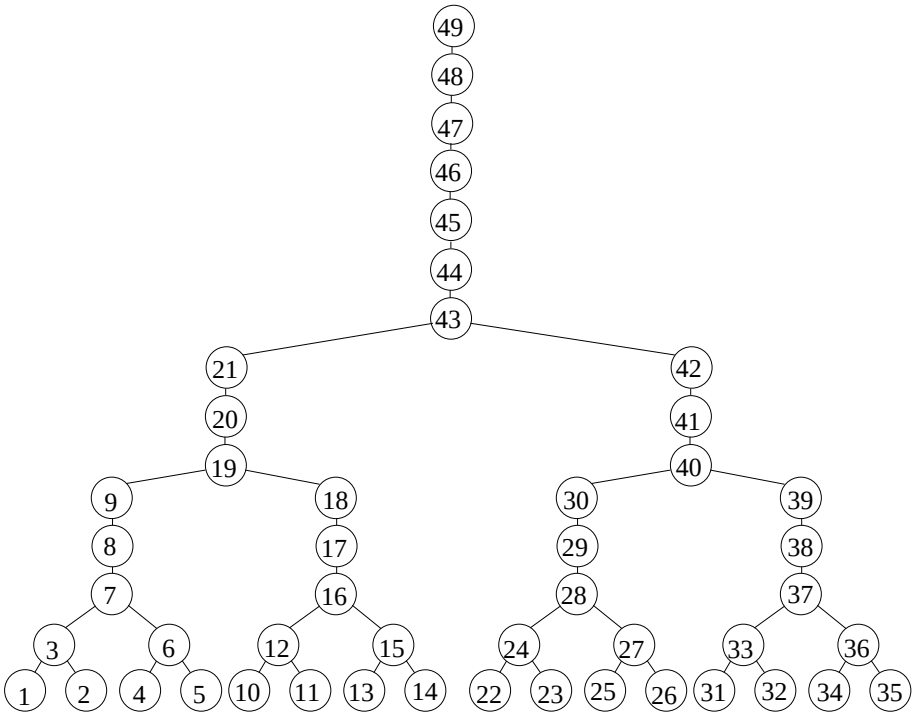
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Řídké matice v řešení soustav lineárních algebraických rovnic

Miroslav Tůma

mirektuma@karlin.mff.cuni.cz



Rozšířené sylaby k předmětu

DRAFT

26. srpna 2020

Obsah

1

Úvod

1.1 Úvod

Tento text je úvodem do pro řešení rozsáhlých a řídkých soustav lineárních algebraických rovnic. Předpokládá se jeho souběžné použití s přednáškami a i z tohoto důvodu jsou jeho některé části shrnuty spíše jako seznam záchytných bodů než jako podrobný popis. První verze textu vznikla pro zimní semestr 2015/2016, kdy začalo přednášení bez používání projektoru, tedy opuštěním konceptu z předchozích let.

Tématicky text vznikl základ textu jako popis klasické teorie přímých metod, jejíž základ se formoval přibližně v šedesátých až devadesátých letech minulého století. Souběžně se ale vyvíjela jak teorie i praxe přibližných rozkladů řídkých matic používaných jako předpodmiňování iteračních metod. Později na tuto klasickou teorii navázala řada výsledků intenzívněji provázaná s technikami aproximace řídkých i hustých matic maticemi nízkých hodností (technikami datové řídkosti). Zahrnování výsledků těchto dvou dalších oblastí je plánovaný směr rozvoje tohoto textu.

1.2 Téma těchto poznámek

1.2.1 Přímé metody

Pojem **přímé metody** má obecně různý obsah v různých kontextech. Zde jej používáme v kontextu řešení soustav lineárních algebraických rovnic. Tyto soustavy vznikají v mnoha praktických aplikacích a způsob řešení přímými metodami je vyjádřením faktu, že postup řešení vychází z principů odvozených z Gaussovy eliminace nebo příbuzného eliminačním procesu. Soudobá prezentace Gaussovy eliminace je obvykle založena na **rozkladu** (faktorizaci) matice soustavy. Pro něj existuje více variant a existují i další možnosti rozkladu, které nejsou odvozeny z Gaussovy eliminace. Výběr varianty rozkladu by měl odpovídat vlastnostem/typu konkrétní matice řešené soustavy, ale souvislosti jsou i směrem k výpočetnímu prostředí. Pro speciální třídy matic jako jsou matice symetrické a pozitivně definitní matice či matice symetrické a indefinitní pak rozklady obvykle využívají specifických vlastností těchto tříd.

1.2.2 Řídké matice

Druhý pojem použitý v názvu předmětu, **řídké matice**, uvozuje, že pro efektivní řešení soustav je třeba opustit klasickou představu o matici jako objektu, kde jsou všechny její prvky rovnocenné. Pro efektivitu algoritmů rozkladu, ale i v mnoha dalších případech práce s maticemi, je podstatná **struktura nulových a nenulových prvků** matice. Tato struktura se dá zachytit pomocí grafů a hypergrafů a ty se pak dají využít v procesu řešení.

Termín řídké matice pak souvisí s **malým množstvím** nenulových prvků v matici, jak o tom budeme hovořit později. Z hlediska aplikací nepředstavuje využívání struktury řídkosti matice okrajový problém, ale je **součástí hlavního proudu výpočetní matematiky** a má silný přesah za samotný rámec rozkladů i do obecných výpočetních postupů přírodovědných a technických aplikací. Analogicky budeme hovořit i o strukturách řídkosti vektorů.

1.3 Návaznost

Předmět vyžaduje k absolvování pouze standardní matematické základy. Přirozeným předpokladem je znalost základních pojmů **lineární algebry** jako jsou matice, vektory, eliminace. Vhodným předpokladem je znalost základních pojmů a motivací **numerické matematiky**. Souvislost, kterou budeme zdůrazňovat, se týká **počítačových architektur**. Přímé metody, ale i metody numerické lineární algebry obecně, se rozvíjejí ruku v ruce s technologickým rozvojem v oblasti výpočetních prostředků. Smysluplnost teorie i praxe vyvíjených metod, které by měly být používány na soudobé výpočetní technice, je touto výpočetní technikou ověřována. I proto zde připomeneme základní prvky používání počítačových architektur. Součástí cvičení je programování některých postupů za účelem ověření zavedených teoretických konceptů.

1.4 Témata po kapitolách

Soupis témat nezahrnuje cílový stav. Předpokládáme další změny v textu především zahrnutím datové řídkosti.

1. Základní použitá terminologie, která se týká především teorie grafů a matic, počítačových architektur a teorie algoritmů. Řídkost matic.

2. Datové struktury pro řídké matice a vektory.

3. Grafy a jejich matice. Od matic ke grafům. Speciální důraz na stromy a kořenové stromy.

4. Řešení soustav lineárních algebraických rovnic Gaussovou eliminační metodou a jejími variantami.

5. Komponenty řídké Choleského faktorizace: maticová a strukturální interpretace, zaplnění při rozkladu.

6. Grafové struktury eliminace: teorie. Eliminační algoritmy.
7. Algoritmická syntéza řešičů. Bloková sloupcová metoda a multifrontální metoda.
8. Nalezení počátečního uspořádání matice pro rozklad. Grafy, přečíslování vrcholů, přeuspořádání matic.
9. Přímé metody řešení soustav lineárních rovnic s řídkou a obecně nesymetrickou maticí.
10. Stabilita řídkých rozkladů. Řešení soustav se symetrickou a indefinitní maticí.
11. Řídká QR faktorizace. Software pro přímé řídké metody.
12. Přibližné maticové rozklady, štěpení a předpodmiňování.

2

Základní použitá terminologie.

V této kapitole připomeneme základní používanou terminologii a značení.

2.1 Terminologie pro množiny, skaláry a matice.

Vektory i **skaláry** značíme obvykle malými písmeny. Jejich rozlišení by mělo být zřejmé z kontextu. **Složky (komponenty)** vektoru budeme značit dolními indexy. Například x_i je i-tá složka vektoru x. Samostatně označované matice nebo množiny budeme obvykle značit velkými písmeny jako jsou A, L nebo U. Jejich složky budou popisovat odpovídajícími malými písmeny s příslušnými dolními indexy jako a_{ij}, l_{ij} nebo u_{kl}. Například, tedy pro matici $A \in R^{n\times n}$ předpokládáme $A = (a_{ij})_{1\leq i,j\leq n}$. Jeden nebo i oba indexy u značení složek vektorů nebo matic jsou někdy nahrazeny intervalem s horním a dolním indexem oddělenými dvojtečkou. Například, podvektor vektoru x se složkami od i do j budeme označovat $x_{i:j}$. Podmatici matice A s řádky o indexech od i_1 do i_2 a sloupci s indexy od j_1 do j_2 zapíšeme $A_{i_1:i_2,j_1:j_2}$. Samotnou dvojtečku použijeme v případě, že uvažujeme celý řádek nebo sloupec. Například, $A_{:i}$ označuje i-tý sloupec matice A a $A_{j:}$ znamená její j-tý řádek. Toto značení je podobné používání těchto objektů v programovém prostředí Matlab^{TM}. V tomto textu bude pro nás praktické používat také druhou konvenci, označení řádků nebo sloupců pomocí "hvězdičkového zápisu". Například, i-tý řádek matice A a j-tý sloupec matice B označíme A_{i*} a B_{*j}. Abychom se vyhnuli nejasnostem, indexy matice budeme někdy oddělovat čárkou. **Pozicí** prvku v matici rozumíme uspořádanou dvojici jeho řádkového a sloupcového indexu, zapsanou třeba (i, j).

Obecně budeme pracovat s reálnými vektory z $\mathbf{R}^n$ a reálnými maticemi z $\mathbf{R}^{n\times n}$. Tento fakt někdy nebudeme připomínat, ačkoli s některými aplikacemi jsou přirozeně spjaty matice a vektory z komplexního oboru a mnoho zde uváděných pojmů, vlastností a algoritmů se dá bez obtíží zobecnit i do komplexního oboru. Hovoříme-li o **číslech** nebo **skalárech**, které reprezentují například prvky matice, myslíme tím vždy prvky $\mathbf{R}$, neuvedeme-li jinak.

Speciální jednotkovou matici (dimenze n) budeme označovat I_n a nulovou matici (se všemi prvky nulovými) označujeme O_n, ale k označení nulového bloku nebo matice budeme používat pro jednoduchost někdy i symbol 0. Následující definice vymezuje některé zde používané třídy matic.

Definice 2.1.1 *Symetrickou matici* $A \in \mathbf{R}^{n\times n}$ *nazveme* **pozitivně definitní** *právě tehdy, platí-li* $x^T Ax > 0$ *pro všechny nenulové vektory* $x \in \mathbf{R}^n$. *Dále tuto matici nazveme* **pozitivně semidefinitní** *právě tehdy, platí-li* $x^T Ax \geq 0$ *pro všechny vektory* $x \in \mathbf{R}^n$.

Definice 2.1.2 **Velikost množiny** X, *neboli počet jejích prvků, budeme označovat výrazem* $|X|$.
Rozkladem množiny X *nazveme množinu* $\mathcal{P}_X = \{P_1, \ldots, P_p\}$ *jejích neprázdných a vzájemně disjunktních podmnožin, které ji sjednocením vytváří. Jinými slovy, rozkladem se nazývá taková množina* $\mathcal{P}_X$ *podmnožin množiny* X, *která splňuje následující podmínky*

- $\emptyset \notin \mathcal{P}_X$,
- $X = \bigcup_{P_i \in \mathcal{P}_X} P_i$,
- $(P_1 \in \mathcal{P}_X \wedge P_2 \in \mathcal{P}_X) \wedge P_1 \neq P_2 \Rightarrow P_1 \cap P_2 = \emptyset$.

Prvky $P \in \mathcal{P}_X$ *sa nazývají třídy rozkladu* $\mathcal{P}_X$.

Definice 2.1.3 *Výrazem* $\binom{X}{2}$ *pro obecně* n*-prvkovou množinu* X *(tedy* $|X| = n$*) označujeme* **množinu všech jejích dvojprvkových podmnožin**. *Formálně tedy píšeme*

$$\binom{X}{2} = \{Y \subseteq X \mid |Y| = 2\}.$$

2.2 Algoritmus a implementace

Důležitou součástí našeho textu jsou výpočetní **algoritmy**. Ty jsou obvykle definovány jako výpočetní procedury transformující množinu zadaných vstupních parametrů na parametry výstupní [35]. Algoritmus lze také chápat jako sekvenci výpočetních kroků vedoucích k řešení problému. Viz například detailní a extenzívní popis algoritmů z obecného pohledu v klasických monografiích [97], [98], [99]. Užitečnými dalšími texty jsou například [35], [100] nebo [115].

Algoritmus s podrobnějším rozpisem jednotlivých kroků budeme nazývat **implementaci**. Zde se obvykle specifikují také důležité datové struktury a jednotlivé kroky se vysvětlují tak, aby byl zřejmý jejich přepis do **počítačového programu**. Hranice mezi algoritmem a implementací **není často příliš zřetelná**, což je ještě zvýrazněno nutností uvažovat v obojím i základní rysy počítačových architektur plánovaných pro implementaci, o kterých bude řeč níže. Dělící čára mezi algoritmem a implementací je tedy obvykle subjektivní.

Pro **popis algoritmů** používáme jednoduchý formalismus, který zachycuje některé společné rysy syntaxe řady programovacích jazyků od Algolu přes Fortran, varianty jazyka C až po moderní skriptovací jazyky jako je Matlab a budeme předpokládat intuitivní srozumitelnost takového formalismu. Obecný cyklus je uvozen klíčovým slovem **for** a ukončen slovem **end** následovanými názvem proměnné tohoto cyklu. Cyklus s podmínkou na ukončení je analogicky vymezen klíčovými slovy **while** a **end while**. Logická podmínka je ohraničena slovy **if** a **end if** s případným uvedením varianty klíčovými slovy **else** nebo

else if. Volané procedury jsou včleněny svými názvy. Některé kroky procedur jsou detailněji popsány slovně. Názvy procedur a operací, kde chceme zdůraznit vstupy a výstupy, obsahují seznam svých parametrů. Nebudeme ale vždy uvádět značení a parametry, které jsou zřejmé z kontextu, jako jsou například daná matice, graf, označení dimenze objektů a podobně. **Návratovou hodnotu** nebo **popis důvodu či způsobu zastavení / návratu** někdy v algoritmech zdůrazníme v příkazu **return** nebo **stop** konkrétním obsahem v závorkách. V algoritmech a definicích budeme občas používat speciální hodnotu **null** pro dodefinování některých zobrazení, funkcí nebo operací. Ve vlastní implementaci či počítačovém programu může být tato hodnota zaznamenána nebo interpretována různě: hodnotou různou od ostatních hodnot nějakého zobrazení (například 0, číslo větší než dimenze matice nebo záporné číslo pro zobrazení do množiny přirozených čísel), chybovým hlášením atd.

2.3 Základní pojmy z počítačových architektur

Zaveďme zde **velmi jednoduchou** představu o výpočetním prostředí, které bude za řadou našich algoritmických úvah. Tradiční (von Neumannovský) model jednoduchého sekvenčního počítače uvažujeme jako propojení tří základních komponent. První z nich je **procesorová jednotka (CPU)**, druhou je **paměť** počítače a poslední je **komunikační systém**, který zabezpečuje jak komunikaci počítače s vnějškem, tak i propojení CPU a paměti. Přestože je tento model velmi zjednodušený, je základem pro popis soudobého počítání na sekvenčních počítačích s rozumně velkou vyrovnávací pamětí, kde je stejně ale řada činností často vykonávána souběžně. Je také východiskem pro výpočty, kde je souběžné zpracování v centru pozornosti. Existuje dále velké množství složitějších modelů počítačů a počítání, které tento model doplňují a přizpůsobují existujícím paralelním počítačovým architekturám, ale to je směr, který v tomto textu nebudeme explicitně sledovat. V následujícím stručně popíšeme komponenty tohoto modelu počítače a naznačíme jejich postupný vývoj od sekvenčního počítání směrem k modernějším počítačovým architekturám, přičemž ale nejmodernější vývoj není naším cílem.

2.3.1 Procesor

Procesorová jednotka obsahuje kromě bloků potřebných pro její vlastní chod a diagnostiku, jako jsou například čítač instrukcí a taktovací jednotka (časovač), také jednotku k provádění vlastních výpočetních operací. Její vývoj od jednoduché aritmeticko-logické jednotky si nyní stručně připomeneme. Uvedeme zde také několik principů, které se postupně a různou měrou integrovaly do dnešních počítačů, kde si přitom nebudeme všímat chronologie vývoje procesorů.

Rychlost provádění operací počítači lze udávat v jejich počtech za časovou jednotku. V celém tomto textu nás hlavně zajímají operace s obecnými numerickými hodnotami a pro ně je běžným modelem výpočtu počítání v **pohyblivé řádové čárce**. Pro popis jeho rychlosti se používají jednotky jako jsou Mflops (milióny operací (M) v pohyblivé řádové čárce (flop) za vteřinu (s)) či Gflops. Bezrozměrová veličina **zrychlení** vyplývající z algoritmické či počítačové inovace, je koeficient, který je poměrem nové rychlosti provádění operací a

původní rychlosti jejich provádění. Poznamenejme, že na moderních počítačových architekturách je měření rychlosti podle počtu operací obvykle značně upozaděno a do hry vstupují veličiny týkající se počtu procesorů, jejich komunikace, latence procesorů i paměti i samotná rychlost ukládání dat.

2.3.2 Vícenásobné funkční jednotky

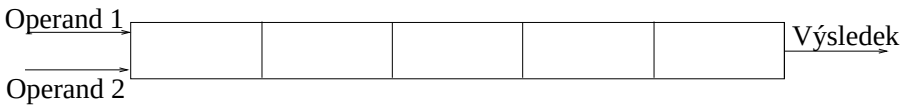
Jeden ze základních způsobů urychlení výpočtů je použití paralelně pracujících výpočetních jednotek. Odedávna měly některé, a postupně prakticky všechny, počítačové procesory možnost souběžného provádění více rozdílných druhů operací, jako jsou sčítání, násobení i logické operace. Jiné počítače již velmi dávno znásobovaly jednotky pro totožné operace, což bylo technologicky dobře proveditelné. Například, první modely japonských superpočítačů ze sedmdesátých a osmdesátých let dvacátého století se vyznačovaly právě velkým počtem funkčních jednotek pro totožné operace. Problém, který ale vzniká při teoretickém zvýšení efektivnosti výpočtu tímto způsobem, je nutnost využít paralelně pracující výpočetní jednotky v maximální míře souběžně. To kladlo a klade nemalé nároky na algoritmy, jejich implementace i na systémový software. Teprve **soulad** všech těchto jednotlivých složek poskytuje vysoké využití celého počítače v počítaných úlohách a tedy i vysoký reálný výkon počítače. Koncept vícenásobných funkčních jednotek se periodicky vracel v průběhu doby v závislosti na technologickém pokroku. Možnost jejich zpracování na systolických polích nebo **superskalárními procesory**, které jsou založeny na plánování operací při běhu programu, využití **výpočetními vlákny** ať již za běhu nebo popisem dlouhými počítačovými instrukcemi nebo využíváním masívního paralelismu grafických výpočetních jednotek jsou moderní postupy, jejichž detailnější popis jde již za hranice tohoto textu.

2.3.3 Pipelining a překrývání operací

Jednoduchý případ **překrytí instrukcí** nazývaný **instrukční pipelining** je motivován bližším pohledem na práci modelu počítače. Výpočetní instrukce se totiž obvykle neprovádějí jako jeden celek najednou v jednom nebo více taktech procesorových hodin, ale vykonávají se v několika fázích. Příkladem může být následující schéma

1. Načtení instrukce,
2. dekódování instrukce,
3. čtení příslušných dat,
4. provádění instrukce (například násobení),
5. a zápis výsledku.

V tomto příkladě je instrukce násobení rozdělena na pět jednodušších operací (**segmentů**), které mohou být prováděny nezávisle, ať už jsou realizovány softwarově nebo technickým vybavením počítače, a mohou pracovat souběžně nad různými daty. V takovém (obvyklém) případě hovoříme o **segmentované instrukci**. Analogicky lze segmentovat i zpracování dat konkrétní instrukcí a v tomto případě hovoříme o **datovém**



Obrázek 2.3.1: *Příklad principu segmentace operací jejich rozdělením na pět jednodušších částí.*

pipeliningu. Příkladem může být zjednodušená aritmetická operaci násobení, při které získáme ze dvou číselných operandů zapsaných na vstupu v exponenciálním tvaru (tvar používaný při počítání v pohyblivé řádové čárce) číselný výsledek například takto:

1. Porovnání exponentů čísel na vstupu,
2. zarovnání mantis vstupů,
3. sečtení exponentů a vynásobení mantis,
4. určení normalizačního faktoru,
5. normalizace výsledku a jeho zapsání v exponenciálním tvaru.

Situaci můžeme znázornit Obrázkem 2.3.1. Zatímco na instrukčním pipeliningu práce nad jedněmi konkrétními daty se může podílet více instrukcí, v datovém pipelingu mohou segmenty jedné instrukce pracovat nad různými daty. Je-li třeba nějaký segment v činnosti pro jedno násobení čísel, nezávislé násobení jiných čísel může využívat jiný segment. Maximální (asymptotické) zrychlení, které z tohoto překryvu vykonávání segmentů instrukcí vyplývá, je rovno počtu segmentů. Nechť je například počet segmentů pro operaci roven k, segmentovaná operace se má vykonat s n datovými jednotkami a všechny tyto operace trvají stejnou dobu. Pak v případě využití segmentace tuto operaci vykonáme v čase $(n + k - 1)\,t$ a bez ní v čase $k\,n\,t$, kde t je doba potřebná na vykonání jednoho segmentu. Výsledné zrychlení v tomto idealizovaném příkladě je tedy $k\,n/(n + k - 1)$.

V závislosti na technické výbavě počítače, se oba typy pipeliningu obvykle kombinují a používají se třebas i mezi odlišnými operacemi, jako jsou sčítání a násobení, kde je většina potřebných segmentů shodná. Terminologicky, jedná-li se o pravidelné překrývání nezávislých operací se stejným počtem segmentů, ve stejném pořadí a přibližně shodným časem na vykonání jednotlivých segmentů, hovoříme obvykle o **pipeliningu**. Jinak používáme spíše obecnější termín **překrývání** operací/instrukcí. Překrývat operace a instrukce je možné ale i na výrazně obecnější úrovni než jsou jen operace v pohyblivé řádové čárce.

2.3.4 Řetězení operací

Řetězení operací je termín, který vyjadřuje přímé propojování výstupu z jedné segmentované operace na vstup jiné segmentované operace místo toho, aby se nejprve čekalo nejprve na její dokončení. Má-li počítačová architektura dostatečnou flexibilitu, může se takto virtuálně prodloužit počet segmentů, které jsou souběžně v činnosti. To vede (asymptoticky, čili pro dostatečně dlouhé vstupní vektory) k dalšímu zrychlení nad standardní

vektorové. Odpovídající zrychlení a rychlost se dříve nazývaly **supervektorové**, ale dnes je takováto možnost součástí všech moderních procesorů. Vzhledem ke složitosti vnitřní stavby procesorů je tento termín spíše historický, protože plánování instrukcí je dnes mnohem sofistikovanější než bylo v počátcích počítání. Analogické principy se používají mnohem explicitněji na zrychlení komunikace v konkrétních topologiích počítačových architektur nebo v obecném síťovém propojení.

2.3.5 Vektorové operace

Princip překrývání na úrovni instrukcí nebo i v širším kontextu je velmi obecná a efektivní možnost zpracování dat / počítání s daty. Naše příklady urychlení datovým a instrukčním pipeliningem nebo jejich kombinací jsou průzračné hlavně v případě zpracování homogenních dat. Matematickým popisem jednoduchých homogenních dat je vektor a to dává název celé strategii - **vektorizaci**, kdy se při formulaci algoritmu využívají co možná nejvíce operace s vektory, jejichž zpracování může operační systém efektivně plánovat. Provádění vektorových operací je obvykle podpořeno prvky hardware jako jsou **vektorové registry**, sloužící k uložení vektorů vstupních dat a výstupních dat určité délky, nicméně v počátcích **vektorových superpočítačů**, jak se počítače se zaměřením na počítání s vektory začaly nazývat, se vektorové operace někdy vykonávaly i přes přímý vstup a výstup z/do paměti, která byla obvykle samozřejmě velmi důmyslně organizovaná. O organizaci paměti budeme hovořit níže. Vektorové instrukce mohou být součástí instrukčního souboru počítače, nebo se sestavují z jednodušších instrukcí programovým vybavením. Zrychlení, kterého se dosáhlo pomocí vektorového provádění některých instrukcí, se historicky nazývalo **vektorovým** zrychlením nebo se hovořilo o vektorovém módu počítače či o vektorové rychlosti provádění operací. V dnešní době jsou vektorové jednotky přirozenou integrovanou součástí CPU a vektorizaci obvykle zabezpečuje nějaká vrstva programového vybavení počítače. Zatímco do tohoto detailu zpracování zde obvykle nenahlížíme, všimněme si jiné podstatné souvislosti. Výše jsme zmínili abstrakci struktury řídkosti vektoru a struktury řídkosti matice. Uvažujeme-li pouze strukturu nulových a nenulových prvků, odhlížíme zároveň od numerických hodnot v maticích a vektorech. Analogicky pro efektivnost je v řadě případů extrémně důležitá taková struktura dat, která umožňuje jejich vektorizovatelnost a která je bližší neodlišování struktury řídkosti od nulových prvků matice. To, jak se s tím vyrovnávají praktické maticové výpočty, budeme diskutovat dále.

2.3.6 Paměť

Paměť se postupem času stala více členěnou a i při velmi hrubém pohledu zahrnuje **více komponent**, jako jsou **registry**, **vyrovnávací paměti** na různých úrovních mezipaměťové i meziprocesorové komunikace (cache), **centrální paměť**, která může být členěna do více segmentů, a **vnější paměť**. V uvedeném pořadí obvykle vzrůstá její velikost a klesá její cena. Na této obecné charakteristice se dlouhodobě nic nemění. Významným faktem, který se prolíná počítáním je to, že **rychlost procesoru** je obvykle výrazně větší, než je vybavovací doba paměti i doba potřebná k čtení z paměti či zapisování do ní a tento rozdíl se technickým vývojem spíše prohlubuje. Obecně rychlost počítání

určují/vymezují stále více (**latence**) na nejrůznějších úrovních výpočetního systému. Aniž bychom zde chtěli jít do detailů, zmiňme, že již v počátcích používání počítačů vznikla nutnost přizpůsobení algoritmů, jejich implementace, ale i systémového software a hardware faktu naprosto rozdílné rychlosti samotných numerických operací a datových přesunů. Následující text stručně komentuje jednotlivé části moderního paměťového systému s přihlédnutím k uvedenému. Jednotlivé komponenty paměťového systému tvoří v počítači obvykle hierarchii, kde registry přímo komunikují s procesorem. Propojením mezi nimi a centrální pamětí je obvykle rychlá vyrovnávací paměť, složená často z více úrovní a rozlišenou na paměť pro instrukce a paměť pro samotná data. Mezi centrální pamětí a vnější pamětí je obvykle také komponenta pracující na principu vyrovnávací paměti.

2.3.6.1 Registry

Registry jsou části paměťového systému, které přímo komunikují s procesorem. Pro vyšší výkon výpočtů je někdy možné využít explicitně příkazy pro registry z vyššího programovacího jazyka. Běžnější je ale jejich programování prostřednictvím překladačů do strojového kódu tak, že se maximálně využijí možnosti překryvu operací nebo vektorizace. To není obvykle práce pro matematika a tvůrce software a zde je to spíše na připomenutí.

2.3.6.2 Vyrovnávací paměť (cache)

Vyrovnávací paměť označuje bloky velmi rychlé paměti umístěné mezi registry a centrální pamětí. Slouží k vyrovnání rychlostí přenosu (latencí přenosu) tím, že data, která se často používají, jsou z ní dostupná s mnohem menší vybavovací dobou, než kdyby se četla z rozsáhlejší, ale pomalejší centrální paměti. Nejsou-li při požadavku procesoru určitá data ve vyrovnávací paměti nalezena, potom hovoříme o výpadku v datech (cache miss, cache fault). Tato data pak musí být získána z pomalejších pamětí této hierarchie, které ale mají větší režii. O důležitosti vyrovnávací paměti se zmíníme jak u schémat na ukládání rozsáhlých matic, tak i u některých algoritmů, které se snaží, aby využití vyrovnávací paměti bylo co nejefektivnější. bez toho, abychom diskutovali různé způsoby mapování mezi vyrovnávací pamětí a centrální pamětí.

2.3.6.3 Centrální paměť

Při čtení z paměti a zapisování do ní hraje velkou roli její hardware a systémový software. Aby se mohly používat fyzicky sousední buňky v paměti, je třeba mezi nimi nějaká prodleva paměťových obvodů, která se nazývá dobou zotavení. Obecněji může být určitá doba na zotavení potřeba i u nesousedních buněk z důvodu konkrétního, ale obvykle hierarchického, uspořádání obvodů, které ovládají paměť. Proto se dají například zapsat dvě čísla za sebou rychleji na dvě nesousední buňky, dokonce s dalším možným zrychlením pomocí překrytí operací zápisu, než na dvě buňky ve stejném fyzickém bloku. S přihlédnutím k velkému rozdílu mezi rychlostí provádění operací v CPU a malé rychlosti (čtení z/zápisu do) paměti se velká základní změna v paměťové organizaci uskutečnila rozdělením centrální paměti na samostatné části, nazývané někdy paměťové banky. Při ukládání po sobě následujících dat se tyto banky cyklicky využívají a tím se celá operace s pamětí zrychlí. Následně se zmenší i pravděpodobnost **paměťových konfliktů**, kdy se obvody

musí zotavit z předcházejících operaci než mohou zapisovat či číst z fyzicky sousedních buněk.

2.3.6.4 Virtuální paměť

Virtuální paměť je koncept mapování a používání paměti, který dává zapomenout na fakt, že velikost paměti je fyzicky omezena menším počtem paměťových jednotek, než je potřeba. Jeho principem je mapování většího prostoru na menší množství fyzické paměti, která je dostupná. Celý obraz tohoto prostoru, který se dělí na menší části, zvané **stránky**, je obvykle uložen na nějakém ještě větším paměťovém médiu (disku, v počítačové síti). Ve fyzické paměti jsou pak přítomny jenom momentálně používané části tohoto obrazu a případně některé další. Jako například stránky, které byly používány bezprostředně předtím nebo stránky, které byly v poslední době použity mnohokrát. Mechanismus řízení virtuální paměti, takzvané **stránkování**, je zabezpečován systémovým software a připomíná techniku řízení vyrovnávací paměti. Vlastní virtualizace paměťového prostoru může být vlastností operačního systému, ale může být také zavedena i jako dodatečné programové vybavení spolu s překladačem z vyššího programovacího jazyka. Analogicky jako u vyrovnávací paměti hovoříme o výpadku stránky (page fault) tehdy, není-li tato přítomna v paměti a musí-li se tedy data získat z nějaké pomalejší části paměťové hierarchie.

2.3.7 Paměť, procesor, algoritmy a implementace

Výše uvedené vysvětlení základní pojmů z oblasti počítačových architektur shrňme následujícím způsobem: K tomu, abychom vytvořili efektivní implementaci, potřebujeme i pro **extrémně jednoduchý** model počítače s jedním procesorem získat **soulad** mezi prací CPU, systémem paměti a použitým algoritmem. V okamžiku, kdy zpracováváme rozsáhlá data, je zapotřebí, aby byla tato data uspořádána tak, aby docházelo k malému množství výpadků stránek i vyrovnávací paměti. Tento požadavek se dá stručně formulovat jako požadavek **časové lokality** dat. Dále je třeba, aby se používaná data přenášela přes vrstvy paměťového systému co nejméně a v okamžiku, kdy se dostanou k procesoru, aby se s nimi vykonalo co nejvíce operací. Jinými slovy je třeba, aby se pokud možno **maximalizoval** poměr počtu operací vůči množství komunikace s konkrétními daty. Protože se komunikace mezi CPU a paměťovým systémem provádí obvykle přesunem bloků, je důležité, aby data, se kterými se takto pracuje, byla zároveň uložena fyzicky/virtuálně blízko sebe. Tento požadavek na blízké uložení dat se také nazývá **prostorová lokalita** dat. K tomu, jaké máme prostředky k dosažení tohoto cíle v konkrétních algoritmech i jejich jednoduchých modelech, se ještě vrátíme. Požadavek prostorové lokality také vyplývá z principu, že počítač vykonává velmi efektivně operace, které se opakují co nejpravidelněji a jsou přitom optimalizovány ke specifickým rysům procesoru, jako je možnost pipeliningu, překrývání operací v širším slova smyslu, vektorovým instrukcím i možnému řetězení. Tento princip nazveme snahou o **regularitu** dat. Od části starostí o lokalitu a regularitu dat nás může oprostit překladač. Nicméně výběr algoritmu, využití možnosti vektorizace či obecné paralelizace jeho případným přeformulováním takovým způsobem, aby explicitní vektory v zápisu algoritmu vystupovaly, musíme zařídit sami.

2.3.8 Efektivita základních lineárně-algebraických operací.

Algoritmy výpočtu můžeme obvykle **implementovat** více různými způsoby, z nichž některé jsou v konkrétních podmínkách vhodnější a jiné méně vhodné. Při znalosti detailů počítačové konfigurace je možné navrhnout velmi efektivní implementaci. Přistupovat detailně ke každé implementaci by si ale vyžádalo nerealisticky mnoho úsilí. To, co je dostatečně efektivní a přitom realistické je implementace vzhledem k nějakému idealizovanému modelu počítače, kde si všímáme jen základních architektonických rysů, jak jsme se o nich zmiňovali výše. Na těchto základech vznikla v komunitě numerické lineární algebry snaha o základní standardizaci implementací některých základních operací lineární algebry pod pojmem BLAS (basic linear algebra subroutines). Vzhledem k tomu, že procedury BLASu jsou zároveň koncipovány jako součást knihovny s určitými univerzálními nároky a mnohdy větším než potřebným množstvím vstupních parametrů, mohou být některé operace implementovány efektivněji, ale obvykle jen ve speciálních případech. Není bez zajímavosti, že procedury BLASu se začaly vytvářet zhruba ve stejné době, kdy vznikal programovací jazyk MATLAB, jehož první verze obsahuje BLASovou standardizaci některých velmi jednoduchých operací. Knihovny obsahující přeložené BLAS procedury jsou součástí mnoha komerčních knihoven, vztažených k určitému počítači, překladači či jejich kombinaci a jsou obvykle mnohem efektivnější než procedury přeložené jednoduše ze zdrojových textů.

2.3.9 BLAS a jeho následníci.

Základní BLAS procedury kromě snahy o maximální lokalitu a regularity operací (vzhledem k počítačovému modelu) splňují i požadavky na **robustnost**, **přenositelnost** a **čitelnost**. Z důvodu potřeby všeobecné srozumitelnosti, ale i pro případnou osobní optimalizaci jsou procedury také přístupné v nějakém vyšším programovacím jazyce Pro jednoduché testy a ladění programů je tedy možné zdrojové kódy překládat společně s aplikačními programy a pro efektivní běhy se poté mohou použít vysoce výkonné knihovní funkce a podprogramy BLASu, které jsou součástí základního programového vybavení počítače.

Historicky, ale i logicky, vznikly celkem tři skupiny algoritmů BLASu, které se týkají hustých vektorů a matic. První z nich, BLAS1, [101] obsahuje nízkoúrovňové operace jako jsou například skalární součin dvou vektorů $x^T y$ nebo operace AXPY $\alpha x + y$ pro vstupní vektory x a y a skalární hodnotu α. Tyto operace, které tvoří BLAS1, budeme nazývat operace typu **vektor-vektor**, neboť právě dva vektory nebo případně jeden vektor tvoří jejich charakteristické operandy. Operace, které zahrnují vektory ve spojení s maticemi, jako jsou například násobení vektoru maticí (GEMV) nebo řešení soustavy lineárních rovnic s trojúhelníkovou maticí, jsou obsahem druhé skupiny, která se nazývá BLAS2 [46]. Třetí skupina, BLAS3 [45], obsahuje operace, ve kterých vystupují dva nebo tři maticové operandy, jako je tomu v příkladě násobení matic (ve skutečnosti jsou obsahem BLASu3 o něco složitější operace; násobení matic (GEMM) například používá kromě škálovacích faktorů tři maticové operandy, kde výsledek násobení přičítá k další matici). Algoritmy v BLASu2 a BLASu3 se také nazývají, po řadě, algoritmy typu **matice-vektor** a **matice-matice**. Zodpovězme si nyní otázku, proč operace rozdělujeme na tyto tři skupiny a

Procedura	*komunikace*	*operace*	přibližný poměr
BLAS 1: AXPY: $y = y + \alpha x$	$3n + 1$	$2n$	$2/3$
BLAS 2: GEMV: $y = Ax$	$n^2 + 2n$	$n(2n - 1)$	2
BLAS 3: GEMM: $C = AB$	$3n^2$	$n^2(2n - 1)$	$2/n$

Obrázek 2.3.2: *Znázornění velikosti komunikace, počtu operací a jejich poměru pro tři procedury BLASu, pro vektory délky n a matice dimenzí $n \times n$. A, B a C jsou matice, x a y jsou vektory a α a β představují skaláry.*

odhlédněme od faktu, že tyto tři skupiny vznikly historicky v tomto pořadí. Uvažujme vybranou operaci každé skupiny a podívejme se na velikost poměru počtu vykonaných operací vůči velikosti komunikace (měřené počtem přenesených skalárů a složek vektorových a maticových operandů z paměti k CPU a zpět do paměti). Výsledek je znázorněn v tabulce na Obrázky 2.3.2.

Vidíme, že uvedený poměr je nejvýhodnější právě pro uvedenou operaci $GEMM$ ze skupiny BLAS3. Obecně, operace z BLASu3 zdaleka nejvíce podporují vyrovnání vysoké rychlosti CPU na jedné straně a znatelně nižší možnosti paměti vybavovat, zapisovat a přenášet data na straně druhé. Blokové operace také snižují i celkovou latenci ve výpočtu. Spolu s důmyslným systémem technického zabezpečení počítače, jako jsou kvalitní virtualizace paměti a výkonný systém hierarchie paměti, přizpůsobení použitých algoritmů (které se může lišit pořadím operací a mít i trochu jiné numerické vlastnosti v důsledku konečné reprezentace reálných čísel) může použití vyšších úrovní BLASu značně přispět k efektivnosti výpočtů.

Cílem standardizace BLASem bylo také umožnit efektivnější implementace dalšího nadstavbového programového vybavení, například základních rozkladů lineární algebry, které jsou na operacích BLASu založeny. Snahou bylo využít zde právě těch efektivnějších operací z BLASu3 jako je tomu například v knihovně LAPACK, která nahradila starší knihovny pro některé operace numerické lineární algebry LINPACK a EISPACK. I u těchto knihoven platí, že jsou vedle sebe k dispozici verze ve zdrojovém kódu i vysoce optimalizované knihovny pro konkrétní počítačové architektury. Přes ohromný pokrok a mnoho různých návrhů, jak systematicky implementovat knihovny pro matice a vektory s nějakou strukturou řídkosti nulových a nenulových prvků, je v této oblasti je stále mnoho nevyřešeného. Částečně to je kvůli mnoha možnostem, jak lze vnitřní strukturu matice využít a částečně i kvůli radikálně novým nárokům soudobých počítačových architektur, které vedou k dalším standardizovaným projektům.

Na standardizaci pomocí BLASu navázala celá řada následných standardizací, které se týkají různých způsobů paralelního počítání. Tyto standardizace tvoří dnes základ dalších nadstavbových knihoven jako jsou, například, ScaLAPACK, Plasma a Magma. Potřebnost takových standardizací je ve velké míře určována soudobým vývojem počítačových architektur.

2.4 Výpočetní složitost

O **efektivitě počítání** jsme hovořili již výše v souvislosti s modelem počítače, kde jsme ji uvažovali z hlediska využití lokality a regularity zpracovávaných dat. Klasickou možností je také založit posuzování efektivity na klasickém pojmu **výpočetní složitosti** podle počtu operací či velikosti paměti potřebné pro konkrétní algoritmus. Oba tyto přístupy mají svá velká omezení právě z výše uvedených důvodů týkajících se velikosti komunikace, latence paměti, latence meziprocesorové komunikace i možností využití segmentace operací. Stanovení výpočetní složitosti algoritmu je pak obvykle součástí mnohem obecnější procedury, **analýzy algoritmu**, ve které se zkoumají všechny prostředky, které konkrétní algoritmus na konkrétní počítačové architektuře potřebuje.

V této sekci si uvedeme základní pojmy týkající se tohoto tématu pouze z pohledu počtu operací nebo velikosti použité paměti. **Složitost nejhoršího případu** je omezení na čas nebo velikost prostoru, které stačí pro vyřešení úlohy s libovolným vstupem algoritmu z množiny jeho přípustných vstupů. V některých situacích je z praktického pohledu zajímavější **průměrná složitost** algoritmu, to jest průměrný spotřebovaný čas nebo prostor, který můžeme předpokládat, že algoritmus využije při nějakém rozložení vstupů algoritmu.

U algoritmů, které v tomto textu popisujeme, je prostorová složitost obvykle snadno nahlédnutelná. Pouze tam, kde není zřejmá, budeme velikost potřebného prostoru zmiňovat a případně uvedeme i techniky, jak tento prostor zmenšit. Přímočaré algoritmy mají někdy vysokou prostorovou složitost, kterou snížíme teprve přechodem ke komplikovanějším algoritmickým postupům. Časová složitost provádění operací je obvykle méně zřejmá a i z tohoto důvodu, hovoříme-li dále o složitosti algoritmů bez bližšího vymezení, máme na mysli jejich časovou složitost.

Pokud nezmíníme jinak, složitost budeme vyjadřovat za předpokladu, že používáme **sekvenční počítač**, který však je v dnešní době iluzí. Přesto je však tento přístup v mnoha případech relevantní. Tuto složitost budeme většinou vyjadřovat počtem elementárních operací se skaláry. Operace mohou být různého typu (aritmetické v pohyblivé řádové čárce, logické, celočíselné kroky při vyhledávacích algoritmech) a, nebude-li to vadit, budeme implicitně používat předpoklad **jednotkové míry operací** (unit-cost random access machine (RAM) model; von Neumann computer), kde všechny tyto operace trvají stejně dlouho. Pro naše účely je tato situace, kdy také nezohledňujeme mimo jiné počet číslic v zápisu čísla logaritmickou mírou, bez újmy na obecnosti. To je dáno i tím, že nás složitost zajímá především asymptoticky v závislosti na velikost vstupních dat.

2.4.1 Popis složitosti algoritmu v asymptotickém případě

Při hodnocení časové nebo paměťové složitosti algoritmu na základě počtu prováděných operací či velikosti potřebné paměti nejsou obvykle podstatné členy nižšího řádu vystupující v jejím přesném popisu jako **funkce** vstupů. Hlavní složkou popisu je **asymptotický** charakter této funkce, tedy to, jak se algoritmus chová v asymptotickém případě při zvětšování velikosti vstupů. Tento charakter je především určen nejrychleji rostoucí komponentou funkce složitosti v závislosti na vstupech. Pro formálnější popis asymptotické závislosti funkcí (velikosti vstupů) budeme používat následujícího značení. Pro jednodu-

chost výkladu zde uvažujeme závislost složitosti na jednom vstupním parametru $n \in \mathbf{N}$. Takovými vstupními parametry jsou například dimenze matice nebo počet jejích nenulových prvků.

Definice 2.4.1 *Řekneme, že reálná funkce $f(n)$ argumentu $n \in \mathbf{N}$ patří do množiny $\Theta(g(n))$ funkce $g(n)$ závisející na témže parametru, existují-li kladné konstanty $l, u \in \mathbf{R}$ tak, že*

$$0 \leq l\ g(n) \leq f(n) \leq u\ g(n)$$

pro všechna dostatečně velká čísla n. Formálně pak pro funkci $f(n)$ splňující uvedený předpoklad budeme psát $f(n) \in \Theta(g(n))$ nebo $f(n) = \Theta(g(n))$.

Někdy říkáme, že množina $\Theta(g(n))$ reprezentuje **asymptoticky těsnou mez** (asymptotically tight bound) pro $f(n)$. Příkladem takového omezení je kvadratická funkce $f(n) = 2n^2 + 4n - 3$, pro kterou platí $f(n) = \Theta(n^2)$.

Analogická mez pro jednostranný horní odhad je uvedena v následující definici.

Definice 2.4.2 *Řekneme, že reálná funkce $f(n)$ argumentu $n \in \mathbf{N}$ patří do $O(g(n))$ funkce $g(n)$ závisející na témže parametru, existuje-li kladná konstanta $u \in \mathbf{R}$ tak, že*

$$0 \leq f(n) \leq u\ g(n)$$

pro všechna dostatečně velká n. Formálně pak pro funkci $f(n)$ splňující uvedený předpoklad budeme psát $f(n) \in O(g(n))$ nebo $f(n) = O(g(n))$.

Obě tyto definice tedy nahrazují funkci f časové nebo paměťové složitosti jednodušší funkcí g, která dává přímočařejší představu o závislosti na vstupních parametrech. V roli funkce g tak budou například pro velikost vstupu daného velikostí n vystupují základní funkce vstupního parametru (etalony) jako jsou $\log n$, n, $n \log n$, n^2, $n^3, \ldots, 2^n, n!, n^n$. Ty tvoří sekvenci stále rychleji rostoucích funkcí a v tomto textu postačí k popisu složitosti většiny algoritmů a implementací.

Velký rozdíl v teoretických, ale i praktických aspektech počítání je mezi algoritmy s asymptotickou složitostí, která se dá vyjádřit **polynomiální funkcí** velikostí vstupů, a mezi algoritmy s asymptotickou složitostí, která roste rychleji než polynomiální funkce. Pro velký význam algoritmů s nejvýše polynomiální složitostí se pro ně zavádí někdy pojem **snadné**. Nicméně, ani existence snadného algoritmu se složitostí například $\Theta(n^3)$ pro velikost vstupu $\Theta(n)$, nemusí být výhrou v praktickém počítání. Řada problémů je pro velká n takovým algoritmem obtížně řešitelná a v praxi je tedy třeba usilovat o nalezení postupu s co možná nejnižší složitostí. V mnoha případech je ideálním postupem algoritmus s nejvýše lineární složitostí, tedy $O(n)$ případě jednoho vstupního parametru n, nebo se složitostí blízké lineární.

Řada našich algoritmů vede na složitost blízkou lineární, kde je multiplikátorem velmi pomalu rostoucí funkce, která se nazývá **inverzní Ackermanova funkce**, uveďme si její definici pro dva vstupní parametry.

Definice 2.4.3 *Definujme pro přirozená čísla $i, j \geq 1$ celočíselnou funkci dvou argumentů $A(i, j)$, $i, j \in \mathbf{N}$ následujícím způsobem*

$$\begin{aligned} A(1,j) &= 2^j \text{ for } j \geq 1 && (2.1)\\ A(i,1) &= A(i-1,2) \text{ for } i \geq 2 && (2.2)\\ A(i,j) &= A(i-1, A(i,j-1)) \text{ for } i,j \geq 2 && (2.3)\\ & && (2.4) \end{aligned}$$

Této funkci budeme říkat **Ackermanova funkce**.

Inverzní Ackermanovu funkci $\alpha(i, j)$ dvou parametrů $i, j \in \mathbf{N}$ pro $i \geq j \geq 1$ pak definujeme vztahem

$$\alpha(i,j) = \min_{k \geq 1}\{k \geq 1 \mid A(k, \lfloor i/j \rfloor) > log_2 j\}. \tag{2.5}$$

Velmi důležitou vlastností inverzní Ackermanovy funkce je její velmi pomalý růst. Jak se dá snadno ukázat, pro naprostou většinu praktických parametrů $i,\ j \in N$ platí $\alpha(i, j) \leq 4$ [152].

2.4.2 Obecné hodnocení složitosti v počítačových vědách

Popišme nyní další pojmy a tvrzení související se složitostí některých algoritmů, se kterými se v tomto textu setkáme a které hledají řešení ve formě minimalizace nějaké účelové funkce vstupu. Takovéto úlohy nazveme obecně **optimalizační**. Samozřejmě i úlohu maximalizace lze v námi studovaných problémech obvykle jednoduše převést na úlohu minimalizace. V popisu použijeme některé termíny z oblasti počítačových věd.

Definice 2.4.4 Rozhodovacím problémem *v nějakém formálním systému nazveme problém, pro který existuje řešení ve formě odpovědi* **ano** *nebo* **ne**.

Definice 2.4.5 *Rozhodovací problém p, viz například formálnější definici v [115] se nazývá* **polynomiálně redukovatelný** *na jiný rozhodovací problém q, existuje-li deterministický polynomiální algoritmus, který konvertuje každý vstup x problému p na vstup y problému q tak, že odpověď rozhodovacího problému q pro vstup y je* **ano** *právě tehdy, je-li odpověď odpověď rozhodovacího problému p pro vstup x je také* **ano**.

Abychom dokázali formálně přesně popsat co znamená řešení problému, pomůžeme si standardním pojmem Turingova stroje, přes nějž jsou formalizovány povolené poměrně jednoduché algoritmické kroky. Bližší popis jeho práce ale vede mimo rámec našich poznámek.

- Výrazem $\mathcal{NP}$ označujeme všechny rozhodovací problémy, pro které je správnost odpovědi **ano ověřitelná** v polynomiálním čase (kde polynom je funkce vstupního parametru) deterministickým Turingovým strojem.
- Výrazem $\mathcal{P}$ pak označujeme všechny rozhodovací problémy, které jsou v polynomiálním čase **řešitelné** deterministickým Turingovým strojem.

Některé důležité problémy výpočetní matematiky jsou označovány jako $\mathcal{NP}$**-úplné**.

Definice 2.4.6 $\mathcal{NP}$**-úplné** *problémy jsou takové* $\mathcal{NP}$ *problémy, na které se všechny problémy v* $\mathcal{NP}$ *polynomiálně redukují.*

Abychom ukázali, že rozhodovací problém p je $\mathcal{NP}$-úplný, musíme tedy ukázat, že $p \in \mathcal{NP}$ a že nějaký jiný problém q, který je $\mathcal{NP}$-úplný, je polynomiálně redukovatelný na p. Jinými slovy, $\mathcal{NP}$-úplné problémy jsou "nejtěžší" mezi $\mathcal{NP}$ problémy.

Třída $\mathcal{NP}$-úplných problémů zahrnuje rozhodovací problémy, pro které není znám žádný polynomiální algoritmus. Existence takového algoritmu pro jeden prvek této třídy $\mathcal{NP}$-úplných problémů by implikovala existenci polynomiálního algoritmu pro jakýkoli NP-úplný problém. Samotná vyjasnění vztahu mezi $\mathcal{P}$ a $\mathcal{NP}$ je slavným otevřeným problémem informatiky.

Problém p je $\mathcal{NP}$**-těžký** právě tehdy, jestliže je každý problém v $\mathcal{NP}$ polynomiálně redukovatelný na p. Na rozdíl od $\mathcal{NP}$-úplných problémů zde nevyžadujeme příslušnost p do třídy problémů $\mathcal{NP}$. Jinými slovy, problém je $\mathcal{NP}$-úplný, je-li $\mathcal{NP}$-těžký a patří do $\mathcal{NP}$. Je-li rozhodovací problém $\mathcal{NP}$-úplný, pak je příslušný optimalizační problém obecně $\mathcal{NP}$-těžký, ale mnoho optimalizačních problémů lze přeformulovat jako problémy rozhodovací o stejné složitosti. Dále, ke každému optimalizačnímu problému existuje jeho rozhodovací varianta. Máme-li, například, v optimalizačním problému minimalizovat funkci $f(n)$, pak jeho rozhodovací variantou je problém, zdali existuje řešení k s velikostí funkční hodnoty $f(k)$ menší než dané číslo L. Jak jsme již zmínili, řada problémů patří do třídy $\mathcal{P}$, ale složitost je stejně příliš vysoká. To implikuje, že v praxi používané algoritmy budou pouze přibližné (heuristiky), které hledají pouze přibližná řešení.

3

Matice a vektory s vnitřní strukturou řídkosti

Koncept **struktury řídkosti** nenulových prvků v maticích a vektorech znamená abstrakci od aktuálních numerických hodnot objektů a uvažováním jejich **hrubší identifikace**, jejich **řídkosti**. Dále uvidíme, že struktura řídkosti matice se dá se dá zachytit a algoritmicky využít pomocí teorie grafů. Číselné hodnoty prvků v základních grafových modelech nejsou brány v potaz, ale mohou být popsány dodatečně, ve formě funkcí nad grafy, které nenulovost prvků popisují.

3.1 Řídké vektory a matice

V literatuře existuje několik různých, ale velmi podobných kritérií pro posouzení, kdy považovat matice nebo vektory za **řídké**. Matice a vektory, které nebudou tato kritéria splňovat, nebudeme považovat za řídké a budeme je nazývat **husté**. Časté je následující kritérium pro označení matice za řídkou.

Definice 3.1.1 *Matici $A \in \mathbf{R}^{n\times n}$ nazveme řídkou, obsahuje-li* **pouze** *$O(n)$ nenulových prvků.*

Tato definice je globální v tom smyslu, že nebere do úvahy rozdíly v počtech prvků na řádcích. Následující Definice 3.1.2 zavádí kritérium, které počítá nenulové prvky v jednotlivých řádcích nebo sloupcích. Vylučuje ale matice, které mají jeden nebo několik řádků nebo sloupců s velkým počtem nenulových prvků a které mohou v některých aplikacích představovat velký výpočetní problém. Příkladem vzniku takového problému jsou husté řádky v (obvykle obdélníkové) matici v problému nejmenších čtverců.

Definice 3.1.2 *Matici $A \in \mathbf{R}^{n\times n}$ považujeme za řídkou, má-li maximální počet nenulových elementů v řádku omezený nějakým celým kladným číslem $r_{\max} \ll n$ nebo má-li maximální počet nenulových elementů ve sloupci omezený nějakým celým kladným číslem $c_{\max} \ll n$.*

γ	$10000^{1+\gamma}$
0.1	25119
0.2	63096
0.3	158489
0.4	398107

Obrázek 3.1.1: *Vztah mezi* γ *a* $10000^{1+\gamma}$ *pro některá* $\gamma \in (0, 1>$.

Podle této definice vidíme, že matice, které vznikají jednoduchými aproximacemi derivací na sítích, kde je počet sousedů vrcholu sítě vždy omezený. To jest, například, matice z pětibodové, sedmibodové či devítibodové diskretizace při numerickém řešení parciálních diferenciálních rovnic metodou konečných rozdílů, můžeme považovat za řídké matice. Podobně je to s maticemi, které vznikají v diskretizacích při řešení úloh metodou konečných prvků. Následující kritérium má výhodu snadnosti použití v **teoretických úvahách**, které se mohou uvažovat například při řešení rovnic vznikajících v řešení elektrických obvodů ale také v teoretických úvahách u diskretizací parciálních diferenciálních rovnic metodou konečných prvků.

Definice 3.1.3 *Matici* $A \in \mathbf{R}^{n\times n}$ *považujeme za řídkou, je-li počet jejích nenulových elementů* $O(n^{1+\gamma})$ *pro nějaké* $\gamma < 1$.

Kritérium z Definice 3.1.3 demonstruje tabulka na Obrázku 3.1.1 udávající pro matici $A \in R^{10000\times 10000}$ počet nenulových prvků $[n^{1+\gamma}]$ pro některá $\gamma \in (0, 1>$. Bez toho abychom nějak shora omezili γ velmi malým číslem, třebas i závislým na dimenzi matice, lze podle tohoto kritéria těžko hovořit o jasné představě, zdali je matice "dostatečně" řídká.

Nejrozšířenějším kritériem řídkosti řídké matice je čistě utilitární pohled založený na následujícím pohledu, který tradice připisuje J.H. Wilkinsonovi.

Definice 3.1.4 *Matici A považujeme za řídkou, můžeme-li při jejím uložení či při operacích s ní* **efektivně využít** *faktu, že část jejích prvků je nulová. Analogicky budeme hovořit o řídkých vektorech.*

Shrňme si důsledky tohoto posledního popisu. Řídkost vztahujeme k **operacím** s maticí. Tatáž matice může být označena jako **řídká vzhledem k jedné operaci** a **hustá k operaci jiné**, také v závislosti na používané výpočetní technice. Můžeme-li řídkosti matice využít při operacích na skalárním počítači, neznamená to, že je užitečné chápat matici jako řídkou na jiné počítačové architektuře, například té s pokročilejšími rysy paralelního zpracování. Situaci si ilustrujme na následujícím příkladě.

Příklad 3.1.1 *Mějme matici* $A \in R^{n\times n}$ *s počtem nenulových prvků* $0.8 \times n^2$. *Zatímco pro* **skalární (sekvenční)** *násobení Av daného vektoru* $v \in R^n$ *se může vyplatit udržovat v paměti pouze nenulové elementy, u operací na* **vektorovém** *počítači tomu tak s velkou pravděpodobností nebude. Zde se může vyplatit uložit celou matici, pokud to dostupná paměť nebo vyrovnávací paměť dovolí, a tak umožnit přímou adresaci prvků matice. Tato*

situace je typická například při počítání s využitím grafických karet, kde latence související s určováním, který prvek je nulový, je obvykle velmi vysoká. Asymptotický výkon počítače při abstrakci od možné řídkosti matice, měřený například v MFLOPsech, může být vyšší.

Definujme si strukturu řídkosti matice formálně.

Definice 3.1.5 *Strukturou řídkosti $\mathcal{S}$ (nebo také $\mathcal{S}(A)$) matice $A \in \mathbf{R}^{m\times n}$ nazveme množinu $\mathcal{S} \in \{1,\dots,m\} \times \{1,\dots,n\}$, jejíž prvky odpovídají nenulovým prvkům matice A.*

Počet nenulových prvků v matici označíme výrazem $|\mathcal{S}(A)|$. Někdy použijeme pro stručnost jen označení $|A|$. Zřejmě lze pro $A \in \mathbf{E}^{m\times n}$ psát

$$\mathcal{S}(A) = \{(i,j) \mid a_{ij} \neq 0,\ i,j = 1, ldots, n\}. \tag{3.1}$$

Pro zachycení řídkosti vektorů budeme používat analogické značení. Pro vektor $x \in \mathbf{R}^n$ definujme **strukturu řídkosti** $Struct(x)$ vektoru x výrazem

$$Struct(x) = \{i \mid x_i \neq 0\}. \tag{3.2}$$

$|Struct(x)|$ nazveme **délkou** nebo také **velikostí** vektoru x. Pojem struktury řídkosti pak můžeme aplikovat na sloupce nebo řádky matice.

Strukturu řídkosti $\mathcal{S}(A)$ matice $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ nazveme **symetrickou**, platí-li následující ekvivalence

$$a_{ij} \neq 0 \Leftrightarrow a_{ji} \neq 0 \text{ pro } i,j = 1,\dots,n. \tag{3.3}$$

3.2 Blokově strukturované matice

V některých případech lze využít toho, že matice má nenulové prvky soustředěny pouze v určitých částech (obdélníkových blocích) matice, které se dají přesně popsat. V tomto případě hovoříme spíše než o obecně řídké matici o matici s **blokovou** strukturou. Definujme si zde jako speciální případ matici blokově diagonální a blokově trojúhelníkovou.

Definice 3.2.1 *Nechť $A \in R^{n\times n}$, $n > 1$ a uvažujme pro $p \in N$, $p \geq 1$ množinu indexů $i_1,\dots,i_p$ takových, že platí $1 = i_1 < \dots < i_p = n$. Označme* **pro jednoduchost v této definici** *výrazem A_{i_k,i_l}, $k = 1,\dots,p-1$, $l = 1,\dots,p-1$ podmatici matice A určenou jejími řádky $i_k,\dots,i_{k+1}-1$ a sloupci $i_l,\dots,i_{l+1}-1$. Zapišme matici A po blocích následujícím způsobem*

$$A = \begin{pmatrix} A_{i_1,i_1} & A_{i_1,i_2} & \cdots & A_{i_1,i_{p-1}} \\ A_{i_2 i_1} & A_{i_2 i_2} & \cdots & A_{i_2 i_{p-1}} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i_{p-1} i_1} & A_{i_{p-1} i_2} & \cdots & A_{i_{p-1} i_{p-1}} \end{pmatrix} \tag{3.4}$$

Řekneme, že A je **blokově diagonální** *nebo je v* **blokově diagonálním tvaru**, *platí-li pro přípustné indexy $A_{ij} = 0, i \neq j$. Řekneme, že A je* **horním blokově trojúhelníkovém tvaru**, *platí-li pro přípustné indexy $A_{ij} = 0, i < j$. Řekneme, že A je* **dolním blokově trojúhelníkovém tvaru**, *platí-li pro přípustné indexy $A_{ij} = 0, i > j$.*

Netriviální bloková struktura matic úzce souvisí s pojmem rozložitelnosti matice, který nyní uvedeme.

Definice 3.2.2 *Řekneme, že matice A dimenze n je* **rozložitelná**, *jestliže ji symetrickou permutací jejích řádků a sloupců můžeme převést na tvar*

$$\begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}, \tag{3.5}$$

kde A_{11} a A_{22} jsou čtvercové netriviální (řádu alespoň 1) matice. Jinými slovy, matice A je rozložitelná právě tehdy, existuje-li permutační matice P odpovídající dimenze taková, že platí

$$P^T AP = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}, \tag{3.6}$$

kde A_{11} a A_{22} jsou čtvercové netriviální reálné matice. Není-li matice A rozložitelná, nazývá se **nerozložitelnou**.

Matice řádu 1 budeme považovat vždy za nerozložitelné. Tvrzení následující poznámky je zřejmé.

Poznámka 3.2.1 *Symetrická rozložitelná matice musí mít blok A_{21} z Definice 3.2.2 nulový a má tedy* **netriviální blokově diagonální tvar.**

Pojem rozložitelnosti matice budeme dále používat a později si jej ještě rozšíříme.

4

Datové struktury pro řídké matice a vektory

Pro práci s řídkými maticemi a řídkými vektory na počítači je zapotřebí dohodnutý způsob jejich ukládání do standardních datových struktur počítače. Zde stručně probereme jak základní principy ukládání řídkých matica uvedeme i některé konkrétní standardní datové struktury, které při implementaci algoritmů s řídkými maticemi lze použít. Přitom zmíníme i ukládání řídkých vektorů, které reprezentují například řádky nebo sloupce řídkých matic.

4.1 Seznam jako schéma pro ukládání řídkých matic

Je-li matice **hustá**, její uložení obvykle využívá standardní koncept hustého **pole**, protože se prvky hustých matic na hustá pole jednoduše zobrazí. **Řídké** matice je třeba uložit tak, aby toto konkrétní uložení bylo paměťově efektivní (jednodušší úloha) a dále umožňovalo také **efektivní algoritmy** nad těmito maticemi (složitější úloha). Přirozenou **abstraktní** datovou strukturou na zachycení řídkých vektorů a matic je **seznam**, který je velmi flexibilní, jak je vidět z následující definice [152].

Definice 4.1.1 Seznam *(list)* $list = (x_1, \ldots, x_k)$ *zachycuje posloupnost prvků. Prvek* x_1 *nazveme* **začátek (první prvek)** *seznamu a prvek* x_k *nazveme* **konec** *seznamu. Obecně budeme zapisovat* $list(i) = x_i$*. Speciálním případem seznamu je seznam prázdný, to jest* $list = ()$*. Délkou seznamu nazveme počet* k*.*

Prvky seznamu mohou být třeba čísla (numerické hodnoty prvků matice), ale i jejich indexy nebo komplikovanější strukturované objekty. Nad seznamem lze definovat formálně řadu operací, viz například [152]. Zmiňme zde některé z nich. K prvkům seznamu potřebujeme především **přistupovat**, ale ne třebas ke všem naráz. Přístupem myslíme možnost čtení/změny příslušného prvku. Seznam je také možné **rozšiřovat** vkládáním prvků na jeho začátek, na konec nebo i dovnitř. Seznam může udržovat nějaké **uspořádání** svých prvků, je-li to potřeba v algoritmech. Seznam může umožňovat vybrat nějakou jeho část, kterou nazveme **podseznam**. Dva seznamy můžeme **zřetězit**

nebo **smísit** tím že z jejich dvou posloupností vytvoříme jednu podle určitého pravidla. Někdy je při zřetězení nebo smíšení třeba některé prvky vynechat. Příkladem může být vynechání prvků se shodnou hodnotou. Různá **omezení** operací seznamu umožňují efektivní provádění konkrétní množiny operací, které nazveme **přípustné** operace, ačkoli i jiné operace je možné provádět, ale některé z nich nebudou efektivní. Podle toho, které operace jsou pro konkrétní seznamy **přípustné** z důvodu zvolené **implementace** s konkrétním uložením abstraktní datové struktury, aby je bylo možné efektivně provádět, rozlišíme různé typy seznamu. Tyto typy se obvykle vyznačují dodatečnými omezeními tak, aby například umožnily implementaci pomocí hustých polí nebo jiných obvyklých prostředků počítačových architektur. Dva takové speciální typy seznamu uvedeme jako definice.

Definice 4.1.2 *Seznam nazveme* **fronta (queue)**, *jestliže umožňuje efektivní provádění*

- *přístupu k prvnímu prvku seznamu,*
- *vynechání prvku ze začátku seznamu a*
- *přidání prvku na konec seznamu.*

Definice 4.1.3 *Seznam nazveme* **zásobník (stack)**, *jestliže umožňuje efektivní provádění*

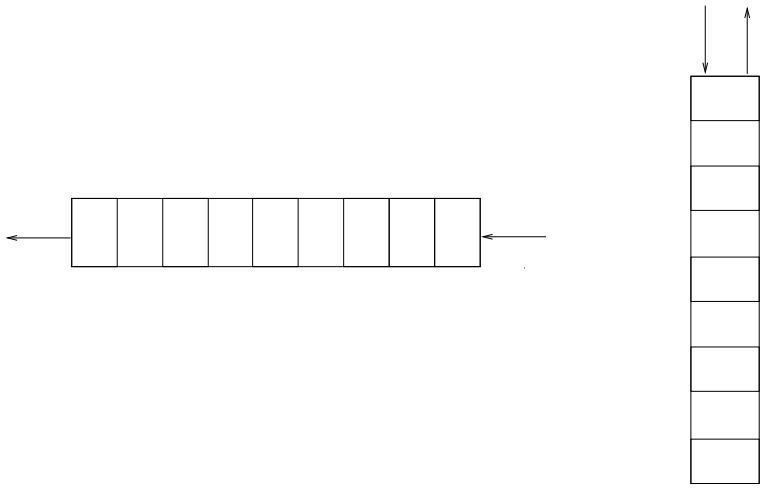
- *přístupu k prvnímu prvku seznamu,*
- *vynechání prvku ze začátku seznamu a*
- *přidání prvku na začátek seznamu.*

Frontu a zásobník schématicky znázorňujeme na Obrázku 4.1.1. Šipkami směřovanými dovnitř struktury, respektive na začátek struktury, jsou naznačeny přípustné (efektivní; jednoduše implementovatelné) operace.

Jak jsme zmínili výše, efektivita se týká jak uložení, tak i práce s daty. Obecně je tedy třeba rozlišit

- efektivitu (**statického**) uložení prvků řídkých matic a dalších pomocných dat,
- efektivitu práce s těmito prvky (**dynamika** uložení) a
- vhodnost uložení seznamu pro použitou počítačovou architekturu (**flexibilita** vzhledem k výpočetním prostředkům).

Efektivitu statického uložení prvků lze měřit velikostí paměti potřebné pro uložení seznamů. Tato velikost by měla být co nejmenší a přitom by rychlost přenosu datových struktur mezi pamětí a výpočetními jednotkami měla být co největší. Jak jsme již viděli dříve, z hlediska latence přenosu dat toto vede k maximálnímu využití hustých bloků dat uvnitř řídké matice. Pro efektivitu **dynamické práce s uloženými daty** je nutné data ukládat dostatečně explicitně tak aby, nebylo nutné složitě nalézat ve vektorech nebo maticích indexy nebo hodnoty jejich nenulových složek a aby bylo možné efektivně přistupovat k řádkům či sloupcům matice podle konkrétního algoritmu. **Vhodnost pro počítačovou architekturu** vyjadřuje, zdali je datová struktura efektivní pro konkrétní počítač, který může obsahovat řadu prvků podporujících souběžné zpracování dat. Zmíněné aspekty budeme v následujícím komentovat u konkrétních návrhů seznamů.



Obrázek 4.1.1: *Schématické znázornění fronty (vlevo) a zásobníku (vpravo). Šipky naznačují jak do nich můžeme vkládat a z nich vybírat data.*

4.2 Seznamy pro řídké matice ve statických polích

Uvažujme uložení řídkých matic jako seznamů ve statických polích programovacího jazyka. V následujících podkapitolách si zavedeme některé základní způsoby takového uložení.

4.2.1 CSR a CSC formáty pro řídké matice a jejich varianty

Jako základní schéma ukládání řídké matice o n řádcích a sloupcích a nz nenulových prvcích uvažujme ukládání informací o nenulových prvcích matice po řádcích, které budeme označovat CSR jako zkratku z **compressed sparse by rows** [139]. Řídká matice je uložena ve standardních polích pomocí tří seznamů. První z nich, který budeme označovat $\mathcal{AA}$, obsahuje sekvenci nenulových prvků matice po řádcích za sebou. Druhý z nich označený $\mathcal{JA}$ obsahuje stejně uloženou sekvenci jejich sloupcových indexů a třetí seznam $\mathcal{IA}$ obsahuje informaci o tom, na kterých pozicích jednotlivé řádky matice v polích $\mathcal{A}$ a $\mathcal{JA}$ začínají ve formě ukazatelů na ně. Zatímco délka pole, kde můžeme uložit první dva seznamy, je rovna (alespoň) nz, délka třetího pole $\mathcal{IA}$ je tradiční **konvencí** rovna $n+1$, kde na pozici $\mathcal{IA}(n+1)$ je hodnota $nz+1$. Tím způsobem je počet nenulových prvků na i-tém řádku matice pro $i = 1, \ldots, n$ dán výrazem

$$\mathcal{IA}(i+1) - \mathcal{IA}(i). \tag{4.1}$$

Ukázka matice o třech řádcích a sloupcích a jejího uložení ve tvaru tří seznamů v polích CSR formátu je na Obrázku 4.2.2.

Variantou tohoto formátu je komprimované schéma ukládání matice **po sloupcích** nazývané CSC, kde jsou po řadě za sebou uloženy seznamy nenulových prvků matice po sloupcích, jejich řádkové indexy a dále ukazatele do těchto polí v poli o délce aspoň $n+1$. Samozřejmě, pro čtvercovou matici A se dá uložení jejích seznamů, které identifikují její nenulové prvky v CSR formátu, interpretovat jako uložení matice A^T s použitím CSC

Matice A:

$$\begin{pmatrix} 1.1 & 2 & \\ 2 & 4.8 & \\ 1 & 3.1 & 5 \end{pmatrix}$$

Uložení A v CSR formátu:

	1	2	3	4	5	6	7
$\mathcal{AA}$:	1.1	2	2	4.8	1	3.1	5
$\mathcal{JA}$:	1	2	1	2	1	2	3
$\mathcal{IA}$:	1	3	5	8			

Obrázek 4.2.2: *Příklad uložení matice v CSR formátu*

formátu. Další varianty jsou například ukládání pouze trojúhelníkové části matice, je-li A symetrická. Snadné je také zobecnění pro obecně obdélníkové matice. Ukládání jednotlivých řádků nebo sloupců matice v komprimované formě představuje základní způsob ukládání řídkých vektorů. Zde pak stačí uvést jen počet jeho nenulových prvků bez nutnosti mít samostatné pole ukazatelů označené výše jako $\mathcal{IA}$. Ukládáme-li pouze strukturu řídkosti, pole s numerickými hodnotami není potřeba.

Společnou vlastností uložení matic jako seznamů prvků a ukazatelů do polí popsaným způsobem je **statičnost** datových struktur. Vkládání nových nenulových prvků do matice je obecně málo efektivní a není to tedy obvykle přípustná operace. Stejně tak může narušovat efektivitu práce s řídkou maticí i odebírání nenulových prvků. Schéma je možné pozměnit, aby bylo dynamičtější, což má v některých případech, jak se o tom i dále zmíníme, smysl, viz například [128], [166]. Na druhou stranu, oproti výtce na statičnost schématu uložení, možnost efektivní práce s celými řádky v CSR formátu nebo sloupci v CSC formátu, či dokonce podmaticemi je vlastnost, která může být v souladu s rychlým zpracováním, přestože formáty CSR a CSC adresují nenulové prvky matic **nepřímo**.

4.2.2 SCSC formát pro řídké matice a grafy

Složitější datová struktura založená na stejném principu jako CSC využívá podobnosti nebo shodnosti **struktury řídkosti** po sobě následujících sloupců a nazveme jej SCSC formát. Schéma bylo používáno především pro ukládání prvků Choleského faktoru, kde je opakování sekvencí nenulových prvků s překrývajícími se množinami řádkových indexů **velmi** běžné a tady jej uvedeme jako připomínku, že schémata uložení mohou být motivována **vlastnostmi používaných algoritmů**. Zde se řádkové indexy se ukládají tak, aby se využily překryvy ve strukturách po sobě následujících sloupců. Při tomto uložení předpokládáme, že nenulové prvky jsou ve svých sloupcích uspořádány podle vzrůstajících řádkových indexů. V opačném případě by schéma nemuselo přinést kýžený efekt dodatečné

$$
\begin{array}{c}
\\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array}\\
\begin{pmatrix}
* & & & & & & & & & & & \\
 & * & & & & & & & & & & \\
 & & * & & & & & & & & & \\
 & & & * & & & & & & & & \\
 & & & & * & & & & & & & \\
 & & & & * & * & & & & & & \\
 & & * & * & & & * & & & & & \\
* & & & & & & & * & & & & \\
 & & & & * & * & * & & * & & & \\
* & * & * & & & * & * & * & * & * & & \\
* & * & * & & & & * & * & * & * & * & \\
* & * & * & * & * & * & * & * & * & * & * & *
\end{pmatrix}
\end{array}
$$

Obrázek 4.2.3: *Příklad struktury řídkosti dolní trojúhelníkové matice A.*

komprese proti CSC formátu uložení. Popišme stručně jak toto schéma může vznikat při vkládání struktury řídkosti jednotlivých sloupců do pole $\mathcal{JA}$. Je-li počáteční část struktury řídkosti nějakého sloupce shodná s momentální koncovou částí pole $\mathcal{JA}$, pak tyto indexy není třeba znovu ukládat a stačí uložit jen zbylou neshodnou část indexů. Pozice prvních řádkových indexů v poli $\mathcal{JA}$ pro jednotlivé sloupce jsou určeny pomocným polem $\mathcal{XA}$. Numerické hodnoty matice jsou tedy stejně jako v CSC formátu do pole $\mathcal{AA}$ a pole ukazatelů $\mathcal{IA}$ k nim umožňuje přístup a zároveň slouží k určení počtu prvků ve sloupci. Toto schéma je zjednodušením Shermanova formátu uložení, kde se diagonální prvky ukládají samostatně [147]. V praxi se používají i další varianty, ale důvodem naší zmínky o tomto formátu bylo především ukázat jakým směrem se statické formáty uložení vyvíjely, specializovaly a nutně i komplikovaly.

Na Obrázku 4.2.3 je znázorněna struktura řídkosti $\mathcal{S}(A)$ dolní trojúhelníkové matice, pro kterou uvádíme pole $\mathcal{IA}$, $\mathcal{JA}$ a $\mathcal{XA}$ jejího uložení v SCSC formátu na Obrázku 4.2.4. Zde vidíme, že například indexy sloupce 10 faktoru se nemusí ukládat do pole $\mathcal{JA}$ samostatně, ale stačí odkaz (39) na konec sloupce 9.

4.3 Spojové seznamy pro řídké matice

Komplexnější uložení řídké matice lze založit na **spojových seznamech**, kde jsou jednotlivé prvky matice spolu s dalšími informacemi uloženy jako samostatné objekty a propojeny navzájem odkazy založenými buď na **explicitních** ukazatelích poskytovaných operačním systémem nebo **interně** prostřednictvím polí ukazatelů vytvořených programem pro odkazy uvnitř nějakého bloku alokované paměti. Způsob odkazů i uložení prvků matice může být velmi různý podle potřeb práce s řídkou maticí i možností hardware a software. Můžeme například rozlišovat **spojový seznam cyklický** či pouze **lineární jednostranný** nebo **lineární oboustranný**. Tuto techniku zachycení řídké matice demonstrujeme pro jeden její řádek a pro jeden typ spojového seznamu na Obrázku 4.3.5.

	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathcal{IA}$:	1	6	11	16	20	24	29	34	38	42	45	47	48

$\mathcal{JA}$:	1	8	10	11	12	2	8	10	11	12	3	7	10
	11	12	4	7	12	5	6	9	12	6	7	9	10
	12	7	9	10	11	12	8	10	11	12	9	10	11
	12	10	11	12	11	12	12						

$\mathcal{XA}$:	1	6	11	16	20	24	29	34	38	39	40	41

Obrázek 4.2.4: *Příklad uložení struktury řídkosti matice A z Obrázku 4.2.3 v SCSC formátu*

Obrázek 4.3.5: *Uložení prvního řádku matice o devíti sloupcích s nenulovými prvky a_{11}, a_{13}, a_{14}, a_{18}, a_{19} pomocí oboustranného necyklického spojového seznamu. Šipky znázorňují ukazatele, které mohou být realizovány různým způsobem.*

$$\boxed{1 \mid a_{11}} \rightleftarrows \boxed{3 \mid a_{13}} \rightleftarrows \boxed{4 \mid a_{14}} \rightleftarrows \boxed{8 \mid a_{18}} \rightleftarrows \boxed{9 \mid a_{19}}$$

Většina dnešních programovacích jazyků umožňuje nejenom přímou implementaci polí, ale i datových struktur spojových seznamů. Přesto se velmi často používá začlenění spojových seznamů do polí z důvodu větší efektivity, což může mít kromě jednoduchého popisu výhodu například i v lepší lokalitě umístěných dat i transparentnějším využíváním vyrovnávacích pamětí. Příklad takového začlenění je uveden na následujícím Obrázku 4.3.6. Pole $\mathcal{FV}$ a $\mathcal{BV}$ délky n zachycují přímý a zpětný ukazatel položek s nenulovými prvky. Jejich pozice udávají též indexy nenulových prvků, ze kterých směřují tyto ukazatele. Pro kompletní informaci o znázorněném řádku matice je třeba mít ještě označeny pozice pro vstup do řetězů přímých i zpětných ukazatelů, ale tyto triviální údaje vztahující se k samotnému programování a vytváření software zde nebudeme formalizovat.
Datové struktury spojových seznamů často umožňují dynamičtější práci s maticí ve smyslu jednoduššího vkládání nových nenulových prvků i jejich vynechávání. To je někdy výhoda,

Obrázek 4.3.6: *Uložení prvního řádku matice z Obrázku 4.3.5, kde spojový seznam je implementován pomocí polí.*

	1	2	3	4	5	6	7	8	9	10	11
V:	a_{11}	a_{13}	a_{14}	a_{18}	a_{19}						
$\mathcal{FV}$:	3		4	8				9			
$\mathcal{BV}$:			1	3				4	8		

za kterou se ale může platit **obtížnějším přizpůsobením soudobým počítačovým architekturám**. Jedna z hlavních linií tohoto textu je ukázat teoretické výsledky pro práci s řídkými maticemi, které často umožní spojové seznamy obejít u naprosté většiny použitých dat.

Historicky vznikla celá řada specializovaných schémat uložení a jejich přehled je možné najít například v [130] či [97]. Mnoho z nich má další důležité analogie, které například slouží pro ukládání matice s blokovou strukturou, která je pro soudobé počítání **velmi podstatná**. Mnoho z nich bylo ale také navrženo bez ohledu na reálný přínos pro implementaci klíčových matematických postupů numerické lineární algebry, pro které navíc ve své době chyběly základní implementační vzory. Protože se ale algoritmy práce s řídkými maticemi výrazně mění, mění se i počítačové architektury a dostáváme se do situace, že uplatnění dnes nalézá i řada nápadů z minulosti.

5

Grafy a jejich matice

Grafy pro nás v tomto textu představují hlavně nástroj pro práci s řídkými maticemi. V následujícím textu zavedeme postupně grafy neorientované i orientované. Někdy se hodí používat i jejich kombinace, **grafy smíšené**, které mohou obsahovat jak hrany neorientované tak i orientované, jak o nich budeme hovořit ve dvou následovně prezentovaných verzích základních typů grafů.

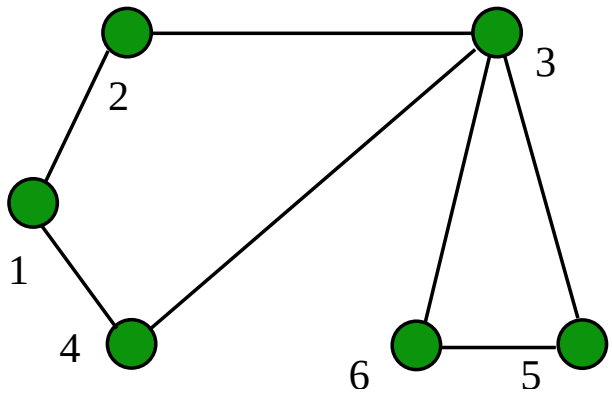
5.1 Neorientovaný graf

Nejjednodušším typem grafu je prostý **neorientovaný graf** z následující Definice 5.1.1.

Definice 5.1.1 Prostý neorientovaný graf *G je uspořádaná dvojice množin (V, E). Množina $V = \{v_1, \ldots, v_n\}$ se nazývá množina* **vrcholů** *grafu G. $E = \{e_1, \ldots, e_m\}$ nazýváme Množina $E = \{e_1, \ldots, e_m\}$* **hran** *grafu G splňuje $E \subseteq \binom{V}{2}$.*

Množinu V o velikosti $|V| = n$ zde často ztotožňujeme s množinou $\{1, \ldots, n\}$ a hrany pak jsou dány neuspořádanými dvojicemi $\{i, j\}$. Příklad prostého neorientovaného grafu $G = (\{1, 2, 3, 4, 5, 6\}, \{\{1, 2\}, \{2, 3\}, \{1, 4\}, \{3, 4\}, \{3, 6\}, \{3, 5\}, \{5, 6\}\})$ je na Obrázku 5.1.1. Vrcholům odpovídají zakreslené body a hranám jejich spojnice.

Poznámka 5.1.1 *V naší definici jsou každé dva vrcholy grafu spojeny nejvýše jednou hranou. Model* **multigrafu** *umožňuje zavést více hran mezi konkrétními vrcholy. Neuvažujeme ani hrany incidentní pouze jednomu vrcholu, které se nazývají* **smyčky**. *Pod neorientovaným grafem bez bližšího určení budeme myslet* **prostý neorientovaný graf** *podle Definice 5.1.1.*



Obrázek 5.1.1: *Příklad prostého neorientovaného grafu* $G = (\{1,2,3,4,5,6\}, \{\{1,2\}, \{2,3\}, \{1,4\}, \{3,4\}, \{3,6\}, \{3,5\}, \{5,6\}\})$

5.1.1 Základní terminologie neorientovaného grafu

Vrcholy v_i a v_j neorientované hrany $\{v_i, v_j\} \in E$ nazýváme jejími **koncovými** vrcholy. Dva vrcholy množiny V, které jsou vrcholy nějaké neorientované hrany, nazveme **sousedními vrcholy** nebo také (vzájemnými) **sousedy** grafu. Je-li vrchol v koncovým vrcholem nějaké neorientované hrany $e \in E$, čili je-li $v \in e$, nazveme jej **incidentní** této hraně. Hranu e budeme pak analogicky nazývat incidentní vrcholu v_i. Vrcholy grafu, které nejsou incidentní žádné hraně grafu, budeme nazývat **izolované** vrcholy. Množinu vrcholů grafu incidentních hranám z nějaké množiny hran E budeme někdy pro její zdůraznění označovat $V(E)$. **Úplným** grafem nazveme takový neorientovaný graf, pro který platí $E = \binom{V}{2}$. Potřebujeme-li zdůraznit u nějaké množiny vrcholů V nebo hran E, ke kterém grafu patří, zdůrazníme to značením jako například $V(G)$ nebo $E(G)$ pro graf G.

Definice 5.1.2 *Graf* $G' = (V', E')$ *nazveme* **podgrafem** *neorientovaného grafu* $G = (V, E)$*, platí-li* $V' \subseteq V$*,* $E' \subseteq E$ *a všechny vrcholy* $V(E')$ *incidentní hranám* E' *podgrafu* G' *patří do* V'*.*

Jestliže E' obsahuje **všechny** hrany grafu G, které mají koncové vrcholy z V', nazveme graf G' podgrafem grafu G **indukovaným** množinou vrcholů $V' \subseteq V(G)$ a budeme jej označovat $G(V')$. Analogicky je definován podgraf indukovaný množinou hran. Výsledkem operace **přidání hrany do grafu** rozumíme nový graf, který je sjednocením původního grafu a grafu indukovaného touto hranou, přičemž sjednocením grafů $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$ rozumíme graf $G = (V_1 \cup V_2, E_1 \cup E_2)$. **Klikou** stupně k grafu G nazveme jeho úplný podgraf s počtem vrcholů (mohutností množiny vrcholů) k.

Množinu sousedů vrcholu $v \in V$ v neorientovaném grafu $G = (V, E)$ označíme výrazem $adj(v)$. To jest,

$$adj(v) = \{u \in V \mid \{u, v\} \in E\}.$$

Budeme-li zde chtít zdůraznit graf, kterého se tato množina týká, uvedeme jej jako dolní index v označení této množiny, jako např. $adj_G(v)$ je množina sousedů vrcholu v v grafu G. Tento dolní index nebudeme uvádět, bude-li zřejmé z kontextu, k jakému grafu se množina

sousedů nějakého vrcholu vztahuje. Velikost množiny sousedů $|adj(v)|$ budeme nazývat **stupněm vrcholu** v v daném grafu a budeme jej někdy označovat jednodušším zápisem $deg(v)$. Stejně jako v jiných případech pro vyhnutí se nejasnostem někdy použijeme označení grafu ve formě dodatečného indexu. Například budeme používat zápis $deg_G(v)$ pro stupeň vrcholu v v grafu G. Pojem množiny sousedů jednoho vrcholu můžeme také zobecnit na množinu sousedů nějaké podmnožiny $U \subseteq V$ následovně.

$$adj_G(U) = \{u \notin U | (\exists v \in U)(u \in adj_G(v))\} \tag{5.1}$$

Množinu $adj_G(U)$ pak nazýváme **množinou sousednosti** množiny U v grafu G.

Definice 5.1.3 *Dva neorientované grafy $G = (V, E)$ a $G' = (V', E')$ nazveme* **izomorfní**, *existuje-li vzájemně jednoznačné zobrazení mezi jejich množinami vrcholů $\alpha : V \to V'$, které zachovává sousednost vrcholů. Pod zachováváním sousednosti vrcholů rozumíme, že $e \in E$ je hrana s koncovými vrcholy u a v právě tehdy, je-li $e' \in E'$ hrana s koncovými vrcholy $\alpha(u)$ a $\alpha(v)$.*

Dva izomorfní grafy můžeme z praktického pohledu často navzájem zaměňovat. Záměna množiny vrcholů $V = \{v_1, \ldots, v_n\}$ za množinu $\{1, \ldots, n\}$ (tedy jejich nové označení) s příslušným zobrazením mezi množinami hran je příkladem takové izomorfie. Tuto izomorfii, kde položíme $V = \{1, \ldots, n\}$ budeme používat velmi často, neboť pak můžeme vrcholy přímo **porovnávat** a často se tak zjednoduší naše vyjadřování.

5.1.2 Sekvence hran a vrcholů neorientovaného grafu

Nechť G je neorientovaný graf. Posloupnost t hran

$$\{i_0, i_1\}, \{i_1, i_2\}, \ldots, \{i_{t-1}, i_t\}$$

incidentních množině vrcholů $\{i_0, \ldots, i_t\}$, nazveme **sledem** mezi vrcholem i_0 a vrcholem i_t, který má délku t, která je tedy dána počtem hran sledu. Triviální sled s $t = 0$, reprezentovaný pouze vrcholem má i_0, má délku 0. Vrcholy i_0 a i_t nazýváme koncové vrcholy sledu a říkáme, že jsou **spojeny/propojeny** tímto sledem. Sled nazýváme **uzavřený**, platí-li $i_0 = i_t$. V opačném případě se sled nazývá **otevřený**. Sled, jehož všechny hrany jsou různé, se nazývá **tah**. Jsou-li všechny vrcholy tahu různé s možnou výjimkou pro koncové vrcholy, tedy pro $i_0 = i_t$, nazývá se tento tah (neorientovanou) **cestou**. Fakt, že mezi dvěma vrcholy grafu u a v existuje neorientovaná cesta budeme zapisovat $u \Leftrightarrow v$ nebo také $u \overset{G}{\Leftrightarrow} v$, když chceme zdůraznit, že uvažujeme cestu v grafu G. Chceme-li navíc ještě omezit množinu **mezilehlých vrcholů** cesty, tedy vrcholů cesty mimo vrcholů koncových, na nějakou množinu $S \subseteq V(G)$, pak cestu mezi dvěma vrcholy v G zapíšeme

$$j \underset{S}{\overset{G}{\Leftrightarrow}} k.$$

Uzavřená cesta se také nazývá **cyklus**. Vrchol grafu i_t nazveme pro danou množinu $S \subseteq V$ grafu **dosažitelným z vrcholu** i_0 **přes množinu** S, existuje-li cesta s koncovými vrcholy i_0 a i_t taková, že všechny existující **mezilehlé** vrcholy této cesty jsou obsaženy

v množině S. Neřekneme-li jinak, připouštíme v našem textu i prázdnou množinu mezilehlých vrcholů, kdy se tato cesta redukuje na jedinou hranu. Formálně si definujeme množinu dosažitelnosti následovně.

Definice 5.1.4 *Nechť V je množina vrcholů grafu G a nechť $i_0 \in V, S \subseteq V$.* **Množinu dosažitelnosti** *vrcholu $i_0 \in V$ přes množinu S v grafu G označenou $Reach(i_0, S, G)$, definujeme vztahem*

$$Reach(i_0, S, G) = \{k \in V \mid k \in V \text{ je dosažitelný z } x \in V \text{ v grafu } G \text{ přes } S\}$$

nebo tedy

$$Reach(i_0, S, G) = \{k \in V \mid i_0 \overset{G}{\underset{S}{\Leftrightarrow}} k\}$$

V množině dosažitelnosti vrcholu tedy formálně předpokládáme že vrchol i_0 může a nemusí být v množině S. Někdy budeme o cestě, ale také o sledu či tahu, hovořit také jako o posloupnosti vrcholů nebo posloupnosti vrcholů a hran, je-li toto vyjádření jednoznačné. V námi zavedeném neorientovaném grafu je délka nejkratšího cyklu alespoň 3. Prostý graf z obrázku 5.1.1 má dva cykly. Jeden má délku čtyři a druhý má délku tři. **Vzdáleností** vrcholů u a v v neorientovaném grafu nazveme délku nejkratší cesty, která má u a v jako své koncové vrcholy, to jest délku takové cesty, která má z nich nejmenší počet hran. Dalším základním grafovým pojmem teorie grafů je souvislost, kterou uvedeme v následující definici.

Definice 5.1.5 *Neorientovaný graf je* **souvislý**, *jestliže jsou jeho každé dva vrcholy spojeny sledem.*

Není-li graf souvislý, budeme jej nazývat **nesouvislým**. Grafu indukovaný jedním vrcholem nebo graf prázdný ($V = E = \emptyset$) jsou triviálně považovány za souvislé. Propojení vrcholů sledem reprezentuje relaci souvislosti nad vrcholy grafu. Tato relace je ekvivalence a indukuje tak rozklad množiny vrcholů na podmnožiny následujícím způsobem:

$$V = V_1 \cup \ldots \cup V_t. \tag{5.2}$$

Podgrafy indukované množinami $V_1, \ldots, V_t$ nazveme **souvislými komponentami** (nebo komponentami souvislosti, případně pouze komponentami) neorientovaného grafu G. Souvislý graf s počtem hran o jedničku menším než je počet jeho vrcholů se nazývá **strom**. Snadno nahlédneme, že strom nemůže obsahovat žádný cyklus. Neorientovaný graf, který neobsahuje cyklus, se také nazývá **acyklický**. Graf, který je acyklický, ale není nutně souvislý, se nazývá **les**.

Dalším důležitým pojmem, který později použijeme, je **vrcholová barevnost** grafu. Je definována jako minimální počet barev, kterými můžeme obarvit vrcholy grafu tak, že žádná hrana mezi dvěma vrcholy nemá oba koncové vrcholy stejné barvy. Důležitým speciálním případem grafu je graf **bipartitní**, což je graf, jehož vrcholová barevnost je rovna dvěma. V následujícím textu, nebude-li moci dojít k nedorozumění, budeme často používat následující zjednodušený zápis. Je-li nějaký vrchol v grafu G prvkem jeho množiny vrcholů nebo hrana e prvkem jeho množiny hran, pak budeme psát $v \in G$, respektive $e \in G$. Stejně tak budeme ztotožňovat graf s jeho množinou vrcholů nebo hran v případech, kde nemůže dojít k nedorozumění.

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

Obrázek 5.1.2: *Příklad matice sousednosti pro prostý neorientovaný graf z obrázku 5.1.1*

5.1.3 Matice neorientovaného grafu

S neorientovanými grafy jsou přirozeně spjaty některé matice. Zde si uvedeme dvě z nich. Podrobný výklad najdeme například v [24].

5.1.3.1 Matice sousednosti neorientovaného grafu

Matice sousednosti grafu je definována následovně.

Definice 5.1.6 *Nechť $G = (V, E)$ je prostý neorientovaný graf s $V = \{1, \ldots, n\}$.* **Maticí sousednosti** *tohoto grafu G nazveme $(0, 1)$ matici $A_G = (a_{ij})$ $(i, j \in V)$, kde a_{ij} je rovno počtu neorientovaných hran $\{i, j\}$ mezi vrcholy i a j.*

Někdy také uvažujeme matici sousednosti s doplněnými diagonálními prvky (jedničkami). Následující příklad demonstruje matici sousednosti.

Příklad 5.1.1 *Prostý neorientovaný graf G z Obrázku 5.1.1 má symetrickou $(0, 1)$ matici sousednosti A_G znázorněnou na Obrázku 5.1.2. Počet nenulových prvků v i-tém řádku této matice je roven stupni $deg_G(i)$ vrcholu i v grafu G.*

Izomorfie dvou neorientovaných grafů se dá jednoduše znázornit pomocí jejich matic sousednosti, což je obsahem následující poznámky.

Poznámka 5.1.2 *Nechť G a G' jsou dva* **prosté neorientované grafy**. *Označme jejich matice sousednosti, po řadě, $A = A_G$ a $A' = A'_G$. Z definice izomorfismu grafu plyne, že G a G' jsou izomorfní právě tehdy, lze-li symetricky permutovat řádky a sloupce matice A tak, že dostaneme matici A'. Jinými slovy, tyto grafy jsou izomorfní právě tehdy, existuje-li permutační matice $P \in \mathbf{R}^{n \times n}$ taková, že $A' = P^T AP$.*

5.1.3.2 Matice incidence neorientovaného grafu

Další důležitou maticí, kterou lze definovat pro daný neorientovaný graf, je neorientovaná **matice incidence** (incidenční matice) grafu.

Definice 5.1.7 *Nechť $G = (V, E)$ je prostý neorientovaný graf s $V = \{1, \ldots, n\}$, $E = \{1, \ldots, m\}$.* **Maticí incidence** *(incidenční maticí, maticí hrany krát vrcholy) grafu G nazveme (obecně obdélníkovou) (0,1) matici $A_G = (a_{ij})$, $(i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\})$, kde $a_{ij} = 1$, je-li j vrcholem hrany i a $a_{ij} = 0$ v opačném případě.*

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

Obrázek 5.1.3: *Příklad matice incidence pro prostý neorientovaný graf z obrázku 5.1.1 s očíslováním hran* $e_1 = \{1,2\}$, $e_2 = \{2,3\}$, $e_3 = \{1,4\}$, $e_4 = \{3,4\}$, $e_5 = \{3,6\}$, $e_6 = \{3,5\}$, $e_7 = \{5,6\}$ *a s řádky odpovídajícími postupně hranám* $e_1, \ldots, e_7$.

Matice incidence grafu G z obrázku 5.1.1 je znázorněna na obrázku 5.1.3. Pozměníme-li tuto matici tak, že v každém řádku se dvěma nenulovými prvky jeden z nich změníme z 1 na -1, hovoříme **orientované** matici incidence (neorientovaného grafu). Samozřejmě, takto definovaná transformace matice incidence není obecně jednoznačná a v praxi se používají její různé varianty. Počet různých možných možností orientace v prostém neorientovaném grafu je roven 2^m, kde m je počet řádků neorientované matice incidence, což je také počet hran odpovídajícího grafu.

5.1.4 Některé vztahy mezi neorientovanými grafy a jejich maticovými reprezentacemi

Mezi maticemi sousednosti a incidence neorientovaného grafu platí následující vztah.

Věta 5.1.1 *Nechť* $G = (V, E)$ *je prostý neorientovaný graf s* $V = \{1, \ldots, n\}$. *Nechť* $A \in \mathbf{R}^{m \times n}$ *je incidenční matice tohoto grafu a* B *jeho matice sousednosti (bez diagonálních prvků). Pak platí*

$$A^T A = D + B, \tag{5.3}$$

kde $D = (d_i)_{i=1,\ldots,n} \in \mathbf{R}^{n \times n}$ *je diagonální matice a* $d_i = deg(i)$ *pro* $i \in V$.

Důkaz: Snadno je vidět, že diagonální prvek matice $A^T A$ se rovná stupni příslušného vrcholu. Uvažme přitom, že sloupce odpovídají jednotlivých vrcholům a diagonální prvek $A^T A$ je dán skalárním součinem sloupce se sebou samým.

Co se týká mimodiagonálních prvků, které odpovídají odlišným vrcholům, je situace následující: Pro dva různé vrcholy i a j je prvek symetrické matice $A^T A$ na pozici $\{i, j\}$ roven jedné právě tehdy, když jsou příslušné vrcholy spojeny hranou. Je to vidět i tak, že příslušný skalární součin dvou sloupců obsahuje jednotkové příspěvky pouze tehdy, když jsou tyto vrcholy propojeny hranou. Tvrzení je tak dokázáno. ■

Poznámka 5.1.3 *Orientovaná matice incidence splňuje mírně pozměněný vztah (5.3). Konkrétně platí*

$$A^T A = D - B. \tag{5.4}$$

Důkaz: Snadno nahlédneme, že **každý mimodiagonální prvek** je daný součinem jedničky a minus jedničky. Pro diagonální prvky matice $A^T A$ se nic nemění. Z tvrzení je zřejmé, že platnost tohoto vztahu nezávisí na konkrétní orientaci prvků matice incidence. ■

Při diskretizacích oblastí se často používá pojem **hranového grafu**, který zavedeme v následující definici.

Definice 5.1.8 *Nechť $G = (V, E)$ je prostý neorientovaný graf. Hranový graf $L(G) = (V^L, E^L)$ definuje $V^L = E$. Množina $E^L \in \binom{V^L}{2}$ obsahuje právě ty dvojice prvků z V^L, které jsou incidentní společnému vrcholu z V, to jest, platí-li*

$$\{e_1, e_2\} \in E^L \Leftrightarrow e_1 \cap e_2 \neq \emptyset.$$

Pro hranový graf platí následující tvrzení, které je analogií tvrzení Věty 5.3:

Věta 5.1.2 *Nechť $A \in \mathbf{R}^{m\times n}$ je incidenční matice prostého neorientovaného grafu G a nechť $B \in \mathbf{R}^{m\times m}$ je matice sousednosti hranového grafu $L(G)$. Pak platí*

$$AA^T = 2I_m + B \tag{5.5}$$

Důkaz: Pro $i \neq j$ je prvek matice B o indexech i a j roven jedné právě tehdy, mají-li hrany e_i a e_j grafu G společný vrchol. Jinak je roven nule. Totéž však můžeme říci o matici AA^T. Hodnoty diagonálních prvků je také snadné ověřit. ■

Druhým základním modelem grafu je graf **orientovaný**. Třída orientovaných grafů se liší od neorientovaných grafů tím, že jejich hrany jsou prvky kartézského součinu a vrcholy, které je vyjadřují, jsou tedy uspořádané.

5.2 Orientovaný graf

Definice 5.2.1 Prostý orientovaný graf *G je uspořádaná dvojice (V, E), kde $V = \{v_1, \ldots, v_n\}$ se nazývá množina vrcholů grafu G a E se nazývá množina orientovaných hran tohoto grafu, kde*

$$E \subseteq V \times V \setminus \{(v_1, v_1), \ldots, (v_n, v_n)\}.$$

Je-li (i, j) hrana z E, pak vrchol i nazýváme jejím počátečním vrcholem a vrchol j jejím koncovým vrcholem. Stejně tak ale můžeme hovořit o dvou koncových vrcholech orientované hrany.

Analogicky jako výše, orientovaným grafem bez bližší specifikace rozumíme **prostý orientovaný graf**, pokud neuvedeme jinak.

5.2.1 Vztah terminologie orientovaného a neorientovaného grafu

Mezi neorientovaným a orientovaným grafem není příliš hluboká propast. Budeme-li v tomto textu rozšiřovat platnost některých definicí, vlastností a tvrzení, které zavádíme pro orientované grafy, i na grafy neorientované, pak tím budeme rozumět jejich platnost pro neorientovaný graf, kde máme neorientovanou hranu $\{i, j\}$ mezi vrcholy i a j právě tehdy, existuje-li v původním grafu **alespoň jedna** z hran (i, j) nebo (j, i). Takto definovaný neorientovaný graf, odvozený z orientovaného grafu G, budeme dále nazývat jeho **symetrizací**. Opačně, některá tvrzení i vlastnosti týkající se neorientovaných grafů můžeme bez problému aplikovat i na orientovaný graf odvozený tak, že každou neorientovanou hranu $\{i, j\}$ mezi vrcholy i a j nahradíme dvojicí orientovaných hran (i, j) a (j, i). Tento odvozený graf se nazývá **symetrickou orientací** původního neorientovaného grafu.

Neorientovanému grafu můžeme také přiřadit orientovaný graf tak, že každou hranu $\{i, j\}$ mezi vrcholy i a j orientujeme výlučně buď jako (i, j) nebo jako (j, i). Tento proces se nazývá **nesymetrická orientace** (stručně jen **orientace**) grafu a není obecně jednoznačný. Nicméně může být užitečný v některých algoritmech pro práci s řídkými maticemi. Výše uvedenou **orientaci matice incidence** můžeme interpretovat také jako orientaci hran grafu, neboli jako vytvoření matice odpovídající nějaké **nesymetrické orientaci grafu.**

5.2.2 Terminologie orientovaného grafu

Vzájemné transformace mezi neorientovanými a orientovanými grafy umožňují jednoduché rozšíření řady pojmů zavedených dříve jako pojmů charakterizující grafy neorientované. V některých případech přidáme ještě další značení a zdůrazníme odlišnost pojmů pro neorientovaný a orientovaný graf.

Důležitým rozšířením z neorientovaných grafů jsou pojmy **orientovaného sledu, orientovaného tahu** a **orientované cesty**, kde v posloupnosti hran, které je definují, uvažujeme pouze orientované hrany. Konkrétně, v nejobecnějším pojmu sledu s vrcholy $i_0, \ldots, l_t$ předpokládáme orientace hran v pořadí $(i_0, i_1), (i_1, i_2), \ldots, (i_{t-1}, i_t)$. Existenci orientované cesty z počátečního vrcholu u do koncového vrcholu v budeme zapisovat $u \Rightarrow v$ nebo také $u \overset{G}{\Rightarrow} v$, když chceme zdůraznit, že uvažujeme cestu v grafu G. Omezení množiny vrcholů cesty pak popíšeme analogicky jako u neorientovaných cest. Hrana (u, v) zapisovaná též jako $u \to v$ nebo $u \overset{G}{\to} v$ je triviální případ takové orientované cesty. Dosažitelnost a množina dosažitelnosti v orientovaném grafu jsou definovány analogicky s tím, že se uvažují pouze orientované cesty. Výše zmíněné označení vrcholů hrany orientovaného grafu G jako počátečním a koncovém vrcholu nebo dvou koncových vrcholech v případě kdy nemůže dojít k záměně, je dáno uvažováním orientované hrany jako hrany v symetrizaci grafu G. Analogicky, pro sled

$$(i_0, i_1), (i_1, i_2), \ldots, (i_{t-1}, i_t)$$

jsou i_0 a i_t jeho koncové vrcholy, nebo také i_0 je počáteční vrchol sledu a i_t je koncový vrchol sledu. Analogicky se rozšiřuje pojem podgrafu indukovaného množinou vrcholů grafu i na orientované grafy. I pojem stromu lze snadno zobecnit na orientované grafy tak,

že za **orientovaný strom** považujeme orientovaný graf, jehož symetrizace je stromem. V orientovaném grafu definujeme pro každý jeho vrchol $v \in V$ následující podmnožiny množiny sousedů vrcholů.

$$\begin{aligned} adj^+(v) &= \{u \in V | (v,u) \in E\} \\ adj^-(v) &= \{u \in V | (u,v) \in E\}. \end{aligned}$$

Množinou sousednosti (sousedů) $adj(v)$ pro $v \in V$ rozumíme sjednocení $adj^+(v) \cup adj^-(v)$ tedy množinu sousedů symetrizace orientovaného grafu.

Podstatným rozšířením pojmu souvislosti z neorientovaných grafů je pojem **silné souvislosti** orientovaných grafů. Dva vrcholy x_1 a x_2 orientovaného grafu G se nazývají **silně souvislé**, platí-li

$$x_1 \overset{G}{\Rightarrow} x_2 \quad \wedge \quad x_2 \overset{G}{\Rightarrow} x_1. \tag{5.6}$$

Vrchol sám se přitom považuje triviálně za silně souvislý sám se sebou. Silná souvislost tak definuje na množině vrcholů ekvivalenci a indukuje obecně rozklad množiny vrcholů na $t, t \geq 1$ tříd ekvivalence vrcholů $V_1, V_2, \ldots, V_t$, kde

$$V = V_1 \cup V_2 \cup \ldots \cup V_t. \tag{5.7}$$

Podgrafy indukované celými množinami vrcholů jednotlivých tříd ekvivalence se nazývají **silné komponenty** orientovaného grafu G. Jsou to tedy **maximální** množiny navzájem silně souvislých vrcholů, kde maximum se uvažuje ve smyslu inkluze, nikoliv ve smyslu nalezení silné komponenty s největší kardinalitou.

Definice 5.2.2 *Orientovaný graf G se nazývá silně souvislý, má-li právě jednu silnou komponentu. Jinými slovy, graf je silně souvislý právě tehdy, jsou-li všechny dvojice jeho vrcholů navzájem silně souvislé.*

5.2.3 Matice spojené s orientovaným grafem

Matici incidence jsme zmínili při zavádění procesu orientace grafu. Co se týká matice sousednosti prostého orientovaného grafu G, pro naše účely bude stačit ji chápat jako matici sousednosti symetrizace grafu G.

5.2.4 Acyklické grafy a kořenové stromy

Velmi významné místo mezi grafy z teoretického i praktického pohledu zaujímají **orientované acyklické grafy**. Následující definice zavádí nejprve **standardní** topologické očíslování grafu.

Definice 5.2.3 Topologickým očíslováním *(orientovaného grafu, vrcholů orientovaného grafu) nazveme takové zobrazení (očíslování) $\alpha : V \to \{1, \ldots, |V|\}$ jeho vrcholů V, že pro všechny jeho hrany (u, v), $u, v \in V$ platí*

$$\alpha(u) < \alpha(v).$$

Věta 5.2.1 *Orientovaný graf je* **acyklický** *právě tehdy, existuje-li jeho topologické očíslování.*

Důkaz: Nechť je daný graf G acyklický. Nejprve ukážeme, že v něm musí existovat vrchol v, pro který platí $adj^-(v) = \emptyset$. Tento vrchol nazveme **zdrojem** acyklického grafu. Zvolme libovolný vrchol $x \in V(G)$. Je-li zdrojem, pak jsme hotovi. Není-li zdrojem, existuje nějaký vrchol, z něhož vede hrana do x. Opakováním tohoto procesu maximálně $(|V| - 1)$-krát nalezneme v grafu s konečným počtem vrcholů zdroj, protože graf je acyklický a nelze se tímto způsobem vrátit do vrcholu v.

Předpokládejme nyní, že graf $G = (V, E)$ je acyklický a uvažujme matematickou indukci podle počtu jeho vrcholů. Pro $|V| = 1$ je tvrzení je zřejmé. Nechť $|V| > 1$. Pak existuje jeho zdroj s a s využitím indukčního předpokladu existence topologického očíslování podgrafu grafu G indukovaného množinou $V \setminus \{s\}$ získáme očíslování tak, že $\alpha(s)$ přiřadíme menší hodnotu než všem vrcholům z tohoto podgrafu, pro který ale topologické očíslování existuje. To jde vždy s využitím čísel z intervalu $\langle 1, , , |V| \rangle$.

Opačně, existuje-li topologické očíslování grafu, graf musí být acyklický, neboť jinak bychom dostali spor porovnáváním hodnot zobrazení α vrcholů existujícího cyklu. ■

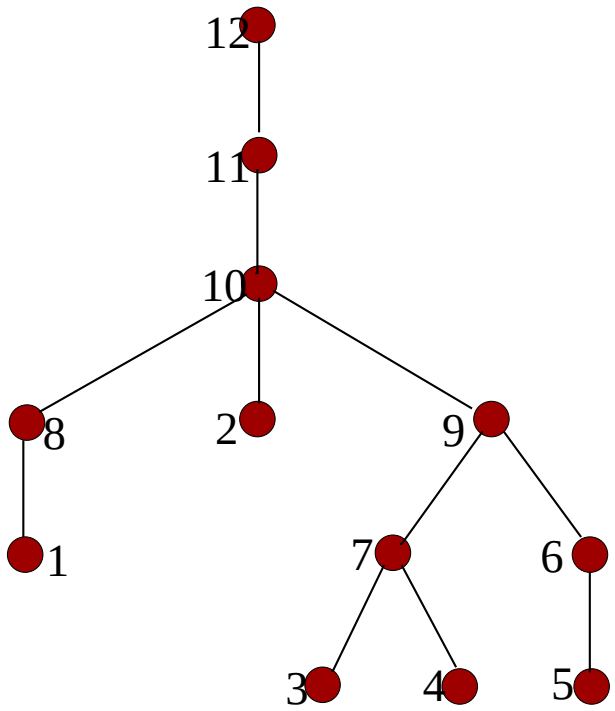
V acyklických neorientovaných grafech můžeme **zavést orientaci** na základě vybraného vrcholu tak, jak nyní popíšeme. Nejprve definice.

Definice 5.2.4 **Kořenovým stromem** *nazveme acyklický a souvislý graf (strom) s jedním vybraným vrcholem r. Ten budeme budeme nazývat* **kořen** *tohoto stromu.*

Výběr kořene znamená **zavedení orientace** v neorientovaném stromu nebo zavedení **nové orientace** stromu, který byl původně orientovaný nebo smíšený. Zatímco první případ je jasný, zavedení nové orientace znamená nejprve **odstranění původní celé nebo "částečné" orientace symetrizací**. Nová orientace, pokud neuvedeme jinak, bude v tomto textu určena tak, že existují jednoznačně určené orientované cesty z **kořene** r ke **všem ostatním vrcholům** $V \setminus \{r\}$. Orientaci můžeme zavést i v obecném nesouvislém orientovaném acyklickém grafu určením vrcholu v každé jeho komponentě, tedy výběrem kořenu jeho komponenty. Kořenové stromy tedy budeme obvykle uvažovat jako grafy orientované. Následující terminologie se týká obecně orientovaných acyklických grafů, které tedy zahrnují i kořenové stromy.

Definice 5.2.5 *Vrchol x orientovaného acyklického grafu G nazveme* **předkem** *vrcholu y, existuje-li v něm orientovaná cesta z x do y. Vrchol x orientovaného acyklického grafu nazveme* **potomkem** *vrcholu y, je-li vrchol y předkem vrcholu x.*

Protože uvažujeme i cesty nulové délky, pak **každý vrchol je sám svým předkem i potomkem**. Tato konvence, jakkoli se zdá být zvláštní, umožní jednodušeji formulovat některá tvrzení. **Vlastní** předek respektive potomek nějakého vrcholu je takový jeho předek respektive potomek, který se liší od sebe sama. Množinu všech předků nějakého vrcholu x budeme označovat $anc(x)$ nebo $anc_G(x)$, chceme-li zdůraznit acyklický graf, ke kterému se pojem předka vztahuje. Analogicky množinu potomků vrcholu x v grafu G budeme značit $desc(x)$ nebo $desc_G(x)$. Pro orientované acyklické grafy $T = (V, E)$ a tedy i kořenové stromy dále zavedeme nejbližšího nevlastního předka nazvaného nazývané **otec** pomocí následujícího zobrazení *parent*.



Obrázek 5.2.4: *Příklad kořenového stromu. Jeho kořen je vrchol* 12.

Definice 5.2.6

$$parent(j) = \begin{cases} i \Leftrightarrow & \exists\, i \in V(T),\ (i,j) \in E(T), \\ \textbf{null} & jinak. \end{cases} \tag{5.8}$$

Vrchol x nazveme **synem** *vrcholu y právě tehdy, je-li vrchol y otcem vrcholu x, to jest, platí-li $y = parent(x)$.*

V případě kořenového stromu $T = (V, E)$, který je souvislý, má hodnotu **null** pouze kořen r. V **orientovaném lese** takových vrcholů může být víc. Na obrázcích kořenových stromů zavedeme konvenci, že vlastní potomky vrcholu nakreslíme vždy níže než samotný vrchol, a pak nemusíme vyznačovat orientaci pomocí šipek. Obrázek 5.2.4 znázorňuje příklad kořenového stromu. Jeho kořenem je vrchol 12. Dále, například, vrchol 10 je předkem vrcholů 2, 8 a 9 a ti jsou zase jeho potomci. Seznam předků vrcholu 10 je $anc(10) = \{10, 11, 12\}$. Hodnoty zobrazení $parent(i)$ pro vrcholy stromu z Obrázku 5.2.4 jsou, po řadě rovny $8, 10, 7, 7, 6, 9, 9, 10, 11, 12$, **null**.

Pro kořenový strom i pro obecný acyklický graf budeme v tomto textu dále používat **alternativní topologického očíslování** dané následující definicí

Definice 5.2.7 Topologickým očíslováním *acyklického grafu nazveme takové zobrazení (očíslování) $\alpha : V \to \{1, \ldots, |V|\}$ jeho vrcholů V, že pro všechny jeho hrany $(parent(u), u)$ $u \in V$, pokud existují, platí*

$$\alpha(u) < \alpha(parent(u)).$$

Zdůrazněme tedy, že bez újmy na obecnosti budeme tedy i v obecném orientovaném acyklickém grafu používat opačnou orientaci než byla diskutována výše tak aby byla

shodná s orientací kořenového stromu jako speciálního případu orientovaného acyklického grafu. To je motivováno kompatibilitou s dalšími částmi textu.

Zavedené značení umožňuje jednoduše vyjádřit následující zřejmé pozorování.

Pozorování 5.2.1 *Topologické očíslování α orientovaného acyklického grafu $G = (V, E)$ má následující tranzitívní vlastnost.*

$$\alpha(v) < \alpha(u) \text{ pro } u \in anc(v),\ u \neq v. \tag{5.9}$$

6

Od matic ke grafům

Zatímco v předcházející kapitole jsme se věnovali grafům a definovali i některé matice, které jsou pro grafy definovány, v této kapitole vyjdeme z pojmu řídké matice a zavedeme si několik variant grafů, který charakterizují **strukturu** jejích nenulových prvků, tedy **strukturu řídkosti matice**. Tyto grafy budeme také nazývat grafové **modely** struktury řídkosti matic. Takto zavedené grafové modely mohou být dále modifikovány podle konkrétních potřeb a případné modifikace modelů zmíníme v příslušných situacích.

6.1 Matice a jejich grafy

Uvažujme obecně obdélníkovou matici $A \in \mathbf{R}^{m\times n}$ a rozlišme tři základní modely, kterými zachytíme její strukturu řídkosti. Grafový model dále ve většině případů snadno rozšíříme s pomocí vrcholového a/nebo hranového ohodnocení tak, aby zachycoval nejenom strukturu, ale i **hodnoty** nenulových prvků. Obecně tedy pak budeme hovořit o grafech (grafových modelech) matice.

6.1.1 Grafový model čtvercové symetrické matice

Neorientovaným **grafovým modelem** nebo také **grafem** symetrické matice $A \in \mathbf{R}^{n\times n}$ (tedy pro $m = n$) nazveme **neorientovaný** graf $G = (V, E)$, kde $V = \{v_1, \ldots, v_n\}$, $E \subseteq \binom{V}{2}$. Přitom položíme

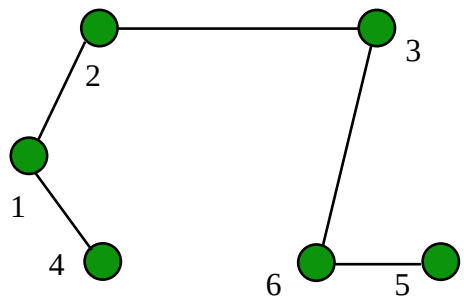
$$E = \{\{v_i, v_j\} | a_{ij} \neq 0 \wedge v_i \neq v_j\}\,.$$

Nenulovým prvkům z matice tedy přiřazujeme bijekcí neorientované hrany tohoto grafu. V následujícím uvádíme příklad symetrické matice a odpovídajícího grafu $G = (V, E) = (\{1, \ldots, 6\}, E)$. Matice je znázorněna na Obrázku 6.1.1 a odpovídající orientovaný graf na Obrázku 6.1.2.

Model neorientovaného grafu můžeme využít k zachycení struktury řídkosti matice A také v případě, je-li matice **symetrická pouze strukturálně** a nikoli numericky. To jest v případě, že existují různé indexy $i_1, j_1 \in \{1, \ldots, n\}$ takové, pro které $a_{i_1 j_1} \neq a_{j_1 i_1}$ a zároveň je splněno, že platí-li pro nějaký prvek a_{ij} matice A $a_{ij} \neq 0$, pak také $a_{ji} \neq 0$.

$$A = \begin{bmatrix} 3 & 2 & & 1 & & \\ 2 & 4 & 5 & & & \\ & 5 & 5 & & & 5 \\ 1 & & & 8 & & \\ & & & & 2 & 8 \\ & & 5 & & 8 & 1 \end{bmatrix}$$

Obrázek 6.1.1: *Symetrická matice* $A \in \mathbf{R}^{n \times n}$.



Obrázek 6.1.2: *Neorientovaný graf* $G = (\{1, \ldots, 6\}, E)$ *odpovídající matici z Obrázku 6.1.1.*

6.1.2 Grafový model čtvercové nesymetrické matice

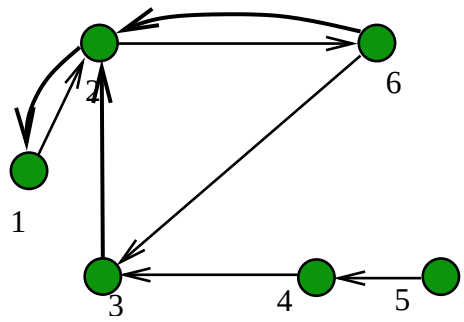
Orientovaným grafem (grafovým modelem) čtvercové a obecně **nesymetrické matice** $A \in \mathbf{R}^{n \times n}$ nazveme prostý orientovaný graf $G = (V, E)$ vytvořený následujícím způsobem. Jeho množina vrcholů je $V = \{v_1, \ldots, v_n\}$ a jeho množina hran je definována předpisem

$$E = \{(v_i, v_j) \mid a_{ij} \neq 0, v_i \neq v_j\}.$$

Obrázek 6.1.3 ukazuje příklad nesymetrické matice, kde pro názornost zapisujeme i její číselné hodnoty, které ale základním modelem prostého grafu bez výše zmíněných ohodnocení nezachytíme. Neorientovaný graf $G = (\{1, \ldots, 6\}, E)$ příslušející této matici je znázorněn na Obrázku 6.1.4.

$$A = \begin{pmatrix} 5.1 & 3.4 & & & & \\ 2.8 & 1.2 & & & & 3.6 \\ & 3.6 & 4.8 & & & \\ & & 8.5 & 5.5 & & \\ & & & 4.0 & 2.6 & \\ & 1.2 & 5.0 & & & 7.1 \end{pmatrix}$$

Obrázek 6.1.3: *Nesymetrická matice* $A \in \mathbf{R}^{n \times n}$.



Obrázek 6.1.4: *Orientovaný graf* $G = (\{1, \ldots, 6\}, E)$ *odpovídající matici z Obrázku 6.1.3.*

Výše uvedená možnost vzít jako množinu vrcholů grafu $V = \{1, \ldots, n\}$ umožňuje hovořit o **očíslování** vrcholů grafu a hodnoty tohoto očíslování ztotožnit s indexy řádků a/nebo sloupců matice. Poznamenejme, že tímto grafovým modelem můžeme také zachytit i strukturu řídkosti symeetrické matice, kde neorientované hraně odpovídají dvě hrany orientované.

6.1.3 Graf matice a její rozložitelnost

Souvislost rozložitelnosti matice s jejím orientovaným grafovým modelem popisuje následující Věta.

Věta 6.1.1 *Matice A dimenze n je rozložitelná právě tehdy, není-li její orientovaný grafový model silně souvislý.*

Důkaz: Je-li matice A rozložitelná, pak existuje podle Definice 3.2.2 permutační matice P taková, že platí

$$P^T AP = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}, \tag{6.1}$$

kde A_{11} a A_{22} jsou čtvercové netriviální matice. Uvažujme orientovaný graf $G = (V, E)$ matice $P^T AP$ a rozklad jeho množiny vrcholů V na dvě neprázdné podmnožiny V_1 a V_2, kde V_1 odpovídá řádkům a sloupcům matice A_{11} a V_2 odpovídá zbývajícím řádkům a sloupcům. Podle struktury řídkosti matice a tedy i grafu G je jasné, že neexistuje orientovaná cesta z žádného vrcholu množiny V_1 do žádného vrcholu množiny V_2. Graf G tedy nemůže být silně souvislý. Protože ale grafy matic A a $P^T AP$ jsou izomorfní, není silně souvislý ani graf matice A.

Předpokládejme nyní, že graf $G = (V, E)$ matice A není silně souvislý. Pak se dají najít dva vrcholy a a b z V takové, že neexistuje orientovaná cesta v grafu G z a do b. Definujme nyní množiny $V_{\Rightarrow b}, V_{a\Rightarrow}$ a V_3 následujícím způsobem. Nechť množina $V_{\Rightarrow b}$ obsahuje vrchol b a všechny vrcholy grafu, pro které existuje orientovaná cesta **do** vrcholu b. Nechť dále množina $V_{a\Rightarrow}$ obsahuje vrchol a a všechny ostatní vrcholy, do kterých vede orientovaná cesta **z** vrcholu a. Množiny $V_{\Rightarrow b}$ a $V_{a\Rightarrow}$ jsou tedy neprázdné. Jsou navíc také disjunktní, neboť z existence vrcholu, který by byl zároveň v obou množinách $V_{\Rightarrow b}$ a $V_{a\Rightarrow}$

bychom dostali spor s předpokladem, že neexistuje orientovaná cesta z a do b. Zaveďme nyní označení

$$V_{other} = V \setminus (V_{\Rightarrow b} \cup V_{a\Rightarrow})$$

a zvolme permutační matici P tak, aby $P^T AP$ obsahovala nejprve řádky a sloupce odpovídající množině $V_{a\Rightarrow}$, a pak postupně řádky a sloupce odpovídající množinám V_{other} a $V_{\Rightarrow b}$. Výslednou permutovanou matici můžeme znázornit následovně:

$$\begin{array}{c} \\ V_{a\Rightarrow} \\ V_{other} \\ V_{\Rightarrow b} \end{array} \begin{array}{c} \begin{array}{ccc} V_{a\Rightarrow} & V_{other} & V_{\Rightarrow b} \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \end{array}.$$

Žádná orientovaná cesta z nějakého vrcholu z $V_{a\Rightarrow}$ do nějakého vrcholu z $V_{\Rightarrow b}$ nemůže existovat, protože pak by existovala orientovaná cesta $a \Rightarrow b$. To implikuje $A_{13} = 0$. Nemůže ale existovat ani žádná orientovaná cesta z nějakého vrcholu z V_{other} do nějakého vrcholu z $V_{\Rightarrow b}$, protože takový vrchol by musel být ve $V_{\Rightarrow b}$ a ne v množině V_{other}. To tedy znamená, že také $A_{23} = 0$ a matice A musí být rozložitelná, jak jsme měli dokázat. ■

6.1.4 Grafový model struktury řídkosti obdélníkové matice

Všimněme si, že ve výše uvedených grafových modelech nezachycujeme **nenulovost diagonálních prvků** matice. To by se dalo provést speciálními hranami, kterým se říká smyčky, ale většinou není nutné grafový model takto doplňovat. Nulovost a nenulovost diagonálních prvků může zachytit pro matici $A \in \mathbf{R}^{m\times n}$ bipartitní graf (to jest graf s chromatickým číslem 2). Tento graf může zachytit i strukturální informaci v matici, která je obdélníková a má obecně nestejný počet řádků m a sloupců n. Formálně jej zapíšeme $G = (R, C, E)$, kde $|R| = m$, $|C| = n$, $E \subset R \times C$ a kde platí

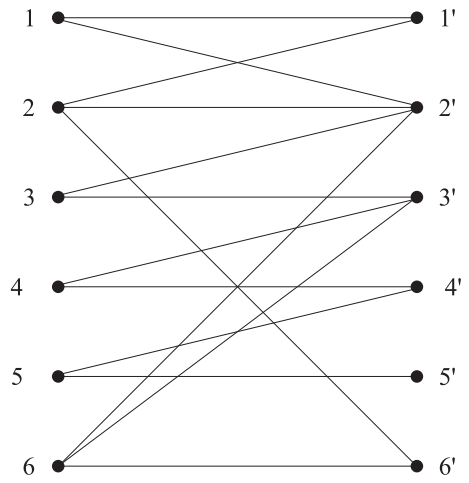
$$E = \{(i, j') \mid i \in R, j' \in C, a_{ij} \neq 0\}.$$

V tomto případě se nenulovost diagonálního prvku a_{ii} vyjadřuje existencí hrany (i, i'). I zde volíme množiny vrcholů jednoduše jako $\{1, \ldots, m\}$ a $\{1', \ldots, n'\}$ místo obecných množin, po řadě, R a C. Důvodem je jednoduchost s jakou můžeme zavést uspořádání vrcholů v těchto množinách. Množinu R budeme někdy nazývat řádkovou množinou bipartitního grafu a C jeho sloupcovou množinou.

Na Obrázku 6.1.5 je znázorněn bipartitní graf $G = (R, B, E)$ odpovídající matici z Obrázku 6.1.3.

6.1.5 Číselné hodnoty prvků matice a diagonální prvky v grafových modelech

Obecně máme v řídké matici jak informaci o její struktuře, tak informaci číselnou, tedy numerické hodnoty jejích prvků. Modely prostých grafů, tak jak jsme o nich hovořili výše, zachycujeme **pouze** strukturu řídkosti matice. Někdy je ale potřeba pojmy prostého



Obrázek 6.1.5: *Bipartitní graf $G = (R, B, E)$ zachycující strukturální informaci v matici z Obrázku 6.1.3 .*

grafu rozšířit zavedením ohodnocení (vah) pro vrcholy a/nebo hrany grafu. Formálně tak přidáváme k definici prostého grafu zobrazení

$$w : \mathbf{R} \to V \quad \text{a/nebo} \quad c : \mathbf{R} \to E.$$

Grafy (grafové modely) doplněné takovým zobrazením nazýváme **ohodnocené** nebo **vážené**. Případně lze zmínit konkrétní typ ohodnocení. Oba typy ohodnocení, **hranové** i **vrcholové**, se mohou kombinovat a jejich hodnoty mohou odpovídat numerickým hodnotám prvků matice. V případě matic symetrických a pozitivně definitních víme, že i transformace matice a příslušná transformace grafového modelu při **přesném** Choleského rozkladu nenulovost diagonálních prvků nemění. V ostatních případech diskutujeme nenulovost diagonálních prvků často samostatně, mimo grafový model.

6.2 Očíslování grafu a přeuspořádání matice

Ze vztahu mezi maticemi a grafovými modely plyne, že očíslování grafového modelu souvisí se zobrazením vrcholů do přirozených čísel příslušných matic. Následující definice tuto souvislost formalizuje.

Definice 6.2.1 **Očíslováním** *vrcholů grafu $G = (V, E)$ o n vrcholech nazveme bijekci $\alpha : V \leftrightarrow \{1, \ldots, n\}$. Bude-li potřeba, graf G s daným očíslováním α budeme značit rozšířenou notací $G = (V, E, \alpha)$.* **Očíslováním** *bipartitního grafu $G = (R, B, E)$ nazveme bijekci $\alpha : R \leftrightarrow \{1, \ldots, m\}$ a $\beta : B \leftrightarrow \{1, \ldots, n\}$. Graf G s tímto dvojím očíslováním α, β budeme analogicky značit $G = (R, B, E, \alpha, \beta)$.*

Přečíslovaným *neorientovaným grafem pak rozumíme graf $G_\alpha = (V_\alpha, E_\alpha)$ izomorfní původnímu grafu, kde $V_\alpha = \alpha(V)$ a $E_\alpha = \{\{\alpha(i), \alpha(j)\} \mid \{i, j\} \in E\}$. Analogicky definujeme přečíslování pro ostatní typy grafů.*

Operace přečíslování prostého orientovaného nebo neorientovaného grafu, ve které získáme G_α, indukuje **symetrickou permutaci** řádků a sloupců matice. Jinými slovy, očíslování α odpovídá permutační matici P takové že

$$\alpha(i) = j \Leftrightarrow P_{ij} = 1. \tag{6.2}$$

O permutované matici PAP^T budeme hovořit jako o matici přeuspořádané nebo nově uspořádané na základě nového očíslování vrcholů grafu. Nebude-li moci dojít k nedorozumění, budeme používat také termíny **nové uspořádání** vrcholů grafu, **přečíslování matice** nebo **přeuspořádání** vrcholů grafu majíce na mysli jednoznačný vztah přečíslování grafu a přeuspořádání matice. V případě bipartitního grafu, kde nezávisle přečíslujeme dvě množiny vrcholů, indukuje tato operace obecně nesymetrickou permutaci (přeuspořádání) matice PAQ, kde P a Q jsou, po řadě, řádkové a sloupcové permutační matice odpovídající daným bijekcím α a β.

7

Obecné schéma řešení soustav lineárních rovnic

V této kapitole si zopakujeme některá fakta týkající se řešení soustav lineárních algebraických rovnic pomocí rozkladů, které vycházejí z Gaussovy eliminace nebo příbuzných algoritmů. Tyto rozklady nejprve připomeneme pro husté matice a poté se zaměříme na matice řídké.

7.1 Gaussova eliminace a LU rozklad husté matice.

Předpokládejme, že matice A soustavy je **hustá**. Eliminační algoritmus řešení soustav lineárních algebraických rovnic

$$Ax = b, \tag{7.1}$$

pro $A \in \mathbf{R}^{n\times n}, x \in \mathbf{R}^n$, nazývaný **Gaussova eliminace** spočívá v (1) transformaci soustavy na ekvivalentní systém s trojúhelníkovou maticí a v (2) zohlednění této redukce při rekonstrukci jejího řešení. Z hlediska výpočetní složitosti hraje centrální roli první krok a proto se budeme věnovat především této fázi eliminace, i když do celého algoritmu řešení soustavy rovnic musíme zahrnout také práci s vektorem pravé strany b. Maticový a algoritmický zápis této transformace ve formě **rozkladu** jsou mnohem novější než původní expozice. Teprve v historicky nedávné době, až koncem padesátých let a začátkem šedesátých let dvacátého století, došlo k přeformulování algoritmu Gaussovy eliminace na dekompoziční (faktorizační) algoritmus, který vyjadřuje matici A ve formě rozkladu na dvě trojúhelníkové matice, tedy ve tvaru $A = LU$ nebo varianty takového rozkladu.

7.1.1 Gaussova eliminace husté matice.

Obsahem následující definice je pojem **silně regulární** matice, který se používá při diskusi o existenci rozkladu.

Definice 7.1.1 *Čtvercová matice $A \in R^{n\times n}$ je* **silně regulární**, *jestliže všechny její hlavní minory (determinanty jejích hlavních podmatic) jsou různé od nuly.*

Příklady jednoduchých matic, které jsou regulární, ale nejsou silně regulární, jsou znázorněny v (7.2).

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{7.2}$$

Následující dvě tvrzení uvedeme bez důkazu a předpokládáme, že je s nimi čtenář obeznámen.

Věta 7.1.1 *Matice* **symetrické a pozitivně definitní** *jsou silně regulární.*

Věta 7.1.2 *Pro silně regulární matici $A \in R^{n\times n}$ existuje jednoznačně určená (regulární) dolní trojúhelníková matice (faktor) L s jednotkovými diagonálními prvky a jednoznačně určená regulární horní trojúhelníková matice (faktor) U takové, že platí*

$$A = LU. \tag{7.3}$$

Poznámka 7.1.1 *Předpoklad, že faktor L má jednotkovou diagonálu (jednotkové diagonální prvky), je standardní a zabezpečuje jednoznačnost faktorů rozkladu. Kdybychom vyjádřili jednotlivé prvky rozkladu pomocí soustavy n^2 rovnic pro $n(n+1)$ neznámých (prvků faktorů U a L), museli bychom pro jednoznačnost n jejich hodnot zvolit. Jednotková diagonála faktoru L je touto standardní volbou.*

Nejčastější prezentace algoritmu Gaussovy eliminace spočívá v systematickém nulování obecně nenulových prvků čtvercové matice $A \in R^{n\times n}$ na jejích poddiagonálních pozicích. Maticově můžeme první krok tohoto procesu, tedy nulování poddiagonálních prvků v prvním sloupci matice A, zachytit následujícím částečným rozkladem

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ a_{21}/a_{11} & 1 & & & \\ a_{31}/a_{11} & & 1 & & \\ \vdots & & & 1 & \\ a_{n1}/a_{11} & & & & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} \equiv M_1 A^{(1)}. \tag{7.4}$$

Příslušný **Schurův doplněk** S matice A vzhledem k prvku a_{11} po prvním kroku procesu označíme také

$$S = A_R^{(1)} = \begin{pmatrix} a_{22}^{(1)} & \dots & \dots & a_{2n}^{(1)} \\ a_{32}^{(1)} & \dots & \dots & a_{3n}^{(1)} \\ \vdots & & \ddots & \vdots \\ a_{n2}^{(1)} & \dots & \dots & a_{nn}^{(1)} \end{pmatrix} \in R^{(n-1)\times(n-1)}, \tag{7.5}$$

kde index R znamená, že částečně eliminovaná matice $A^{(1)}$ je redukovaná. Tuto redukovanou matici budeme někdy také nazývat **aktivní část částečně eliminované matice**, protože na ní se soustředí aktivita dalších kroků rozkladu (v této variantě, která je založena na výpočtu posloupnosti Schurových doplňků). Nulováním poddiagonálních prvků matice $A^{(1)}$ v jejím druhém sloupci získáme matici $A^{(2)}$, což vyjádříme následujícím rozkladem.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & a_{32}/a_{11} & 1 & \\ & \vdots & & 1 \\ & a_{n2}/a_{11} & & & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & \dots & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} \equiv M_2 A^{(2)}. \tag{7.6}$$

Analogicky s předcházejícím označíme Schurův doplněk S matice $A_R^{(1)}$ vzhledem k prvku $a_{22}^{(1)}$ výrazem $A_R^{(2)}$ a analogicky budeme postupovat i v dalších krocích. Opakovanou aplikací tohoto postupu získáme po $n-1$ krocích posloupnost dolních trojúhelníkových matic $M_1, M_2, \dots, M_{n-1}$ s jednotkovou diagonálou, které nazýváme sloupcové eliminační matice a také získáme posloupnost částečně eliminovaných matic $A^{(1)}, A^{(2)}, \dots, A^{(n-1)}$. Tyto posloupnosti spojuje vztah

$$A = M_1 M_2 \dots M_{n-1} A^{(n-1)}, \tag{7.7}$$

Součin dolních trojúhelníkových matic $M_1 M_2 \dots M_{n-1}$ je také dolní trojúhelníková matice, která má také jednotkovou diagonálu. Výslednou matici označíme L. Zároveň si ale všimněme následující velmi podstatné věci, kterou zapíšeme jako poznámku

$$L = \begin{pmatrix} 1 & & & \\ a_{21}^{(0)}/a_{11}^{(0)} & 1 & & \\ a_{31}^{(0)}/a_{11}^{(0)} & & & \\ \vdots & \vdots & \ddots & \\ a_{n1}^{(0)}/a_{11}^{(0)} & & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 & & & \\ & 1 & & \\ & a_{32}^{(1)}/a_{22}^{(1)} & & \\ \vdots & \vdots & \ddots & \\ & a_{n2}^{(1)}/a_{22}^{(1)} & \dots & 1 \end{pmatrix} \dots = \begin{pmatrix} 1 & & & \\ a_{21}^{(0)}/a_{11}^{(0)} & 1 & & \\ a_{31}^{(0)}/a_{11}^{(0)} & a_{32}^{(1)}/a_{22}^{(1)} & & \\ \vdots & \vdots & \ddots & \\ a_{n1}^{(0)}/a_{11}^{(0)} & a_{n2}^{(1)}/a_{22}^{(1)} & \dots & 1 \end{pmatrix} \tag{7.8}$$

Poznámka 7.1.2 *Poddiagonální prvky, které explicitně vystupují v i-tém sloupci matice M_i, $i = 1, \dots, n-1$, jsou prvky dolního trojúhelníkového faktoru L.*

To tedy znamená, že když používáme klasickou implementaci Gaussovy eliminace založenou na násobení eliminačními maticemi zleva, můžeme jejich součin L pomocí skalárů, které jsou mezivýsledky této eliminace. Matice $A^{(n-1)}$ je horní trojúhelníková a budeme ji dále označovat U. Její prvky získáváme **také** během procesu rozkladu. Konkrétně, jako naddiagonální prvky v i-tém řádku matice $A^{(i)}$, $i = 0, \dots, n-1$. Postupem Gaussovy eliminace jsme tedy zároveň získali rozklad $A = LU$, který nazýváme LU rozkladem a **to nám umožňuje hovořit o LU rozkladu namísto Gaussovy eliminace** maje na mysli výše uvedenou vzájemnou **ekvivalenci.** Připomeňme ještě, že rozklad používá pouze operace, které nemění hodnost matice. Faktory silně regulární matice jsou tedy také regulární. Z hlediska algoritmu je proveditelnost rozkladu ekvivalentní nenulovosti prvních $n-1$ diagonálních prvků faktoru U, což je mírně slabší podmínka než silná regularita. Následující jednoduché tvrzení nám tento problém přiblíží.

Lemma 7.1.1 *Není-li matice silně regulární, pak je některý diagonální prvek faktoru U roven nule.*

Důkaz: Není-li matice silně regulární, znamená to, že některý z jejích hlavních minorů je nulový. Označme si k, $1 \leq k \leq n$, nejmenší index takový, že minor čtvercové podmatice určené prvními k řádky a sloupci je nulový. Je-li $k = 1$, tvrzení je triviální. Pro $k > 1$ to znamená, že podmatice faktorů $L_{1:k-1,1:k-1}$ a $U_{1:k-1,1:k-1}$ jsou regulární. Pak ale nulovost minoru podmatice $A_{1:k,1:k}$, čili její singularita, je ekvivalentní tomu, že prvek $u_{k,k}$ je roven nule. V opačném případě bychom totiž $A_{1:k,1:k}$ rozložili na součin dvou regulárních trojúhelníkových faktorů $A_{1:k,1:k} = L_{1:k,1:k}U_{1:k,1:k}$ a to by byl spor. Tvrzení lemmatu plyne použitím matematické indukce. ■

7.1.2 Přímý a zpětný chod pro nalezení řešení soustavy lineárních rovnice.

Jsou-li faktory LU rozkladu spočteny, řešení soustavy (7.1) se získá v následujících dvou krocích. V prvním kroku nazývaném **přímý chod**, nalezneme vektor $y \in \mathbf{R}^n$ řešením soustavy lineárních rovnic

$$Ly = b \tag{7.9}$$

s dolní trojúhelníkovou maticí L. V druhém kroku získáme vektor $x \in \mathbf{R}^n$ soustavy 7.1 řešením soustavy lineárních rovnic

$$Ux = y \tag{7.10}$$

s horní trojúhelníkovou maticí U, což se nazývá **zpětný chod.**

Je-li Gaussova eliminace prezentována jako nulování poddiagonálních prvků matice, pak se přímý chod někdy provádí současně s konstrukcí faktoru U implicitní aplikací matice L^{-1} tak, že se vektor b pravé strany zahrne do **rozšířené soustavy** jako další sloupec matice soustavy. Pak můžeme podle (7.4) po aplikaci první sloupcové eliminační matice M_1 psát

$$\begin{pmatrix} A & b \end{pmatrix} = M_1 \begin{pmatrix} A^{(1)} & M_1^{-1}b \end{pmatrix}. \tag{7.11}$$

a celkově dostaneme

$$\begin{pmatrix} A & b \end{pmatrix} = M_1 M_2 \ldots M_{n-1} \begin{pmatrix} A^{(n-1)} & L^{-1}b \end{pmatrix}. \tag{7.12}$$

Pro nalezení řešení x pak už stačí jen provést zpětný chod.

7.1.3 Varianty výpočtu LU rozkladu.

Pojetí Gaussovy eliminace jako rozkladu, ve kterém nejprve získáme faktory L a U a oddělíme od jejich konstrukce přímý a zpětný chod zvyšuje flexibilitu algoritmů. To také motivovalo studium různých dalších algoritmických variant LU rozkladu, které se liší **vzájemným** pořadím operací, ale poskytují **matematicky ekvivalentní** výsledné faktory i v případě nepřesné aritmetiky počítače za předpokladu, že operace jsou prováděny striktně ve stejném pořadí bez dodatečných změn ve výpočtu vyvolaných konkrétní architekturou či jejím programovým vybavením. Podotkněme, že tento předpoklad není

ale v praxi na moderních počítačových architekturách většinou splněný. V praxi mohou být některé algoritmy vhodnější než jiné ale hlavně z důvodu různé organizace datových přesunů během výpočtu faktoru. Algoritmy se mohou se také lišit odlišným pořadím operací či různými způsoby ukládáním mezivýsledků. Velký význam volbě varianty se začal přikládat především v období od začátku 80. let minulého století v souvislosti s rozvojem počítání na vektorových počítačích [47], [127] a později i důrazem na přizpůsobení algoritmů hierarchické struktuře paměti, která má na dosažení vysokého výkonu implementací podstatný vliv. Ale volba vhodné varianty je aktuální i dnes. V následujícím si proto některé varianty rozkladů popíšeme. Nejprve pro obecně husté matice a poté i pro řídké matice.

7.1.3.1 LU rozklad v generickém schématu

Bližší pohled na LU rozklad ukazuje, že fixujeme-li roli indexů danou operací, pak je možné jej přepsat generickým schématem se třemi vnořenými smyčkami, jak je znázorněno níže:

Algoritmus 7.1.1 *Generické schéma algoritmu LU rozkladu.*

1. **for** ————————-
  for ————————-
    for ————————-
      $a_{ij} = a_{ij} - l_{ik}a_{kj}$
    end
  end
end

Základní aritmetické operace zůstává stále stejná a je celkem je šest možností jak přiřadit indexy i, j a k jednotlivým cyklům. Meze jednotlivých cyklů jsou pak jednoznačně určeny. Každá z možností odpovídá jiné variantě základního postupu, ale tyto varianty je možné sdružit po dvou do skupin označených jako řádkové, sloupcové nebo podmaticové schéma LU rozkladu, když nepřihlížíme ke vzájemnému pořadí operací dvou vnitřních cyklů. Jednotlivá schémata (algoritmy) si nyní přiblížíme.

7.1.4 Podmaticový algoritmus LU rozkladu

Počítání faktorů v $n-1$ krocích podle (7.4), (7.6), (7.7) vytváří v každém vnějším kroku (kroku vnějšího cyklu) jeden sloupec faktoru L a jeden řádek faktoru U. Schématicky lze krok rozkladu matice

$$A = \begin{pmatrix} a_{11} & u^T \\ v & C \end{pmatrix}, \tag{7.13}$$

kde jsme položili

$$v = a_{2:n,1},\ C = A_{2:n,2:n},\ u^T = a_{1,2:n} \tag{7.14}$$

zapsat

$$A = M_1 \begin{pmatrix} a_{11} & u^T \\ & C - vu^T/a_{11} \end{pmatrix} = \begin{pmatrix} 1 & \\ v/a_{11} & I \end{pmatrix} \begin{pmatrix} a_{11} & u^T \\ & C - vu^T/a_{11} \end{pmatrix} \equiv \begin{pmatrix} 1 & \\ v/a_{11} & I \end{pmatrix} \begin{pmatrix} a_{11} & u^T \\ & S \end{pmatrix}, \tag{7.15}$$

kde matice S je **Schurův doplněk** matice A vzhledem k prvku a_{11}. Výpočet, kde se stejný princip rozkladu aplikuje na podmatici S a postupně se v $n-1$ krocích spočtou faktory L a U, se nazývá **podmaticový** LU rozklad, nebo také LU rozklad s vnějším součinem. Je snadno vidět, že podmaticový algoritmus má v generickém schématu index vnějšího cyklu k. Přiřazení indexů i a j vnitřním cyklům, které vytváří dvě varianty podmaticového algoritmu, je pak obvykle sladěno s konkrétním uložením dat matice A. Pro větší názornost uvedeme tento LU rozklad také jako algoritmické schéma s podrobným rozpisem počítáním prvků faktorů L a U.

Algoritmus 7.1.2 *Podmaticový LU rozklad pro výpočet $A = LU, A = (a_{i,j}), L = (l_{i,j}), U = (u_{i,j})$. Tato konkrétní varianta (podle pořadím dvou vnitřních cyklů) se nazývá algoritmus s řádkově orientovaným vnějším součinem a je někdy označovaná jako varianta* **kij***).*

```
1.  L = I_n, U = O_n
2.  for k = 1 : n − 1
3.      for i = k + 1 : n
4.          l_{i,k} = a_{i,k} a_{k,k}^{-1}
5.          for j = k + 1 : n
6.              a_{i,j} = a_{i,j} − l_{i,k} a_{k,j}
7.          end
8.      end
9.      u_{k,k:n} = a_{k,k:n}
10. end
11. u_{n,n} = a_{n,n}
```

Samozřejmě platí tvrzení následující Věty.

Věta 7.1.3 *Matice A s nenulovým prvkem $a_{1,1}$ je regulární právě tehdy, je-li regulární i* **Schurův doplněk** *matice A vzhledem k tomuto prvku.*

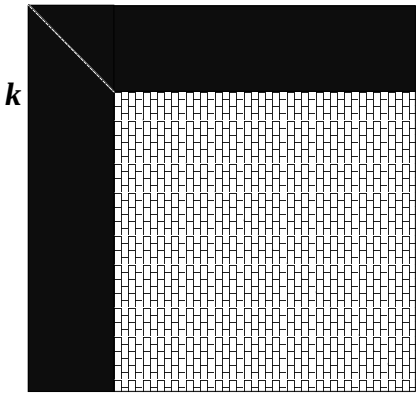
Výpočet faktorů v podmaticové variantě rozkladu je schématicky nakreslen na Obrázku 7.1.1, kde upravovaný Schurův doplněk znázorňujeme znázorňujeme pomocí výplně **cihličkami**. **Tmavá výplň** se používá pro označení vypočtených částí matice, které se stávají součástí faktorů L a U a v dalších krocích výpočtu se již nepoužívají.

7.1.4.1 Sloupcový a řádkový algoritmus LU rozkladu

Jeden krok rozkladu matice podle generického schématu, kde vnější index je j, budeme nazývat **sloupcový** algoritmus LU rozkladu. Blokově jej můžeme zachytit následujícím způsobem.

Pro $j = 1$ je situace jasná, protože mimoddiagonální prvky v prvním sloupci L jsou prvky prvního sloupce matice A dělené $u_{11} = a_{11}$. Předpokládejme nyní pro $1 < j < n$, že jsme v prvních $j-1$ krocích rozkladu spočítali prvních $j-1$ sloupců faktorů L a U matice $A \in R^{n\times n}$. To lze schématicky zapsat

$$\begin{pmatrix} L_{j-1} \\ L'_{j-1} \end{pmatrix} U_{j-1} = \begin{pmatrix} A_{1:j-1,1:j-1} \\ A_{j:n,1:j-1} \end{pmatrix}, \tag{7.16}$$



Obrázek 7.1.1: *Schéma podmaticového algoritmu LU rozkladu.*

kde

$$L_{j-1} \in R^{(j-1)\times(j-1)},\ L'_{j-1} \in R^{(n-j+1)\times(j-1)},\ U_{j-1} \in R^{(j-1)\times(j-1)} \tag{7.17}$$

již byly spočítány. Následující krok rozkladu, který získá j-tý sloupec l faktoru L a j-tý sloupec u faktoru U, zapíšeme

$$u_{1:j-1} = L_{j-1}^{-1} A_{1:j-1,j},\ u_{jj} = a_{j,j} - (L'_{j-1})_{j,1:j-1} u_{1:j-1},\ l_{j+1:n} = A_{j+1:n,j} - L'_{j-1} u_{1:j-1}. \tag{7.18}$$

Všimněme si přitom, že sloupcový LU rozklad se skládá ze dvou úplně odlišných částí. První z nich je založena na aplikaci L_{j-1}^{-1}, čili na **přímém chodu** s dosud vypočítanou částí faktoru L. Druhá z nich počítá poddiagonální část sloupce faktoru L jako lineární kombinaci sloupce A a sloupců z části dosud spočítaného faktoru L.

Posledním schématem nezmíněným dosud podrobněji je algoritmus **řádkový**, který má vnější index generického schématu i. Jeho formální popis zde není třeba uvádět, protože jej můžeme interpretovat také jako sloupcový algoritmus rozkladu matice A^T.

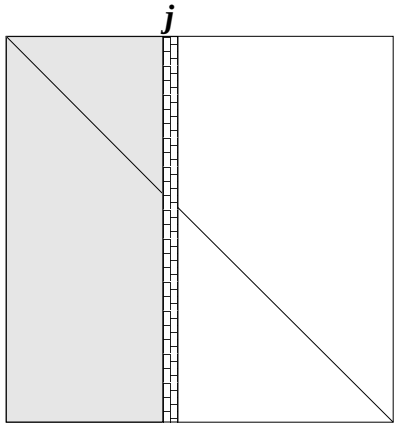
Znázornění sloupcového a řádkového algoritmu je, po řadě, na Obrázku 7.1.2 a obrázku 7.1.3 Na obou obrázcích vybarvujeme **šedě** ty části matice, které jsou již úplně vypočteny, patří tedy do faktorů, a dále se používají jen pro výpočet dalších prvků faktorů. Část matice, která se do příslušného kroku algoritmu dosud nepoužila, **nevybarvujeme**. Dvě možnosti, které máme v každém algoritmu pro přiřazení indexů vnitřním cyklům, schématické znázornění neovlivní.

7.1.4.2 Metoda ovroubení a další postupy

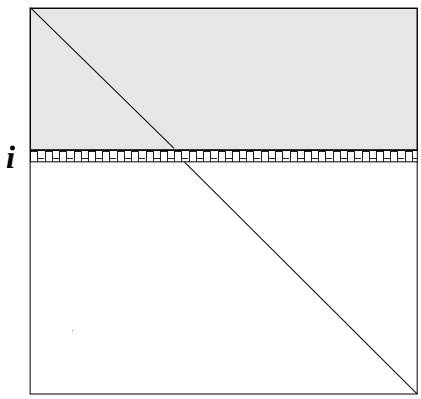
Výše uvedené algoritmy nejsou jediné možné, ale pak je zapotřebí vyjít za hranici generického schématu s operací

$$a_{i,j} = a_{i,j} - l_{i,k} a_{k,j} \equiv a_{i,j} = a_{i,j} - a_{i,k} a_{k,k}^{-1} a_{k,j}, \tag{7.19}$$

která fixuje indexy i, j a k. Varianta LU rozkladu nazývaná **metoda ovroubení** počítá v každém kroku rozkladu řádek faktoru L a sloupec faktoru U. Namísto detailního algoritmického postupu pro počítání prvků faktorů uvedeme maticový zápis, ze kterého postupy vycházejí. Tímto zápisem je následující rovnost, která platí pro $k = 2, \ldots, n$.



Obrázek 7.1.2: *Schéma sloupcového algoritmu LU rozkladu.*



Obrázek 7.1.3: *Schéma řádkového algoritmu LU rozkladu.*

$$A_{1:k,1:k} = \begin{pmatrix} A_{1:k-1,1:k-1} & A_{1:k-1,k} \\ A_{k,1:k-1} & a_{k,k} \end{pmatrix} = \begin{pmatrix} L_{1:k-1,1:k-1} & 0 \\ L_{k,1:k-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1:k-1,1:k-1} & U_{1:k-1,k} \\ 0 & u_{k,k} \end{pmatrix} \tag{7.20}$$

Pro tato k můžeme spočítat nový řádek $L_{k,1:k-1}$ faktoru L a nový sloupec $U_{1:k-1,k}$ faktoru U postupně z následujících dvou vztahů

$$\begin{aligned} L_{k,1:k-1}U_{1:k-1,1:k-1} &= A_{k,1:k-1}, \\ L_{1:k-1,1:k-1}U_{1:k-1,k} &= A_{1:k-1,k}. \end{aligned}$$

Diagonální prvek $u_{k,k}$ faktoru U pak splňuje vztah

$$u_{k,k} = a_{k,k} - L_{k,1:k-1}U_{1:k-1,k}.$$

Algoritmus započne výpočet položením

$$u_{1,1} = a_{1,1}.$$

V následujícím textu probereme podrobněji problém symetrické faktorizace. Jak jsme viděli výše, sloupcový i řádkový algoritmus LU rozkladu v sobě kombinují dva různé postupy. Symetrický případ vede k algoritmům, které tyto dva postupy oddělí. Zjednodušení vlivem symetrie matice zároveň umožní do popisu schémat rozkladu jednodušeji zahrnout řídkost matice.

7.2 Choleského rozklad řídké matice.

Jak jsme viděli výše, je-li matice A symetrická, může se stát, že A je sice regulární, ale není silně regulární. To se nestane, je-li A navíc i pozitivně definitní. LU rozklad pak nenaruší symetrii rozkladu a zachová i pozitivní definitnost, v případě, když opustíme předpoklad jednotkových diagonálních prvků matice L, a to je obsahem následujícího textu.

Lemma 7.2.1 *Uvažujme jeden krok podmaticové varianty rozkladu symetrické a pozitivně definitní matice A. Schurův doplněk matice A vzhledem k (kladnému) prvku $a_{1,1}$ je pozitivně definitní.*

Důkaz: Pro vektor $\begin{pmatrix} \alpha & z^T \end{pmatrix}^T$ můžeme napsat

$$\begin{aligned} x^T A x &= \begin{pmatrix} \alpha & z^T \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2:n} \\ a_{2:n,1} & A_{2:n,2:n} \end{pmatrix} \begin{pmatrix} \alpha \\ z \end{pmatrix} \\ &= \alpha^2 a_{1,1} + \alpha a_{1,2:n} z + \alpha z^T a_{2:n,1} + z^T A_{2:n,2:n} z \\ &= (\alpha + a_{1,1}^{-1} a_{1,2:n} z)^T a_{1,1} (\alpha + a_{1,1}^{-1} a_{1,2:n} z) + z^T (A_{2:n,2:n} - a_{2:n,1} a_{1,1}^{-1} a_{1,2:n}) z \end{aligned}$$

Zvolme libovolné $z \neq 0$ a položme $\alpha = -a_{1,1}^{-1} a_{1,2:n} z$. Ve předchozí rovnosti pak dostaneme

$$x^T A x = z^T S z,$$

kde

$$S = A_{2:n,2:n} - a_{2:n,1}a_{1,1}^{-1}a_{1,2:n}$$

je příslušný Schurův doplněk, který tak musí být pozitivně definitní. ■

V rozkladu symetrické a pozitivně definitní matice A platí pro jednoznačně určené trojúhelníkové matice L a U z Věty 7.1.2 vztahy

$$U = DL^T, A = LDL^T \tag{7.21}$$

kde $D = diag(U)$ je diagonální matice, jejíž všechny diagonální prvky jsou kladné. Hovoříme pak o LDL^T rozkladu a někdy jej nazýváme také **bezodmocninová Choleského faktorizace** nebo **symetrická Gaussova eliminace**.

LDL^T rozklad můžeme upravit do tvaru, kde rozložená matice má jen dva faktory:

$$LDL^T = \tilde{L}\tilde{L^T},$$

kde $\tilde{L} = LD^{1/2}$. Pro rozklady symetrických a pozitivně definitních matic je vzájemný přechod mezi LDL^T rozkladem a rozkladem $\tilde{L}\tilde{L^T}$ jednoznačně definován a druhému z těchto rozkladů se říká **Choleského faktorizace** nebo také odmocninová Choleského faktorizace. V tomto textu budeme často hovořit genericky o Choleského faktorizaci (odmocninové nebo bezodmocninové).

Je-li matice A řídká, struktura řídkosti dolní trojúhelníkové matice je v obou případech stejná. L a $\tilde{L}$ se liší pouze škálováním numerických hodnot diagonální maticí. Nebude-li moci dojít k záměně, dolní trojúhelníkový faktor budeme tak v obou případech označovat L. To bude například kdykoli, kdy budeme hovořit pouze o struktuře nenulových prvků. Není-li A pozitivně definitní, LDL^T rozklad nemusí existovat. Taková situace může nastat například, když provádíme rozklad pouze **přibližně** a modifikujeme jej tak změnami prvků matice. Místo matice A se pak rozkládá modifikovaná matice $A + E$, pro nějakou matici modifikace E a ta nemusí být pozitivně definitní. Existuje-li pak přesto LDL^T, pak mohou být některé prvky jeho diagonálního faktoru D záporné. V takových případech může tedy existovat (bezodmocninový) LDL^T rozklad, ale obecně nikoli odmocninová Choleského faktorizace. Příklad takového rozkladu je na následujícím obrázku

$$\begin{pmatrix} -2 & -6 & 4 \\ -6 & -21 & 15 \\ 4 & 15 & -13 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -2 & -1 & 1 \end{pmatrix} \begin{pmatrix} -2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} 1 & 3 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

Pro řešení soustav se symetrickými a obecně indefinitními maticemi se ale z důvodů jeho možné neexistence a stability často uvažuje jiná základní forma rozkladu odvozená z Gaussovy eliminace, kterou zmíníme později a kterou budeme terminologicky odlišovat od dosud diskutovaných variant symetrického rozkladu.

V textu níže postupně probereme sloupcové, řádkové a podmaticové algoritmy Choleského faktorizace, kde příslušná varianta je nazvána podle toho, jak se v obecném LU rozkladu počítá faktor L. Poznamenejme dále, že metoda ovroubení se v symetrickém případě redukuje na řádkový algoritmus.

7.2.1 Podmaticový Choleského rozklad

Podmaticový Choleského rozklad se liší od počítání podmaticové varianty LU rozkladu jen tím, že se počítají pouze dolní trojúhelníkový faktor L a diagonální faktor D. V případě odmocninového Choleského rozkladu faktor L pak škálujeme převrácenými hodnotami odmocnin diagonálních prvků. Pro větší názornost jsou v následujícím příkladu znázorněny dva kroky podmaticové varianty odmocninového Choleského rozkladu symetrické a silně regulární matice A, kde prvek faktoru L na pozici $(i,j), i,j \in \{1,\dots,n\}$, $j \le i$ je označen $l_{i,j}$.

$$
\begin{aligned}
A &= \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3:n} \\ a_{2,1} & a_{2,2} & a_{2,3:n} \\ a_{3:n,1} & a_{3:n,2} & A_{3:n,3:n} \end{pmatrix} \\
&= \begin{pmatrix} \sqrt{a_{1,1}} & 0 & \\ \frac{a_{2,1}}{\sqrt{a_{1,1}}} & \sqrt{a^{(1)}_{2,2}} & \\ \frac{a_{3:n,1}}{\sqrt{a_{1,1}}} & \frac{a^{(1)}_{3:n,2}}{\sqrt{a^{(1)}_{2,2}}} & I_{n-2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & A^{(2)}_{3:n,3:n} - \frac{a_{3:n,1}a_{1,3:n}}{a_{1,1}} - \frac{a^{(1)}_{3:n,2}a^{(1)}_{2,3:n}}{a^{(1)}_{2,2}} \end{pmatrix} \begin{pmatrix} \sqrt{a_{1,1}} & \frac{a_{2,1}}{\sqrt{a_{1,1}}} & \frac{a_{1,3:n}}{\sqrt{a_{1,1}}} \\ 0 & \sqrt{a^{(1)}_{2,2}} & \frac{a^{(1)}_{2,3:n}}{\sqrt{a^{(1)}_{2,2}}} \\ & & I_{n-2} \end{pmatrix} \\
&= \begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3:n,1} & l_{3:n,2} & I_{n-2} \end{pmatrix} \begin{pmatrix} l_{1,1} & l_{2,1} & l_{1,3:n} \\ 0 & l_{2,2} & l_{2,3:n} \\ 0 & 0 & I_{n-2} \end{pmatrix}
\end{aligned}
$$

Je-li matice A řídká, i počítaný faktor rozkladu je často řídký, ale obvykle s jinou strukturou řídkosti než měla matice A, jak o tom budeme dále hovořit.

7.2.2 Sloupcový Choleského rozklad

Sloupcový Choleského rozklad založený na pořadí indexů v generickém schématu jki nebo jik je přepsán pomocí sloupcových operací jako Algoritmus 7.2.1. Pro názornost uvádíme opět jeho odmocninovou variantu.

Algoritmus 7.2.1 Sloupcový algoritmus Choleského faktorizace.
Input: *Řídká matice A.*
Output: *Odmocninový Choleského faktor $L = (l_{ij})$ matice $A = (a_{ij})$.*

1. **for** $j = 1 : n$ **do**
2. *Spočtěte pomocný vektor $t_{j:n}$*

$$
\begin{pmatrix} t_j \\ \vdots \\ t_n \end{pmatrix} = \begin{pmatrix} a_{jj} \\ \vdots \\ a_{nj} \end{pmatrix} - \sum_{\{k|l_{jk}\neq 0\}} l_{jk} \begin{pmatrix} l_{jk} \\ \vdots \\ l_{nk} \end{pmatrix} \tag{7.22}
$$

3. *Získejte sloupec faktoru L škálováním vektoru $t_{j:n}$*

$$
\begin{pmatrix} l_{jj} \\ \vdots \\ l_{nj} \end{pmatrix} = \frac{1}{\sqrt{t_j}} \begin{pmatrix} t_j \\ \vdots \\ t_n \end{pmatrix} \tag{7.23}
$$

4. **end** j

V popisu je použitý pomocný vektor $\left(t_1, \ldots, t_n\right)^T$. **Řídkost rozkládané matice** se v algoritmu projeví na první pohled tak, že ve sloupcových vektorech vystupují pouze nenulové prvky a také tím, že v sumě úprav uvažujeme pouze takové, které mají nenulový součinitel l_{jk}. Teoretické poznatky, které se týkají struktur řídkosti v průběhu rozkladu, budeme diskutovat v dalším textu.

7.2.3 Řádkový Choleského rozklad

Řádková faktorizace symetrické matice, která odpovídá generickému schématu s pořadím indexů, počítá v každém hlavním kroku algoritmu jeden řádek faktoru L. Základní schéma postupu uvádíme jako Algoritmus 7.2.2.

Algoritmus 7.2.2 Řádkový algoritmus Choleského faktorizace.
Input: *Řídká matice A.*
Output: *Choleského faktor L matice A.*

1. **for** $i = 1 : n$ **do**
2. *Vyřešte systém s trojúhelníkovou maticí*

$$L_{1:i-1,1:i-1} \begin{pmatrix} l_{i1} \\ \vdots \\ l_{i,i-1} \end{pmatrix} = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{i,i-1} \end{pmatrix} \tag{7.24}$$

3. *Spočtěte diagonální prvek* $l_{ii} = \sqrt{\left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\right)}$
4. **end** i

Jakkoli je řádkový algoritmus jednoduchý, základní problém může být také v jeho efektivnosti v případě řídké matice. Zatímco sloupcový algoritmus uvedený v předcházející sekci vyžaduje v každém kroku pouze úpravu pomocí lineární kombinace některých z předcházejících sloupců, což je postup dobře implementovatelný na moderních počítačových architekturách, řádkový algoritmus ve svém kroku 2 obsahuje řešení soustav s trojúhelníkovou a stále se zvětšující maticí. To obecně vede k rekurentní aplikaci přímého chodu. O některých technikách, které umožňují efektivnější implementaci, budeme hovořit dále.

7.3 LU rozklad řídké matice

Choleskému rozkladu řídké matice jsme věnovali speciální pozornost, protože v sobě obsahuje i základní skladebné prvky obecnějšího řídkého LU rozkladu. Obecnější schéma, které těchto prvků využívá uvedeme později v souvislosti s konkrétními implementacemi.

7.4 LU rozklad a přeuspořádání matice

Předpoklad silné regularity kladený na matici se může zdát velmi silný. Tato vlastnost totiž svazuje dohromady dvě vlastnosti matice dohromady: její **regularitu** a **konkrétní dané pořadí** řádků a sloupců v matici. Nicméně platí následující Věta 7.4.1, kterou si pro názornost dokážeme.

Věta 7.4.1 *Ke každé regulární matici A existuje řádková permutační matice P taková, že matice PA je silně regulární. Tato permutační matice se dá nalézt v průběhu LU rozkladu.*

Důkaz: Uvažujme podmaticovou variantu LU rozkladu, ve kterém před provedením k-tého kroku pro nějaké $1 \leq k \leq n$ permutujeme (přeuspořádáme) řádky příslušné matice $A \equiv A^{(0)}, A^{(1)}, \ldots, A^{(k-1)}$ tak, že diagonální prvek je v každé z těchto matic nenulový. Podle Lemmatu 7.1.1 je vlastnost silné regularity ekvivalentní tomu, že taková permutace existuje pro každé k, $1 \leq k \leq n$. To, že pro nějaké takové k nelze takovou permutaci najít a pro všechny předcházející kroky jsme takovou permutaci našli znamená, že v k-tém sloupci C_k matice $C = P_{k-1}A^{(k-1)}$, kde P_{k-1} označuje dosavadní řádkovou permutaci danou předcházejícími kroky rozkladu, jsou všechny prvky v $C_{k:n,k}$ nulové.

Případ $k = 1$ je triviální. Pro $k > 1$ to ale znamená, že k-tý sloupec matice C můžeme vyjádřit pomocí předcházejících sloupců neboť stačí vyjádřit $C_{1:k-1,k}$ pomocí sloupců $C_{1:k-1,1:k-1}$. a matice C není proto regulární a tudíž ani matice A není regulární a to je spor. ■

V praktických algoritmech LU faktorizace tedy silnou regularitu rozkládané matice ovlivníme řádkovou **permutací**. Analogicky je možné silné regularity docílit permutací sloupců. Z praktického hlediska je ale často výhodné hledat ještě obecnější permutaci matice A, kde se permutují jak její řádky, tak její sloupce. Předpokládejme, že rozkládaná matice je získána z matice A nesymetrickou permutací řádků a sloupců a že ji můžeme zapsat ve tvaru PAQ, kde P a Q jsou permutační matice. Pak tedy máme rozklad

$$PAQ = LU. \tag{7.25}$$

Řešení soustavy lineárních rovnic (7.1) pak získáme **permutovaným** přímým chodem jako řešení soustavy s dolní trojúhelníkovou maticí

$$Ly = Pb \tag{7.26}$$

a **permutovaným** zpětným chodem, to jest řešením soustavy

$$UQ^T x = y. \tag{7.27}$$

Volbě permutace (přeuspořádání) se budeme věnovat později, kde budeme brát do úvahy i strukturu nulových a nenulových prvků matice a kde zároveň ukážeme několik rozdílných přístupů k hledání konkrétního přeuspořádání, motivovaného i jinými cíli než jen dosažením silné regularity.

7.5 Choleského rozklad, LU rozklad a grafové modely

Náš úvod do LU, LDL^T a Choleského rozkladů nechal zatím stranou otázku, jak jsou struktury eliminační matice a jejich změny zachyceny pomocí grafů. V této sekci zavedeme dva druhy grafových modelů. Oba mohou být jak neorientované (pro diskutované symetrické rozklady) orientované (pro rozklady nesymetrické) nebo i bipartitní. První druh těchto grafových modelů zachycuje strukturu Schurova doplňku, který je v podmaticovém schématu právě tou maticí, se kterou se pracuje. Tyto grafy se nazývají nazývat **eliminační grafy** a budeme je značit

$$G^{(k)} = (V^{(k)}, E^{(k)}) \equiv (\{k+1, \ldots, n\}, E(A_R^{(k)})),\ k = 0, \ldots, n-1. \tag{7.28}$$

Pro připomenutí redukovaných částečně eliminovaných matic, které vyjadřují strukturu Schurových doplňků, uveďme následující příklad.

Příklad 7.5.1 *Eliminační graf $G^{(2)}$ odpovídající matici $A_R^{(2)}$ pro $A \in \mathbf{R}^{n\times n}$ je grafem*

$$G(\begin{pmatrix} a_{33}^{(2)} & \ldots & a_{3n}^{(2)} \\ \vdots & \ddots & \vdots \\ a_{n3}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix})$$

s množinou vrcholů $V^{(2)} = \{3, \ldots, n\}$

V textu budeme také používat **rozšířené eliminační grafy**, což jsou grafové modely částečně eliminovaných matic, kde jsou doplněné prvky příslušných sloupců faktoru L a symetrizovány. Konkrétně

$$\begin{aligned} G_E^{(0)} &= G(A), \\ G_E^{(k)} &= (V, E_E^{(k)}) \equiv (V, E(A^{(k)} + M_1 + \ldots M_k + M_1^T + \ldots M_k^T)),\ k = 1, \ldots, n-1. \end{aligned}$$

Z definice obou variant eliminačních grafů je vidět, že jejich vrcholy jsou uspořádány, což je dáno zavedením množiny vrcholů $V = \{1 \ldots, k\}$ respektive $V = \{1 \ldots, n\}$. V dalším textu budeme vždy, neřekneme-li jinak, za množinu vrcholů grafových modelů o $|V|$ vrcholech považovat

$$V = \{1, \ldots, |V|\}.$$

7.6 Nenulovost prvků matic a vektorů

V této kapitole zdůrazníme jeden podstatný fakt, který se týká jak symetrických tak i nesymetrických rozkladů v případě, že matici považujeme za řídkou. Jsou-li prvky matice obecná reálná čísla, pak to, které z nich je vhodné považovat za nenulové, nemusí být zřejmé. V některých algoritmech je totiž možné jejich chod urychlit tím, že některé nenulové, ale "malé" prvky zanedbáme. Budeme pak hovořit o neúplných nebo, obecněji, o přibližných rozkladech. Rozhodnutí o nenulovosti a potažmo o řídkosti matice může

tedy záviset na **motivaci** a je součástí výpočetních algoritmů pracujících s řídkými maticemi. Opačně, pro jednodušší popis indexových množin maticových prvků s nenulovými hodnotami lze v konkrétních situacích pominout, že některé z takto určených prvků mají aktuálně hodnotu nula tak, abychom **zjednodušili datové struktury** nebo algoritmy. Jindy můžeme také předpokládat předpokládat, že některé prvky matice mají sice nejprve hodnotu nula, ale ta se může v průběhu algoritmu manipulujícího s touto maticí změnit. Chceme-li mít již od počátku definitivní datovou strukturu pro uložení matice v tomto případě, budeme v ní s některými prvky hned pracovat jako s nenulovými. Tento přístup k rozdělení prvků na nenulové a ty ostatní je v souladu s naším pohledem na **aplikační** definici řídkosti matice, kde chceme využít faktu nulovosti části jejích prvků v praxi. Malé množství prvků s nulovou hodnotou, kterým tímto **apriorním** popisem upřeme jejich nulovost, efektivitě popisu nemusí ublížit a může naopak leccos zjednodušit. Označíme-li některé prvky matice za nenulové a jejich množinu pak používáme jako strukturu řídkosti této matice, může být tato počáteční volba **subjektivní**.

Velkou motivací pro apriorní stanovení nulovosti a nenulovosti prvků matice na jejích pozicích je situace, kdy chceme používat algoritmy, které pracují pouze s její strukturou řídkosti vyjádřenou **grafovým modelem**. Ten bez vrcholového a hranového ohodnocení nerozliší nenulovost prvků při změnách jejich hodnot. Na druhou stranu modelování struktury řídkosti matice a její změny v rozkladech pomocí grafového modelu může celý rozklad zpřehlednit a umožnit efektivní implementaci. Z toho důvodu budeme ve větší části tohoto textu používat následující předpoklad, který je formulován pro hodnoty reprezentující například prvky matic.

Předpoklad 7.6.1 Předpoklad o nevyrušení: *Výsledek operace sčítání, odčítání, nebo násobení hodnot považovaných za nenulové je opět nenulová hodnota.*

Tento předpoklad umožňuje mimo jiné říci, že struktura řídkosti součtu C matic A a B je dána sjednocením struktur řídkosti matic A a B. Za předpokladu o nevyrušení se pak jednodušeji modelují změny struktury řídkosti při jednoduchých aritmetických operacích s maticemi pomocí grafů. Ve většině algoritmů je případů, kdy dojde ke skutečnému vynulování prvků při operaci sčítání matic, velmi málo a mohou být zanedbány ve prospěch jednoduššího popisu. Navíc, ačkoli se pro konkrétní matici mohou některé její prvky stát nulovými, často (ale ne vždy) existuje takové přiřazení nenulových hodnot prvkům matice, že v používaných algoritmech k žádnému vynulování nedojde.

S označením množiny vrcholů

$$V = \{1, \ldots, |V|\}$$

můžeme zavést alternativní označení pro struktury řídkosti řádků a sloupců striktně trojúhelníkové části matice, které budeme v dalším textu používat. Uvažujme neorientovaný grafový model matice G matice A. Množiny

$$ladj_G(i) = \{1, \ldots, i-1\} \cap \{j \mid j \in adj_G(i)\} \quad \text{a} \quad hadj_G(i) = \{i+1, \ldots, n\} \cap \{j \mid j \in adj_G(i)\}.$$

vyjadřují, po řadě, struktury řídkosti řádku a sloupce striktně trojúhelníkové části matice A. Alternativně označíme po řadě, strukturu řídkosti řádku a sloupce matice A výrazy

$$row_A(j) \equiv \mathcal{S}(A_{j,1:j-1}) = \{k \mid k < j,\ a_{jk} \neq 0\},\ 1 \leq j \leq n. \tag{7.29}$$

$$col_L(j) \equiv \mathcal{S}(L_{j+1:n,j}) = \{k \mid k > j,\ l_{kj} \neq 0\},\ 1 < j \leq n. \tag{7.30}$$

Vyjádření pro *ladj* a *row* a také pro *hadj* a *col* jsou při platnosti předpokladu o nevyrušení ekvivalentní.

8

Komponenty řídkého Choleského rozkladu

Předpokládejme, že matice soustavy $A \in R^{n \times n}$ je řídká a symetrická a diskutujme její Choleského rozklad. Většina teorie zde uvedená nepotřebuje předpokládat, že je matice pozitivně definitní, protože se týká struktury řídkosti jejího dolního trojúhelníkového faktoru. Nicméně vždy **předpokládáme nenulovost diagonálních prvků** a vždy předpokládáme symetrický rozklad diskutovaný výše a jeho existenci. Struktura řídkosti faktoru L je stejná pro odmocninovou i bezodmocninovou variantu ve formě LDL^T rozkladu, pro jejíž formální existenci v přesné aritmetice stačí silná regularita matice A a kde mohou být na diagonále i záporné prvky, tak i pro její odmocninovou LL^T variantu, pro jejíž existenci matice pozitivně definitní musí být.

V této kapitole a v několika dalších předpokládáme, že maticové rozklady, pokud je lze provést, provádíme matematicky přesně. Později stručně zmíníme některé aspekty přesnosti a stability diskutovaných metod, které se týkají počítáním v aritmetice s konečnou přesností. Ještě později budeme diskutovat i přibližné (neúplné) rozklady, které nacházejí své použití v urychlování iteračních metod a kde je problém existence ještě komplikovanější a vede k různým modifikacím základního schématu.

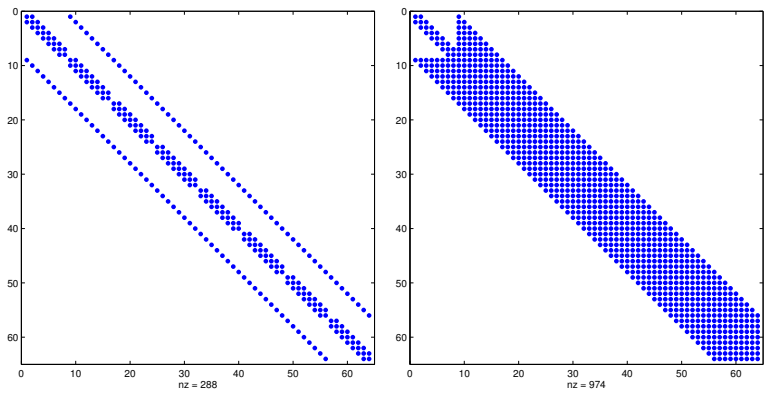
8.1 Zaplnění v Choleského rozkladu

Pokud neřekneme jinak (ale budeme občas zdůrazňovat), předpokládáme, že pro Choleského rozklad matice A platí **předpoklad o nevyrušení**. Pro řídkou symetrickou matici to znamená, že v jejím neorientovaném grafovém modelu $G(A) = (V, E)$ pro $1 \leq i, j \leq n$ platí ekvivalence

$$\{i, j\} \in E \Leftrightarrow a_{ij} \neq 0 \tag{8.1}$$

a platné jsou i analogické vztahy pro grafové modely částečně eliminovaných matice včetně grafů faktorů jejího LDL^T rozkladu.

Často se v analýze řídkého Choleského rozkladu předpokládá, že rozkládaná matice je **nerozložitelná**. Symetrická a rozložitelná matice je totiž blokově diagonální a úvahy, týkající se struktury řídkosti faktorů rozkladu, je možné aplikovat na jednotlivé bloky



Obrázek 8.1.1: *Příklad vzniku zaplnění v rozkladu matice z diskretizace na pravidelné dvourozměrné síti. Vlevo je znázorněna struktura řídkosti původní matice. Vpravo je znázorněna struktura řídkosti matice* $L + L^T$.

zvlášť. Choleského faktor původní symetrické rozložitelné matice je pak také rozložitelná matice, jejíž diagonální bloky lze sestavit z Choleského faktorů diagonálních bloků původní matice. V tomto textu to předpokládat nebudeme i z důvodu kompatibility s některými pojmy používanými pro popis rozkladů nesymetrických matic.

Následující pozorování vychází přímo z předpokladu o nevyrušení, protože **nenulovost** prvků A se pak beze změny přenáší do faktorů.

Pozorování 8.1.1 *Za předpokladu o nevyrušení pro Choleského rozklad čtvercové symetrické matice A platí následující vztah mezi její strukturou řídkosti a strukturou řídkosti jejího Choleského faktoru*

$$\mathcal{S}(A) \subseteq \mathcal{S}(L + L^T), \tag{8.2}$$

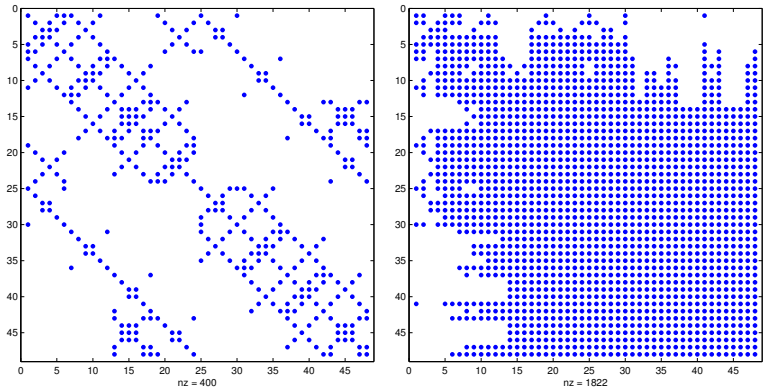
Ve faktoru L se ale mohou objevit nové nenulové prvky. Těm budeme říkat **prvky zaplnění (fill-in)** nebo krátce **zaplnění**. Pro algoritmy rozkladu je klíčové, kde a v jaké velikosti zaplnění vzniká. Velmi často je prvků zaplnění řádově více než prvků původní matice. Demonstrujme si tento jev nejprve na Obrázcích 8.1.1, 8.1.2 a 8.1.3, kde vlevo znázorňujeme původní strukturu a vpravo strukturu se zaplněním.

Definice 8.1.1 *Matici $A^{(n-1)}$ a matici $F = L + L^T$, která má stejnou strukturu řídkosti, budeme nazývat* **maticemi zaplnění.** *Jejich neorientovaný graf nazveme* **grafem zaplnění** *matice A.*

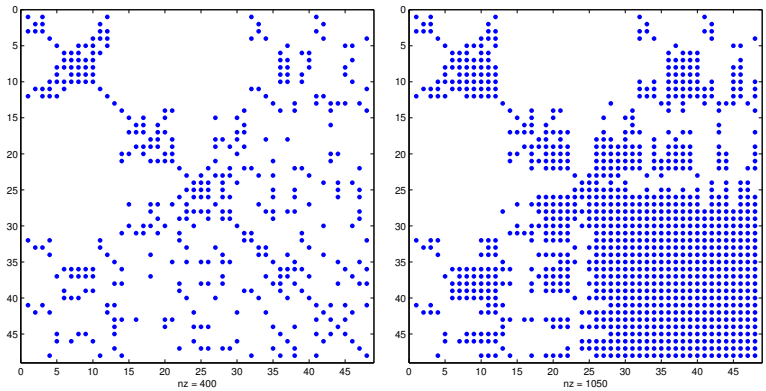
Další ukázka zaplnění vzniklé při rozkladu řídké matice prostřednictvím struktur řídkosti jejích faktorů je na Obrázku 8.1.4, kde jsou původní nenulové prvky znázorněny hvězdičkami a nové prvky zaplnění (fill-in) jsou označeny symbolem f.

8.1.1 Vznik zaplnění: lokální pohled

Vznik zaplnění v Choleského rozkladu si nejprve přiblížíme obrázkem. Příklad pozice (i, j), $i \leq n, j \leq n$ pro matici $A \in \mathbf{R}^{n \times n}$ na které vznikne prvek zaplnění při přechodu



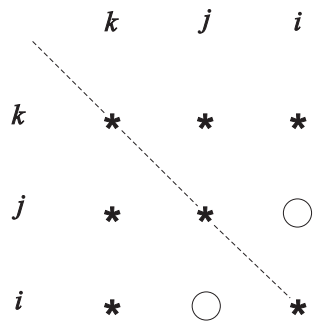
Obrázek 8.1.2: *Příklad vzniku zaplnění v rozkladu matice se složitější strukturou řídkosti řídkosti. Vlevo je znázorněna struktura řídkosti původní matice. Vpravo je znázorněna struktura řídkosti matice* $L + L^T$.



Obrázek 8.1.3: *Příklad zaplnění při rozkladu matice se složitější strukturou řídkosti z Obrázku 8.1.2, Matice je ale nejprve přeuspořádána tak, aby v počátečních krocích rozkladu bylo zaplnění malé. Vlevo je znázorněna struktura řídkosti původní matice. Vpravo je znázorněna struktura řídkosti matice* $L + L^T$.

$$
\begin{array}{c}
\\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} 1&2&3&4&5&6&7&8&9&10&11&12 \end{array}\\
\begin{pmatrix}
* & & & & & & & * & & * & * & * \\
 & * & & & & & & & & * & * & * \\
 & & * & & & & * & & & * & * & * \\
 & & & * & & & * & & & & & * \\
 & & & & * & * & & & * & & & * \\
 & & & & * & * & & & & * & & * \\
 & & * & * & & & * & & * & & & * \\
* & & & & & & & * & & * & * & * \\
 & & & & * & & * & & * & & & * \\
* & * & * & & & * & & * & & * & & * \\
* & * & * & & & & & * & & & * & * \\
* & * & * & * & * & * & * & * & * & * & * & *
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c}
\\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} 1&2&3&4&5&6&7&8&9&10&11&12 \end{array}\\
\begin{pmatrix}
* & & & & & & & * & & * & * & * \\
 & * & & & & & & & & * & * & * \\
 & & * & & & & * & & & * & * & * \\
 & & & * & & & * & & & & & * \\
 & & & & * & * & & & * & & & * \\
 & & & & * & * & & & f & * & & * \\
 & & * & * & & & * & & * & f & f & * \\
* & & & & & & & * & & * & * & * \\
 & & & & * & f & * & & * & f & f & * \\
* & * & * & & & * & f & * & f & * & f & * \\
* & * & * & & & & f & * & f & f & * & * \\
* & * & * & * & * & * & * & * & * & * & * & *
\end{pmatrix}
\end{array}
$$

Obrázek 8.1.4: *Příklad struktur řídkosti matice A (vlevo) a matice zaplnění F (vpravo; prvky zaplnění jsou označeny f).*



Obrázek 8.1.5: *Příklad pozice $\{i, j\}$ v matici, na které vznikne nový prvek zaplnění.*

mezi výše definovanými maticemi $A^{(k-1)}$ a $A^{(k)}$ pro nějaké k, $1 \leq n-1$, je znázorněn na Obrázku 8.1.5 kroužkem. Symetrická pozice (j, i), která patří do L^T je označena také kroužkem.

Následující lemma formálně popisuje, kdy vznikne prvek zaplnění na konkrétní pozici (i, j) matice $A^{(k)}$. Pro $i > j$ patří tato pozice do její dolní trojúhelníkového části, čili do faktoru L.

Lemma 8.1.1 (Lemma o zaplnění) *Nechť $i, j, k \in \{1, \ldots, n\}, k < \min\{i, j\} \leq n$. Pak za předpokladu o nevyrušení platí následující tvrzení.*

$$a_{ij}^{(k)} \neq 0 \Longleftrightarrow a_{ij}^{(k-1)} \neq 0 \vee (a_{ik}^{(k-1)} \neq 0 \wedge a_{kj}^{(k-1)} \neq 0)$$

Důkaz: Pro diagonální prvky je tvrzení triviální. Platí-li $a_{ij}^{(k)} \neq 0$ a zároveň $a_{ij}^{(k-1)} = 0$, pak muselo dojít na pozici (i, j) eliminační matice ke změně hodnoty v k-tém kroku rozkladu. Tedy při vytváření $A^{(k)}$ částečným rozkladem $A^{(k-1)}$. Pak ovšem musí platit $a_{ik}^{(k-1)} \neq 0$ a zároveň $a_{kj}^{(k-1)} \neq 0$, neboť prvky v rozkladu se upravují podle vztahu

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)} \equiv a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k-1)} / a_{kk}^{(k-1)}.$$

Opačně, pravá strana dokazované ekvivalence nutně implikuje nenulovost $a_{ij}^{(k)}$. Lemma je tak dokázáno. ■

Lemma o zaplnění se dá jednoduše vyjádřit také pomocí grafů souvisejících s rozkladem. Hrany odpovídající zaplnění (**hrany zaplnění**) jsou postupně znázorněny v eliminačních grafech $G^{(k)}$, $k = 0, \ldots, n-2$ a celou situaci lze vidět v rozšířeném eliminačním grafu $G_E^{(n-1)} \equiv G(F)$. Protože předpokládáme nevyrušení v operacích rozkladu, žádná z těchto hran nemizí. Přepisem Lemmatu o zaplnění v grafových pojmech pro graf zaplnění je pak následující tvrzení a vidíme, jak předpoklad o nevyrušení usnadňuje formulaci souvisejících tvrzení tím, že je můžeme formulovat rovnou pro graf zaplnění.

Lemma 8.1.2 (Grafová forma lemmatu o zaplnění) *Nechť* $i, j \in \{1, \ldots, n\}, k < \min\{i, j\}$. *Za předpokladu o nevyrušení pak platí*

$$\{i, j\} \in E(F) \Longleftrightarrow \{i, j\} \in E(A) \vee (\{i, k\} \in E(F) \wedge \{k, j\} \in E(F)),$$

kde $G(F) = (V, E(F))$ *je grafový model matice zaplnění* F.

8.1.2 Zaplnění ve struktuře Schurova doplňku.

Uvažujme k-tý krok algoritmu podmaticového rozkladu matice A pro $1 \leq k \leq n-1$. Schurův doplněk v matici $A^{(k)}$ získáme v rozkladu úpravou vnějším součinem vektorů. V následujícím schématu zapisujeme i příslušné indexy řádků a sloupců a ukazujeme jen příslušný Schurův doplněk.

$$A^{(k)} = A^{(k-1)} - \begin{array}{c} \\ 1 \\ \vdots \\ k \\ k+1 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & \ldots & k & k+1 & \ldots & n \end{array} \\ \begin{pmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & \frac{a_{k+1,k}^{k-1} a_{k,k+1}^{k-1}}{a_{k,k}^{k-1}} & \ldots & \frac{a_{k+1,k}^{k-1} a_{k,n}^{k-1}}{a_{k,k}^{k-1}} \\ & & & \vdots & \ddots & \vdots \\ & & & \frac{a_{n,k}^{k-1} a_{k,k+1}^{k-1}}{a_{k,k}^{k-1}} & \ldots & \frac{a_{n,n}^{k-1} a_{n,n}^{k-1}}{a_{k,k}^{k-1}} \end{pmatrix} \end{array} \tag{8.3}$$

Z podmaticové varianty rozkladu snadno nahlédneme, že za předpokladu o nevyrušení jsou v $A^{(k)}$ nenulové prvky určitě na pozicích

$$\{\{i, j\} | \{i, k\} \in E, \{k, j\} \in E,\ i > k,\ j > k\}. \tag{8.4}$$

Na některých z těchto pozic nenulové prvky už původně mohly být. Na jiných pozicích se objeví nově jako prvky zaplnění. Formálně definujeme množinu těchto **nových** pozic nenulových prvků v matici (hranách $E(F)$ grafu $G(F)$) výrazem

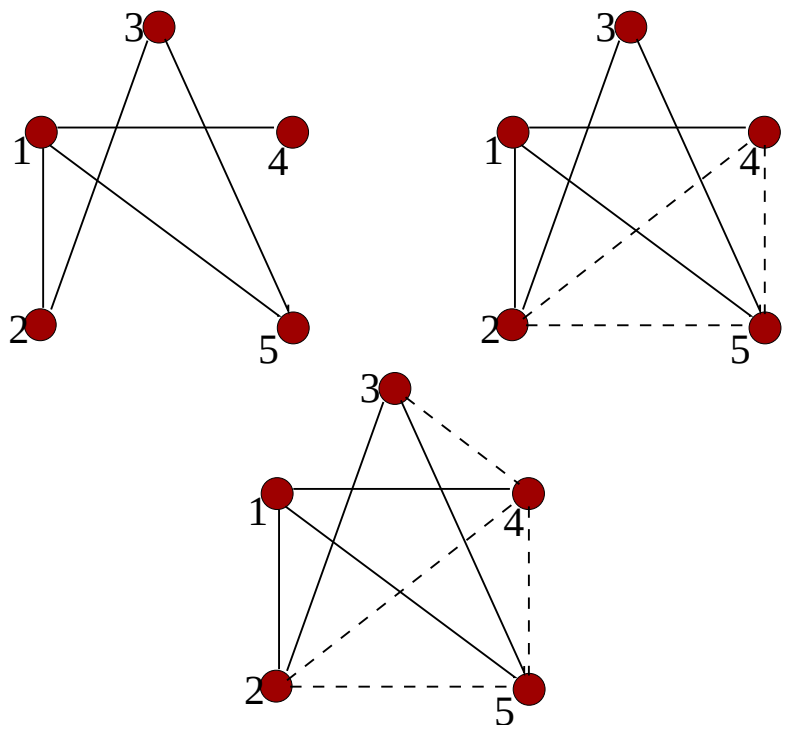
$$D^{(k)}(G) = \{\{i, j\} | \{i, k\} \in E, \{k, j\} \in E, \{i, j\} \notin E, i \neq j,\ i > k,\ j > k\} \tag{8.5}$$

jako **deficit** k-tého kroku rozkladu. Obrázek 8.1.6 ukazuje postupný vznik zaplnění v matici. Obrázek 8.1.7 znázorňuje vývoj příslušných rozšířených eliminačních grafů. Strukturální změnu mezi grafy $G^{(k-1)}$ a $G^{(k)}$ formalizuje Lemma 8.1.3.

$$
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5
\end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array}\\
\begin{pmatrix}
* & * & & * & * \\
* & * & * & & \\
 & * & * & & * \\
* & & & * & \\
* & & * & & *
\end{pmatrix}
\end{array}
\quad
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5
\end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array}\\
\begin{pmatrix}
* & * & & * & * \\
* & * & * & f & f \\
 & * & * & & * \\
* & f & & * & f \\
* & f & * & f & *
\end{pmatrix}
\end{array}
$$

$$
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5
\end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array}\\
\begin{pmatrix}
* & * & & * & * \\
* & * & * & f & f \\
 & * & * & f & * \\
* & f & f & * & f \\
* & f & * & f & *
\end{pmatrix}
\end{array}
$$

Obrázek 8.1.6: *Původní matice (vlevo nahoře) a vývoj zaplnění během prvních dvou kroků rozkladu. Nové prvky zaplnění jsou označeny symbolem f. Všimněme si, že v dalších krocích rozkladu už žádné prvky zaplnění nepřibudou.*



Obrázek 8.1.7: *Vývoj rozšířených eliminačních grafů odpovídající Obrázku 8.1.6 počínaje grafem dané matice.*

Lemma 8.1.3 *Eliminační graf* $G^{(k)} = (V^{(k)}, E^{(k)})$ *pro* $k \in \{0, \ldots, n-1\}$ *se dá s pomocí pojmu deficitu vztahem* $G^{(k)} = (V^{(k)}, E^{(k-1)}(V^{(k)}) \cup D^{(k)})$.

Důkaz: Tvrzení je zřejmé ze zápisu algoritmů rozkladu. Přiřazení

$$a_{ij} = a_{ij} - l_{ik} a_{kj}^{(k-1)}$$

v obecném kroku rozkladu doplní nenulové prvky v $A^{(k)}$ právě na ony původně nulové pozice, pro které platí $(a^{(k-1)})_{ik} \neq 0$ a $(a^{(k-1)})_{kj} \neq 0$ zároveň. ■

Přechod mezi strukturou řídkosti nenulových prvků dvou následujících eliminačních matic se dá popsat pomocí odpovídajících grafů také tak, že pro $1 \leq k \leq n-1$ v $G^{(k-1)}$ vrcholy sousedící a vrcholem k, které jsou větší než k, vytvoří v eliminačním grafu $G^{(k)}$ **kliku**, neboli úplný podgraf na těchto sousedních vrcholech. Některé hrany této kliky existovaly v $G^{(k-1)}$, ostatní hrany odpovídají novým hranám zaplnění, tedy příslušnému deficitu.

8.1.3 Zaplnění a cesty v grafu rozkládané matice

Následující věta představuje jinou charakterizaci zaplnění po provedení k-tého kroku rozkladu pro $k = 1, \ldots, n-1$. Konkrétně popisuje zaplnění nikoli lokálně, ale existenci prvku zaplnění charakterizuje **globálně** pomocí struktury řídkosti matice A. Tvrzení formulujeme pomocí grafového modelu matice $A \in \mathbf{R}^{n \times n}$. Nenulovost prvků v postupně tvořených maticích odpovídá existenci určitých neorientovaných cest v grafovém modelu $G(A)$ rozkládané matice A.

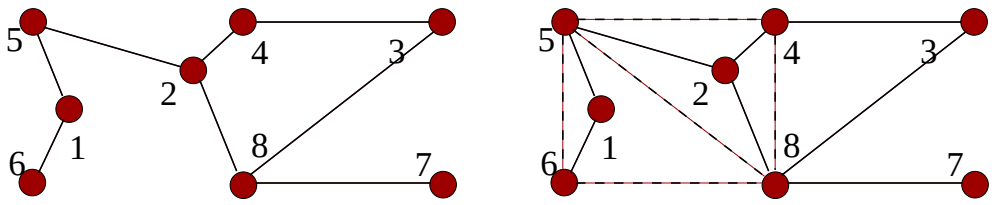
Věta 8.1.1 (Věta o zaplnění) *Nechť* $i, j, k \in \{1, \ldots, n\}, k < \min\{i, j\} \leq n$ *a platí předpoklad o nevyrušení. Pak* $a_{ij}^{(k)} \neq 0$ *právě tehdy, existuje-li v neorientovaném grafu* G *matice* A *taková cesta*

$$i \overset{G}{\Leftrightarrow} j,$$

že všechny její vrcholy různé od i *a* j *jsou menší než* $\min\{i, j\}$.

Demonstrace: Ukažme proces vzniku nenulových prvků nejprve na příkladě. Uvažujme matici A z (8.6). Vlevo je původní matice s nenulovými prvky označenými hvězdičkami, vpravo včetně prvků zaplnění označených f.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\begin{pmatrix}
* & & & & * & * & & \\
 & * & & * & * & & & \\
 & & * & * & & & & * \\
 & * & * & * & & & & \\
* & * & & & * & & & \\
* & & & & & * & & \\
 & & & & & & * & * \\
 & * & * & & & & * & *
\end{pmatrix} \end{array}
\qquad
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\begin{pmatrix}
* & & & & * & * & & \\
 & * & & * & * & & & \\
 & & * & * & & & & * \\
 & * & * & * & f & & & f \\
* & * & & f & * & f & & f \\
* & & & & f & * & & f \\
 & & & & & & * & * \\
 & * & * & f & f & f & * & *
\end{pmatrix} \end{array}
\tag{8.6}$$



Obrázek 8.1.8: *Neorientované grafy $G(A)$ a $G(L + L^T)$ s prvky zaplnění vyznačenými čárkovanými spojnicemi v druhém z nich.*

Odpovídající neorientované grafy jsou na Obrázku 8.1.8, přičemž $G(A)$ je na levé straně a $G(L+L^T)$, který obsahuje také hrany odpovídající prvkům zaplnění, je na pravé straně.

Existence prvku zaplnění na pozici, která odpovídá hraně mezi vrcholy 6 a 8, v grafu $G(L + L^T)$ odpovídá podle Věty 8.1.1 cestě

$$8 \leftrightarrow 2 \leftrightarrow 5 \leftrightarrow 1 \leftrightarrow 6.$$

Obecně taková cesta nemusí být jednoznačná. Cesty z tvrzení Věty 8.1.1 budeme nazývat **cesty zaplnění**. Popišme nyní proces vzniku zaplnění v cestách zaplnění. Hrany v rozšířených eliminačních grafech vznikají postupně z důvodu eliminace řádků a sloupců matice A v daném pořadí. První krok, eliminace prvního řádku a sloupce, implikuje vznik hrany mezi jejími sousedy 5 a 6. Druhý krok má za následek vznik hrany mezi vrcholy 5 a 8. V důsledku prvních dvou kroků jsou tedy od grafu $G_E^{(2)}$ vrcholy 6 a 8 sousedy vrcholu 5. Důsledkem eliminace pátého řádku a sloupce tedy bude hrana mezi těmito vrcholy, která bude podle předpokladu o zaplnění i hranou v $G(F) \equiv G(L + L^T)$.

■

Důkaz: Označme diskutovanou cestu v $G(A)$ z tvrzení věty následujícím způsobem

$$(i, p_1, \ldots, p_t, j),$$

kde

$$p_l \leq k,\ l \in \{1, \ldots, t\}. \tag{8.7}$$

Množina mezilehlých vrcholů cesty $\{p_1, \ldots, p_t\}$ přitom může být i prázdná. Předpokládejme $i, j, k \in \{1, \ldots, n\}, k < \min\{i, j\} \leq n$ a nechť $a_{ij}^{(k)} \neq 0$. Nechť dále platí $i > j$. K důkazu použijeme matematickou indukci podle sloupcového indexu j tohoto prvku. Pro $j = 1$ je tvrzení zřejmé, neboť struktura řídkosti sloupce $A_{*1}^{(k)}$ je shodná se strukturou řídkosti sloupce $A_{*1}^{(0)} \equiv A_{*1}$ původní matice a existuje v ní triviální cesta, tedy hrana, $(i, 1)$ délky 1. Předpokládejme, že tvrzení platí pro všechny sloupcové indexy nejvýše rovné nějakému j. Uvažujme nenulový prvek $a_{i,j+1}^{(k)}$. Podle Lemmatu 8.1.1, platí buď $a_{i,j+1}^{(0)} \equiv a_{i,j+1} \neq 0$ nebo existuje takový nenulový index r, $r < \min\{i, j+1\}, r \leq k$, že platí $a_{ir}^{(r-1)} \neq 0$ a zároveň $a_{r,j+1}^{(r-1)} \neq 0$. V prvním případě je cesta triviálně určena, v druhém případě ji získáme spojením dvou cest, které podle indukčního předpokladu existují. V $G(A)$ tedy existuje hledaná cesta taková, že má počáteční bod v i a koncový bod v $j + 1$ a indukční krok je dokázán.

Opačně, nechť existuje cesta $(i, p_1, \ldots, p_t, j)$ v $G(A)$ taková, že $p_l \leq k$, $l \in \{1, \ldots, t\}$.. Důkaz provedeme matematickou indukcí podle její délky. Je-li její délka 1, to jest jestliže se tato cesta redukuje na hranu (i, j), pak stačí položit $k = 0$ a tvrzení je triviálně platné. Předpokládejme nyní, že tvrzení platí pro všechny cesty délky nejvýše t pro $t \geq 1$. Označme $p_s = \max\{p_l \mid l \in \{1, \ldots, t\}\}$ a definujme si formálně $p_0 = i$ a $p_{t+1} = j$. Pak ovšem dle indukčního předpokladu aplikovaného na cesty $i, \ldots, p_s$ a $p_s, \ldots, j$ platí $a_{ip_s}^{(p_s-1)} \neq 0$ a $a_{p_s j}^{(p_s-1)} \neq 0$. Podle Lemmatu 8.1.3 dostáváme $a_{ij}^{(p_s)} \neq 0$ a tedy i $a_{ij}^{(k)} \neq 0$ pro všechna přípustná $k \geq p_s$. Uvedená ekvivalence je tedy dokázána. ■

Jednodušší formou Věty o zaplnění je následující tvrzení, které ukazuje situaci v grafu $G(L + L^T)$ a nebere do úvahy, ve kterém eliminačním grafu se hrany zaplnění objeví poprvé.

Důsledek 8.1.1 (Grafová forma Věty o zaplnění) *Nechť $i, j \in \{1, \ldots, n\}$ a platí předpoklad o nevyrušení. Pak $l_{ij} \neq 0$ ($\{i, j\} \in E(F)$) právě tehdy, existuje-li v $G(A)$ cesta zaplnění*

$$i \overset{G}{\Leftrightarrow} j.$$

Ačkoli popis vznikajícího zaplnění využívá ve formulaci nenulovost prvků dané matice A (vyjádřenou grafem), tedy využívá známá data, místo nenulovosti prvků v částečně eliminovaných maticích, stále je **implicitní**, protože diskutuje strukturu nenulovosti na základě existence cest v dané matici A, které nejsou jednoduchým způsobem k dispozici. Celý popis je proto potřeba dále zjednodušit. V následující sekci ukážeme vznik zaplnění nikoli prostřednictvím vnějších součinů podmaticové varianty rozkladu, ale z pohledu, jak struktura řídkosti jednoho spočítaného sloupce faktoru L ovlivní strukturu Schurova doplňku.

8.1.4 Zaplnění a replikace struktur sloupců faktoru L

V této kapitole popíšeme způsob, jakým jeho struktura řídkosti ovlivní strukturu řídkosti některých dalších sloupců v Schurově doplňku. Označení pomocí zobrazení *parent* zavedené v Definici 8.1.2 je shodné s označením zavedeným pro orientované acyklické grafy. Jak dále uvidíme, tato shoda ve značení není náhodná.

Definice 8.1.2 *Označme výrazem výrazem parent(j) pro $j \in \{1, \ldots, n\}$ řádkový index prvního nenulového poddiagonálního prvku ve sloupci j matice L. Neexistuje-li takový nenulový prvek, položme parent$(j) = 0$. Dále označme parent$^2(j) \equiv$ parent(parent(j)), parent$^3(j) \equiv$ parent(parent(parent(j))) atd.*

Uvažujme spočítaný j-tý sloupec faktoru $L_{j:n,j}$, který se v našem popisu rozkladu zapíše také jako $A_{j:n,j}^{(j-1)}$ pro $j = 1, \ldots, n-1$. Tento sloupec je při konstrukci $A_{j:n,j}^{(j)}$ použit k aktualizaci prvků ve sloupci *parent*(j). To je hned vidět ze sloupcové varianty rozkladu v Algoritmu 7.2.1, ale je to zřejmé i z popisu zaplnění v Schurově doplňku diskutovaném výše. Zároveň ale prvky sloupce $L_{j:n,j}$ neovlivní **hodnoty ani nenulovost prvků (strukturu)** v žádném sloupci s indexem větším než j a menším než *parent*(j) jak vyplývá nejen z Algoritmu 7.2.1, ale i z toho, že v deficitu j-tého kroku rozkladu

nemá žádná hrana koncový vrchol s indexem menším než $parent(j)$. Aktualizace struktury řídkosti sloupce $parent(j)$ v příslušné eliminační matici z pohledu grafového modelu spočívá v tom, že se **sjednotí struktura řídkosti sloupce** $parent(j)$ v příslušné eliminované matici a struktura řídkosti části sloupce L_{*j} s řádkovými indexy mezi $parent(j)$ a n včetně. Tato inkluze je obsahem následujícího Lemmatu 8.1.2.

Pozorování 8.1.2 *Za předpokladu o nevyrušení platí*

$$\mathcal{S}(L_{j+1:n,j}) \subseteq \mathcal{S}(L_{parent(j):n,parent(j)}).$$

Celý proces ovlivnění dalších sloupců i jejich struktur sloupcem $L_{j:n,j}$ faktoru je postupně znázorněn na Obrázku 8.1.9. Situace je nejprve zobrazena vlevo nahoře. Následuje znázornění, jak je ovlivněna struktura řídkosti sloupce $parent(j)$ a poté ovlivnění sloupce $parent(parent(j))$. **Ten ale mohl získat také příspěvky do struktury řídkosti od jiných sloupců** k, pro které $j = parent(k)$.

Obrázek 8.1.9 ukazuje nejenom postupné slévání struktur a příslušné ovlivňování numerických hodnot opakovanou aplikací principu z Pozorování 8.1.2. Zároveň ale ukazuje, že pomocí sekvence $parent(j), parent(parent(j) \equiv parent^2(j), \ldots$ zavedené v Definici 8.1.2 je možné se dostat k pozici každého nenulového prvku $l_{ij}, i > j$ výše uvedeným způsobem replikace struktur. Následující věta tento fakt formalizuje. Detailnější grafový popis uvedeme později.

Věta 8.1.2 *(Věta o replikaci sloupcové struktury) Nechť $l_{ij} \neq 0$ pro $j < i \leq n$. Pak existuje $p > 0$ takové, že $parent^p(j) = i$. Zároveň platí $l_{is} \neq 0$ pro $s = j$, $parent(j)$, $parent^2(j) \ldots, parent^p(j)$.*

Důkaz: Dříve než uvedeme jednoduchý důkaz, demonstrujme si situaci na Obrázku 8.1.10. Zde vidíme, jak replikace sloupců zároveň implikuje nenulovost prvků v i-tém řádku Choleského faktoru.

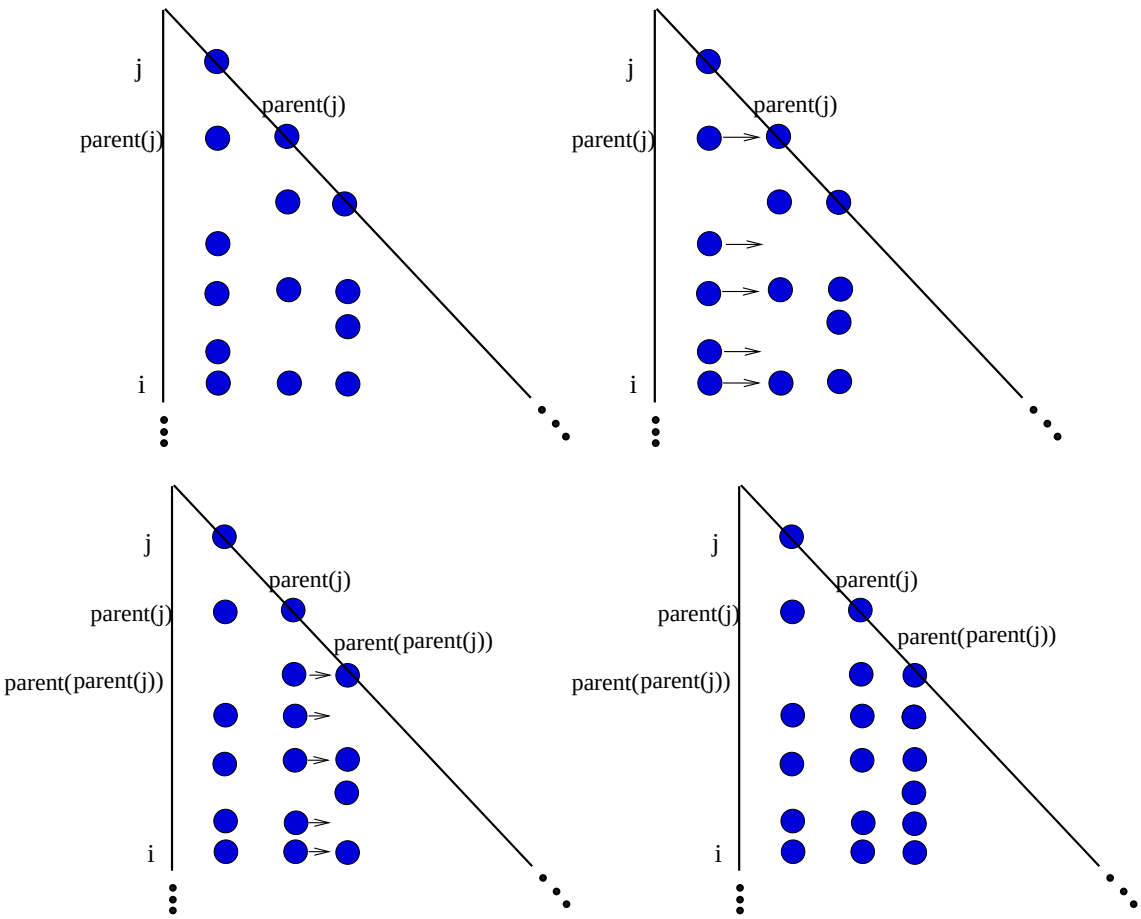
Formálně, platí-li $i = parent(j)$, tvrzení věty je dokázáno. V opačném případě existuje takový index s, $j < s < i$, že platí $s = parent(j)$. Replikace sloupcové struktury implikuje $l_{is} \neq 0$. Aplikací indukčního argumentu na tento prvek dostaneme tvrzení po konečném počtu kroků. ■

Následující teoretické výsledky ukazují vznik a rozšiřování zaplnění během rozkladu podrobněji. Lze je chápat jako rozšíření tvrzení **Věty o zaplnění** s využitím rekurentní aplikace zobrazení *parent* v jejich formulaci. Zároveň tyto výsledky ukazují, jak **některé** cesty z této Věty najít. Nejprve uvedeme Lemma 8.1.4 o propojení vrcholů cestou v jistém podgrafu grafu $G(A)$ matice A.

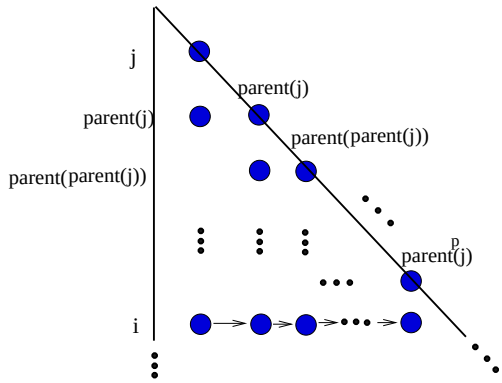
Lemma 8.1.4 *Nechť k, j, $k < j$ jsou vrcholy grafu $G(A)$. Tyto vrcholy jsou propojeny neorientovanou cestou v podgrafu $G(A)$ indukovaném vrcholy $1, \ldots, j$ zapsanou jako*

$$j \underset{\{1,\ldots,j\}}{\overset{G(A)}{\Longleftrightarrow}} k$$

právě tehdy, existuje-li přirozené číslo p, $p > 0$ takové, že $parent^p(k) = j$.



Obrázek 8.1.9: *Znázornění replikace sloupcových struktur v Choleského faktorizaci řídké matice.*



Obrázek 8.1.10: *Znázornění vzniku zaplnění na řádku i v sekvenci replikací sloupců faktoru v Choleského faktorizaci řídké matice.*

Důkaz: Předpokládejme nejprve existenci takové cesty v $G(A)$. První implikaci lemmatu lze dokázat matematickou indukcí podle délky této cesty. Pro cesty délky 1, to jest platí-li $a_{jk} \neq 0$ a tedy i $l_{jk} \neq 0$, tvrzení plyne z Věty 8.1.2 o replikaci sloupcové struktury. Uvažujme cestu délky t a předpokládejme, že výsledek platí pro všechny délky cesty menší než t. Nechť m je maximální vrchol na této cestě. Platí-li $m \leq k$, pak tvrzení platí i pro cestu délky t. V opačném případě, protože musí být $m > k$ a $m < j$ existují indexy $q_1, q_2 > 0$ takové že, $parent^{q_1}(m) = j$ a $parent^{q_2}(k) = m$. Propojením těchto cest dostaneme tvrzení první implikace lematu.

Opačně, nechť existuje přirozené p, $p > 0$ takové, že $parent^p(k) = j$ a diskutujme cestu mezi k a j, která existuje v grafu zaplnění. Podle Věty o zaplnění můžeme každou její hranu nahradit cestou v $G(A)$ s mezilehlými vrcholy menšími než oba vrcholy krajní. Propojením těchto cest získáme výsledek a lemma je tak dokázáno. ■

Jak jsme již zmínili, cesta z Věty o zaplnění není obecně jednoznačná. Následující Věta 8.1.3 říká, jak jedna taková cesta vypadá.

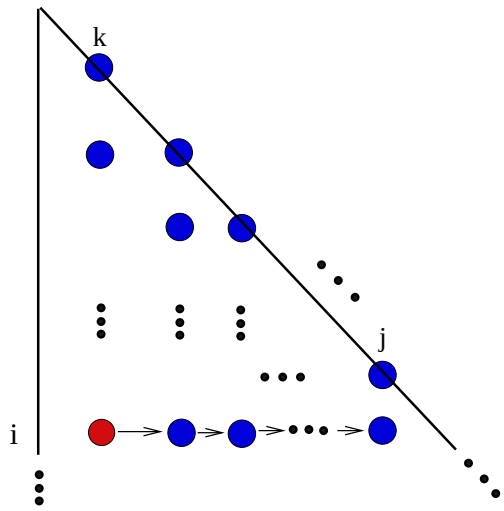
Věta 8.1.3 *(***Věta o zaplnění s použitím replikace struktur sloupců***) Nechť $a_{ij} = 0$ pro $j < i \leq n$. Pak $l_{ij} \neq 0$ právě tehdy, existuje-li $k < j$, pro které platí $a_{ik} \neq 0$ a zároveň existuje $p > 0$ takové, že*

$$parent^p(k) = j \quad .$$

Důkaz: Nechť $l_{ij} \neq 0$ pro nějaký vrchol j, kde $j < i \leq n$. Podle Věty o zaplnění existuje v $G(A)$ cesta zaplnění

$$i \overset{G}{\Leftrightarrow} j.$$

Označme si vrcholy této cesty $(i, p_1, \ldots, p_t, j)$, kde mezilehlé vrcholy splňují $p_i \leq k$, $i = 1, \ldots, t$. Musí platit $t > 0$, protože předpokládáme, že $a_{ij} = 0$. Uvažujme prvek matice $a_{ip_t} \neq 0$. Z Lemmatu 8.1.4 dostáváme, že existuje takové $q > 0$ takové, že $parent^q(p_t) = j$, protože podle tohoto Lemmatu $p_t \in \{1, \ldots, p_t\}$ a existuje neorientovaná cesta $i \underset{\{1,\ldots,p_t\}}{\overset{G(A)}{\Longleftrightarrow}} p_t$



Obrázek 8.1.11: *Znázornění vztahující se k Větě 8.1.3. Za platnosti jejích předpokladů prvek l_{ij} bude nenulový právě tehdy, existuje-li nenulový prvek a_{ij} znázorněný vlevo dole a příslušná sekvence charakterizovaná zobrazením parent.*

mezi i a p_t v podgrafu grafu $G(A)$ indukovaném množinou $\{1, \ldots, p_t\}$. Položením $k = p_t$ získáváme tvrzení první implikace.

Obráceně, nechť existují index $k < j$, pro který platí $a_{ik} \neq 0$ a $p > 0$ s vlastnostmi uvedenými ve Větě. Podle Lemmatu 8.1.4 existuje cesta $k \xleftrightarrow[\{1,\ldots,j\}]{G(A)} j$, která je tedy celá v podgrafu $G(\{1, \ldots, j\})$. To dává dohromady s hranou $\{i, k\}$ cestu zaplnění mezi vrcholy i a j. Proto platí $l_{ij} \neq 0$ a Věta je tak dokázána.

■

Demonstrujme si tvrzení Věty na Obrázku 8.1.11 rozšířením zobrazení z Obrázku 8.1.10. K tomu, abychom dostali nenulový prvek l_{ij} musí za předpokladu o nevyrušení platit $a_{ij} \neq 0$ a existovat příslušná sekvence charakterizovaná zobrazením *parent*.

8.2 Eliminační strom

V předcházejícím textu jsme viděli, že v procesu replikace struktur hrají důležitou roli **první nenulové poddiagonální prvky** ve sloupcích faktoru L. Jejich struktury řídkosti významnou měrou určují, jak vypadá struktura řídkosti celého faktoru. Orientovaný graf, který nyní definujeme a který budeme nazývat eliminační strom je založen právě na těchto prvcích a zavedeném zobrazení *parent*.

Definice 8.2.1 *Eliminační strom $T(A) = (V, E(A))$ symetrické matice A dimenze n pro kterou existuje nějaké přiřazení numerických hodnot jejím prvků tak, že existuje její Choleského rozklad $A = LDL^T$ definujeme jako její podgraf určený množinou hran*

$$E(A) = \{(i, j) \mid i = \min\{k \mid k > j \wedge l_{kj} \neq 0\}.$$

V Definici 8.2 si všimněme označení $E(A)$ množiny hran, které můžeme doplnit spodním indexem, je-li třeba vyjasnit, o jaký strom se jedná. Vidíme, že eliminační strom je speciální případ **orientovaného acyklického grafu**. Na rozdíl od obecného grafového pojmu stromu **nevyžadujeme**, aby eliminační strom byl souvislý (ve smyslu jeho symetrizace). Je-li eliminační strom souvislý, je zároveň kořenový strom s kořenem n a orientací zavedenou v Definici 8.2, která odpovídá výše zavedené orientaci kořenových stromů. S použitím označení pro orientované acyklické stromy můžeme hranu (i, j) eliminačního stromu také zapsat $(parent(j), j)$. Vrchol $parent(j)$ je **otcem** vrcholu j ve smyslu definice pro acyklické grafy a vrchol j je **synem** vrcholu $parent(j)$. Analogicky budeme hovořit i o předcích a potomcích vrcholů eliminačního stromu. Ačkoliv je eliminační strom obvykle zaváděn pro symetrické matice v případě, kdy jsou pozitivně definitní, zde používáme obecnější definici i proto, že tento pojem je založen pouze na struktuře mimodiagonálních prvků nějakého existujícího LDL^T rozkladu. Budeme-li hovořit o eliminačním stromu matice bude za tím vždy předpoklad jeho existence. Vlastnost souvislosti eliminačního stromu je popsána v následujícím tvrzení, které je snadné dokázat.

Lemma 8.2.1 *Je-li symetrická matice A nerozložitelná, $T(A)$ je zároveň stromem ve smyslu teorie grafů. To znamená, že jeho symetrizace je souvislá a acyklická.*

Je-li tedy matice z Definice nerozložitelná, eliminační strom je zároveň kořenový strom s kořenem $r = n$. Je-li tato matice rozložitelná, zkonstruovaný eliminační strom bude stále acyklický, ale nebude souvislý. Ve smyslu teorie grafů bude tedy eliminační strom **lesem**.

Snadno nahlédneme, že terminologie orientovaných acyklických grafů použitá na eliminační strom je plně kompatibilní s terminologií zavedenou v souvislosti s replikacemi nenulových prvků v průběhu faktorizace. Následující pozorování dává do souvislosti dříve ukázané vlastnosti s pojmy předka a potomka zavedenými pro acyklické grafy.

Pozorování 8.2.1

$$i = anc_T(j) \Leftrightarrow j \in desc_T(i) \Leftrightarrow (\exists p > 0)(parent^p(j) = i)$$

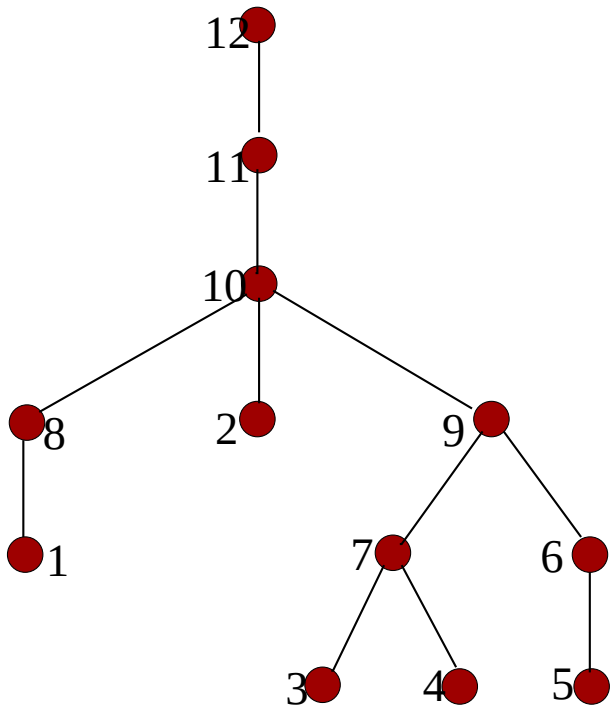
pro vrcholy i, j, $i > j$ eliminačního stromu T a přirozené číslo $p > 0$.
Další jednoduché tvrzení je v následujícím lemmatu.

Lemma 8.2.2 *Je-li vrchol i předek vrcholu $j \neq i$ v eliminačním stromu T matice A, pak platí $i > j$.*

Důkaz: Je-li vrchol i vlastní předek vrcholu j, pak v orientovaném eliminačním stromu T existuje cesta $i \Rightarrow j$ nenulové délky. Pro každou hranu (k, l) této cesty musí platit $k > l$ podle Definice 8.2. Z tranzitivity uspořádání pak vyplývá, že $i > j$, což jsme měli dokázat. ■

Protože eliminační strom je speciálním případem kořenového stromu a tedy i orientovaného acyklického grafu, pak následující definice pouze doplňuje terminologii orientovaného acyklického grafu.

Definice 8.2.2 Listy *kořenového stromu nazveme takové jeho vrcholy, které nemají vlastní potomky.*



Obrázek 8.2.12: *Příklad eliminačního stromu matice z Obrázku 8.1.4.*

Definice 8.2.3 **Podstromem** $T(j)$ *orientovaného acyklického grafu* T *ve vrcholu* j *nazveme jeho podgraf, který je indukován množinou* **všech** *vrcholů, do kterých vede cesta z vrcholu* j *v* T. $T(j)$ *je kořenovým stromem s kořenem* j. *Platí tedy, že množina vrcholů* $T(j)$ *je ekvivalentní s množinou* $desc_T(j)$. **Velikostí** $T(j)$, *kterou budeme značit* $|T(j)|$, *rozumíme počet vrcholů* $T(j)$.

V grafickém znázornění eliminačního stromu použijeme výše uvedenou konvenci, že orientace hran je často vyznačena nikoli šipkami, ale **polohou.** Hrana eliminačního stromu bude v této konvenci vždy orientována od vrcholu zobrazeného výše k vrcholu zobrazenému níže. Někdy budeme eliminační strom chápat jako neorientovaný ve smyslu jeho symetrizace. Podobně jako u grafových modelů matice budeme, pokud to bude možné, terminologii zjednodušovat a hovořit například o eliminačním stromu T nebo jen o eliminačním stromu namísto úplného značení $T(A)$. Nebude-li moci dojít k záměně, budeme také hovořit o eliminačním stromu grafu $G(A)$ nebo grafu $G(F) \equiv G(L + L^T)$ namísto o eliminačním stromu matice A. Nebude-li moci dojít k záměně, budeme také hovořit o podstromu $\bar{T}$ nějakého stromu T jako o množině vrcholů, která ho určuje s indukovanými hranami z původního stromu T.

Příklad 8.2.1 *Na Obrázku 8.2.12 je znázorněn eliminační strom matice z Obrázku 8.1.4.*

Pojem eliminačního stromu umožňuje přeformulovat některá dříve uvedená tvrzení kompaktněji, kde nahradíme, mimo jiné, rekurzívní aplikaci zobrazení *parent* vlastnostmi eliminačního stromu a jeho podstromů. Věta 8.1.2 o replikaci sloupcové struktury se dá vyjádřit například následujícím způsobem.

Lemma 8.2.3 *Platí-li pro prvek faktoru L Choleského rozkladu $l_{ij} \neq 0, i > j$ (tedy $\{i,j\} \in G(F)$ neboť předpokládáme nevyrušení) matice A, pak je vrchol i předkem vrcholu j v eliminačním stromu $T(A)$. Tvrzení můžeme také zapsat stručněji následovně*

$$l_{ij} \neq 0 \Rightarrow i \in anc_T(j). \tag{8.8}$$

Důkaz: Důkaz provedeme matematickou indukcí podle sloupcového indexu j. Pro $j = n$ tvrzení zjevně platí. Indukčním předpokladem je, že tvrzení platí pro všechny sloupce s indexy $j+1, \ldots, n$. Nechť nyní $l_{ij} \neq 0$. Je-li $i = parent(j)$ nebo $i = j$, pak je tvrzení jasné z definice zobrazení *parent*. V opačném případě existuje index k, pro který platí $k = parent(j)$, $j < k$. Musí ale platit celkem $j < k < i$, neboť jinak bychom dostali spor s definicí zobrazení *parent*. Neboť $l_{kj} \neq 0$ a $l_{ij} \neq 0$, pak podle Lemmatu 8.1.3 máme $l_{ik} \neq 0$. Z indukčního předpokladu dostaneme $i \in anc(k)$ a $k = parent(j)$, což dohromady dává $i \in anc(j) \equiv anc_T(j)$. Tvrzení pak vyplývá z principu matematické indukce. ■

8.2.1 Konstrukce eliminačního stromu

Eliminační strom je zaveden v Definici 8.2 pomocí struktury řídkosti matice faktoru L. Ta ale není obvykle k dispozici. Jeho konstrukce na základě původní matice, tedy na základě vstupních dat, je základním předpokladem pro efektivní řídkou Choleského faktorizaci. Následující Věta 8.2.1 vznikla přeformulováním Lemmatu 8.1.4 s využitím pojmu eliminačního stromu.

Věta 8.2.1 *Vrchol i je předek vrcholu j, $i > j$ v eliminačním stromu $T(A)$ právě tehdy, existuje-li v $G(\{1, \ldots, i\})$ neorientovaná cesta*

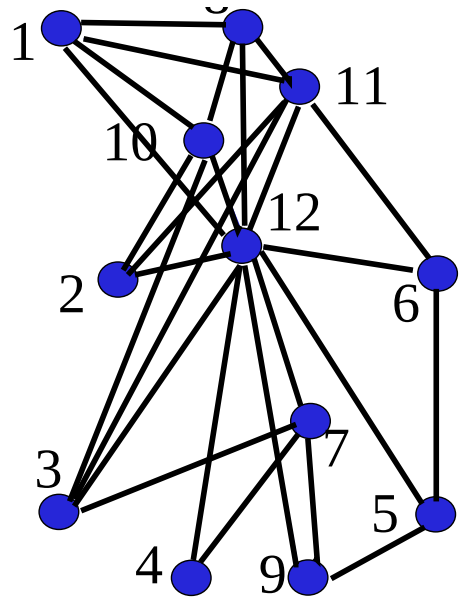
$$j \underset{\{1,\ldots,i\}}{\overset{G(A)}{\Longleftrightarrow}} i.$$

Důkaz: Nechť taková cesta existuje. Důkaz Věty provedeme matematickou indukcí podle její délky. Je-li její délka 1, tato cesta je hranou a tvrzení je zřejmé. V opačném případě nechť tvrzení platí pro všechny cesty délky menší než t, $2 \leq t \leq n-1$ a uvažujme cestu délky t. Nechť m je maximální vrchol této cesty. Je-li $m < j$, pak musí být $l_{ij} \neq 0$ podle Věty o zaplnění a tvrzení platí. V případě $m \geq j$ je podle indukčního předpokladu i předkem m a m je předkem j, což dává tvrzení. Opačně, nechť je vrchol i předek vrcholu j a uvažujme cestu $i \Rightarrow j$ v eliminačním stromu T. Každá její hrana (k,l) odpovídá nenulovému prvku l_{kl} faktoru L a tím i cestě v grafu přes vrcholy s menším ohodnocením než k i l. Spojením cest dohromady získáme hledanou cestu, jejíž vrcholy mají ohodnocení nejvýše j a Věta je tedy dokázána. ■

Následující zřejmý Důsledek 8.2.1 ihned implikuje způsob, jak můžeme eliminační strom zkonstruovat.

Důsledek 8.2.1 *Předek i vrcholu j, $i > j$ v eliminačním stromu $T(A)$ je jeho otcem (splňuje tedy $i = parent(j)$) právě tehdy, je-li vrchol j maximální prvek nějaké komponenty grafu $G(\{1, \ldots, i\}) \setminus \{i\}$ tak, že existuje neorientovaná cesta*

$$j \underset{\{1,\ldots,i\}}{\overset{G(A)}{\Longleftrightarrow}} i.$$



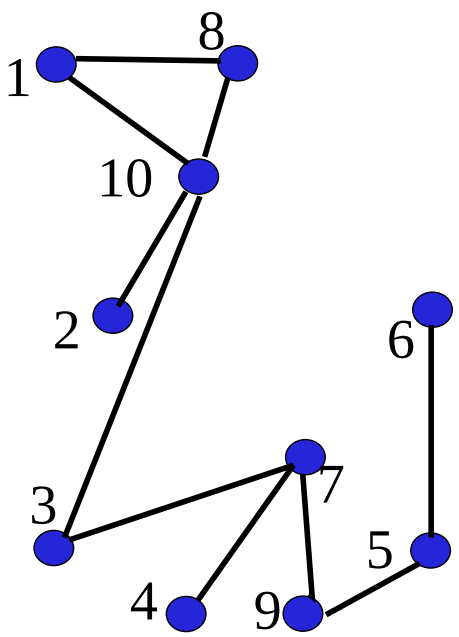
Obrázek 8.2.13: *Znázornění grafu matice z Obrázku 8.1.4 (vlevo - bez zaplnění).*

Uvažujme nyní graf matice z Obrázku 8.1.4 vlevo, tedy bez zaplnění, který znázorňujeme na Obrázku 8.2.13. A na Obrázku 8.2.14 je potom znázorněme odvozený graf $G(\{1, \ldots, 10\})$.

Vidíme, že vrchol 10 je skutečně předkem všech vrcholů $1-8$, jak je vidět na Obrázku 8.2.12, což demonstruje tvrzení Věty 8.2.1. Také vidíme, že když z toho grafu odebereme vrchol 10, pak maximální vrcholy ve vzniklých komponentách jsou právě vrcholy 8, 2 a 9, tedy synové vrcholy 10, což demonstruje Důsledek 8.2.1. Zkoumáním komponent souvislosti postupně se zvětšujícího grafu je tedy možné eliminační strom nalézt. Tato konstrukce je důležitá, že se dá zobecnit pro obecnější nesymetrický eliminační strom a využít k jeho konstrukci, jak dále uvidíme. V následujícím textu uvedeme jinou konstrukci eliminačního stromu v Algoritmu 8.2.1, která je založena na tom, že postupně procházíme vrcholy grafu a snažíme se je přidávat ke komponentám postupně tvořeného eliminačního stromu. Tato konstrukce je nejenom efektivní a její další výhoda je, že umožňuje velmi dobře využít klasické výsledky teorie kořenových stromů. Algoritmus 8.2.1 samozřejmě předpokládá platnost předpokladu o nevyrušení.

Algoritmus 8.2.1 Nalezení eliminačního stromu
Input: *Řídká SPD matice $A \in R^{n \times n}$ a její neorientovaný graf.*
Output: *Eliminační strom uložený pomocí vektoru parent.*

1. **for** $i = 1 : n$ **do**
2. $\quad parent(i) = 0$
3. $\quad$ **for** k *taková, že* $k \in adj(i) \wedge k < i$ **do**
4. $\quad\quad j = k$
5. $\quad\quad$ **while** $(parent(j) \neq 0 \wedge parent(j) \neq i)$ **do**
6. $\quad\quad\quad j = parent(j)$
7. $\quad\quad$ **end while**



Obrázek 8.2.14: *Znázornění grafu matice z Obrázku 8.1.4 (vlevo - bez zaplnění).*

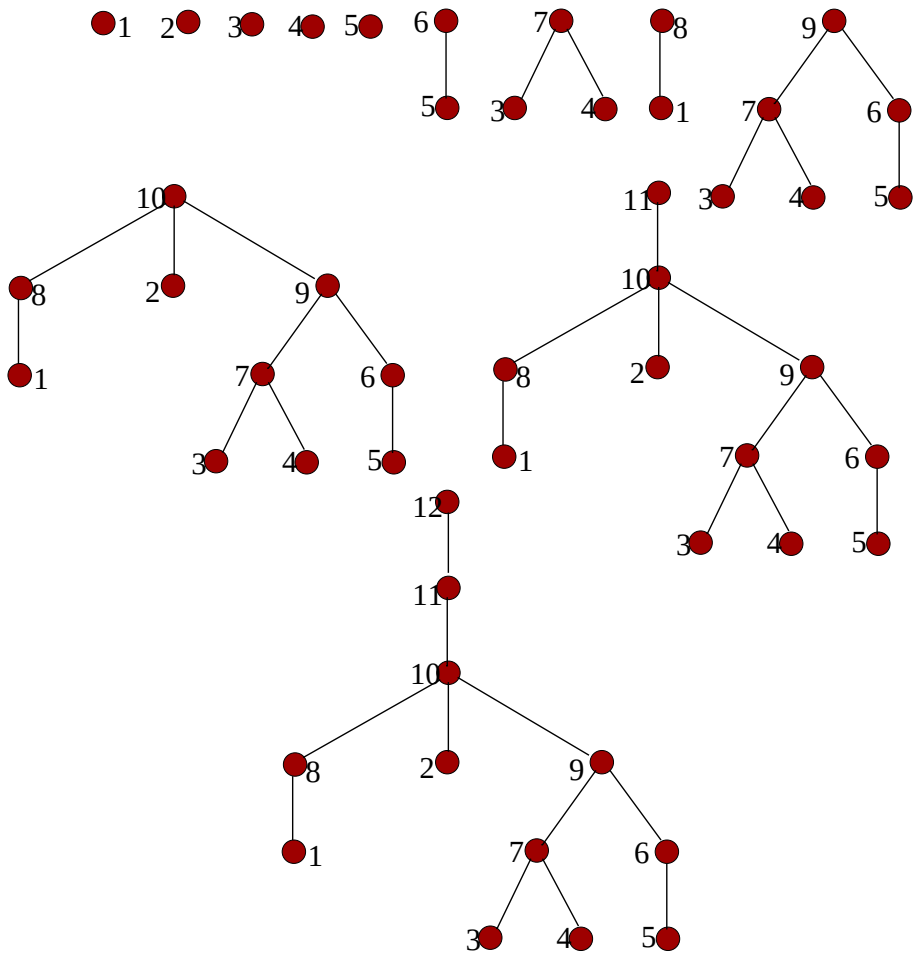
```
8.        if parent(j) = 0 then
9.            parent(j) = i
10.       end if
11.    end k
12. end i
```

Ve vnějším cyklu algoritmu se postupně procházejí řádky matice A a uvažují jejich nenulové prvky. Poznamenejme, že v kroku 3 Algoritmu 8.2.1 není potřeba, aby indexy nenulových prvků $k \in adj(i) \wedge k < i$ byly procházeny v určitém pořadí. V důkazu Lemmatu 8.2.4 ukážeme, že Algoritmus 8.2.1 opravdu konstruuje eliminační strom.

Lemma 8.2.4 *Uvažujme Algoritmus 8.2.1 a jeho i-tý krok pro $i = 1, \ldots, n$. Po ukončení i-tého kroku algoritmu pro $i = 1, \ldots, n$ je zkonstruován eliminační strom $T(A_{1:i,1:i})$.*

Důkaz:

Budeme postupovat matematickou indukcí podle i. Pro $i = 1$ je tvrzení zřejmé. Uvažujme $i > 1$ a předpokládejme, že tvrzení platí pro $j = 1, \ldots, i-1$. Je-li nenulový prvek a_{ik} matice A pro $k < i$ (při předpokladu o nevyrušení tedy i $k \in adj(i)$) prvním poddiagonálním prvkem v jejím k-tém sloupci, pak vzhledem k indukčnímu předpokladu to je ekvivalentní faktu, že doposud platí $parent(i) = 0$. Položení $parent(k) = i$ na řádku 9 Algoritmu 8.2.1 odpovídá definici eliminačního stromu. Není-li prvním poddiagonálním prvkem, pak první poddiagonální prvek ve sloupci k už musel být uvažován v řádkovém cyklu některého předcházejícího kroku. Označme jeho řádkový index $j, j < i$ a platí tedy $parent(k) = j$. Podle Věty 8.1.2 ale platí také $l_{ij} \neq 0$, protože nenulovost tohoto prvku

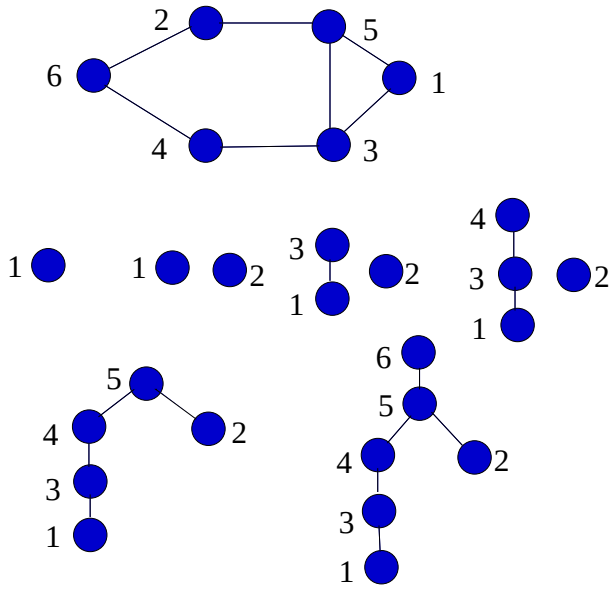


Obrázek 8.2.15: *Znázornění konstrukce eliminačního stromu matice z Obrázku 8.1.4, který je vcelku na Obrázku 8.2.12.*

v matici faktoru L vyplývá z příslušné replikace struktury řídkosti sloupce. Po konečném počtu těchto kroků, kde v algoritmu předefinováváme sloupcový index algoritmickou konstrukcí $j = parent(j)$, získáme prvek řádku i a sloupce j, pro který platí $l_{ij} \neq 0$ a zároveň je i prvním poddiagonálním prvkem sloupce j. Ten odpovídá hraně dané vztahem $i = parent(j)$, která buď byla implikována replikací nenulovosti při kterém se vyšlo z nějakého $k', k' \neq 0$, které jsme na i-tém řádku uvažovali dříve a pro které $a_{ik'} \neq 0$ nebo musíme položit $i = parent(j)$. To odpovídá rozhodování na řádku 8 algoritmu. Eliminační strom byl tak rozšířen, indukční krok je tím dokázán. Z principu matematické indukce plyne tvrzení tohoto lemmatu. ■

Příklad 8.2.2 *Obrázek 8.2.15 znázorňuje postupnou konstrukci eliminačního stromu procesem přidávání* **maximálně** *jedné hrany pro každý řádek. Po k-tém kroku konstrukce se eliminační strom skládá ze všech komponent, které obsahují vrcholy* $1, \ldots, k$.

Příklad 8.2.3 *Na Obrázku 8.2.16 je jiná ukázka grafu $G(A)$ matice A a také částečně zkonstruovaných eliminačních stromů po provedení i-tého kroku Algoritmu 8.2.1 pro $i =$*



Obrázek 8.2.16: *Ukázka grafu $G(A)$ o šesti vrcholech a šesti postupně tvořených eliminačních stromů pro její hlavní podmatice. Částečné eliminační stromy jsou seřazeny zleva doprava a shora dolů.*

$1, \ldots, 6$. *Na rozdílů od předcházejícího zobrazení zde opakovaně znázorňujeme i komponenty, které nejsou ovlivněny přidáním hrany v i-tém kroku cyklu. Tento příklad použijeme ještě níže pro demonstraci obecně rychlejšího algoritmu konstrukce eliminačního stromu popsaného jako Algoritmus 8.2.2.*

Procházení cest ve vytvářeném eliminačním stromu směrem ke kořeni znázorněným přiřazením $j = parent(j)$ na řádku 6 Algoritmu 8.2.1 může být časově náročné. To platí obzvláště v případě konstrukce eliminačního stromu, jehož nějaká komponenta má velkou **výšku**, což je pojem, který nyní zavedeme. Definujme nejprve pomocnou funkci

$$ht_T : V \ \rightarrow \{1, \ldots, n-1\} \cup \{0\}$$

předpisem

$$ht_T(i) = \begin{cases} ht_T(i) = 0 & \text{je} - \text{li } i \text{ listem v } T, \\ 1 + \max\{ht_T(j) |\ parent(j) = i\} & \text{jinak.} \end{cases} \tag{8.9}$$

Výšku $height(T)$ eliminačního stromu T pak zavedeme přiřazením

$$height(T) = \begin{cases} \max_{i=1,\ldots,n} ht_T(i) & pro\ T \neq \emptyset, \\ -1 & jinak. \end{cases} \tag{8.10}$$

Důsledek 8.2.2 *Je-li eliminační strom souvislý, pak je zřejmě*

$$height(T) = ht_T(n).$$

Příklad struktury řídkosti matice $A \in \mathbf{R}^{n\times n}$ pro $n = 7$, kde velká výška jejího eliminačního stromu brání jeho efektivní konstrukci, je následující.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} * & * & * & * & * & * & * & * \\ * & * & & & & & & \\ * & & * & & & & & \\ * & & & * & & & & \\ * & & & & * & & & \\ * & & & & & * & & \\ * & & & & & & * & \\ * & & & & & & & * \end{pmatrix} \end{array} \tag{8.11}$$

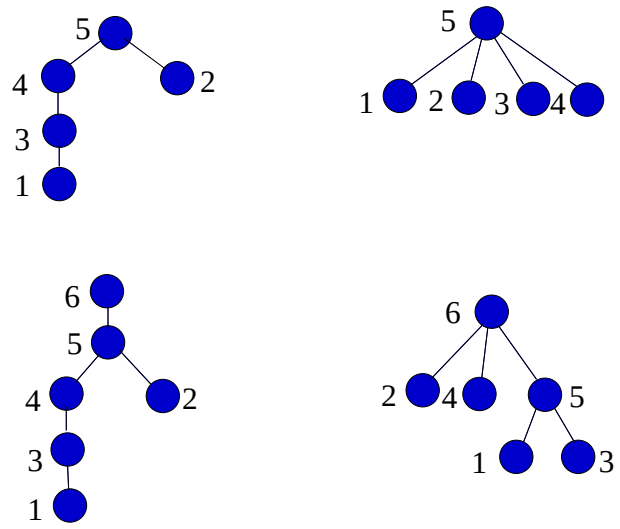
Eliminační strom je určen hranami

$$parent(i) = i + 1,\ i = 1, \ldots, n.$$

Při jeho konstrukci Algoritmem 8.2.1 pro každý řádek musíme projít celý dosud zkonstruovaný strom. To znamená, že složitost konstrukce je $O(n^2)$, čili je úměrná dimenzi kvadraticky. Zefektivnění konstrukce eliminačního stromu, které částečně eliminuje dlouhé sekvence hledání v Algoritmu 8.2.1 vlivem jeho velké výšky, je založeno na **kompresi částí cest** v eliminačním stromu. V praxi se implementuje s použitím pomocného vektoru *ancestor*, který pro vrchol j uchová nějakého obecně jiného předka než je $parent(j)$, který může mít menší vzdálenost ke kořeni komponenty dosud zkonstruovaného eliminačního stromu. Konkrétně, vektor *ancestor* uchovává **zkratky** cest, které už byly někdy použity při procházení částečně zkonstruovaným eliminačním stromem. Podgraf popsaný vektorem *ancestor* místo vektoru *parent* se nazývá **virtuální eliminační strom** a jeho komponenty mají pokud možno malou výšku. Příslušný postup nalezení eliminačního stromu, který je tak obvykle efektivnější než Algoritmus 8.2.1, zde uvádíme jako Algoritmus 8.2.2.

Algoritmus 8.2.2 Nalezení eliminačního stromu s urychlením kompresí cest
Input: *Řídká SPD matice $A \in R^{n\times n}$ a její neorientovaný graf.*
Output: *Eliminační strom uložený pomocí vektoru parent.*

1. **for** $i = 1 : n$ **do**
2. $\quad parent(i) = 0$
2. $\quad ancestor(i) = 0$
3. $\quad$ **for** *k taková, že $k \in adj(i) \wedge k < i$* **do**
4. $\quad\quad j = k$
5. $\quad\quad$ **while** *$(ancestor(j) \neq 0 \wedge ancestor(j) \neq i)$* **do**
6. $\quad\quad\quad l = ancestor(j)$
7. $\quad\quad\quad ancestor(j) = i$
8. $\quad\quad\quad j = l$
9. $\quad\quad$ **end while**
10. $\quad\quad$ **if** $ancestor(j) = 0$ **then**
11. $\quad\quad\quad ancestor(j) = i$



Obrázek 8.2.17: *Ukázka grafu matice $G(A)$ a dvou postupně tvořených eliminačních stromů (dole) a virtuálních eliminačních stromů (nahoře) pro její pátou a šestou hlavní podmatici.*

12. $parent(j) = i$
13. **end if**
14. **end** k
15. **end** i

Příklad 8.2.4 *Na Obrázku 8.2.17 jsou znázorněny postupně tvořené částečné eliminačních stromy (popsané vektorem parent) a částečné virtuálních eliminačních stromy (popsané vektorem ancestor) po provedení pátého a šestého kroku Algoritmu 8.2.2. Ostatní částečné eliminační stromy a částečné virtuální eliminační stromy jsou shodné s eliminačními stromy z předcházejícího příkladu. Původní graf matice $G(A)$ přitom je stejný jako na obrázku 8.2.16.*

Obecně existují ještě další sofistikované možnosti jak konstrukci eliminačního stromu obohatit, které mohou být efektivní v konkrétních situacích a které mohou být například popsány formalismem sjednocování množin z [153], ale zde se jimi nebudeme zabývat.

8.2.2 Lokalizace cest grafového modelu vzhledem k eliminačnímu stromu

Uveďme některá další tvrzení, která, neorientovanou cestu z Věty 8.2.1 a obecně cesty zaplnění dále lokalizují. Nejprve uvedeme souvislost nenulovosti prvků Choleského faktoru s podstromy eliminačního stromu, které tvoří jen část množiny předků vrcholu eliminačního stromu, která je uvažována ve Větě 8.2.1. Tato Věta 8.2.2 není vůbec samozřejmá a uvedeme ji včetně formálního důkazu.

Věta 8.2.2 *Pro $i > j$ platí $l_{ij} \neq 0$ za předpokladu o nevyrušení právě tehdy, existuje-li v grafu G matice A neorientovaná cesta*

$$j \overset{G(A)}{\Longleftrightarrow} i$$

taková, že všechny její mezilehlé vrcholy patří do podstromu $T(j)$, kde T je eliminační strom matice A.

Důkaz: Předpokládejme nejprve, že taková neorientovaná cesta existuje. Označme ji $P = (i, p_1, \ldots, p_t, j)$, kde

$$p_l < j,\ l \in \{1, \ldots, t\}. \tag{8.12}$$

Neboť j je vlastní předek všech mezilehlých vrcholů cesty P, které jsou ve stejné komponentě souvislosti jako j. Protože platí $l_{ip_1} \neq 0$, pak je podle Věty 8.1.2 také $l_{ij} \neq 0$.

Opačně, nechť $l_{ij} \neq 0$. Podle Věty 8.1.1 o zaplnění existuje v G cesta $P = (i, p_1, \ldots, p_t, j)$, taková, že

$$p_l < j,\ l \in \{1, \ldots, t\} \tag{8.13}$$

a dodatečně položíme

$$i = p_0, \quad j = p_{t+1}.$$

Je-li $t = 0$, tvrzení zřejmě platí. Uvažujme $t > 0$ a předpokládejme sporem, že některé mezilehlé vrcholy cesty P nepatří do $T(j)$. Nechť s je největší index nějakého mezilehlého vrcholu pro který $p_s \notin T(j)$. Tudíž $p_{s+1} \in T(j)$. Protože ale platí $\{p_s, p_{s+1}\} \in E(G)$ a p_{s+1} nemůže být předek p_s, protože pak by p_s patřil do $T(j)$. pak musí p_s být předek p_{s+1}. Oba vrcholy p_s a j jsou tedy předci vrcholu p_{s+1}. Ten druhý z nich podle předpokladu o výše uvedené maximalitě indexu s. V každém případě je tedy p_s vlastním předkem j, protože j nemůže být předek vrcholu p_s. To je ve sporu s předpokladem $p_s < j$. ■

Ukázkami cest, které vyhovují předchozímu tvrzení pro prvky Choleského faktoru matice z Obrázku 8.1.4 $l_{11,9}$ a $l_{10,9}$ jsou, po řadě, cesty

$$9 \overset{G(A)}{\Longleftrightarrow} 5 \overset{G(A)}{\Longleftrightarrow} 6 \overset{G(A)}{\Longleftrightarrow} 11$$

a

$$9 \overset{G(A)}{\Longleftrightarrow} 7 \overset{G(A)}{\Longleftrightarrow} 3 \overset{G(A)}{\Longleftrightarrow} 10.$$

O vrcholech v podstromech eliminačního stromu můžeme dokázat ještě následující tvrzení, které odpovídá tomu, že konstrukce eliminačního stromu v každém kroku, ve kterém je do něj přidána nová hrana, se propojují jeho do té chvíle nesouvislé komponenty grafu dané matice. Toto tvrzení je zřejmé z Věty 8.2.1, která říká, že vrcholu eliminačního stromu musí být propojen se všemi ostatními vrcholy podstromu tohoto vrcholu cestami v grafu dané matice, ale přesto je vyslovíme a i formálně dokážeme.

Věta 8.2.3 *Podgraf grafu $G(A)$, indukovaný v $G(A)$ vrcholy podstromu $T(j)$ eliminačního stromu $T(A)$ pro $j = 1, \ldots, n$ je v $G(A)$ souvislý.*

Důkaz: Důkaz provedeme indukcí podle počtu vrcholů v $T(j)$. Pro jednovrcholový podgraf eliminačního stromu je tvrzení zjevně pravdivé. Uvažujme $t > 1$. Jsou-li $s_1, \ldots, s_p$ jeho synové, pak tvrzení platí pro všechny podgrafy $T(s_1), \ldots, T(s_p)$ podle indukčního předpokladu. Dále, $\{s_i, j\}, i = 1, \ldots, p$ jsou hrany v grafu zaplnění a pro každou z nich existuje podle Věty 8.2.2 cesta v $G(A)$ přes vrcholy v $T(s_i)$. Příslušný podgraf je tedy souvislý. ■

Zřejmým důsledkem této věty je následující tvrzení, které ukazuje vztah k vrcholům větším než vybraný vrchol j, tedy k $hadj(j)$.

Důsledek 8.2.3 *Vrcholy podgrafu $T(j)$ eliminačního stromu $T(A)$ pro $j = 1, \ldots, n$ tvoří souvislou komponentu grafu v takovém podgrafu $G(A)$, který obsahuje všechny vrcholy $G(A)$ mimo vrcholy $adj(T(j))$.*

Množinu $adj(T(j))$ v tomto důsledku můžeme také nahradit její nadmnožinou, která je rovna **množině vlastních předků** podstromu $T(j)$, neboť žádný z nich neovlivní vlastnost souvislosti mezi vrcholy $T(j)$.

8.2.3 Eliminační strom a stromový paralelismus

Jiná formulace předchozích lokalizačních vět přímo ukazuje možné využití eliminačního stromu pro nezávislé výpočty vztažené na jeho podstromy. Navozujeme ji následující Větou 8.2.4.

Věta 8.2.4 *Nechť $T(j)$ a $T(k)$ jsou dva disjunktní podstromy eliminačního stromu T. Potom za předpokladu o nevyrušení pro libovolné vrcholy $s \in T(j)$ a $t \in T(k)$ platí $l_{st} \neq 0$.*

Důkaz: Nejprve podotkněme, že vzhledem k tomu, že L je dolní trojúhelníková matice, předpokládáme bez újmy na obecnosti, že $s > t$. Důkaz provedeme sporem. Kdyby takové vrcholy podle zadání Věty existovaly, pak by podle Lemmatu 8.2.3 musel být vrchol s předkem vrcholu t. To je ale ve sporu s disjunktností $T(j)$ a $T(k)$. ■

Důsledkem tohoto tvrzení je, že výpočty, které probíhají na disjunktních podstromech eliminačního stromu, mohou být prováděny nezávisle. Hovoříme přitom o stromovém paralelismu. Dále uvidíme, že algoritmy řídké Choleského faktorizace mohou tohoto paralelismu využívat.

8.3 Struktura řídkosti řádků a sloupců Choleského faktoru

Jádrem této sekce je podrobnější popis řádkové a sloupcové struktury řídkosti Choleského faktoru L založený především na pojmu eliminačního stromu.

8.3.1 Struktura řídkosti řádků Choleského faktoru

Pro zjednodušení diskuse o **řádkové struktuře** Choleského faktoru použijme výše zavedené označení indexů nenulových **mimodiagonálních** prvků na řádku faktoru L. Konkrétně uvažujme

$$row_L(j) \equiv \mathcal{S}(L_{j,1:j-1}) = \{k \mid k < j,\ l_{jk} \neq 0\},\ 1 \leq j \leq n. \tag{8.14}$$

Následující tvrzení je pouze jiná formulace Věty 8.1.3 s použitím zavedeného pojmu eliminačního stromu.

Věta 8.3.1 *Uvažujme Choleského faktor spočítaný za předpokladu nevyrušení. Pro indexy i, j, $i > j$ platí $l_{ij} \neq 0$ právě tehdy, je-li vrchol j předkem nějakého vrcholu k v eliminačním stromu T, pro který platí $a_{ik} \neq 0$.*

Důkaz: Nechť $l_{ij} \neq 0$. K důkazu první implikace použijeme Větu o zaplnění, podle které existuje cesta $P = (i, p_1, \ldots, p_t, j)$ v $G(A)$ taková, že platí

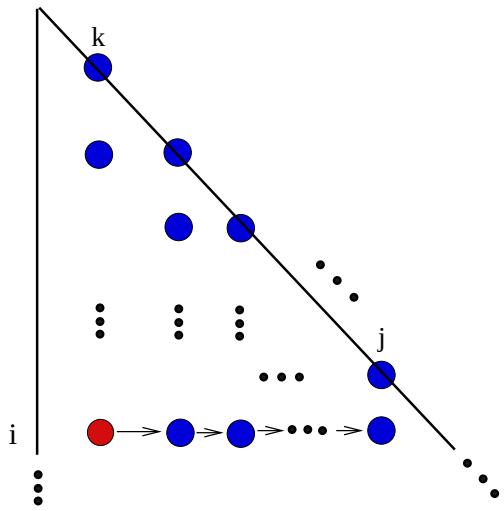
$$p_l < j,\ l \in \{1, \ldots, t\}. \tag{8.15}$$

Pro cestu $P = (i, j)$ je tvrzení platné (s prázdnou množinou mezilehlých vrcholů), neboť $a_{ij} \neq 0$ a tedy můžeme položit $k = j$. Pro $t > 0$ zvolme $k = p_1$ a máme tedy $a_{ik} \neq 0$. To, že j je předkem vrcholu k dokážeme matematickou indukcí podle počtu q vrcholů v P, které jsou mezi k a j. Jsou-li v P prvky k a j sousedi, pak je tvrzení dokazované implikace pravdivé, neboť zřejmě $j \in anc(k)$. Předpokládejme, že tvrzení platí pro všechny cesty P, které mají tento počet mezilehlých vrcholů menší než q. Nechť námi zkoumaná cesta má počet vrcholů mezi k a j roven q. Nechť m je takový z těchto mezilehlých vrcholů, který má maximální index. Je-li $m = k$, pak platí $l_{jk} \neq 0$, protože mezi j a k máme cestu z Věty o zaplnění. Vztah $j \in anc(k)$ potom vyplývá z tvrzení Věty 8.2.3. V opačném případě podle indukčního předpokladu aplikovaného na cesty (k, m) a (m, j) plyne $j \in anc(m)$ a $m \in anc(k)$ z čehož vyplývá tvrzení dokazované implikace.

Nechť pro nějaké k platí $j \in anc(k)$ a $a_{ik} \neq 0$. Pak také platí $l_{ik} \neq 0$. Podle Lemmatu 8.2.3 platí také $i \in anc(k)$. Pro $i > j$ musí být vrchol j mezilehlým vrchol na jednoznačně určené cestě od i do k v T, jinak dostaneme spor s definicí eliminačního stromu. Nyní dokážeme, že pro všechny vrcholy l této cesty od i do k v T platí $l_{il} \neq 0$. Důkaz provedeme matematickou indukcí podle indexu k. Pro $k = n$ tvrzení zjevně platí. Nechť tvrzení platí pro všechny indexy $k+1, \ldots, n$. Nechť $l_{ik} \neq 0$ a $i > k$. Je-li $i = parent(k)$, pak je tvrzení jasné z definice zobrazení $parent$. V opačném případě existuje index l, pro který je splněn vztah $l = parent(k)$, $k < l < i$. Neboť $l_{lk} \neq 0$ a zároveň je $l_{ik} \neq 0$, pak z Lemmatu 8.1.3 vyplývá $l_{il} \neq 0$. Z indukčního předpokladu pak dostáváme tvrzení dokazované implikace. Věta je tak dokázána. ■

Pro názornější interpretaci tvrzení Věty 8.3.1 si definujme pojem ořezaného podstromu eliminačního stromu T.

Definice 8.3.1 *Podstrom $T_p(i)$ eliminačního stromu T nazveme* **ořezaným podstromem** *T ve vrcholu i, jestliže má kořen i, jeho vrcholy jsou podmnožinou podstromu $T(i)$ a dále pro každý list k podstromu $T_p(i)$ platí, že všechny vrcholy na orientované cestě z i do k patří také do $T_p(i)$. $T_p(i)$ je podgraf eliminačního stromu, který je svými vrcholy indukován. Obecně je tedy $T_p(i)$ podgraf $T(i)$.*



Obrázek 8.3.18: *Znovu použitý příklad, který tentokrát demonstruje tvrzení Věty 8.3.1.*

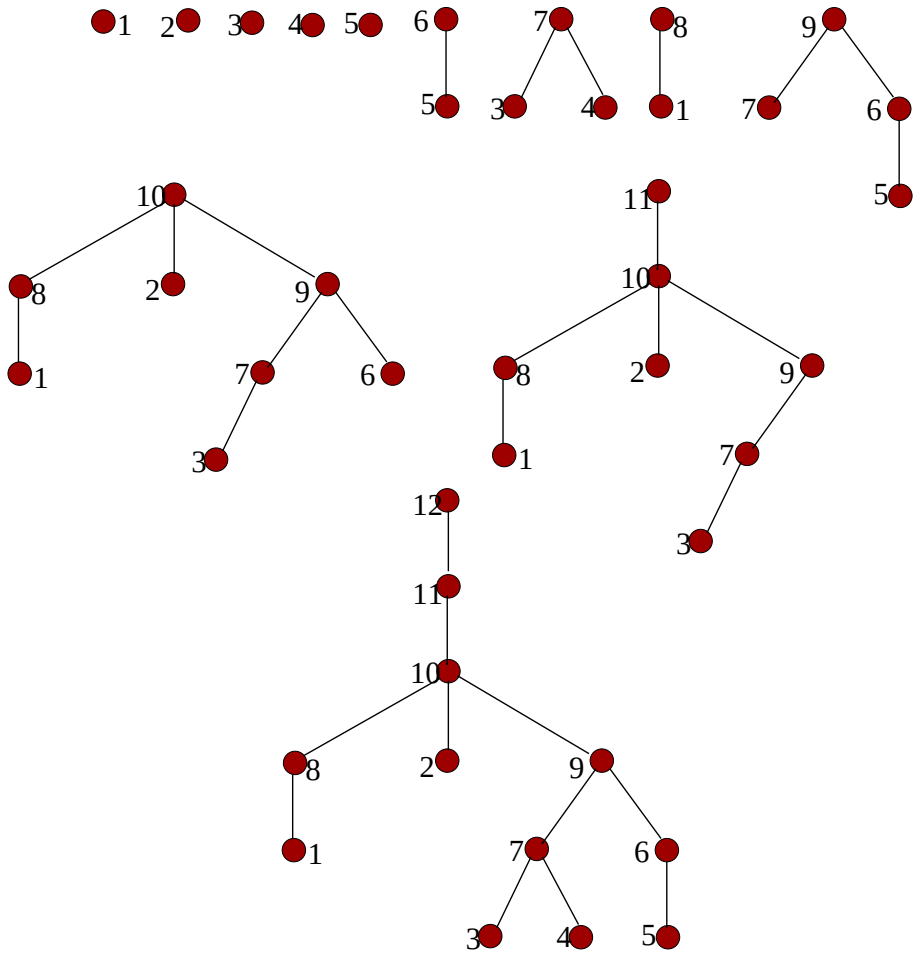
Věta 8.3.1 konkrétně říká, že struktura řídkosti řádku Choleského faktoru je dána určitým ořezaným podstromem. Formálně tuto úvahu popíšeme jako Důsledek 8.3.1 Věty 8.3.1 a zároveň zde definujeme pojem **řádkový podstrom** eliminačního stromu.

Důsledek 8.3.1 *Nechť $n \geq i > j \geq 1$. Uvažujme ořezaný podstrom $T_p(i)$ eliminačního stromu T, kde vrchol j patří do jeho množiny vrcholů právě tehdy, je-li buď $a_{ij} \neq 0$ nebo $j \in anc_{T(i)}(k)$ pro nějaké k, kde $a_{ik} \neq 0$. Tento ořezaný podstrom budeme značit $T_r(i)$ a nazývat i-tý řádkový podstrom eliminačního stromu T. Vrcholy $row_L(i) \cup \{i\}$ jsou právě vrcholy $T_r(i)$.*

Vrcholy řádkového podstromu $T_r(i)$ jsou tedy dány sjednocením všech vrcholů $T(i)$ na cestách z vrcholu i do všech vrcholů daných sloupcovými indexy i-tého řádku matice A. K tomu, abychom řádkový podstrom $T_p(i)$ eliminačního stromu T ve vrcholu i jednoznačně určili stačí ale sjednotit pouze vrcholy na cestách do jeho listů, které tvoří pouze podmnožinu vrcholů daných sloupcovými indexy i-tého řádku matice A. Přitom vrchol j je listem $T_r(i)$ právě tehdy, je-li $a_{ij} \neq 0$ a platí-li zároveň $a_{ik} = 0$ pro všechny vlastní potomky vrcholu j v eliminačním stromu T. Tvrzení Věty 8.3.1 lze také ilustrovat již dříve použitým Obrázkem 8.3.18, kde je symbolicky znázorněno, jak nenulovost prvku a_{ik} i-tého řádku matice A implikuje nenulovost prvků l_{ij} v řádku i faktoru L, s indexy $j \in anc(k), j \leq i$.

Příklad 8.3.1 *Všechny řádkové podstromy eliminačního stromu z Obrázku 8.2.12 jsou znázorněny na Obrázku 8.3.19.*

Předcházející diskuse implikuje, že graf zaplnění i eliminační strom zůstanou stejné, vynecháme-li z $G(A)$ všechny hrany, které nejsou listy nějakého řádkového podstromu. Příslušné nenulové pozice ve faktoru jsou totiž dány zaplněním v průběhu eliminace. Tomuto jednoznačně určenému grafu, odvozenému z G se někdy říká **eliminační redukce** (eliminační skelet) $G(A)$ a budeme jej značit G^-, respektive $G^-(A)$. Tento graf může



Obrázek 8.3.19: *Řádkové podstromy $T_r(i)$ eliminačního stromu z Obrázku 8.2.12 uspořádané ve třech řadách zleva doprava a shora dolů pro $i = 1, \ldots, 12$.*

$$
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10\\ 11\\ 12
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array}\\
\begin{pmatrix}
* & & & & & & & * & & * & * & * \\
 & * & & & & & & & & * & * & * \\
 & & * & & & & * & & & * & * & * \\
 & & & * & & & * & & & & & * \\
 & & & & * & * & & & * & & & * \\
 & & & & * & * & & & & * & & \\
 & & * & * & & & * & & * & & & \\
* & & & & & & & * & & & & \\
 & & & & * & & * & & * & & & \\
* & * & * & & & * & & & & * & & \\
* & * & * & & & & & & & & * & \\
* & * & * & * & * & & & & & & & *
\end{pmatrix}
\end{array}
$$

Obrázek 8.3.20: *Struktura matice, ve které jsou pouze ty nenunulové mimodiagonální prvky, které odpovídají eliminačnímu skeletu jejího neorientovaného grafového modelu.*

pomoci snížit časovou náročnost v celé řadě algoritmů, kterým stačí jako vstupní data struktury řádkových podstromů a případně pro urychlení i eliminační strom. Problém hledání listů řádkových podstromů, které jsou podstatné k nalezení eliminační redukce, budeme diskutovat později. Na následujícím Obrázku 8.3.20 znázorňujeme eliminační skelet matice z Obrázku 8.1.4 vlevo ve formě jejích nenulových mimodiagonálních prvků matice.

8.3.2 Nalezení velikostí řádků faktoru L a jejich struktury řídkosti

Řádkové podstromy lze použít k nalezení počtů nenulových prvků na řádcích faktoru L. Celkový počet $|L|$ nenulových prvků faktoru L pak jej jejich součtem, tedy

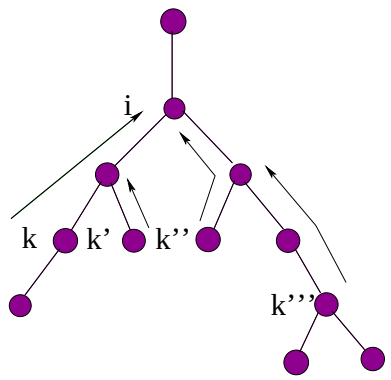
$$|L| = \sum_{i=1}^{n} |row_L(i)| + n. \tag{8.16}$$

Tato velikost tedy reprezentuje užitečnou informaci pro určení velikosti paměti pro uložení faktoru. Tato velikost je stejná pro odmocninovou i bezodmocninovou variantu rozkladu, protože v druhém případě sice neukládáme jednotkové diagonální prvky L, ale musíme také uložit diagonální faktor D.

Následující Algoritmus 8.3.1 počítá nejen počty mimodiagonálních prvků na řádcích, ale zároveň nalezne i strukturu řádků faktoru. Je také vidět, že by algoritmus mohl zároveň poskytnout i počty nenulových prvků faktoru L ve sloupcích, viz [107], ale to zde pro jednoduchost nebudeme uvádět. Počty mimodiagonálních prvků na řádcích L a struktury řídkosti těchto řádků jsou v algoritmu, po řadě, označeny v souladu s naší terminologií

$$|row_L(i)|, \quad row_L(i), \quad i = 1, \ldots, n.$$

Algoritmus 8.3.1 postupně prochází řádkové podstromy dříve zkonstruovaného eliminačního



Obrázek 8.3.21: *Schématické znázornění Algoritmu 8.3.1 pro počítání vrcholů v i-tém řádkovém podstromu. Šipky ukazují směr jeho procházení. Předpokládáme, že listy $T_r(i)$ jsou k, k', k'' a k'''. Mezi prohledávanými prvky i-tého řádku (vrcholy příslušného grafu) ale mohou být nenulové prvky i s jinými sloupcovými indexy.*

stromu matice. Souběžný výpočet byl navržen v [18] Vícenásobnému zápočtu vrcholů v tomto procházení se vyhne použitím pomocného vektoru *mark*.

Algoritmus 8.3.1 Počty a struktury řídkosti mimodiagonálních prvků řádků Choleského faktoru L.
Input: *Řídká symetrická matice dimenze n splňující podmínky z definice eliminačního stromu, její neorientovaný graf a eliminační strom $T(A)$.*
Output: *Počty mimodiagonálních prvků $|row_L(i)|$ v řádcích faktoru L a struktury řídkosti $row_L(i)$ těchto řádků pro $i = 1, \ldots, n$*

1. **for** $i = 1 : n$ **do**
2. $|row_L(i)| = 0$
3. $mark(i) = i$
4. **for** k *taková, že* $k \in adj(i) \wedge k < i$ **do**
5. $j = k$
6. **while** $(mark(j) \neq i)$ **do**
7. $|row_L(i)| = |row_L(i)| + 1$
7. $row_L(i) = row_L(i) \cup \{j\}$
8. $mark(j) = i$
9. $j = parent(j)$
10. **end while**
11. **end** k
12. **end** i

Příklad 8.3.2 *Schématické znázornění postupu Algoritmu 8.3.1 je na Obrázku 8.3.21. Šipky ukazují směr procházení od listů řádkového podstromu k jeho kořeni.*

Za jednoduchost Algoritmu 8.3.1 platíme časovou složitostí, která je $O(|L|+n)$. Princip efektivnějšího postupu, který není závislý na počtu nenulových prvků Choleského faktoru

bude zmíněn později. Struktury řídkosti řádků jsou podstatné v řádkové variantě Choleského rozkladu, ale mohou být užitečné i v jiných situacích. V případě, kdy jsou známy listy řádkových podstromů, ale i tehdy, když procházíme řádkové podstromy s počátky hledání v mimodiagonálních prvcích dané matice A, je možné i pro řádkový algoritmus nepředpočítávat struktury řídkosti před rozkladem, ale počítat je zároveň s ním.

8.3.3 Struktura řídkosti sloupců Choleského faktoru

Uvažujme výše zavedené označení množiny indexů aplikované na nenulové mimodiagonální prvky Choleského faktoru L.

$$col_L(j) \equiv \mathcal{S}(L_{j+1:n,j}) = \{k \mid k > j,\ l_{kj} \neq 0\},\ 1 < j \leq n. \tag{8.17}$$

Tvrzení v Pozorování 8.1.2 lze pak přepsat na

$$col_L(j) \subseteq col_L(parent(j)) \cup \{parent(j)\}. \tag{8.18}$$

Zatímco Pozorování 8.1.2 definuje sloupcové struktury rekurzívně, Věta 8.3.2 rozšiřuje tento lokální popis a popisuje strukturu $col_L(j)$ mimodiagonálních prvků j-tého sloupce faktoru L pro $1 \leq j \leq n$ explicitně.

Věta 8.3.2 *Struktura řídkosti sloupce j faktoru L matice A je dána množinou sousednosti vrcholů podstromu $T(j)$ eliminačního stromu T. To lze zapsat*

$$col_L(j) = adj_{G(A)}(T(j)). \tag{8.19}$$

Důkaz: Předpokládejme nejprve, že $i \in col_L(j)$. Podle Věty 8.3.1 pak platí, že j, $j < i$ je předkem nějakého vrcholu k v eliminačním stromu T, kde $a_{ik} \neq 0$. Jinak zapsáno platí

$$i \in col_L(j) \Rightarrow (\exists k \in T(j))(i \in adj_{G(A)}(k)).$$

Obráceně, uvažujme $i \in adj_{G(A)}(T(j))$. Odsud plyne, že v i-tém řádku L je prvek na pozici $j \neq i$, $j < i$ nenulový. To lze také zapsat $j \in row_L(i)$ a tedy i $i \in col_L(j)$. Ekvivalence množin z této věty je tak dokázána. ■

Jiný, ale stále rekurzívní popis množiny $col_L(j)$ pro daný sloupec j, je uveden ve Větě 8.3.3. Algoritmická atraktivnost tohoto popisu je v tom, že pro získání struktury řídkosti nějakého sloupce stačí obecně sjednotit struktury řídkosti jenom některých předcházejících sloupců.

Věta 8.3.3 *Za předpokladu o nevyrušení platí následující ekvivalence množin*

$$col_L(j) = \left(adj_{G(A)}(j) \cup \bigcup_{\{k|j=parent(k)\}} col_L(k) \right) \setminus \{1, \ldots, j\}. \tag{8.20}$$

Důkaz: Uvažujme nejprve výraz na pravé straně. Z definice množiny sousednosti plyne

$$adj_{G(A)}(j) \subseteq col_L(j) \cup \{1, \ldots, j-1\}.$$

Podle (8.18) platí $col_L(k) \subseteq col_L(j) \cup \{j\}$ pro každého syna k vrcholu j. Dohromady tedy po odečtení množiny $\{1, \ldots, j-1\}$ dostáváme, že všechny prvky množiny na pravé straně (8.20) patří i do $col_L(j) \equiv adj_{G(A)}(T(j))$.

Obráceně, nechť $i \in col_L(j)$. Protože $l_{ij} \neq 0$ pak je samozřejmě i $i > j$ jak ostatně vyplývá i z Lemmatu 8.2.2. Podle Věty 8.3.1 je vrchol j předkem nějakého vrcholu k' v eliminačním stromu T, pro který platí $a_{ik'} \neq 0$. Je-li $k' = j$, pak platí $i \in adj_{G(A)}(j)$. Jinak existuje syn k vrcholu j v T, který je obecně předkem vrcholu k'. Podle Věty 8.1.3 máme také $l_{ik} \neq 0$ a tedy $i \in adj_{G(A)}(T(k))$ a Věta je dokázána. ■

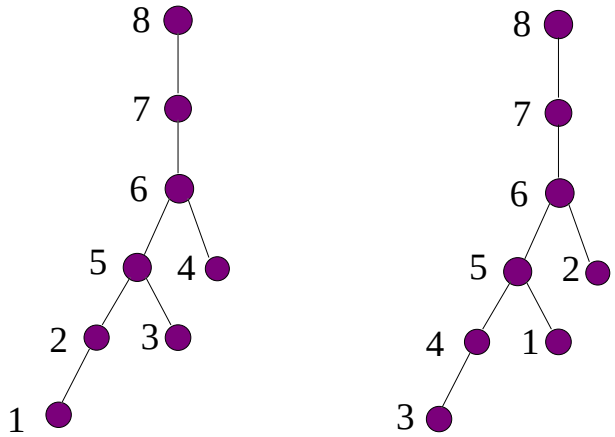
8.3.4 Nalezení struktur řídkosti sloupců Choleského faktoru

Nalezení sloupcové struktury řídkosti faktoru vychází z Věty 8.3.3. Algoritmus 8.3.2 tento postup nalezení struktur řídkosti sloupců Choleského faktoru popisuje. Pomocnou datovou strukturou použitou v tomto algoritmu jsou seznamy synů $son(j)$ vrcholů j pro $j = 1, \ldots, n$. Tyto seznamy se inicializují prázdnými množinami a aktualizují v průběhu algoritmu. Reprezentují tedy jinou strukturu zachycující eliminační strom, který je tak vytvářen v průběhu Algoritmu 8.3.2.

Algoritmus 8.3.2 Nalezení sloupcových struktur faktoru L.
Input: *Neorientovaný graf $G(A)$ matice A.*
Output: *Sloupcové struktury řídkosti $col_L(j)$ faktoru L pro $j = 1, \ldots, n$.*

1. *for* $j = 1$ *to* n *do*
2. $son(j) = \emptyset$
3. *end* j
4. *for* $j = 1$ *to* n *do*
5. $col_L(j)$ = *množina indexů množiny* $adj_G(j) \setminus \{1, \ldots, j-1\}$
6. *for* $k \in son(j)$ *do*
7. $col_L(j) = col_L(j) \cup col_L(k) \setminus \{j\}$
8. *end* k
9. *if* $col_L(j) \neq 0$ *then*
10. $p = \min\{i \mid i \in col_L(j)\}$
11. $son(p) = son(p) \cup \{j\}$
12. *end if*
13. *end* j

Algoritmus 8.3.2 se někdy nazýval **symbolická faktorizace**. Tento název je historický a pochází z doby, kdy tento postup byl naprosto dominantní částí symbolického předzpracování Choleského rozkladu řídké matice. V dnešní době se tímto pojmem popisuje symbolické předzpracování složené z obvykle mnohem více částí, nehledě na odlišnou terminologii vzniklou v dalších vědeckých školách.



Obrázek 8.4.22: *Příklad topologických očíslování kořenového stromu. Strom na levé straně je kořenový strom s topologickým očíslováním vrcholů. Strom na pravé straně má vrcholy očíslované* **jiným** *topologickým očíslováním.*

Podstatnou podmínkou počítání sloupcových struktur je takové pořadí počítání struktur sloupců, které splňuje podmínku jejich rekurzívního vyčíslování a tedy využití vztahu z Věty 8.3.2. V okamžiku počítání struktury řídkosti konkrétního sloupce musí již být vyčísleny struktury řídkosti sloupců odpovídajících synům tohoto sloupce v eliminačním stromu. Jak plyne z Lemmatu 8.2.2, pořadí $1, \ldots n$ toto splňuje, neboť eliminační strom má vrcholy očíslovány vrcholy topologickým očíslováním, viz Definici 5.2.7. Obecně lze ale použít v Algoritmu 8.3.2 jakékoli jiné topologické očíslování vrcholů eliminačního stromu. Některá z těchto očíslování mohou být z různých důvodů výhodnější a přečíslováním vrcholů eliminačního stromu se budeme zabývat v následujícím textu.

8.4 Přečíslování vrcholů eliminačního stromu.

Jedno ze zásadních témat spojených nejen s eliminačními stromy, ale i s obecnějšími orientovanými acyklickými grafy, je spojeno s přečíslováním jejich vrcholů. To umožňuje v mnoha případech získat efektivnější algoritmy nebo implementace Choleského nebo obecnějších rozkladů. Přečíslování vrcholů eliminačního stromu $T(A)$ odpovídá symetrické permutaci matice A a v mnoha případech umožní získat efektivnější algoritmy nebo implementace rozkladu.

8.4.1 Obecná topologická očíslování a zaplnění

Nejprve si na příkladě ukažme dvě různá topologická očíslování kořenového stromu, která jsou speciálním případem očíslování orientovaného acyklického grafu.

Příklad 8.4.1 *Příklady dvou topologických očíslování kořenového stromu jsou na Obrázku 8.4.22.*

Nové očíslování eliminačního stromu indukuje permutaci (přeuspořádání) matice a naopak. Pokud nebude moci dojít k nedorozumění, budeme používat v tomto smyslu také

termín přečíslování (nové očíslování) (eliminačního stromu, grafu) alternativně k pojmu přeuspořádání matice. Budeme také například hovořit o **topologickém přečíslování matice**, kterým rozumíme symetrickou permutaci matice indukovanou nějakým novým topologickým očíslováním eliminačního stromu. Věta 8.4.1, ukazuje ekvivalenci mezi velikostí zaplnění v Choleského rozkladu matice A a velikostí zaplnění v Choleského rozkladu topologicky přečíslované matice. Tato vlastnost zabezpečuje, že **z hlediska počtu operací** máme volné ruce k přeuspořádání matice založenému na topologickém přečíslování jejího eliminačního stromu. V následujícím textu uvidíme, že řada algoritmů, které se týkají řídkého maticového rozkladu, často kvůli efektivnosti předpokládá, že je příslušný eliminační strom očíslován **specifickým** topologickým očíslováním.

Věta 8.4.1 *Nechť A je daná matice a $T(A)$ je její eliminační strom. Nechť je dále P permutační matice příslušné dimenze, kde je vztah mezi vrcholy eliminačního stromu matice P^TAP a vrcholy eliminačního stromu $T(A)$ dán topologickým přečíslováním $\alpha : V(A) \leftrightarrow V(P^TAP)$. Pak jsou grafy zaplnění Choleského faktorizace matice A a matice P^TAP izomorfní.*

Důkaz: Položme $\alpha(A) = P^TAP$. Nechť $\alpha(F)$ je graf zaplnění matice $\alpha(A)$. Dále označme $1, \ldots, n$, respektive $\alpha(1), \ldots, \alpha(n)$, vrcholy grafu matice A, respektive $\alpha(A)$. Nechť i a j jsou dva vrcholy grafu $G(A)$. Nechť dále $r = \alpha(i)$ a $s = \alpha(j)$. Ukážeme, že $\{i, j\} \in G(F)$ právě tehdy, když $\{r, s\} \in G(\alpha(F))$.

Nejprve nechť $\{i, j\} \in G(F)$ a $i > j$, tedy podle předpokladu o nevyrušení i $l_{ij} \neq 0$. Podle Věty 8.2.3 je i vlastním předkem j v eliminačním stromu $T(A)$. Neboť α je topologické přeuspořádání grafu, pak platí $r > s$. Podle Věty 8.2.2 existuje v grafu $G(A)$ cesta i, p_1, $\ldots, p_t, j$ pro kterou platí $\{p_1, \ldots, p_t\} \in T(j)$. Protože je nové očíslování topologické, pak všechny tyto mezilehlé vrcholy musí mít obrazy také menší než s. Z Věty o zaplnění pak plyne, že $\{r, s\} \in G(\alpha(F))$.

Obráceně, nechť $\{r, s\} \in G(\alpha(F))$. Nechť přitom platí $r > s$. Podle Věty o zaplnění existuje v $G(\alpha(A))$ cesta $r, p_1, \ldots, p_t, s$ taková, že indexy všech (případných) mezilehlých vrcholů jsou menší i než s. Neboť zkoumané přeuspořádání je topologické, vrcholy $p_1, \ldots, p_t$ nemohou patřit do $anc(j)$. Jinak by totiž vztah mezi jejich obrazy vyvracel předpoklad, že očíslování je topologické. Podle Důsledku 8.2.3 tyto všechny vrcholy musí patřit do $T(j)$. Jinými slovy, v $G(A)$ existuje cesta mezi r a s přes vrcholy z $T(j)$. Dále víme, že i leží vně $T(j)$. Následně, $i > j$. Podle Věty 8.2.1 platí $\{i, j\} \in G(F)$ a věta je dokázána. ■

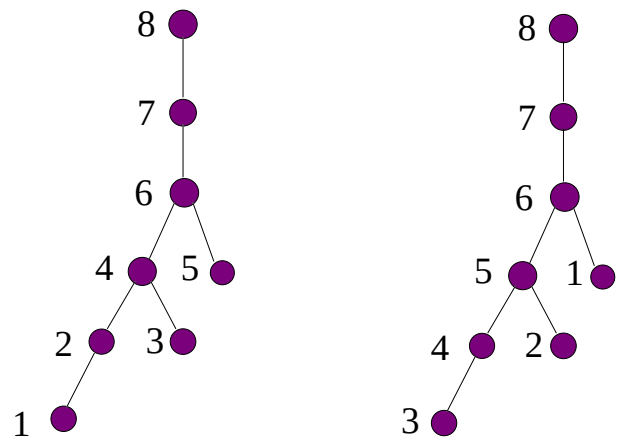
8.4.2 Postordering

Speciální třídu topologických očíslování orientovaného acylického grafu tvoří množina očíslování, které jsou nazývány **postordering**. Definujeme si toto očíslování následovně.

Definice 8.4.1 *Topologické očíslování α kořenového stromu $T = (V = \{1, \ldots, n\}, E)$ nazveme postordering, tvoří-li množina vrcholů* **v každém jeho podstromu**

$$T(j), \quad \text{pro} \quad j = 1, \ldots, n$$

interval přirozených čísel množiny $\{1, \ldots, n\}$.



Obrázek 8.4.23: *Dva různé postorderingy kořenového stromu.*

Pozorování 8.4.1 *Je zřejmé, že v očíslování $\alpha : V \leftrightarrow \{1, \ldots, n\}$, které je postorderingem, je obrazem vrcholu $\alpha(1)$ vždy* **list** *kořenového stromu.*

Příklad 8.4.2 *Žádné očíslování stromu na Obrázku 8.4.22 není postorderingem. Dva různé postorderingy stromu z Obrázku 8.4.22 jsou znázorněny na Obrázku 8.4.23.*

8.4.3 Procházení kořenového stromu a topologická očíslování

Přirozený postup jak získat postordering nebo i jiná topologická očíslování eliminačního stromu je proces procházení, který nyní zavedeme pro obecný pojem neorientovaného stromu, tedy neorientovaného acyklického a souvislého grafu.

Definice 8.4.2 Procházení neorientovaného *a stromu (stromu v grafovém smyslu, tedy souvislého) $T = (V, E)$ je proces, při kterém každý vrchol V stromu* **navštívíme** *právě jednou. Operace navštívení vrcholu $v \in V$ znamená zařazení v do množiny navštívených vrcholů V_v podle následujícího popisu.*

V každém okamžiku procházení máme vrcholy rozdělené na navštívené V_v a ostatní, dosud nenavštívené vrcholy V_n. Množiny V_v a V_n tvoří v každém kroku procházení rozklad množiny V.

- *První krok procházení zařadí jeden zvolený vrchol (počáteční vrchol procházení) do množiny V_v.*
- *Každý další krok procházení*
  - *Vybere* **hranu procházení** *$\{x, y\} \in E$, která má jeden koncový vrchol x v množině V_v navštívených vrcholů a druhý z nich, y, v množině V_n nenavštívených vrcholů.*
  - *Přesune y z V_n do V_v.*
- *Procházení je ukončeno jestliže žádnou další hranu procházení nelze najít.*

Procházení je možné snadno formulovat pro jiné druhy grafu. Například, procházení orientovaného stromu T lze převést na procházení jeho symetrizace a neuvažovat orientaci hran. Jiným způsobem procházení v případě orientovaného stromu je uvažovat místo symetrizace takové kroky procházení, kde hrana procházení mezi vrcholem $x \in V_v$ a $y \in V_n$ je buď neorientovaná nebo orientovaná hrana z $V_v \times V_n$ s počátečním vrcholem x. Pak se může stát, že nelze všechny vrcholy zařadit mezi navštívené. V každém případě lze ale projít pouze vrcholy, které patří souvislé komponentě grafu. Eliminační strom můžeme procházet tím, že uvažujeme jeho symetrizaci a procházíme všechny jeho komponenty.

Definice 8.4.3 *Vrcholy stromu, které lze konkrétním procházením z určitého vrcholu $x \in V$ navštívit (zařazené do V_v), se nazývají vrcholy* **dosažitelné** *z vrcholu x.*

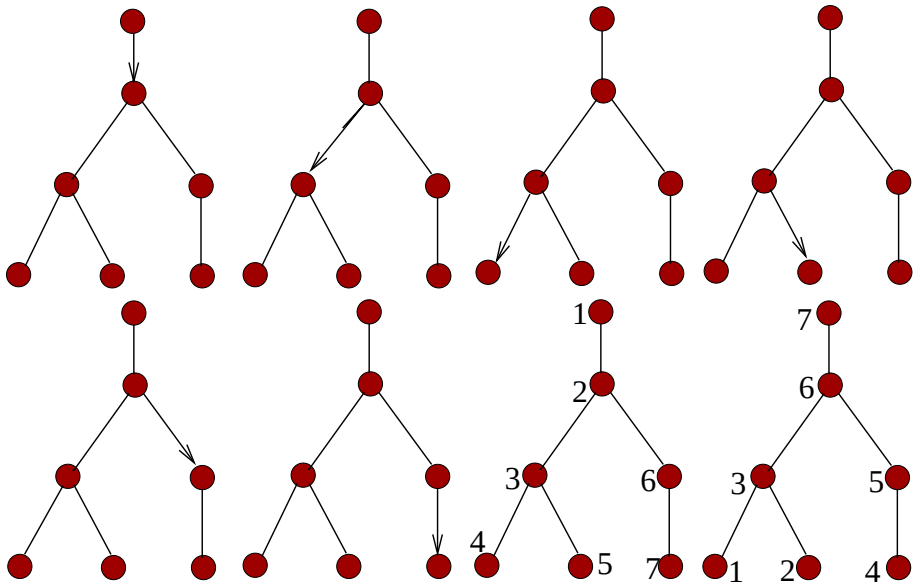
V praxi nás zajímají především **systematické** způsoby procházení stromů. Například, v **procházení stromu do hloubky** se za hranu procházení volí vždy taková hrana, jejíž navštívený vrchol je **poslední dosud navštívený vrchol** stromu a pro který hrana procházení existuje. Zatímco navštívený vrchol hrany procházení z V_v je jednoznačně určen, není tomu tak pro vrchol z V_n. V tomto smyslu není ani procházení do hloubky jednoznačně určeno, jak ještě zmíníme dále. V **procházení stromu do šířky** zase volíme nejprve všechny hrany procházení takové, že vrcholy z V_v mají co nejmenší vzdálenost od počátku procházení. Ani v tomto případě není procházení jednoznačné.

Důležitým aspektem procházení je vytvoření **záznamu** o procházení. Záznamy seřazené v získaném pořadí jsou ve tvaru **posloupnosti zaznamenaných vrcholů**. Pro vytvoření záznamu v konkrétním procházení do hloubky je také více možností. Jednou možností je vložit vrchol do záznamu (označkovat) v okamžiku, kdy se tento vrchol stává vrcholem hrany procházení a patří ještě do V_n. Přečíslování stromu, které získaná posloupnost vrcholů určuje nazveme **preordering** stromu. Druhou možností je vložit do tvořené posloupnosti záznamů záznam o vrcholu j právě tehdy, když tento vrchol vytvoří podstrom, ve kterém jsou už v posloupnosti zaznamenaných vrcholů všechny vrcholy kromě j. Takto získané přečíslování nazveme **postordering**. Souvislost s předcházející definicí postorderingu uvedeme níže. Nově vytvořené posloupnosti záznamů pro strom se všemi vrcholy dosažitelnými reprezentují pro $V = \{1, \ldots, n\}$ bijekce nad touto množinou a odpovídají tedy přečíslování vrcholů množiny V.

Konstrukce preorderingu a postorderingu pro procházení stromu do hloubky se dá velmi efektivně implementovat algoritmicky. Konkrétní Algoritmus 8.4.1 procházení stromu do hloubky uvádíme jako rekurzívní proceduru *df_traverse*. Připomeňme, že u stromu chápaného jako grafový pojem a nikoli jako eliminační strom předpokládáme souvislost a tedy i dosažitelnost všech vrcholů jeho procházením.

Algoritmus 8.4.1 Algoritmus $depth\_first\_traverse(j, kpre, kpost)$**: Procházení neorientovaného stromu T do hloubky s počátečním vrcholem procházení j.**
Input: *Neorientovaný strom T.*
Output: *Dvě různé posloupnosti zaznamenaných vrcholů T:* **preordering, postordering**.

1. $kpre = 1, kpost = 1$ *% nastavení globálních čítačů zaznamenaných vrcholů*
2. $kpre = kpre + 1$, *%přesuň j do V_v*



Obrázek 8.4.24: *Schématické znázornění procházení stromu do hloubky. Na grafech zleva a doprava a shora dolů jsou postupně znázorněny hrany procházení šipkou od vrcholu z V_v k vrcholu z V_n. Na předposledním z nich je znázorněno očíslování procházeného stromu preorderingem, na posledním z nich je znázorněno očíslování postorderingem.*

3. $preordering(kpre) = j$ *%přesuň j do V_v*
4. **for** i *taková, že* $i \in adj_T(j)$, $i \in V_n$ **do**
5. $\quad depth\_first\_traverse(i, kpre, kpost)$
6. **end** i
7. $kpost = kpost + 1$
8. $postordering(kpost) = j$

Jak jsme zmínili výše, Algoritmus 8.4.1 procházení stromu do hloubky s počátečním vrcholem procházení j nemá v kroku 3 jednoznačně určeno pořadí procházení vrcholů, pro které je j rodičem a je tedy více možností, jak budou posloupnosti záznamů a tedy odpovídající nová očíslování vrcholů vypadat. Jinými slovy, takto zavedená očíslování preordering a postordering nejsou **jednoznačná**. Souvislost T implikuje, že po ukončení Algoritmu 8.4.1 platí $kpre = kpost = |V| + 1$.

Klíčem pro implementaci rekurzívního Algoritmu 8.4.1 bez rekurze je použít datovou strukturu zásobníku. Následující obrázek znázorňuje procházení grafu do hloubky včetně očíslování vrcholů preorderingem a postorderingem na posledních dvou podobrázcích.

Uvažujme nyní eliminační strom, který jsem zavedli jako acyklický, orientovaný a obecně nesouvislý graf. Každá komponenta eliminačního stromu je kořenový strom, jejíž kořen je k, což je počet jejích vrcholů. Vrchol, řekněme j, komponenty eliminačního stromu T je v prohledávání do hloubky označen jako navštívený a zaznamenán do posloupnosti tvořící postordering teprve tehdy, když byl takto označen i celý podstrom $T(j)$ komponenty eliminačního stromu T. Eliminační strom očíslujeme postorderingem projdeme-

li všechny jeho komponenty z jejích kořenů. Snadno nahlédneme, že postordering takto získaný formálně procházením symetrizace kořenového stromu z jeho kořene splňuje vlastnosti z Definice 8.4.1, protože prohledávání do hloubky zabezpečuje, že v každém podstromu tvoří vrcholy v pořadí postorderingu interval přirozených čísel. To vyjádřeme i formálně.

Pozorování 8.4.2 *Očíslování vrcholů kořenového stromu nalezené jako postordering Algoritmem 8.4.1 při procházení z kořene splňuje Definici 8.4.1.*

V následujícím textu probereme další témata týkající se řídkého Choleského rozkladu. Související tvrzení a algoritmy budou **často předpokládat**, že příslušný eliminační strom matice je očíslován postorderingem a budeme tehdy hovořit o matici přeuspořádané postorderingem.

8.5 Listy řádkových podstromů

Důležitou roli v řídké Choleského faktorizaci hrají listy řádkových podstromů

$$T_r(i),\ i = 1, \ldots, n$$

eliminačního stromu T. Jak jsme viděli, tyto listy **nejsou obecně** totožné s listy podstromů $T(i)$ eliminačního stromu T pro $1 \leq i \leq n$, protože řádkové podstromy jsou obecně ořezané podstromy. Nicméně oba druhy podstromů jsou spojeny vztahy z následující Věty 8.5.1.

Věta 8.5.1 *Nechť jsou vrcholy eliminačního stromu T matice A uspořádány* **postorderingem**. *Uvažujme řádkový index i, $1 < i \leq n$. Označme dále*

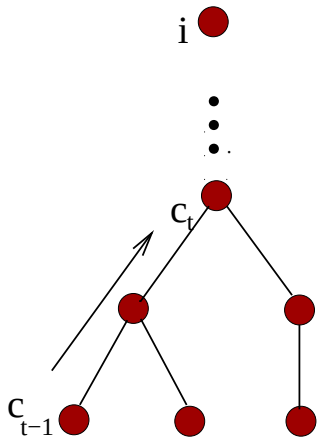
$$adj_{G(A)}(i) \cap \{1, \ldots, i-1\} = \{c_1, \ldots, c_s\},\ 0 < c_1 < \ldots < c_s < i,\ s \geq 1.$$

Pak platí, že vrchol c_t pro $t \in \{1, \ldots, s\}$ je listem řádkového podstromu $T_r(i)$ eliminačního stromu T právě tehdy, je-li $t = 1$ nebo platí-li

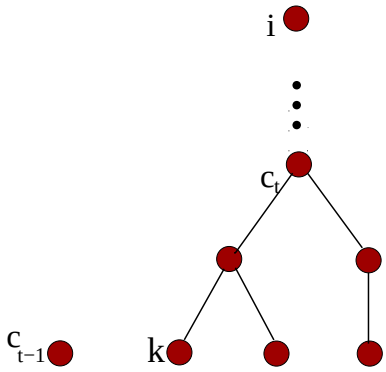
$$t > 1 \ \wedge\ c_{t-1} \notin T(c_t). \tag{8.21}$$

Důkaz: Sloupcový index prvního nenulového prvku c_1 na řádku i je vždy listem řádkového podstromu $T_r(i)$. Kdyby totiž nebyl c_1 listem $T_r(i)$, pak by v podstromu $T(c_1)$ musel existovat list řádkového podstromu $T_r(i)$. Podle vlastností topologického uspořádání (a tedy i postorderingu) by byl menší než c_1 a reprezentoval by tedy nenulový prvek na řádku i matice A s indexem menším než c_1. To by byl spor a proto se soustřeďme na případ $t > 1$.

Nechť platí $c_{t-1} \in T(c_t)$ a zároveň je c_t listem $T_r(i)$. Situaci si znázorněme na Obrázku 8.5.25. Protože c_{t-1} je sloupcovým indexem nenulového prvku řádku i matice A, pak pro všechny jeho předky $k, k < i$, mezi kterými je podle předpokladu i c_t, platí $l_{ik} \neq 0$. Pro všechna taková k musí platit $c_{t-1} \leq k < c_t$ (vlastnost topologického uspořádání vrcholů eliminačního stromu) a jsou tedy nejen v $T(c_t)$ ale i v $T_r(i)$. Pak tedy ale c_t nemůže být listem řádkového podstromu $T_r(i)$.



Obrázek 8.5.25: *Znázornění situace v $T(i)$, kdy pro $t > 1$ platí $c_{t-1} \in T(c_t)$ a zároveň se předpokládá, že c_t listem řádkového podstromu $T_r(i)$.*



Obrázek 8.5.26: *Znázornění situace v $T(i)$, kdy pro $t > 1$ platí $c_{t-1} \notin T(c_t)$ a zároveň se předpokládá, že c_t není listem řádkového podstromu $T_r(i)$.*

Opačně, nechť c_t není listem řádkového podstromu $T_r(i)$ a zároveň $c_{t-1} \notin T(c_t)$. Samozřejmě nemůže platit $t = 1$. Situaci pro $t > 1$ opět znázorníme graficky. Viz Obrázek 8.5.26. Pro $t > 1$ tedy existuje list k podstromu $T_r(i)$, do kterého vede z kořene stromu T cesta přes vrchol c_t. Pro tento list platí $a_{ik} \neq 0$, tedy $k \in adj_{G(A)}(i) \wedge k < i$. Z toho plyne, že všechny vrcholy s indexy l, $k \leq l < c_t$ jsou v podstromu $T(c_t)$ (vlastnost postorderingu). Protože tohle platí pro každý takový list k' podstromu $T_r(i)$, do kterého vede z kořene stromu T cesta přes vrchol c_t, pak mezi potomky $k, k', \ldots$ musí být i vrchol c_{t-1}, to jest největší z těchto potomků, který tedy musí být synem vrcholu c_t v i-tém řádku matice A a opět máme spor s předpokladem. Věta je tedy dokázána. ■

Očíslování vrcholů eliminačního stromu postorderingem umožňuje velmi efektivní algoritmické testování podmínky z Věty 8.5.1. Důsledek 8.5.1 tento postup naznačuje.

Důsledek 8.5.1 *Nechť jsou vrcholy eliminačního stromu T matice A uspořádány postorderingem. Uvažujme řádkový index i, $1 < i \leq n$. Nechť dále*

$$adj_{G(A)}(i) \cap \{1, \ldots, i-1\} = \{c_1, \ldots, c_s\},\ 0 < c_1 < \ldots < c_s < i,\ s \geq 1.$$

Vrchol c_t pro $t \in \{1, \ldots, s\}$ je listem řádkového podstromu $T_r(i)$ eliminačního stromu T právě tehdy, platí-li

$$t = 1 \quad \text{nebo} \quad t > 1 \wedge c_{t-1} < c_t - |T(c_t)| + 1.$$

Platnost Důsledku 8.5.1 je zřejmá z Věty 8.5.1. Z vlastností postorderingu totiž plyne, že v případě platnosti podmínky v této Větě musí být dříve než c_t očíslovány všechny vrcholy v podstromu $T(c_t)$.

Abychom mohli test z Důsledku 8.5.1 efektivně implementovat, je potřeba znát velikosti podstromů eliminačního stromu. To je ale jednoduchá úloha. Algoritmus 8.5.1 počítá velikosti podstromů kořenového stromu. Má-li eliminační strom více komponent, postup se aplikuje na všechny jeho komponenty ve formě kořenových stromů samostatně. Pro správný výpočet velikostí podstromů kořenového stromu stačí předpokládat, že eliminační strom má vrcholy očíslovány obecným topologickým očíslováním, což splňuje i prvotní očíslování vrcholů eliminačního stromu bez dodatečného přečíslování.

Algoritmus 8.5.1 Nalezení velikostí podstromů $|T(i)|$ v kořenovém stromu.
Input: *Kořenový strom T s topologickým očíslováním vrcholů.*
Output: *Velikosti podstromů $|T(i)|$ pro $i = 1, \ldots, n$*

1. **for** $i = 1 : n$ **do**
2. $\quad |T(i)| = 1$
3. **end** i
4. **for** $i = 1 : n - 1$ **do**
5. $\quad k = parent(i)$
6. $\quad |T(k)| = |T(k)| + |T(i)|$
7. **end** i

Známe-li uspořádání prvků na řádcích matice podle jejich sloupcových indexů od nejmenšího po největší. Pak může být hledání listů eliminačních podstromů založené na procházení poddiagonální části matice A po řádcích s využitím Důsledku 8.5.1 přímočaré. Takový postup ale nemusí vždy vyhovovat, protože uspořádání nenulových prvků v poddiagonální části matice A podle sloupcových indexů může nalezení listů řádkových podstromů zpomalit. Navíc, pro další úkoly může být výhodnější procházení struktury řídkosti matice po sloupcích a některé takové úkoly popíšeme v dalším textu. Proto je užitečné tvrzení Věty 8.5.2.

Věta 8.5.2 *Vrchol j je listem nějakého řádkového podstromu eliminačního stromu T očíslovaného postorderingem právě tehdy, existuje-li vrchol $i \in adj_{G(A)}(j), i > j$ takový, že $i \notin adj_{G(A)}(k)$ pro všechna $k \in T(j) \setminus \{j\}, i > k$.*

Důkaz: Je-li j listem nějakého řádkového podstromu, například $T_r(i)$, pak platí $i \in adj_{G(A)}(j), i > j$. V množině $T(j) \setminus \{j\}$ pak nemůže existovat žádný vrchol k takový, že by platilo $i \in adj_{G(A)}(k), i > k$. Z jeho existence by totiž plynulo, že všichni jeho předchůdci, mezi které při předpokladu postorderingu patří i j, jsou v řádkovém podstromu $T_r(i)$ a j by tedy nemohl být listem $T_r(i)$.

Opačně, nechť j není listem žádného řádkového podstromu eliminačního stromu T. Předpokládejme, že existuje vrchol $i \in adj_{G(A)}(j), i > j$ (to jest, j je vrchol řádkového podstromu $T_r(i)$), pro který platí vztah $i \notin adj_{G(A)}(k), i > k$ pro všechna $k \in T(j) \setminus \{j\}$. Protože j není listem $T_r(i)$, pak podle Věty 8.3.1 musí existovat nějaké $k \in T(j) \setminus \{j\}$ takové, že $a_{ik} \neq 0$. Z předpokladu o nevyrušení tedy dohromady plyne, že $l_{ik} \neq 0$ a zároveň $l_{ij} \neq 0$. Předpokládali jsme ale, že j je vrchol řádkového podstromu $T_r(i)$ a přitom existuje vrchol k takový, že $i \in adj_{G(A)}(k)$, $i > k$, $i > j$, který je potomkem vrcholu j, tedy patří do $T(j) \setminus \{j\}$. To je ve sporu s naším předpokladem a věta je tak dokázána. ■

Všimněme si, že tvrzení Věty 8.5.2 hovoří právě o tom, jak najdeme při procházení po sloupcích listy řádkových podstromů kořenového stromu. Uvažujeme-li sloupce j, pak jej zařadíme mezi listy právě tehdy, když při procházení tohoto sloupce trojúhelníkové matice narazíme na první index $i > j$ splňující tvrzení. Postordering přitom implikuje, že všechny diskutované vrcholy k splňující $k \in T(j) \setminus \{j\}, i > k$ už byly zpracovány. Algoritmus 8.5.2 nalezení listů řádkových podstromů kořenového stromu je tedy založen na Větě 8.5.2 a spoléhá se i na vlastnosti z Důsledku 8.5.1, ve které je vyjádřena vzdálenost mezi listy konkrétního řádkového podstromu. Výstupem je vektor $isleaf$ délky n, který pro každý vrchol i udává, zdali je ($isleaf(i)$=**true**) nebo není ($isleaf(i)$=**false**) listem nějakého řádkového podstromu.

Algoritmus 8.5.2 Nalezení listů řádkových podstromů.
Input: *Kořenový strom T s vrcholy očíslovanými postorderingem.*
Output: *Logický vektor $isleaf$ popsaný výše.*
1. **for** $j = 1 : n$ **do**
2. $isleaf(j)$=**false**
3. $prev\_nonz(j)$=*0*
4. **end** j
5. **for** $j = 1 : n$ **do**

```
6.     spočítej |T(j)|
7. end j
8. for j = 1 to n do
9.     for i takové, že i > j ∧ a_ij ≠ 0 (j ∈ adj_G(A)(i)) do
10.        k = prev_nonz(i)
11.        if k < j − |T(j)| + 1 then
12.            isleaf(j)=true
13.        end if
14.        prev_nonz(i) = j
15.    end i
16. end j
```

V Algoritmu 8.5.2 jasně vidíme, že je-li konkrétní index j listem nějakého řádkového podstromu, existuje řádek, kde je podmínka z Důsledku 8.5.1 splněna.

8.6 Supervrcholy

Podstatnou složkou algoritmů Choleského rozkladu i obecnějších rozkladu řídkých matic je využití **hustých bloků** struktury řídkosti faktoru L, který je obecně hustší než rozkládaná matice. Využití takových bloků vede obecně k větší výpočetní efektivnosti na moderních počítačových architekturách, protože je možné maticovými operaci mnohem lépe překrýt přenosy dat i paměťovou latenci. Souvisejícím efektem může být úspora paměti při ukládání řídké matice s hustými podbloky použitím menšího množství indexů v datových strukturách matice s blokovou strukturou.

Vznik hustých bloků v řídké Choleského faktorizaci matice A je možné vidět jako důsledek replikace sloupcových struktur při rozkladu. Předpokládejme, že eliminační strom rozkládané matice A je očíslován postorderingem. Jak jsme viděli, to umožňuje efektivně nalézt listy jeho řádkových podstromů. Následující pojem **supervrcholu** označuje konkrétní množinu indexů po sobě následujících sloupců, ale nebude-li moci dojít k nedorozumění, budeme o něm hovořit také jako o množině sloupců matice nebo o množině vrcholů příslušného neorientovaného grafového modelu, kde $V = \{1, \ldots, n\}$.

Definice 8.6.1 *Nechť $s, t \in N$, $1 \leq s \leq n, 1 \leq t \leq n, s+t-1 \leq n$. Řekneme, že množina sloupců matice Choleského faktoru L s indexy*

$$\{s, s+1, \ldots, s+t-1\}, \text{ kde } s, t \in N,\ 1 \leq s, t \leq n, s+t-1 \leq n, \tag{8.22}$$

je **supervrchol** *faktoru L, jestliže tuto množinu nejde zvětšit přidáním sloupce $s-1$ pro $s > 1$ nebo přidáním sloupce $s+t$ pro $s+t-1 < n$ a zároveň indexy sloupců $s, s+1, \ldots, s+t-1$ splňují rovnost*

$$col_L(s) \cup \{s\} = col_L(s+t-1) \cup \{s, \ldots, s+t-1\}. \tag{8.23}$$

Vrchol s a vrchol $s+t-1$ nazveme počátečním vrcholem respektive koncovým vrcholem supervrcholu. Supervrcholem může být i triviální množina S, která obsahuje jen vrchol s (jeden sloupec faktoru) a kde tedy $t = 1$.

$$
\begin{array}{c|ccccccccccc}
 & \dots & \dots & s & \dots & \dots & s' & \dots & \dots & \dots & \dots & \\
\hline
\vdots & \ddots & \dots & & & & & & & & & \\
\vdots & \vdots & \ddots & & & & & & & & & \\
s & & & * & & & & & & & & \\
\vdots & & & * & \ddots & & & & & & & \\
\vdots & & & * & * & \ddots & & & & & & \\
s' & & & * & * & * & * & & & & & \\
\vdots & & & . & \dots & \dots & . & \ddots & & & & \\
\vdots & & & * & * & * & * & & \ddots & & & \\
\vdots & & & * & * & * & * & & & \ddots & & \\
\vdots & & & . & \dots & \dots & . & & & & \ddots & \\
\vdots & & & * & * & * & * & & & & & \ddots
\end{array}
$$

Obrázek 8.6.27: *Příklad struktury řídkosti supervrcholu, kde s' označuje $s+t-1$.*

Fakt, že množina sloupců definující supervrchol nejde uvedeným způsobem zvětšit nazveme, že tato množina po sobě následujících sloupců z Definice 8.6.1 je **maximální ve smyslu inkluze**.
Příklad struktury řídkosti supervrcholu je na Obrázku 8.6.27. Z Definice 8.6.1 hned plyne, že indexy všech vrcholů, které vytvářejí jeden supervrchol, můžeme uložit do paměti počítače zaznamenáním počátečního a koncového vrcholu a počtu prvků množiny $\mathcal{S}(L_{*s+t-1})$ a úspora paměti při ukládání indexů je tedy hned patrná.

8.6.1 Charakterizace supervrcholů počtem nenulových prvků

Supervrchol lze charakterizovat následující nutnou a postačující podmínkou založenou na počtu nenulových prvků ve sloupcích faktoru L.

Věta 8.6.1 *Množina sloupců matice s indexy $S = \{s, s+1, \dots, s+t-1\}$ je* **supervrchol** *Choleského faktoru L právě tehdy, je-li takovou maximální množinou sloupců ve smyslu inkluze s po sobě následujícími indexy, pro něž v eliminačním stromu tohoto faktoru platí vztah*

$$\text{vrchol } s+k-1 \text{ je syn vrcholu } s+k,\ k = 1, \dots, t-1$$

a zároveň platí

$$|\, col_L(s)\,| = |\, col_L(s+t-1)\,| + t - 1. \tag{8.24}$$

Důkaz: Je-li S je supervrchol, pak Definice 8.6.1 implikuje, že jeho libovolné dva indexy i a j, $i > j$ splňují $i \in col_L(j)$, protože ve sloupci s jsou na těchto řádkových pozicích nenuly. Vrchol i tedy musí být předkem vrcholu j a v eliminačním stromu tedy existuje orientovaná cesta z i do j. Konkrétně, pro $i = j+1$, kde $j = s, \dots, s+t-2$ je vrchol i otcem vrcholu j a v eliminačním stromu tedy existuje cesta z vrcholu $s+t-1$ do vrcholu s je

$$s+t-1 \to \dots \to s-1 \to s.$$

Protože ale pro libovolné dva vrcholy i a j, $i > j$ platí replikace sloupcové struktury řídkosti

$$i \in col_L(j) \Rightarrow col_L(j) \setminus \{1, \dots j\} \subseteq col_L(i), \tag{8.25}$$

pak pro vrcholy supervrcholu platí

$$|\, col_L(s+i) \,| \geq |\, col_L(s+i-1) \,| - 1,\ i = 1, \dots, t-1 \tag{8.26}$$

s rovností právě tehdy, je-li

$$col_L(s+i) = col_L(s+i-1) \setminus \{s+i\}, \tag{8.27}$$

tedy v případě supervrcholu. První implikace je tedy dokázána.

Opačně, nechť množina indexů $S = \{s, s+1, \dots, s+t-1\}$ je takovou maximální množinou sloupců ve smyslu inkluze, pak to, že S je supervrchol plyne z nerovnosti (8.26) a nutné a postačující podmínky na rovnost v ní. ■

Charakterizace supervrcholů ve Větě 8.6.1 dává jednoduchý návod jak je nalézt. Stačí porovnat **počty** nenulových prvků ve sloupcích rozkladu, jejichž indexy tvoří **souvislé úseky orientovaných cest** v eliminačním stromu. Efektivnější způsob nalezení supervrcholů se dá použít, omezíme-li se na jejich následující speciální variantu.

8.6.2 Fundamentální supervrcholy

Definice 8.6.2 *Nechť $s, t \in N$, $1 \leq s \leq n, 1 \leq t \leq n, s+t-1 \leq n$ Řekneme, že množina $\{s, s+1, \dots, s+t-1\}$ indexů sloupců matice Choleského faktoru L je* **fundamentální supervrchol** *Choleského faktoru L, jestliže je zároveň maximální množinou indexů sloupců ve smyslu inkluze s po sobě následujícími indexy, pro které je vrchol $s+i-1$* **jediný** *syn vrcholu $s+i$ v eliminačním stromu pro $i = 1, \dots, t-1$ a přitom platí*

$$col_L(s) \cup \{s\} = col_L(s+t-1) \cup \{s, \dots, s+t-1\}. \tag{8.28}$$

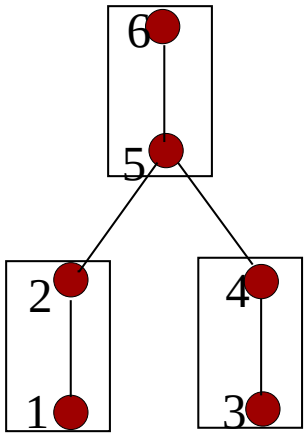
V praxi se ukazuje, že rozdíl mezi počtem supervrcholů a fundamentálních supervrcholů nebývá příliš velký a omezení na fundamentální supervrcholy tedy neovlivní příliš efektivitu Choleského rozkladu. Další podstatný důvod pro uvažování fundamentálních supervrcholů je to, že jsou nezávislé na konkrétním zvoleném postorderingu. Na Obrázku 8.6.28 je znázorněna matice, kde se liší její maximální supervrcholy a fundamentální supervrcholy.

Pro fundamentální supervrcholy platí Věta 8.6.2, která dává do souvislosti jejich počáteční vrcholy s listy řádkových podstromů.

Věta 8.6.2 *Sloupec s indexem s, $1 \leq s \leq n$ je počátečním vrcholem fundamentálního supervrcholu právě tehdy, má-li vrchol s alespoň dva syny v příslušném eliminačním stromu T nebo je-li s* **list nějakého řádkového podstromu** *eliminačního stromu.*

Důkaz: Má-li vrchol s alespoň dva syny v eliminačním stromu, pak musí být prvním vrcholem fundamentálního supervrcholu, do kterého patří. Nechť s je listem řádkového podstromu $T_r(i)$ pro nějaké $i > s$. V případě, že s je listem T, pak je tvrzení zřejmé,

$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{pmatrix}
* & * & & & * & * \\
* & * & & & * & * \\
 & & * & * & * & * \\
 & & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & *
\end{pmatrix}
\end{array}
$$



Obrázek 8.6.28: *Příklad matice a jejího eliminačního stromu s fundamentálními supervrcholy vyznačenými rámečky. Její maximální supervrcholy jsou jen dva:* $\{1, 2\}$, $\{3, 4, 5, 6\}$.

protože takový vrchol musí být počátečním vrcholem jakéhokoli supervrcholu. Má-li s právě jednoho syna, pak postordering (který v této kapitole předpokládáme) implikuje, že vrchol $s - 1$ je synem vrcholu s.

Podle Věty 8.3.1 dále platí pro list $T_r(i)$

$$a_{is} \neq 0 \text{ a } a_{i,s-1} = 0,$$

to jest, i není v $col_L(s-1)$, ale je v $col_L(s)$. Odsud plyne

$$\mathcal{S}(L_{*s-1}) \subsetneqq \mathcal{S}(L_{*s}) \cup \{s-1\} \tag{8.29}$$

a vrcholy s a $s-1$ nemohou patřit do stejného supervrcholu a tedy ani do stejného fundamentálního supervrcholu. Index s tedy začíná nový fundamentální supervrchol.

Předpokládejme nyní, že s je počátečním indexem fundamentálního supervrcholu S. Má-li vrchol s alespoň dva syny nebo žádného syna, tvrzení platí. Má-li jen jednoho syna, pak je podle vlastností postorderingu tento potomek $s-1$. Podotkněme, že v případě obecného topologického očíslování eliminačního stromu tomu tak nemusí být. Potom ale existuje vrchol i, že platí

$$i \notin col_L(s-1) \wedge i \in col_L(s) \tag{8.30}$$

z podmínky maximality S vzhledem k inkluzi, protože jinak by bylo možné supervrchol rozšířit o index $s-1$. To ale znamená, že s je listem i-tého řádkového podstromu $T_r(i)$ eliminačního stromu T a Věta je dokázána. ■

K tomu, aby se dala použít podmínka z Věty 8.6.2 k určení fundamentálních supervrcholů, stačí použít algoritmus pro nalezení listů řádkových podstromů uvedený dříve. Nalezení počátečních vrcholů supervrcholů společně s testováním podmínky z Věty 8.6.1 pak poskytne efektivní algoritmus hledání supervrcholů.

8.7 Počty nenulových prvků v řádcích a sloupcích faktoru L

Nalezení počtu prvků na řádcích a ve sloupcích Choleského faktoru jsme probírali již výše. Konkrétně, Algoritmus 8.3.1 počítá nenulové prvky na řádcích faktoru souběžně se strukturami řídkosti řádků a velmi jednoduše by mohl počítat také počty nenulových prvků ve sloupcích faktoru. Důvodem návratu k tématu v tomto textu není ani tak větší efektivnost jejich výpočtu, která je zřejmá, ale elegance celého postupu. Počítání prvků na řádcích a ve sloupcích faktoru L probereme samostatně a uvedeme pro přehlednost v samostatných schématech, ačkoli v praxi se více výpočtů z této kapitoly slučuje pro větší efektivnost v kombinovaných algoritmech.

8.7.1 Efektivní nalezení počtu prvků na řádcích L

V této sekci popíšeme algoritmus pro nalezení počtů mimodiagonálních nenulových prvků na řádcích faktoru L, který je efektivnější než Algoritmus 8.3.1. Zatímco v něm se explicitně procházely jednotlivé řádkové podstromy eliminačního stromu, což vedlo automaticky ke složitosti, ve které vystupoval člen s počtem nenulových prvků faktoru, nový algoritmus je založen pouze na sčítání délek cest v určitých rozkladech řádkových podstromů. Nejprve definujme pojem hloubky vrcholu v kořenovém stromu.

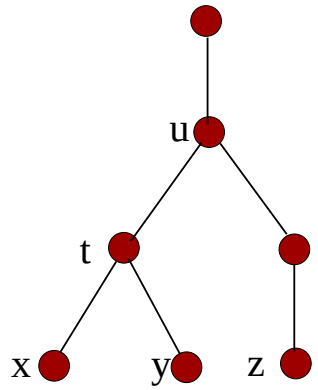
Definice 8.7.1 Hloubka *$depth(x)$ vrcholu x v kořenovém stromu s kořenem r je definována následujícím rekurentním vztahem.*

$$depth(x) = \begin{cases} 0 & pro\ x = r, \\ depth(parent(x)) + 1 & jinak. \end{cases} \tag{8.31}$$

Následující Algoritmus 8.7.1 počítá hloubky vrcholů v kořenovém a topologicky očíslovaném stromu. Topologické uspořádání je potřebné k tomu, aby byl algoritmus korektní.

Algoritmus 8.7.1 Algoritmus pro nalezení hloubek vrcholů depth v kořenovém a topologicky očíslovaném stromu určeném zobrazením parent.
Input: *Topologicky očíslovaný kořenový strom T.*
Output: *Hloubky vrcholů $depth(i)$ stromu T pro $i = 1, \ldots, |T|$*

1. $depth(|T|) = 0$
2. **for** $j = |T| - 1 : 1$ **do**
3. $\quad depth(j) = depth(parent(j)) + 1$
4. **end** j



Obrázek 8.7.29: *Příklad kořenového stromu, kde platí* $lca(x, y) = t$, $lca(x, t) = t$, $lca(x, z) = u$.

Pojem nejmenšího společného předka dvou vrcholů v obecném acyklickém grafu je zaveden v následující definici.

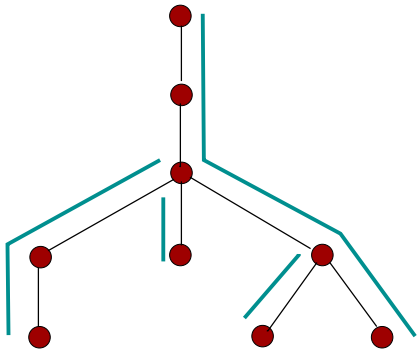
Definice 8.7.2 Nejmenší společný předek $lca(x, y)$ *dvou vrcholů* x *a* y *orientovaného acyklického grafu je takový jejich společný předek* $z \in anc(x) \cup anc(y)$, *od kterého je dělí v tomto grafu nejmenší vzdálenost. Obecně takový vrchol nemusí pro libovolné dva vrcholy existovat, je-li orientovaný acyklický graf nesouvislý. V kořenovém stromu takový vrchol vždy existuje a je jednoznačně určený, protože cesty od kořene ke všem jeho vrcholům jsou také jednoznačně určeny.*

Na následujícím obrázku je jednoduchý kořenový strom v jehož popisu uvádíme příklady nejmenších společných předků. Věta 8.7.1 ukazuje, že kořenový strom lze rozložit na množinu disjunktních cest a že v tomto rozkladu hrají roli některé jeho vrcholy a nejmenší společné předky jejich dvojic.

Věta 8.7.1 *Nechť* $P = \{p_1, p_2, \ldots, p_k\}$, $p_1 < p_2 < \ldots < p_k$ *jsou nějaké vrcholy kořenového stromu* R *očíslovaného postorderingem, které zahrnují všechny jeho listy a také jeho kořen. Nechť* $q_i = lca(p_i, p_{i+1})$ *jsou nejmenší společné předky vrcholů* p_i *a* p_{i+1} *pro* $1 \leq i < k$. *Pak je každá hrana* (t, s) *kořenového stromu* R *součástí* **právě jedné cesty** *z* q_j *do* p_j *pro nějaké* $1 \leq j < k$.

Důkaz: Nejprve dokážeme, že každá taková hrana (t, s) je součástí **alespoň** jedné cesty z tvrzení Věty. Nechť p_j je potomek vrcholu s z množiny P, který je ze všech takových potomků největší. Alespoň jeden takový potomek musí existovat, protože P obsahuje všechny listy kořenového stromu R. Navíc, protože s není kořen stromu R, existuje vrchol p_{j+1} v P. Podle definice postorderingu, protože p_{j+1} není potomek s, musí platit $p_j \leq s < p_{j+1}$. Nejmenší společný předek $q_j = lca(p_j, p_{j+1})$ vrcholů p_j a p_{j+1} musí pak být vlastní předek vrcholu s a tedy i předek vrcholu p_{j+1} a hrana (t, s) je tedy hranou cesty z q_j do p_j.

Nyní ukážeme, že (t, s) nemůže být na žádné jiné cestě z q_i do p_i pro nějaké $i \in \{1, \ldots, |P|\}$ různé od j. Jestliže $s \notin anc_R(p_i)$, pak je toto tvrzení zřejmé. Nechť $s \in$



Obrázek 8.7.30: *Rozklad kořenového stromu na cesty podle Věty 8.7.1. Cesty jsou znázorněny silnějšími spojnicemi vedle hran a vrcholů.*

$anc_R(p_i)$. Pak nutně platí $p_i \leq s$. Protože předpokládáme, že $i \neq j$, máme také $p_{i+1} \leq s$, protože p_j byl největší potomek s z množiny P. Vrchol p_{i+1} je proto také v podstromu $T(s)$ a tudíž $s \in anc_R(p_{i+1})$. Následně, $s \in anc_R(q_i)$ a (t, s) proto nemůže být na na cestě z q_i do p_i pro žádné i různé od výše určeného j, což jsme měli ukázat. ■

Rozklad kořenového stromu na cesty podle Věty 8.7.1, kde P obsahuje listy a kořen tohoto stromu je demonstrován na Obrázku 8.7.30. Následující Věta 8.7.2 ukazuje, jak se pomocí rozkladu kořenového stromu na cesty najdou počty mimodiagonálních prvků na řádcích $row_L(i)$, $i = 1, \ldots, n$.

Věta 8.7.2 *Nechť $P = \{p_1, p_2, \ldots, p_k\}$, $p_1 < p_2 < \ldots < p_k$ jsou indexy množiny vrcholů $ladj_{G(A)}(i) \cup \{i\}$, odpovídající nenulovým prvkům řádku dolní trojúhelníkové části matice A pro nějaké $i \in \{1, \ldots, n\}$. Předpokládejme, že eliminační strom $T(A)$ je očíslovaný postorderingem. Pak je počet $|row_L(i)|$ mimodiagonálních nenulových prvků na řádku i Choleského faktoru L dán vztahem*

$$|row_L(i)| = \sum_{i=1}^{k-1} \left(depth(p_i) - depth(lca(p_i, p_{i+1})) \right). \tag{8.32}$$

Důkaz:

Řádkové podstromy eliminačního stromu uspořádaného postorderingem lze podle Věty 8.7.1 rozložit na množinu disjunktních cest. Délka každé z nich je dána podle Definice 8.7.1 rozdílem hloubek jejích koncových vrcholů. Při sčítání délek podle tvrzení Věty je tedy každý vrchol, který odpovídá některému z uvažovaných nejmenších společných předků s výjimkou kořene, započítán právě jednou. A to tehdy, když není koncovým vrcholem nějaké cesty z rozkladu uvažovaného podstromu. Kořen řádkového podstromu započítán není, což odpovídá definici $|row_L(i)|$, kde není index diagonálního prvku. ■

Uvedený rozklad na cesty lze použít ke stanovení počtů mimodiagonálních prvků na řádcích Choleského faktoru L prohledáváním příslušných řádkových podstromů. Teď už ale nikoli tak, že bychom všechny cesty explicitně procházeli. K efektivní implementaci je zapotřebí efektivně hledat **listy řádkových podstromů** a zároveň nalézt **nejmenší**

společné předky některých jeho vrcholů. Efektivní nalezení listů řádkových podstromů v případě eliminačního stromu očíslovaného postorderingem jsme diskutovali výše. Nalezení nejmenších společných předků zmíníme ještě později. Následující Algoritmus 8.7.2 počítá počty mimodiagonálních nenulových prvků na řádcích faktoru L s využitím rozkladu řádkových podstromů eliminačního stromu přičemž listy těchto řádkových podstromů jsou počítány zároveň. Vstupem pro algoritmus není graf $G(A)$, ale graf $G(A) \cup T(A)$, který má stejný graf zaplnění. V grafu $G(A)$ je totiž eliminační strom pouze implicitně a to není vhodné pro přímočarou aplikaci Věty 8.7.1, kde je potřeba mít k dispozici všechny hrany řádkových podstromů explicitně. Pro jednoduchost dále předpokládáme, že graf matice $G(A)$ je souvislý.

Algoritmus 8.7.2 Algoritmus pro nalezení počtů mimodiagonálních nenulových prvků $|row_L(i)|$, $i = 1, \ldots, n$ na řádcích L.
Input: *Souvislý graf $G(A) \cup T(A) = (V, E)$, funkce $lca(.,.)$ nejmenšího společného předka, kterou zmíníme později.*
Output: *Počty mimodiagonálních nenulových prvků $|row_L(i)|$, $i = 1, \ldots, n$ faktoru L.*

```
1. for j = 1 : n do
2.     prev_leaf(j) = 0
3.     prev_nonz(j) = 0
4.     |row_L(j)| = 0
5. end j
6. for j = 1 : n do
7.     for i taková, že i > j ∧ j ∈ adj_G(A)(i) do
8.         k = prev_nonz(i)
9.         if k < j − |T(j)| + 1 then
10.            p = prev_leaf(i)
11.            if p = 0 then
12.                |row_L(i)| = |row_L(i)| + depth(j) − depth(i)
13.            else
14.                q = lca(p, j)
15.                |row_L(i)| = |row_L(i)| + depth(j) − depth(q)
16.            end if
17.            prev_leaf(i) = j
18.        end if
19.        prev_nonz(i) = j
20.    end i
21. end j
```

Algoritmus 8.7.2 prochází maticí A po sloupcích. Pro zaznamenání sloupcového indexu posledního dosud navštíveného nenulového prvku na řádku se používá pracovní pole *prev_nonz*. To nám slouží k tomu, abychom rozhodli, zdali je prvek listem řádkového podstromu testem podle Důsledku 8.5.1. Bezprostředně předcházející listy tohoto řádkového podstromu jsou ukládány pomocí pole *prev_leaf*. Tak jsou vždy k dispozici oba parametry funkce *lca*,které potřebujeme.

Poznámka 8.7.1 *Poznamenejme, že místo grafu $G(A) \cup T(A)$ je v praxi mnohem vhodnější vyjít z grafu $G(A)^- \cup T(A)$, kde $G(A)^-$ je eliminační skelet grafu $G(A)$. Ten obsahuje všechny listy řádkových podstromů a jeho eliminační strom je také $T(A)$.*

8.7.2 Efektivní nalezení počtu prvků ve sloupcích L

Stejně jako u algoritmu pro počty prvků v řádcích, i zde budeme diskutovat efektivnější postup než ten, který vznikne procházením řádkových podstromů jako v Algoritmu 8.3.1 a který vede na složitost s členem úměrným $|L|$. Definujme **rozdíly** $\delta(j)$ mezi veličinami $|col_L(j)|$ a $\sum_{j=parent(k)} |col_L(k)|$ následujícím vztahem

$$\delta(j) = |col_L(j)| - \sum_{j=parent(k)} |col_L(k)|, \tag{8.33}$$

který zachycuje informaci o **velikostech překryvů** sloupcových struktur vrcholu j a jeho synů. Budeme-li znát tyto rozdíly, pak počty mimodiagonálních nenulových prvků ve sloupcích faktoru L lze spočítat Algoritmem 8.7.3, kde předpokládáme topologického uspořádání vrcholů eliminačního stromu.

Algoritmus 8.7.3 Spočítání počtu prvků ve sloupcích faktoru $L\mathbf{R}^{|V|\times|V|}$ ze znalosti rozdílové funkce $\delta(j)$ definované výše.
Input: *Rozdílová funkce $\delta(j)$ mezi $|col_L(j)|$ a $\sum_{j=parent(k)} |col_L(k)|$ pro $j = 1, \ldots, |V|$.*
Output: *Počty prvků ve sloupcích faktoru L.*

1. **for** $j = 1 : |V|$ **do**
2. $\quad |col_L(j)| = \delta(j)$
3. **end** j
4. **for** $j = 1 : |V|$ **do**
5. $\quad |col_L(parent(j))| = |col_L(parent(j))| + \delta(j)$
6. **end** j

Otázkou je, jak tyto rozdíly spočítat. Možným postupem je složit její hodnotu z příspěvků jednotlivých řádkových podstromů $T_r(i)$, $i \in \{1, \ldots, |V|\}$. eliminačního stromu T.

Uvažujme rozdíl $\delta(j)$ pro nějaké $j \in \{1, \ldots, |V|\}$. Budeme ji inicializovat nulou. Jak uvidíme, to odpovídá tomu, že počítáme jen mimodiagonální prvky sloupců faktoru L. Rozlišme tři základní případy výpočtu aktualizace $\delta(j)$ na základě vztahu k řádkovým podstromům $T_r(i)$, $i = 1, \ldots, |V|$.

1. Pro $j \notin T_r(i)$ se $\delta(j)$ na základě vztahu k $T_r(i)$ nezmění, protože řádek i faktoru nepřispívá nenulovými prvky ani ke sloupci j, ani ke sloupcům žádného z jeho synů.

2. Je-li j list řádkového podstromu $T_r(i)$, pak $\delta(j)$ na základě vztahu k $T_r(i)$ vzroste o jedničku, protože list j nemá žádné syny v $T_r(i)$.

3. Nechť jsou vrchol j a d jeho synů obsaženy v $T_r(i)$. Pak je přírůstek k hodnotě $\delta(j)$ na základě vztahu k $T_r(i)$ roven

$$1 - d \equiv -(d-1).$$

Algoritmicky se tato změna při získávání rozdílů od listů směrem ke kořenům dá zachytit následujícím způsobem.

- Pro $d = 1$, to jest pro **interní vrcholy na cestách** v $T_r(i)$, se $\delta(j)$ při postupu těmito cestami směrem ke kořeni nezmění.
- Pro $d > 1$ platí, že vrchol j je **nejmenším společným předkem** právě $d - 1$ dvojic listů řádkového podstromu $T_r(i)$. Stačí tedy aktualizovat $\delta(j)$ odečtením jedničky pokaždé, když je vrchol j nejmenším společným předkem dvou různých listů kořenového podstromu.

Algoritmus 8.7.2 začleňuje naše úvahy o počítání sloupcových součtů pomocí rozdílů δ. Počítání nenulových prvků na řádcích i ve sloupcích se dá spojit do jednoho algoritmu, ale to zde pro větší názornost neděláme.

Algoritmus 8.7.4 Algoritmus pro nalezení počtů prvků ve sloupcích faktoru L. Funkce vyčíslení nejmenších společných předků $lca(p, q)$ vrcholů p a q je integrována pomocí množinových operací $find$ a $link$.
Input: *Graf $G(A) \cup T(A) = (V, E)$. Množinové operace $find$ a $link$.*
Output: *Rozdílová funkce zavedená výše.*

```
1. for j = 1 to |V| do
2.     prev_leaf(j) = 0
3.     prev_nonz(j) = 0
4.     δ(j) = 0
5. end j
6. for j = 1 to |V| do
7.     for i taková, že i > j ∧ a_ij ≠ 0 do
8.         k = prev_nonz(i)
9.         if k < j − |T(j)| + 1 then
10.            p = prev_leaf(i)
11.            δ(j) = δ(j) + 1
12.            if p ≠ 0 then
13.                q = find(p)
14.                δ(q) = δ(q) − 1
15.            end if
16.            prev_leaf(i) = j
17.        end if
18.        prev_nonz(i) = j
19.    end i
20.    link(j, parent(j))
21. end j
```

V Algoritmu 8.7.2 je operace nejmenšího společného předka $lca(.,.)$ implementována prostřednictvím množinových operací $find$ a $link$ postupně aplikovaných na počáteční množinu vrcholů $\{1, \ldots, |V|\}$. Její prvky jsou postupně propojovány hranami. V každém okamžiku tato množina představuje les, tedy acyklický a obecně nesouvislý graf na těchto vrcholech. Každá komponenta tohoto lesa má svého reprezentanta, kterým je její největší vrchol. Na počátku jsou tedy všechny vrcholy této množiny reprezentanty sebe sama. Operace $find(p)$ nalezne pro vrchol p reprezentativní vrchol podstromu, ve kterém tento vrchol je obsažen. Operace $link(x, y)$ zkombinuje množiny s reprezentativními vrcholy x a y do jedné, kde reprezentativní vrchol je větší z nich. Samotné propojení vrcholů v komponentách může být ale značně odlišné od samotného eliminačního stromu. Jednou z možností je například reprezentace s kompresí cest, kterou jsme uvedli při diskusi o konstrukci eliminačního stromu, ale způsobů může být mnohem víc, viz klasické publikace [106], [108], [80]. To, že operace $find(p)$ v Algoritmu najde nejmenšího společného předka vrcholů p a j vyplývá z toho, že oba dva vrcholy patří do $T_r(i)$ a zároveň do jedné komponenty vytvářeného lesa.

Lze ukázat, že složitost naposledy uvedených algoritmů pro počítání řádkových a sloupcových součtů je $O(m + m\alpha(m, n))$, což je výrazně méně než pro Algoritmus 8.3.1. Jak jsme již uvedli, tuto složitost můžeme dále zmenšit nahrazením grafu $G(A)$ jeho eliminačním skeletem $G^-(A)$. Výsledná složitost (pro nalezení jak řádkových tak sloupcových velikostí L) pak je $O(m + m^-\alpha(m^-, n))$, kde $m^- = |E(G^-(A))|$. To, že tento postup je efektivní ukazují například experimenty v [80].

9

Syntéza řídkého Choleského rozkladu

Tato kapitola navazuje na znalost některých základních komponent řídkého Choleského rozkladu z kapitoly 8 a také na diskusi o obecných schématech Choleského rozkladu a LU faktorizace jakožto variant Gaussovy eliminace v kapitole 7. V ní jsme uvedli několik možných variant Choleského rozkladu pro husté i řídké matice, které odpovídají různému pořadí cyklů generického schématu. Poté jsme diskutovali základní komponenty řídkého Choleského rozkladu, které pracují především se strukturami řídkosti původní matice i faktoru. V této kapitole na tento popis naváže uvedením několika variant řídkého Choleského rozkladu, kde už ale podrobnějí zmíníme výše probrané komponenty.

9.1 Obecné schéma řídkého Choleského rozkladu

Celkové schéma Choleského rozkladu lze zapsat ve třech krocích následujícím způsobem.

Algoritmus 9.1.1 Choleského rozklad řídké matice.
Input: *Řídká matice A.*
Output: *Choleského faktor L matice A.*

1. *Nalezení počátečního uspořádání matice*
2. *Symbolická faktorizace*
3. *Numerická faktorizace*

Počáteční uspořádání matice budeme diskutovat později souhrnně pro více druhů řídkých rozkladů. **Symbolická faktorizace** zahrnuje obvykle řadu komponent, které uvažují strukturu řídkosti a její změny, diskutovaných výše.

První dva kroky rozkladu se někdy nazývají souhrnně **analýza**. Třetí krok se někdy nazývá stručně **faktorizace (rozklad)** a bere do úvahy numerické hodnoty matice. Jednotlivé kroky nyní popíšeme podrobněji s tím, že obecný postup mívá více alternativ podle zvolené varianty rozkladu.

$$\begin{pmatrix} * & * & * & * & * \\ * & * & & & \\ * & & * & & \\ * & & & * & \\ * & & & & * \end{pmatrix} \qquad \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \\ * & * & * & * & * \end{pmatrix}$$

Obrázek 9.1.1: *Příklad dvou šípových matic.*

$$\begin{pmatrix} * & * & * & * & * \\ * & * & f & f & f \\ * & f & * & f & f \\ * & f & f & * & f \\ * & f & f & f & * \end{pmatrix} \qquad \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \\ * & * & * & * & * \end{pmatrix}$$

Obrázek 9.1.2: *Příklad zaplnění ve dvou šípových maticích z Obrázku 9.1.1. Nové prvky zaplnění jsou označeny symbolem f.*

9.1.1 Nalezení počátečního uspořádání

Prvním krokem řídké Choleského rozkladu je nalezení **počáteční uspořádání** matice. Toto uspořádání odlišujeme od jiných postupů, jako je například výše zmíněné uspořádání matice postorderingem (indukované jejím eliminačním stromem). Motivace pro volbu konkrétního druhu počátečního uspořádání se mohou lišit, ale vždy je za nimi snaha **zmenšit velikost zaplnění** v rozkladu. Na Obrázku 9.1.1 jsou znázorněny struktury řídkosti dvou matic. A na Obrázku 9.1.2 je navíc znázorněno vzniklé zaplnění.

Zatímco první matice na Obrázku 9.1.1 se v průběhu rozkladu úplně zaplní, druhá z nich svou strukturu nezmění. Mezi strukturami řídkosti obou matic je ale úzký vztah. Označíme-li A matici se strukturou řídkosti z Obrázku 9.1.1 vlevo, pak matice vpravo na tomto obrázku vpravo má strukturu řídkosti permutované matice $P^T AP$ s permutační maticí

$$P = \begin{pmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}.$$

Vidíme tedy, že počáteční permutací matice je možné velikost zaplnění výrazně ovlivnit a příslušným technikám věnujeme v tomto textu celou kapitolu. U obecnějších rozkladů přitom uspořádání musí být volena tak, aby vyhovala obvykle kombinovanému kritériu, které uvažuje nejenom velikost zaplnění.

9.1.2 Symbolická faktorizace v širším smyslu

Druhý krok rozkladu obvykle slouží k **nalezení základních algoritmických a implementačních struktur**. Jeho základem je obvykle grafový model struktury řídkosti matice a jejích faktorů. Využívá přitom výše popsané algoritmy, jako jsou hledání eliminačního stromu, nalezení struktury řídkosti Choleského faktoru, nalezení počtů nenulových prvků na řádcích a ve sloupcích faktoru, přečíslování vrcholů eliminačního stromu na základě eliminačního stromu či nalezení supervrcholů. Souhrnně se celý tento druhý krok nazývá **symbolická faktorizace**, byť se symbolickou faktorizací historicky mínilo pouze nalezení sloupcové struktury faktoru užitím Algoritmu 8.3.2 založeném na vztahu (8.20).

9.1.3 Numerická faktorizace

Posledním krokem schématu je **numerická faktorizace**, jejímž obsahem je nalezení numerických hodnot prvků faktorů rozkladu. Ve smyslu výše uvedeného textu nebudeme dále diskutovat rozdíl mezi odmocninovým a bezodmocninovým rozkladem a budeme předpokládat, že rozklad existuje, což je splněno pro matice symetrické a pozitivně definitní, ale může být splněno i pro mnohé obecně symetrické matice. Obecně tedy vyžadujeme silnou regularitu matice pro symetrický rozklad. Související problémy stability rozkladu budeme diskutovat později.

Základní schémata numerické faktorizace založená na řádkovém přístupu, sloupcovém přístupu či podmaticovém přístupu jsme probrali výše. Každý z nich může vyžadovat výpočet jiné množiny prvků ze symbolické faktorizace z minulé kapitoly. Některé základní algoritmy nyní popíšeme podrobněji.

9.2 Supervrcholová sloupcová Choleského faktorizace

Tento postup označuje sloupcový algoritmus uvedený výše, kde jsou symbolické kroky propojeny s hledáním supervrcholů. Po nalezení eliminačního stromu, jeho postorderingu a nalezení supervrcholů je celý sloupcový algoritmus založen na blocích. V terminologii numerické lineární algebry to znamená, že operace s numerickými hodnotami budou založeny na operacích s maticemi, které vedou k vysokému výpočetnímu výkonu použitím aritmetiky knihovny BLAS3.

Sloupcový algoritmus Choleského faktorizace uvedený výše v každém kroku počítá sloupec faktoru, v našem případě blokový sloupce faktoru, pomocí lineární kombinace předcházejících blokových sloupců. V dalším textu nebudeme fakt, že se samotná aritmetika rozkladu týká bloků, příliš zdůrazňovat.

Algoritmus 9.2.1 Supervrcholová sloupcová Choleského faktorizace.
Input: *Řídká matice A.*
Output: *Choleského faktor L matice A.*

1. *Nalezení eliminačního stromu*
2. *Nalezení postorderingu*
3. *Kombinace počátečního uspořádání*

4. Nalezení počtu prvků ve sloupcích
5. Optimalizace postorderingu
6. Nalezení supervrcholů, jejich optimalizace a odhad velikosti pracovního prostoru pro blokový rozklad
7. Nalezení sloupcové struktury faktoru L pro supervrcholový rozklad
8. Supervrcholová **numerická** *faktorizace*

Pro efektivní implementaci řídkého rozkladu potřebujeme znát strukturu nenulových prvků sloupce, protože po sloupcích faktor počítáme. Tu nám poskytne proces symbolické faktorizace v užším smyslu, jak jsme o ní hovořili výše, tedy Algoritmus 8.3.2, kde je v grafové terminologii hlavním principem **slévat struktury řídkosti synů vrcholů**. Pro poslední krok Algoritmu **??** ale také potřebujeme znát, které prvky řádku Choleského faktoru vystupují v sumě (7.22), jinými slovy, potřebujeme znát struktury řídkosti řádků Choleského faktoru. Jak vidíme z následující Věty 9.2.1, počet prvků v řádcích Choleského faktoru ve sloupcovém algoritmu určuje počet aktualizací právě počítaného sloupce některým z dříve vypočtených sloupců faktoru tak, jak je vidět z následující Věty 9.2.1.

Věta 9.2.1 *Nechť $j > k$. Numerické hodnoty sloupce L_{*j} závisí na hodnotách sloupce L_{*k} právě tehdy, je-li $l_{jk} \neq 0$.*

Důkaz: Tvrzení plyne ze sloupcové formulace Choleského faktorizace. Hlavní úprava v tomto algoritmu je totiž dána vztahem

$$A_{*j} = A_{*j} - \sum_{k<j} l_{jk} A_{*k}.$$

Algoritmus 7.2.1 pak upřesňuje počet netriviálních úprav. ■

Základní metoda pro zjištění nenulových prvků v nějakém i-tém řádku, spočívá ve výpočtu s použitím řádkového podstromu $T_r(i)$ eliminačního stromu. Ten můžeme zjistit přímo z eliminačního stromu či během samotné numerické faktorizace procházením cest mezi vrcholy množiny $adj_{G^-(A)}(i)$ nebo $adj_{G(A)}(i)$ a vrcholem i v T [105]. V případě sloupcového algoritmu je ale možné použít ještě jednodušší postup, který je ve své obecnosti zároveň historicky nejstarší a v praxi je stále nejběžnější. Tento postup je založen na metodě uvedené v programovém souboru YSMP (Yale Sparse Matrix Package) a byl poté použit v celé řadě dalších implementací. Vlastní algoritmus potřebuje, aby nenulové prvky sloupců faktoru L byly uspořádané podle vzrůstajícího řádkového indexu, což lze v rámci rozkladu jednoduše zařídit. Postup nalezení řádků je pak velmi podobný simulaci řádkového přístupu do matice A v Algoritmu 17.3.2. V průběhu algoritmu procházíme souběžně sloupce vytvářeného faktoru L tak, že pro každý z nich je v pracovním poli *start* index prvního nenulového prvku sloupce, který ještě nebyl prvkem l_{jk} nějaké řádkové struktury, přes kterou sčítáme v (7.22). Všechny tyto sloupcové indexy jsou konkrétní řádky uloženy pomocí spojových seznamů, které obsahují vždy pouze první dosud nepoužitý index ze sloupce. Tyto seznamy jsou průběžně aktualizovány. Tyto spojové seznamy se dají všechny uložit do **jednoho pole** délky n, protože každý sloupcový první

nepoužitý index může být v těchto spojových seznamech nejvýše jednou. Každý krok sloupcové faktorizace, spočítá sloupec j faktoru podle (7.22) a pak jednoduše projde ten spojový seznam, ve kterém je index j, a provede jeho aktualizaci nahrazením použitých indexů nejbližšími dalšími. Konkrétně, spojový seznam, ve kterém je sloupec j je v tomto vnějším kroku Algoritmu 7.2.1 aktualizován dokud existuje nějaký následník v příslušném sloupci. Také řádkový index následníka vrcholu j v j-tém sloupci, tedy index $parent(j)$, je přidán do spojového seznamu následníků tohoto řádku a je incializována komponenta $start(i)$.

9.3 Multifrontální metoda

V této sekci probereme velmi významný postup, který je založen na podmaticovém algoritmu Choleského rozkladu a který se nazývá multifrontální metoda. Tato metoda není jediným postupem založeným na podmaticové variantě rozkladu, ale je jednoznačně nejvíce prozkoumanou a nejpoužívanější metodou Choleského rozkladu.

Stejně jako v předcházející podkapitole této kapitoly i zde algoritmus Choleského rozkladu symetrické a silně regulární matice $A \in R^{n\times n}$ přepíšeme do tvaru, kde využíváme maticové a vektorové operace. Jeden krok podmaticového algoritmu můžeme schématicky přepsat následujícím způsobem

$$\begin{aligned}
A &= \begin{pmatrix} a_{11} & v \\ v^T & C \end{pmatrix} \\
&= \begin{pmatrix} \sqrt{a_{11}} & \\ v^T/\sqrt{a_{11}} & I \end{pmatrix} \begin{pmatrix} I & \\ & C - v^T v/a_{11} \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & v/\sqrt{a_{11}} \\ & I \end{pmatrix} \\
&= \begin{pmatrix} \sqrt{a_{11}} & \\ v^T/\sqrt{a_{11}} & I \end{pmatrix} \begin{pmatrix} I & \\ & S \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & v/\sqrt{a_{11}} \\ & I \end{pmatrix} \\
&= \begin{pmatrix} \sqrt{a_{11}} & \\ v^T/\sqrt{a_{11}} & L_S \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & v/\sqrt{a_{11}} \\ & L_S^T \end{pmatrix},
\end{aligned}$$

kde matice S, která je Schurovým doplňkem matice A vzhledem k podmatici $A_{11} \equiv a_{11}$. Tato matice je upravena **vnějším součinem** vektorů v^T a v škálovaným inverzí diagonálního prvku a_{11} a může být dále rozložena dalšími kroky Choleského rozkladu. Všimněme si ale, že první sloupec už je definitivně spočítaný. Analogicky můžeme napsat situaci po $k-1$ krocích podmaticového algoritmu, kde vlastně ukazujeme, jak vzniká obecný Schurův doplněk

$$\begin{aligned}
A &= \begin{pmatrix} A_{11} & V \\ V^T & C \end{pmatrix} \\
&= \begin{pmatrix} L_{11} & \\ V L_{11}^{-T} & I \end{pmatrix} \begin{pmatrix} I & \\ & C - V^T L_{11}^{-T} L_{11}^{-1} V \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{11}^{-1} V \\ & I \end{pmatrix}.
\end{aligned}$$

Podstatné pozorování je, že výsledek je týž, když budeme příslušný Schurův doplněk počítat přímo touto maticovou operací nebo postupně v $j-1$ krocích a můžeme tedy psát

(v zápise prvků využíváme symetrie)

$$C - V^T L_{11}^{-T} L_{11}^{-1} V = C - \sum_{k=1}^{j-1} \begin{pmatrix} l_{j\,k} \\ l_{j+1,k} \\ \vdots \\ l_{n\,k} \end{pmatrix} \begin{pmatrix} l_{j\,k} & l_{j+1,k} & \dots & l_{n\,k} \end{pmatrix} \tag{9.1}$$

Stejně jako ve speciálním případě už je část faktoru, v tomto případě jeho prvních $j-1$ sloupců, definitivně spočítána. Diskutujme nyní obecný j-tý sloupec, který bude v tomto kroku definitivně spočítán. Strukturu tohoto sloupce jsme vyjádřili pomocí podstromu eliminačního stromu $T(j)$ ve Větě 8.3.2 ve tvaru

$$col_L(j) = adj_{G(A)}(T(j)). \tag{9.2}$$

Označme si nyní prvky struktury řídkosti j-tého sloupce faktoru L podrobněji vztahem

$$\mathcal{S}(L_{*j}) \equiv col_L(j) \cup \{j\} = \{j, i_1, \dots, i_r\}. \tag{9.3}$$

Frontální matice, kterou zavedeme v následující definici, je přesně ta matice, ve které po j-tém kroku rozkladu vznikne j-tý sloupec Choleského rozkladu jako její první sloupec. Dále uvidíme, že její zbylá část bezprostředně vystupuje v dalších krocích rozkladu.

Definice 9.3.1 Frontální matici $\mathcal{F}_j$ *j-tého podstromu $T(j)$ eliminačního stromu T definujeme následovně.*

$$\begin{aligned} \mathcal{F}_j &= \begin{pmatrix} a_{jj} & a_{i_1 j} & \dots & a_{i_r j} \\ a_{i_1 j} & & & \\ \vdots & & 0 & \\ a_{i_r j} & & & \end{pmatrix} - \sum_{k \in T(j) \setminus \{j\}} \begin{pmatrix} l_{jk} \\ l_{i_1 k} \\ \vdots \\ l_{i_r k} \end{pmatrix} \begin{pmatrix} l_{jk} & l_{i_1 k} & \dots & l_{i_r k} \end{pmatrix} \\ &\equiv \begin{pmatrix} a_{jj} & a_{i_1 j} & \dots & a_{i_r j} \\ a_{i_1 j} & & & \\ \vdots & & 0 & \\ a_{i_r j} & & & \end{pmatrix} + \bar{\mathcal{U}}_j \end{aligned}$$

Frontální matice $\mathcal{F}_j$ je tedy složena ze dvou částí. Její první částí je podmatice určená řádkem a sloupcem matice A (používáme indexy $i_1, \dots, i_r$ definované pomocí $col_L(j)$, take některé prvky mohou být nulové), druhou částí je souhrn všech dosavadních předcházejících aktualizací této podmatice prvky faktoru. Uvažujme teď provedení j-tého kroku Choleského rozkladu, po kterém je definitivně spočítán j-tý sloupec Choleského faktoru. Toto provedení rozloží příslušnou frontální matici s aktualizacemi akumulovanými z řádků a sloupců, které odpovídají $T(j) \setminus \{j\}$.

$$\mathcal{F}_j = \begin{pmatrix} l_{jj} & \dots \\ l_{i_1 j} & \\ \vdots & I_r \\ l_{i_r j} & \end{pmatrix} \begin{pmatrix} 1 & \\ & \mathcal{U}_j \end{pmatrix} \begin{pmatrix} l_{jj} & l_{i_1 j} & \dots & l_{i_r j} \\ & & I_r & \end{pmatrix} \tag{9.4}$$

Následující tvrzení se vztahuje k tvaru frontální matice po jednom kroku rozkladu, jak jej máme uveden v (9.4).

Věta 9.3.1 *Platí následující vztah*

$$\mathcal{U}_j = -\sum_{k \in T(j)} \begin{pmatrix} l_{i_1 k} \\ \vdots \\ l_{i_r k} \end{pmatrix} \begin{pmatrix} l_{i_1 k} & \dots & l_{i_r k} \end{pmatrix}. \tag{9.5}$$

Důkaz: Vztah (9.4) si můžeme přepsat na následující tvar

$$\mathcal{F}_j = \begin{pmatrix} l_{jj} \\ l_{i_1,j} \\ \vdots \\ l_{i_r,j} \end{pmatrix} \begin{pmatrix} l_{jj} & l_{i_1,j} & \dots & l_{i_r,j} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & U_j \end{pmatrix} \tag{9.6}$$

Porovnejme teď levou a pravou stranu rovnosti (9.6) po vynechání prvního řádku a sloupce. Na levou stranu přitom použijeme její definici a získáme

$$-\sum_{k \in T(j)\setminus\{j\}} \begin{pmatrix} l_{i_1 k} \\ \vdots \\ l_{i_r k} \end{pmatrix} \begin{pmatrix} l_{i_1 k} & \dots & l_{i_r k} \end{pmatrix} = \begin{pmatrix} l_{i_1,j} \\ \vdots \\ l_{i_r,j} \end{pmatrix} \begin{pmatrix} l_{i_1,j} & \dots & l_{i_r,j} \end{pmatrix} + \mathcal{U}_j, \tag{9.7}$$

což poskytuje výsledek. ■

Zdůrazněme různou roli matic $\mathcal{U}_j$ a $\bar{\mathcal{U}}_j$ pro $j = 1, \dots, n$. Matice $\mathcal{U}_j$, která se nazývá také **aktualizační matice** multifrontální metody, obsahuje všechny příspěvky od řádků a sloupců, které ovlivňují výpočet **následujících řádků a sloupců** s indexy většími než j, tedy konkrétně právě těch, které jsou v $col_L(j)$. Matice $\bar{\mathcal{U}}_j$ pak slouží k akumulaci těchto aktualizačních matic a po přidání příslušného řádku a sloupce matice A slouží k výpočtu j-tého sloupce. V multifrontální metodě využijeme vztahu mezi frontálními a aktualizačními maticemi, ale nejprve zavedeme formální popis kombinování matic, určenými svými řádkovými a sloupcovými indexy pomocí nové asociativní operace **rozšířeného součtu** řídkých matic.

Definice 9.3.2 *Nechť $B \in \mathbf{R}^{k \times k}$ a $C \in \mathbf{R}^{l \times l}$ jsou podmatice nějakých matic z $\mathbf{R}^{n \times n}$ určené, po řadě, indexovými množinami jejich řádků a sloupců $\mathcal{K} = \{\kappa_1 < \dots < \kappa_{|\mathcal{K}|}\}$ a $\mathcal{L} = \{\lambda_1 < \dots < \lambda_{|\mathcal{L}|}\}$, kde $|\mathcal{K}| = k$ a $|\mathcal{L}| = l$.* **Rozšířeným součtem** *těchto podmatic označeným $A \diamond B$ nazveme matici $E \in \mathbf{R}^{p \times p}$, která je podmaticí matice z $R^{n \times n}$, je určena indexovou množinou $\mathcal{K} \cup \mathcal{L}$ svých řádků a sloupců, kde $|\mathcal{K} \cup \mathcal{L}| = p$ a tato indexová množina je opět vzestupně uspořádána. Její jednotlivé prvky se získají součtem odpovídajících prvků (prvků se stejnými řádkovými a sloupcovými indexy) v B a C, pro indexy, které jsou jak v $\mathcal{K}$ tak i v $\mathcal{L}$ a jednotlivými prvky z B nebo C v opačném případě.*

Následující příklad ukazuje rozšířený součet podmatic B a C.

Příklad 9.3.1 *Obrázek 9.3.3 znázorňuje dvě podmatice matice z $\mathbf{R}^{9 \times 9}$ a jejich rozšířený součet.*

$$B = \begin{array}{c} 2 \\ 3 \\ 5 \end{array} \begin{pmatrix} 1.1 & 1.1 & 2.2 \\ 1.1 & 1.1 & 2.2 \\ 2.2 & 2.2 & 3.3 \end{pmatrix} \quad C = \begin{array}{c} 1 \\ 3 \\ 5 \\ 9 \end{array} \begin{pmatrix} 1.1 & 1.1 & 2.2 & 1.1 \\ 1.1 & 1.1 & 3.3 & 1.1 \\ 2.2 & 3.3 & 1.1 & 4.4 \\ 1.1 & 1.1 & 4.4 & 1.1 \end{pmatrix}$$

$$E = \begin{array}{c} 1 \\ 2 \\ 3 \\ 5 \\ 9 \end{array} \begin{pmatrix} 1.1 & & 1.1 & 2.2 & 1.1 \\ & 1.1 & 1.1 & 2.2 & \\ 1.1 & 1.1 & 2.2 & 5.5 & 1.1 \\ 2.2 & 2.2 & 5.5 & 4.4 & 4.4 \\ 1.1 & & 1.1 & 4.4 & 1.1 \end{pmatrix}$$

Obrázek 9.3.3: *Příklad dvou podmatic, B a C, nějakých matic z $\mathbf{R}^{9\times 9}$ a jejich rozšířeného součtu E. Indexy řádků vymezující podmatice B, C a E na obrázcích, indexy sloupců jsou samozřejmě stejné.*

Věta 9.3.2 ukazuje zmíněnou souvislost mezi frontální maticí a aktualizačními maticemi $\mathcal{U}_k$ podrobněji s využitím struktury eliminačního stromu. Konkrétně se zde říká, že v součtu ve výrazu (9.4) **stačí uvažovat pouze několik sčítanců**. Větu 9.3.2 můžeme také chápat jako konkrétní zobecnění symbolické faktorizace, kde ale zároveň pracujeme s numerickými hodnotami rozkladu.

Věta 9.3.2 *Nechť $c_1, \ldots, c_s$ jsou indexy synů vrcholu j eliminačního stromu T. Pak platí:*

$$\mathcal{F}_j = \begin{pmatrix} a_{jj} & a_{ji_1} & \ldots & a_{ji_r} \\ a_{i_1 j} & & & \\ \ldots & & 0 & \\ a_{i_r j} & & & \end{pmatrix} \diamond \mathcal{U}_{c_1} \diamond \ldots \diamond \mathcal{U}_{c_s}. \tag{9.8}$$

Důkaz: Frontální matice $\mathcal{F}_j$ byla definována pomocí matice $\bar{\mathcal{U}}_j$, která zahrnuje všechny aktualizace od sloupců v $T(j) \setminus \{j\}$. Množina $T(j) \setminus \{j\}$ je ale **disjunktní** sjednocení množin $T(c_1), \ldots, T(c_s)$ a $\bar{U}_j$ může být vyjádřena jako sjednocení aktualizací od podstromů $T(c_1), \ldots, T(c_s)$. Podle Věty 9.5 jsou tyto aktualizace rovny aktualizačním maticím $\mathcal{U}_{c_1}, \ldots, \mathcal{U}_{c_s}$, což ve formalismu rozšířeného součtu dává tvrzení této věty. ■

Následující Algoritmus 9.3 shrnuje celý postup.

Algoritmus 9.3.1 Algoritmus multifrontální metody Choleského faktorizace.
Input: *Řídká matice A.*
Output: *Choleského faktor L matice A.*

1. **for** $j = 1 : n$ **do**
2. *Nechť $\mathcal{S}(L_{*j}) = \{j, i_1, \ldots, i_r\}$*
3. *Nechť $c_1, \ldots, c_s$ jsou synové j v T*
4. *Vytvoř $\bar{\mathcal{U}} = \mathcal{U}_{c_1} \diamond \ldots \diamond \mathcal{U}_{c_s}$*
3. *Definuj*

$$\mathcal{F}_j = \begin{pmatrix} a_{jj} & a_{ji_1} & \dots & a_{ji_r} \\ a_{i_1 j} & & & \\ \dots & & 0 & \\ a_{i_r j} & & & \end{pmatrix} \diamond \bar{U}$$

3. *Definuj aktualizační matici $\mathcal{U}_j$ jedním krokem LDL^T rozkladu:*

$$\mathcal{F}_j = \begin{pmatrix} l_{jj} & \dots \\ l_{i_1 j} & \\ \vdots & I_r \\ l_{i_r j} & \end{pmatrix} \begin{pmatrix} 1 & \\ & \mathcal{U}_j \end{pmatrix} \begin{pmatrix} l_{jj} & l_{i_1 j} & \dots & l_{i_r j} \\ & & I_r & \end{pmatrix} \tag{9.9}$$

Tím získáme jeden sloupec faktoru L a příslušný prvek diagonální matice.

4. **end** j

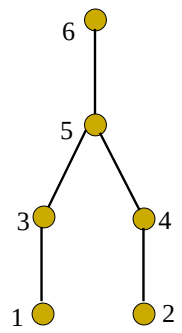
Stejně jako v případě Algoritmu 8.3.2 i Algoritmus funguje správně pro **libovolné topologické** pořadí, tedy i v pořadí $1, \dots, n$, ve kterém byl vytvořen eliminační strom. Pak jsou totiž všechny aktualizační matice v jeho kroku 4 dispozici pro další výpočet. K tomu, aby celý postup byl ještě efektivnější, je ale vhodné, aby tyto matice byly v algoritmu také **rychle dostupné** v následujícím smyslu: Protože uchovávání více frontálních matic může vyžadovat velké množství prostoru, je vhodné, aby potřebné aktualizační matice nebyly umístěny někde hluboko v použitém pracovním prostoru, ale aby byly okamžitě k dispozici. Tvrzení Věty 9.3.3 dává návod, jak toto umožnit.

Věta 9.3.3 *Nechť je eliminační strom matice A v multifrontální metodě uspořádán* **postorderingem** *a předpokládejme, že po spočítání j-tého sloupce z frontální matice $\mathcal{F}_j$ odložíme aktualizační matici $\mathcal{U}_j$ pro další použití do* **zásobníku**. *(S výjimkou $j \equiv n$ musí být tyto matice nenulové* **pro nerozložitelné** *A). Pak platí, že matice $\mathcal{U}_j$ potřebné ke konstrukci $\mathcal{F}_j$ leží vždy navrchu tohoto zásobníku.*

Důkaz: Tvrzení je zřejmé z definice postorderingu, pro který platí, že vrcholy v každém podstromu tvoří interval. Z toho vyplývá, že při tvorbě frontální matice $\mathcal{F}_j$ budeme používat ty právě ty matice $\mathcal{U}_j$ potomků vrcholu j, které byly odloženy jako dosud poslední. ■

Demonstrujme si multifrontální metodu na příkladu. Uvažujme následující matici znázorněnou včetně prvků zaplnění a její eliminační strom

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} * & & * & & * & \\ & * & & * & & * \\ * & & * & & * & \\ & * & & * & * & f \\ * & & * & * & * & f \\ & * & & f & f & * \end{pmatrix} \end{array}$$



Multifrontální metoda postupně generuje následující posloupnost frontálních a aktualizačních matic.

$$\mathcal{F}_1 = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \end{array}, \quad \mathcal{U}_1 = \begin{array}{c} \\ 3 \\ 5 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 3 & 5 \end{array} \\ \begin{pmatrix} * & * \\ * & * \end{pmatrix} \end{array},$$

$$\mathcal{F}_2 = \begin{array}{c} \\ 2 \\ 4 \\ 6 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 2 & 4 & 6 \end{array} \\ \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \end{array}, \quad \mathcal{U}_2 = \begin{array}{c} \\ 4 \\ 6 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 4 & 6 \end{array} \\ \begin{pmatrix} * & * \\ * & * \end{pmatrix} \end{array},$$

$$\mathcal{F}_3 = \begin{array}{c} \\ 3 \\ 5 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 3 & 5 \end{array} \\ \begin{pmatrix} * & * \\ * & * \end{pmatrix} \end{array} \oplus \mathcal{U}_1, \; \mathcal{U}_3 = \begin{array}{c} \\ 5 \end{array}\!\!\begin{array}{c} 5 \\ \begin{pmatrix} * \end{pmatrix} \end{array}.$$

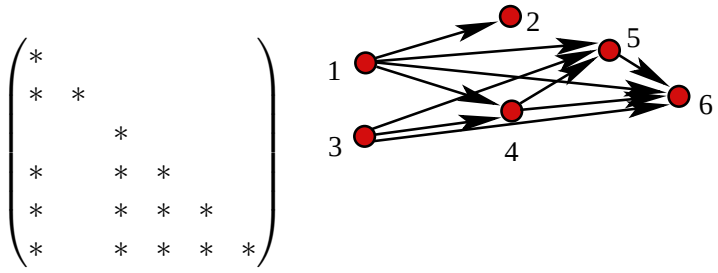
Na příkladu vidíme a můžeme si zopakovat, že

- První řádky a sloupce frontálních matic $\mathcal{F}_k$ odpovídají nenulám v L_{*k}. Aktualizační matice $\mathcal{U}_k$ přispívá do $\mathcal{F}_{parent(k)}$. To je první frontální matice, do které $\mathcal{U}_k$ přispívá. Do žádných matic s indexy ostře mezi k a $parent(k)$ nepřispívá.

- Celá matice $\mathcal{U}_k$ se tedy stává součástí frontální matice $\mathcal{F}_{parent(k)}$.

- Je-li matice přeuspořádaná podle nějakého postorderingu eliminačního stromu matice A, pak všechny matice $\mathcal{U}_k$ mohou být uloženy v zásobníku. Potřebné aktualizační matice budou vždycky navrchu zásobníku.

Dalším význačným rysem multifrontální metody je použití **hustých matic** ve výpočtu. To, společně s použitím supervrcholů, zabezpečuje velkou efektivitu rozkladu na soudobých počítačových architekturách.



Obrázek 9.4.4: *Příklad dolní trojúhelníkové matice a orientovaného grafu $G(L^T)$ její transpozice.*

9.4 Řádkový Choleského rozklad

Tento rozklad je poskytuje faktor L po řádcích. Pro implementaci Algoritmu 7.2.2 potřebujeme nejprve stejně jako ve sloupcovém algoritmu spočítat velikost faktoru L.

Jakkoli je samotný algoritmus jednoduchý, základní problém řádkového Choleského rozkladu je v jeho efektivnosti, kterou budeme v této sekci komentovat. Sloupcový supervrcholový algoritmus uvedený v předcházející sekci vyžaduje v každém kroku pouze úpravu pomocí lineární kombinace některých z předcházejících sloupců. To je postup dobře implementovatelný na moderních počítačových architekturách a snadno rozšířitelný například efektivní blokovou verzi, kterou jsme výše předpokládali. Řádková struktura řídkosti použitá ve sloupcovém algoritmu se pak dá jednoduše odvodit ze znalosti sloupcových struktur řídkosti.

V případě řádkového algoritmu nezbývá, než tuto strukturu spočítat samostatně. Jedna možnost je použít řádkové podstromy eliminačního stromu. Další možností je použít pro hledání řádkových struktur následující postup, který je založen na hledání řešení soustavy s trojúhelníkovou maticí. Řádkový algoritmus v sobě totiž obsahuje **řešení soustav s trojúhelníkovou a stále se zvětšující maticí**. Předpokládejme, že $L \in R^{n\times n}$ je regulární matice v dolním trojúhelníkovém tvaru. Příklad takové matice a orientovaného acyklického grafového modelu její transpozice, je na Obrázku 9.4.4.

Věta 9.4.1 je speciálním případem obecného tvrzení Věty 5.1 z [78].

Věta 9.4.1 *Nechť $L = (l_{ij})_{i,j\in\{1,\dots,n\}} \in R^{n\times n}$ je regulární dolní trojúhelníková matice a $b = (b_i)_{i\in\{1,\dots,n\}} \in R^n$. Za předpokladu o nevyrušení při řešení soustavy rovnic*

$$Lx = b$$

pro $x = (x_i)_{i\in\{1,\dots,n\}} \in R^n$ platí $x_i \neq 0$ právě tehdy, jestliže v grafu $G(L^T)$ existuje cesta $j \Rightarrow i$ z vrcholu $j \in \{1,\dots,n\}, j < i$, pro který $b_j \neq 0$, to jest, právě tehdy jestliže platí

$$i \in Reach(\{j \mid b_j \neq 0\}, V(G(L)), G(L^T)).$$

Důkaz: Důkaz proveďme matematickou indukcí podle indexu $k \in \{1,\dots,n\}$ odpovídající složce řešení x_k. Pro $k = 1$ je tvrzení zřejmé. Předpokládejme, že tvrzení platí pro všechna $i < k$, kde $k \in \{1,\dots,n-1\}$ a uvažujme složku x_k řešení x. Regulární matice L má nenulové diagonální prvky a složka x_k je tedy nenulová právě tehdy, je-li nenulový výraz

$$b_k - L_{k,1:k-1} L^{-1}_{1:k-1,1:k-1} b_{1:k-1} \equiv b_k - L_{k,1:k-1}(x_1, \ldots, x_{k-1})^T, \tag{9.10}$$

jak je vidět z následující rovnosti

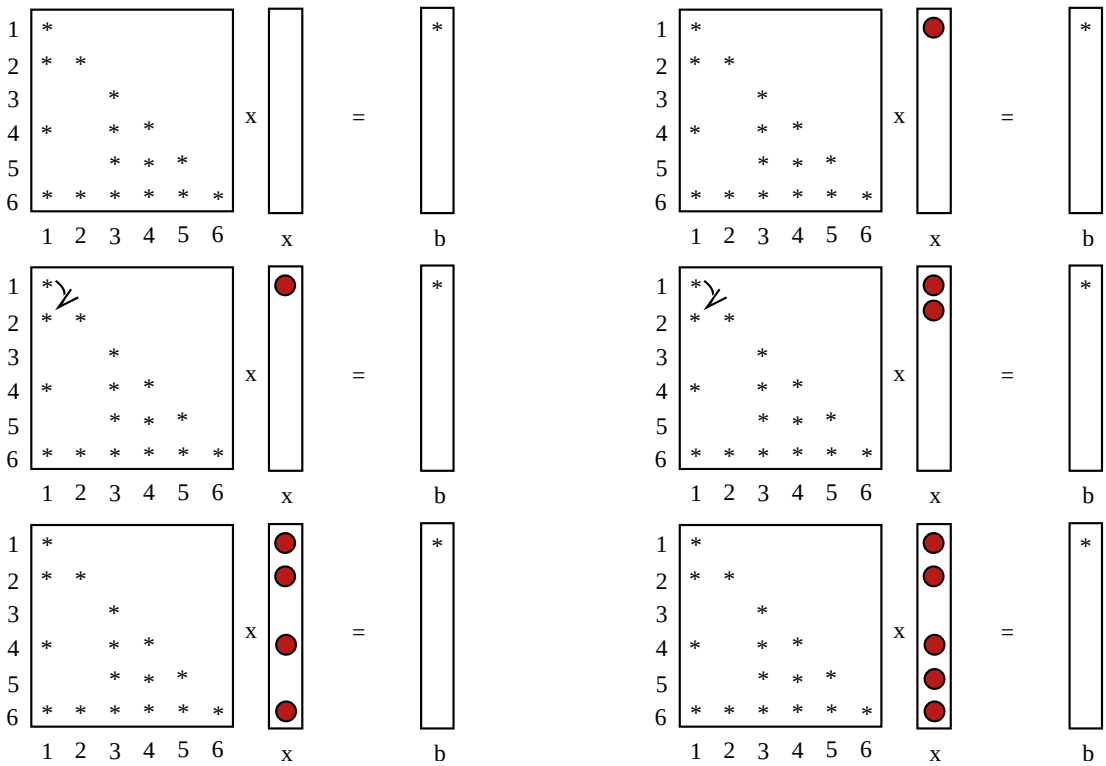
$$\begin{pmatrix} L_{1:k-1,1:k-1} & \\ L_{k,1:k-1} & l_{k,k} \end{pmatrix} \begin{pmatrix} x_{1:k-1} \\ x_k \end{pmatrix} = \begin{pmatrix} b_{1:k-1} \\ b_k \end{pmatrix} \tag{9.11}$$

Neboť předpokládáme nevyrušení nenulových prvků, pak výraz (9.10) bude nenulový právě tehdy, je-li buď $b_k \neq 0$, nebo existuje-li takové $s, s < k$, pro které je zároveň $l_{ks} \neq 0$ a $x_s \neq 0$. V prvním případě je hledanou cestou cesta nulové délky k, v druhém případě získáme tuto cestu podle indukčního předpokladu propojením cesty $j \Rightarrow s$ a hrany $s \to k$ pro nějaké $j \in \{1, \ldots, n\}$. Tvrzení tedy platí i pro $i = k$ a Věta je dokázána. ■

Tvrzení Věty 9.4.1 si ukážeme na příkladech. Nejprve na schématu 9.12, kde je znázorněna soustava rovnic s dolní trojúhelníkovou maticí. Vektor řešení má nenulové komponenty právě ve všech vrcholech, které jsou dosažitelné v grafu $G(L^T)$ z vrcholu 1, pro který $b_1 \neq 0$.

$$\begin{pmatrix} * & & & & \\ & * & & & \\ * & & * & & \\ & & & * & \\ & & * & & * \end{pmatrix} \begin{pmatrix} * \\ * \\ * \end{pmatrix} = \begin{pmatrix} * \\ \\ \\ \\ \\ \end{pmatrix} \tag{9.12}$$

Postup, který z toho vyplývá pro postupné hledání struktury řídkosti nějakého řádku je znázorněn níže ještě podrobněji jako postupné prohledávání grafu $G(L^T)$ v několika fázích pro vektor pravé strany s první komponentou nenulovou.



Analogické tvrzení je samozřejmě možné zformulovat i pro řešení soustavy rovnic s horní trojúhelníkovou matici. Ačkoli postup vyplývající z Věty 9.4.1 je použitelným symbolickým procesem, opakované řešení soustav s trojúhelníkovými maticemi, čili opakované substituční kroky, které potřebujeme pro numerickou faktorizaci v řádkovém Choleského rozkladu, jsou obecně velmi problematické. Nicméně, v některých situacích jako je výpočet v paralelním výpočetním prostředí, může být tento postup velmi užitečný. Širší diskuse na toto téma přesahuje rámec tohoto textu.

10

Přímé metody řešení soustav lineárních rovnic s řídkou a obecně nesymetrickou maticí

Hlavní náplní této kapitoly je úvod do teorie a algoritmů řídkého LU rozkladu, který reprezentuje nejnáročnější složku těchto metod. Zapišme jej ve tvaru

$$A = LU,$$

kde L je dolní trojúhelníková matice s jednotkovou diagonálou a U je horní trojúhelníková matice. V celé kapitole předpokládáme, pokud nezmíníme jinak, že matice soustavy je silně regulární. Pro existenci rozkladu, který se dá použít pro řešení soustav lineárních rovnic, je takový rozklad bez újmy na obecnosti. Jak jsme totiž zmínili ve Větě 7.4.1 a v diskusi pod ní, pro každou regulární matici A existuje LU rozklad permutované matice PAQ pro nějaké permutační matice P a Q. Dokonce nemusíme uvažovat oboustranné permutace a pro každou regulární matici A existuje LU rozklad matice PA, kde P je nějaká permutační matice. Nicméně, analogicky k Choleského rozkladu, permutace matice vede obecně k maticím jejichž faktory mohou mít jiné zaplnění než má původní nepermutovaná matice a toho se může v praxi využít.

Z hlediska počítání v aritmetice s konečnou přesností je LU rozklad na rozdíl od Choleského rozkladu pouze **podmíněně zpětně stabilní**. To znamená, že i v případě dobře zvoleného počátečního uspořádání, které jsme zmínili výše u Choleského rozkladu a které budeme dále diskutovat, je někdy potřeba stabilitu LU rozkladu ovlivnit dodatečným uspořádáním. Tento proces známe jako **výběr hlavního prvku** nebo **pivotaci**. Pivotace v průběhu rozkladu může efektivitu LU rozkladu výrazně komplikovat a proto bylo pro jeho algoritmy navrženo více strategií i grafových modelů. Stejně jako v případě symetrických a silně regulárních matic se budeme nejprve věnovat teoretickému a algoritmickému popisu LU rozkladu.

10.1 Řídká faktorizace nesymetrických matic a grafové modely

Uvažujme model orientovaného grafu obecně řídké a nesymetrické matice A a kroky LU rozkladu, kterými vytváříme posloupnost eliminačních matic

$$A \equiv A^{(0)}, \ldots, A^{(n-1)},$$

orientovaných eliminačních grafů

$$G^{(0)}, \ldots, G^{(n-1)}$$

a rozšířených orientovaných eliminačních grafů

$$G_E^{(0)}, \ldots, G_E^{(n-1)}$$

Za předpokladu o nevyrušení platí, že struktura rozšířeného eliminačního grafu $G_E^{(n-1)}$ je také strukturou řídkosti matice zaplnění $F = L + U$. V popisu LU rozkladu hrají podstatnou roli, stejně jako v Choleského rozkladu, deficity vrcholů v eliminačních grafech, které vyjadřují neexistenci nějakých hran před provedením konkrétního kroku rozkladu. Ty také mohou být užitečné k teoretickým diskusím o **eliminovatelnosti matice bez vzniku zaplnění**. Pojem orientovaného deficitu v obecném orientovaném grafu je obsahem následující definice.

Definice 10.1.1 *Uvažujme orientovaný graf $G = (V, E)$ a nějaký jeho vrchol $k \in V \equiv \{0, \ldots, n-1\}$. Množinu hran*

$$Df_G(k) = \{(i,j) | (i,k) \in E, (k,j) \in E, (i,j) \notin E, i \neq j, i > k, j > k\}, \tag{10.1}$$

nazveme **orientovaným deficitem** *vrcholu $k \in V$ v G.*

Orientovaný eliminační graf $G^{(k)} = (V^{(k)} \equiv \{k+1, \ldots, n\}, E^{(k)})$ pro $k \in \{0, \ldots, n-1\}$ se dá analogicky jako v případě symetrické eliminace, viz Lemma 8.1.3, vyjádřit vztahem

$$G^{(k)} = (V^{(k)}, E^{(k-1)}(V^{(k)}) \cup Df_G(k)).$$

Na Obrázku 10.1.1 je znázorněn orientovaný deficit prvního kroku rozkladu, tedy $Df_G(1)$.

10.1.1 LU rozklad a základní tvrzení o zaplnění

Grafové zaplnění v závislosti na hranách grafu $G(A)$ původní matice A charakterizuje následující nesymetrická analogie Věty 8.1.1.

Věta 10.1.1 *Nechť $i, j, k \in \{1, \ldots, n\}, k < \min\{i, j\},\ k \leq n-1$. Pak $a_{ij}^{(k)} \neq 0$ právě tehdy, existuje-li orientovaná cesta $i \Rightarrow j$, kterou označíme $(i, p_1, \ldots, p_t, j)$ v $G(A)$, taková, že*

$$p_l \leq k,\ 1 \leq l \leq k, \tag{10.3}$$

kde množina mezilehlých vrcholů cesty $\{p_1, \ldots, p_t\}$ může být i prázdná.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} * & & * & & * & \\ * & & f & & f & \\ & & & & & * \\ * & & f & & f & \\ & & & & & \\ * & & f & & f & \end{pmatrix} \end{array} \tag{10.2}$$

Obrázek 10.1.1: *Příklad orientovaného deficitu struktury řídkosti nesymetrické matice. Prvky deficitu vrcholu* 1 *jsou označeny symbolem* f.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} * & & * & & * & \\ * & * & & & * & * \\ & & * & * & & \\ * & & * & * & & \\ & & * & & * & \\ * & * & & & & * \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} * & & * & & * & \\ * & * & f & & * & * \\ & & * & * & & \\ * & & * & * & f & \\ & & * & f & * & \\ * & * & f & f & f & * \end{pmatrix} \end{array} \tag{10.4}$$

Obrázek 10.1.2: *Příklad struktury řídkosti nesymetrické matice (vlevo) a struktury řídkosti matice* $A^{(n-1)}$. *Prvky zaplnění jsou označeny symbolem* f.

Demonstrujme si Větu 10.1.1 na Obrázku 10.1.2. Na levé straně obrázku je struktura řídkosti původní matice A, na jeho pravé straně je struktura řídkosti matice zaplnění $A^{(n-1)}$, která má stejnou strukturu řídkosti jako matice zaplněné $F = L + U$. Prvky zaplnění jsou označeny symbolem f a získáme je postupnou aplikací obecného Lemmatu o zaplnění 8.1.1. Za předpokladu o nevyrušení vznikají v průběhu rozkladu postupně nové prvky (hrany (i, j) orientovaného modelu grafu) právě tam, kde existuje k, $k < \min\{i, j\}$ takové, že $(i, k) \in G(L)$ a $(k, j) \in G(U)$.

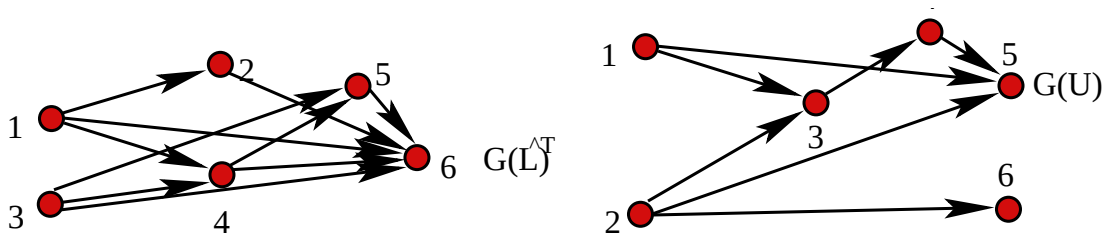
Prvek faktoru L na pozici $(6, 4)$ je prvkem zaplnění plyne z toho, že v grafu $G(A)$ existuje cesta z podmínky Věty 10.1.1. Takovou cestou je například cesta

$$6 \Rightarrow 4 \;\equiv\; 6 \to 1 \to 3 \to 4, \tag{10.5}$$

ale jinou takovou cestou je například cesta

$$6 \Rightarrow 4 \;\equiv\; 6 \to 2 \to 1 \to 3 \to 4. \tag{10.6}$$

V předchozím textu jsme viděli, že efektivní strukturou pro znázornění a algoritmizaci faktorizace symetrických a pozitivně definitních matic je eliminační strom. Pro LU faktorizaci nesymetrické matice máme více možností, jak strukturální změny v průběhu rozkladu zachytit. Tyto možnosti si postupně probereme.



Obrázek 10.1.3: *Orientované acyklické grafy $G(L^T)$ a $G(U)$ pro matici z Obrázku 10.1.2.*

10.1.2 LU rozklad a acyklické orientované grafy

První grafový model, který budeme diskutovat pro LU rozklad silně regulární matice A, je založen na dvojici acyklických orientovaných grafů, které zachytí strukturu řídkosti faktorů L a U samostatně. Uvažujme strukturu matice a matice zaplnění z Obrázku 10.1.2. Oba grafy $G(L)$ a $G(U)$ jsou orientované acyklické grafy. Na Obrázku 10.1.3 zachycujeme orientované grafy faktorů $G(L^T)$ (který se liší od grafu $G(L)$ jen orientací hran) a $G(U)$.

Předpoklad o nevyrušení nás opravňuje psát pro grafy

$$G(L^T) = (V, E(L^T)), \qquad G(U) = (V, E(U))$$

vztahy

$$E(L^T) = \{(i,j) \,|\, l_{ij} \neq 0,\ i \neq j\},\ E(U) = \{(i,j) \,|\, u_{ij} \neq 0,\ i \neq j\}. \tag{10.7}$$

10.1.2.1 Struktury řídkosti řádků faktoru L LU rozkladu a orientované acyklické grafy

Pro určení řádkových i sloupcových struktur řídkosti platí tvrzení analogická rozkladům symetrickým, která ale uvažují nenulovost v kombinaci dvou faktorů. Nejprve uvedeme nutnou a postačující podmínku pro platnost vztahu $l_{ij} \neq 0$ na základě existence **hrany v $G(A)$ a cesty v $G(U)$**. Tato podmínka je zobecněním charakterizace struktur řídkosti řádků faktoru L v symetrických rozkladech.

Věta 10.1.2 *Nechť $A = LU$ je LU faktorizace a platí předpoklad o nevyrušení. Uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$. Nechť $i > j$. Pak $l_{ij} \neq 0$ právě tehdy, existuje-li nějaké k, $k \leq j$, že $a_{ik} \neq 0$ a v $G(U)$ existuje orientovaná cesta $k \Rightarrow j$.*

Důkaz: Uvažujme matici $A(1:i, 1:i)$ pro $i > 1$. Určitě pro i-tý řádek jejího faktoru L platí vztah

$$\begin{pmatrix} l_{i1} \\ \vdots \\ l_{i,i-1} \end{pmatrix} = U^{-T}_{1:i-1,1:i-1} \begin{pmatrix} a_{i1} \\ \vdots \\ a_{i,i-1} \end{pmatrix}. \tag{10.8}$$

Graf $G(U^T_{i-1,1:i-1})$ je orientovaný a acyklický. Podle Věty 9.4.1 je $l_{ij} \neq 0$ právě tehdy, jestliže existuje $a_{ik} \neq 0$ pro nějaké $k \leq j \leq i-1$ takové, že

$$j \in Reach(k, \{1, \ldots, i-1\}, G(U_{1:i-1,1:i-1})).$$

Věta je tak dokázána. ■

Tvrzení Věty 10.1.2 si můžeme demonstrovat na matici z Obrázku 10.1.2. Faktor L má zaplnění na pozici $(6, 4)$, neboť

$$a_{ij} \equiv a_{62} \neq 0$$

a v $G(U)$ existuje cesta

$$k \equiv 2 \to 3 \to 4 \equiv j.$$

Analogické tvrzení pro platnost vztahu $u_{ij} \neq 0$, které můžeme chápat také určování sloupcové struktury faktoru U LU rozkladu uvádíme bez důkazu ve Větě 10.1.3.

Věta 10.1.3 *Nechť $A = LU$ je LU faktorizace a platí předpoklad o nevyrušení. Uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$. Nechť $i < j$. Pak $u_{ij} \neq 0$ právě tehdy, existuje-li nějaké k, $k \leq i$, že $a_{kj} \neq 0$ a v $G(L)$ existuje orientovaná cesta $i \Rightarrow k$.*

10.1.2.2 Struktury řídkosti sloupců faktoru L LU rozkladu a orientované acyklické grafy

Sloupcové struktury faktoru L a řádkové struktury faktoru U v LU rozkladu silně regulární matice lze vyjádřit podobně jako v Choleského rozkladu. Algoritmus jejich počítání nazveme také symbolickou faktorizací jako v případě symetrické a silné regulární matice. Prvním krokem k odvození tohoto vyjádření je formální charakterizace struktury řídkosti faktoru L v následující Větě 10.1.4.

Věta 10.1.4 *Nechť $A = LU$ je LU rozklad a platí předpoklad o nevyrušení. Pak platí*

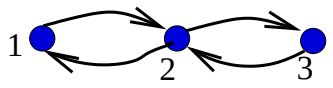
$$\mathcal{S}(L_{*j}) = \mathcal{S}(A_{*j}) \cup \bigcup \{\mathcal{S}(L_{*k}) \mid k < j,\ u_{kj} \neq 0\} \setminus \{1, \ldots, j-1\} \tag{10.9}$$

Platnost uvedeného tvrzení snadno nahlédneme, protože je jen jinou interpretací počítání sloupce faktoru L na základě dříve spočítaných prvků rozkladu. Podobně můžeme vyjádřit strukturu řádků faktoru U LU rozkladu.

Věta 10.1.5 *Nechť $A = LU$ je LU rozklad a platí předpoklad o nevyrušení. Pak platí*

$$\mathcal{S}(U_{i*}) = \mathcal{S}(A_{i*}) \cup \bigcup \{\mathcal{S}(U_{k*}) \mid k < i,\ l_{ik} \neq 0\} \setminus \{1, \ldots, i-1\} \tag{10.10}$$

Obě tato vyjádření nejsou z praktického pohledu žádnou výhrou, neboť v nich vystupují obecně celé grafy faktorů, které mohou obsahovat mnoho nenulových prvků. Kýžené vyjádření musí být založeno na nějaké grafové redukci, která umožní proces slévání struktur řídkosti vyjádřit efektivněji. V symetrickém případě je touto redukcí eliminační strom. K nalezení symbolické faktorizace vedlo pozorování procesu replikace struktur sloupců faktoru. Následující podsekce se redukci vhodné k efektivnímu vyjádření struktury řídkosti faktorů LU rozkladu bude věnovat.



Obrázek 10.1.4: *Příklad grafu, pro který existují dvě tranzitívní redukce, které nejsou ani podgrafem daného grafu.*

10.1.2.3 Tranzitivní redukce orientovaného acyklického grafu a LU rozklad

Následující Pozorování 8.1.2 je nesymetrickou analogií tvrzení o replikaci sloupcových struktur faktoru L LU rozkladu.

Pozorování 10.1.1 *Nechť* $1 \leq \ldots \leq k < j \leq \ldots \leq n$ *a* $(k, j) \in E(G(U))$. *Pak platí*

$$\mathcal{S}(L_{*k}) \setminus \{1, \ldots, j-1\} \subseteq \mathcal{S}(L_{*j}). \tag{10.11}$$

Například pro matici na Obrázku 10.1.2 platí

$$\mathcal{S}(L_{*1}) \setminus (2, 1) \subseteq \mathcal{S}(L_{*3}).$$

V případě LU rozkladu je tedy situace složitější než v rozkladu symetrickém, protože se zde prolínají tvrzení o grafech dvou faktorů. Nicméně toto pozorování naznačuje, že strukturu závislostí je možné zjednodušit redukcí takovou orientovaných acyklických grafů $G(L)$ a $G(U)$, která zachová vlastnosti podstatné pro replikaci struktur ve faktorech LU rozkladu. Redukcí, která umožní zjednodušit symbolickou faktorizaci je **tranzitivní redukce** orientovaného grafu, definovaná následovně.

Definice 10.1.2 *Graf* $G^0 = (V, E^0)$ *nazveme tranzitivní redukcí orientovaného grafu* $G = (V, E)$ *jestliže*

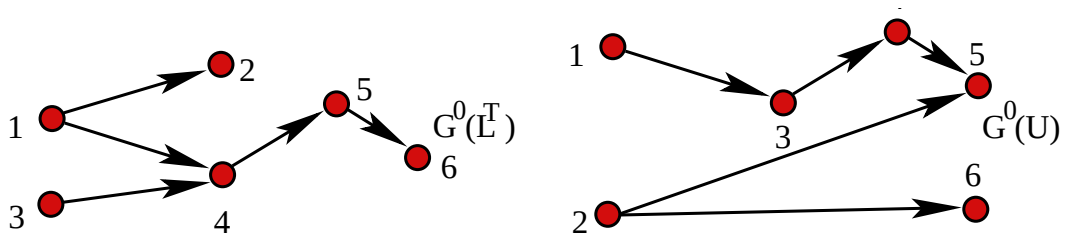
- *Pro libovolné vrcholy* $u, v \in V$ *platí* $u \Rightarrow v$ *v grafu* G^0 *právě tehdy, když* $u \Rightarrow v$ *v* G,
- *žádný graf s množinou vrcholů* V *a menším počtem hran než* $|E^0|$ *předcházející podmínku nesplňuje.*

Tranzitivní redukce orientovaného grafu G nemusí být obecně ani jednoznačná ani podgrafem grafu G. Příkladem může být graf na Obrázku 10.1.4. Tranzitívní redukcí je zde cyklus, který propojuje vrcholy $1, 2, 3$ v různém pořadí.

V případě orientovaného acyklického grafu je situace jednodušší. Příslušné tvrzení uvedeme bez důkazu.

Věta 10.1.6 *Tranzitivní redukce orientovaného acyklického grafu je jednoznačně určena a je podgrafem grafu* G.

Tranzitivní redukce orientovaného acyklického grafu přirozeně rozšiřuje pojem eliminačního stromu zavedený pro symetrický rozklad silně regulární matice. Je to onen podgraf, který zachovává strukturu cest v původním grafu: vrcholy jsou propojené cestou v tranzitivní redukci právě tehdy, jsou-li propojeny cestou v původním grafu.



Obrázek 10.1.5: *Tranzitivní redukce $G^0(L^T)$ a $G^0(U)$ orientovaných acyklických grafů $G(L^T)$ a $G(U)$ k Obrázkům 10.1.2 a 10.1.3.*

Věta 10.1.7 *Tranzitivní redukcí orientovaného acyklického grafu $G(U) \equiv G(L^T)$ symetrické nerozložitelné matice je eliminační strom.*

Důkaz: Eliminační strom T splňuje požadavek, že z existence hrany (i, j) v $G(L^T)$ pro $i > j$ plyne existence cesty $i \Rightarrow j$ v T jak je hned vidět z procesu replikace hran v symetrické faktorizaci bez vyrušení. Podle Věty 10.1.6 je tranzitívní redukce $G(L^T)$ jednoznačně určená a má mít nejmenší počet hran a je podgrafem $G(L^T)$. Vynecháním jakékoli hrany v eliminačním stromu T porušíme podmínku dosažitelnosti z Definice 10.1.2. ■

Na Obrázku 10.1.5 jsou znázorněny tranzitivní redukce orientovaných acyklických grafů $G(L^T)$ a $G(U)$ pro matice z Obrázku 10.1.2 a grafy faktorů z Obrázku 10.1.3.

Tranzitivní redukce orientovaných acyklických grafů umožňují zefektivnit nalezení struktur matic, které souvisejí s řídkým rozkladem. Nejprve uveďme Větu 10.1.8, jejíž tvrzení je zřejmé z Definice 10.1.2.

Věta 10.1.8 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků a nechť $l_{ij} \neq 0$, $i > j$. Pak v $G^0(L)$ existuje orientovaná cesta $i \Rightarrow j$.*

Důkaz: Vztah $l_{ij} \neq 0$ znamená za předpokladu o nevyrušení, že platí $i \to j$ v $G(L)$. $G^0(L)$ je tranzitívní redukce $G(L)$, což implikuje $i \Rightarrow j$ v $G^0(L)$. ■

Věta 10.1.8 tedy ukazuje, jak se dá zjednodušit výpočet struktur sloupců faktoru L z Věty 10.1.4 a poté i analogicky řádků faktoru U. Místo, aby se použily pro hledání cest všechny mimodiagonální nenulové prvky matic L a U, tedy stačí uvažovat pouze ty z nich, které odpovídají hranám v tranzitivních redukcích jejich grafů a využít princip replikace z Pozorování 10.1.1. Hran v tranzitivních redukcích **je obvykle výrazně méně** než v původních grafech podobně jako eliminační strom zredukoval cesty, které určují zaplnění, na nejnutnější minimum. Přepisem Věty 10.1.4 pomocí tranzitivní redukce získáme následující tvrzení.

Věta 10.1.9 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků a uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$ a tranzitivní redukci grafu $G(U)$ $G^0(U) = (V, E_U^0)$. Pak platí*

$$\mathcal{S}(L_{*j}) = \mathcal{S}(A_{*j}) \cup \bigcup \{\mathcal{S}(L_{*k}) \mid (k, j) \in E_U^0\} \setminus \{1, \ldots, j-1\} \tag{10.12}$$

Důkaz: Uvažujme struktury řídkosti sloupců faktoru L a uvažujme dále nějakou hranu (k, j), která existuje v $G(U)$, ale není v $G^0(U)$. Opakovanou aplikací Pozorování 10.1.1 na cestu $k \Rightarrow j$ v $G^0(U)$ vidíme, že i struktura řídkosti L_{*k} je obsažena ve sjednocení na pravé straně v (10.12) a platí tedy

$$\mathcal{S}(L_{*j}) \subseteq \mathcal{S}(A_{*j}) \cup \bigcup \{\mathcal{S}(L_{*k}) \mid (k, j) \in E_U^0\} \setminus \{1, \ldots, j-1\}.$$

Pravá strana tohoto výrazu je zřejmě obsažena v levé straně a první část tvrzení Věty je dokázána. ■

Následující tvrzení pro struktury řídkosti řádků faktoru U se dokáže analogicky.

Věta 10.1.10 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků a uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$ a tranzitívní redukci $G^0(L) = (V, E_L^0)$. Pak platí*

$$\mathcal{S}(U_{i*}) = \mathcal{S}(A_{i*}) \cup \bigcup \{\mathcal{S}(U_{k*}) \mid (i, k) \in E_L^0\} \setminus \{1, \ldots, i-1\} \tag{10.13}$$

Stejně tak je možné modifikovat Věty 10.1.2 a 10.1.3, kde jsou cesty v acyklických grafech $G(L)$ a $G(U)$ nahrazeny cestami v jejich tranzitivních redukcích. Uvedení těchto variant není samoúčelné, protože praktické procedury hledání struktur faktorů jsou obvykle založeny právě **na cestách v těchto redukcích**.

Věta 10.1.11 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků a uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$. Nechť $i > j$. Pak $l_{ij} \neq 0$ právě tehdy, existuje-li nějaké k, $k \leq j$, že $a_{ik} \neq 0$ a v $G^0(U)$ existuje orientovaná cesta $k \Rightarrow j$.*

Věta 10.1.12 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků a uvažujme graf $G(A)$ s $V = \{1, \ldots, n\}$. Nechť $i < j$. Pak $u_{ij} \neq 0$ právě tehdy, existuje-li nějaké k, $k \leq i$, že $a_{kj} \neq 0$ a v $G^0(L)$ existuje orientovaná cesta $i \Rightarrow k$.*

Algoritmus 10.1.1 počítá **oba symbolické faktory po řádcích** tak, jak bylo navrženo v [79], ale existují i jiné možnosti postupu.

Algoritmus 10.1.1 Symbolická eliminace silně regulární matice A na základě tranzitívních redukcí grafů faktorů L a U.
Input: *Silně regulární matice A.*
Output: *Struktury řídkosti jejích LU faktorů L a U.*

1. **for** $i = 1 : n$ **do**
2. *Spočítej $\mathcal{S}(L_{i*})$ procházením $G^0(U_{1:i-1,1:i-1})$ z vrcholů $\mathcal{S}(A_{i*})$.*
3. *Získej $Adj_{G^0(L_{1:i,1:i})}(i)$ a tím i $G^0(L_{1:i,1:i})$ tranzitívní redukcí $\mathcal{S}(L_{i*})$.*
4. *Spočítej $\mathcal{S}(U_{i*})$ jako sjednocení předcházejících řádků s pomocí $G^0(L_{1:i,1:i})$.*
5. *Získej $Adj_{G^0(U_{1:i,1:i})}(i)$ tranzitívní redukcí $\mathcal{S}(U_{*i})$.*
6. **end** i

$$
\begin{array}{c}
\\ j \\ s \\ k
\end{array}
\begin{array}{c}
\begin{array}{cccc} j & s & k & \end{array} \\
\begin{pmatrix}
* & & & & & & \\
 & * & & * & & * & \\
 & & * & & & & \\
 & * & & * & & & \\
 & & & & * & & \\
 & * & & & & * & \\
 & & & & & & *
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c}
\\ j \\ s \\ k
\end{array}
\begin{array}{c}
\begin{array}{cccc} j & s & k & \end{array} \\
\begin{pmatrix}
* & & & & & & \\
 & * & & * & & * & \\
 & & * & & & & \\
 & * & & * & & f & \\
 & & & & * & & \\
 & * & & f & & * & \\
 & & & & & & *
\end{pmatrix}
\end{array}
\tag{10.14}
$$

Obrázek 10.1.6: *Příklad struktury řídkosti nesymetrické matice (vlevo) a struktury řídkosti matice $L + U$ (vpravo). Prvky zaplnění jsou označeny symbolem f.*

10.1.2.4 Konstrukce tranzitivní redukce orientovaného acyklického grafu

Tranzitívní redukce diskutovaných orientovaných acyklických grafů jsou sice jednoznačně určeny, ale v praxi je obvykle náročné je získat. Proto se spíše než přesné tranzitívní redukce počítají jejich aproximace, které vzniknou z grafů $G(L)$ a $G(U)$ pouze **částečnou redukcí** jejich hran, neboli pouze jejich ořezáním. Uvažujme Obrázek 10.1.6, kde je redukce (vynechání hrany) provedena na základě platnosti vztahu $l_{sj} * u_{js} \neq 0$. Podrobněji: hrana (j, k) je nahrazena ve výsledném redukovaném grafu nahrazena cestou

$$j \to s \to k$$

a takovou redukcí můžeme postupně získat tranzitivní redukci $G^0(L)$. K tomu, aby ve faktoru L vzniklo zaplnění na pozici (k, s), je ale zapotřebí nejenom aby platilo $l_{sj} \neq 0$, ale i $u_{js} \neq 0$, tedy dohromady $l_{sj} * u_{js} \neq 0$. Takovéto redukci se říká **symetrická redukce**. Výsledná struktura řídkosti je znázorněna na Obrázku 10.1.7. Zdůrazněme, že to je struktura řídkosti, kterou použijeme pro tranzitívní redukci **grafů** faktorů L a U. Pro zachycení numerických hodnot faktorů musíme samozřejmě použít neredukované grafy a matice.

Opakem tranzitivní redukce orientovaného grafu v určitém slova smyslu je **tranzitivní uzávěr grafu**, který zmíníme později.

10.1.3 LU rozklad na základě sloupcového eliminačního stromu

Velkým problémem řídkého LU rozkladu v praxi je zabezpečení splnění předpokladu silné regularity. Jak víme, silná regularita matice odpovídá tomu, že se nestane, že v některém kroku rozkladu spočítáme nulový diagonální prvek faktoru U. V praxi potřebujeme ale ještě víc, konkrétně omezit velikost růstového faktoru rozkladu, protože LU rozklad je pouze **podmíněně zpětně stabilní**. To můžeme provést dvěma základními způsoby. Za prvé, budeme provádět pivotaci. Jedna z jeho variant, nazývaná **částečná pivotace**, znamená, že se hledá LU rozklad

$$PA = LU, \tag{10.16}$$

$$
\begin{array}{c}
\\ j \\ \\ s \\ \\ k \\ \\
\end{array}
\begin{array}{c}
\begin{array}{cccccc} \quad & j & \quad & s & \quad & k \end{array} \\
\begin{pmatrix}
* & & & & & \\
 & * & & * & & \\
 & & * & & & \\
 & * & & * & & f \\
 & & & & * & \\
 & & & f & & * \\
 & & & & & & * \\
\end{pmatrix}
\end{array}
\tag{10.15}
$$

Obrázek 10.1.7: *Příklad struktury řídkosti redukované struktury matice A odpovídající grafu $G^0(L+U)$. Prvky zaplnění jsou označeny symbolem f.*

kde P je permutační matice odpovídající dimenze. Samozřejmě je možné uvažovat o nalezení vhodné permutační matice dříve, než se bude rozklad provádět. To je ale obecně proveditelné pouze ve velmi speciálních případech. Druhým způsobem omezení růstového faktoru je rozklad modifikovat a počítat jej tedy pouze nepřesně. To můžeme vyjádřit vztahem

$$A = LU + E, \tag{10.17}$$

kde E je matice, která obsahuje chybu rozkladu. O tomto druhém způsobu budeme hovořit v dalších částech textu a budeme přitom hovořit o **neúplném** nebo **přibližném** rozkladu.

Jakákoli pivotace může být velmi neefektivní, je-li přímo propojena s numerickou faktorizací. Existuje ale způsob, jak oddělit symbolickou faktorizaci od numerické v případě částečné pivotace. Takový postup je založen na **sloupcovém eliminačním stromu**. Sloupcovým eliminačním stromem nazýváme eliminační strom matice A^TA. Následující věta [76] takový postup motivuje.

Věta 10.1.13 *(George, Ng (1985)) Předpokládejme, že matice A má všechny diagonální prvky nenulové. Nechť je dále $\tilde{L}\tilde{L}^T$ symetrický rozklad silně regulární matice A^TA bez vyrušení. Pak*

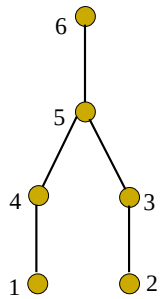
$$\mathcal{S}(L+U) \subseteq \mathcal{S}((\tilde{L}) + (\tilde{L})^T)$$

pro libovolnou permutační matici P takovou, že $PA = LU$.

Připomeňme, že má-li matice A plnou sloupcovou hodnost, pak je A^TA pozitivně definitní a tedy i silně regulární. Na Obrázku 10.1.8 si postup vytvoření struktury řídkosti faktorů na základě Věty 10.1.13 znázorníme. Vlevo je zobrazena struktura řídkosti matice A, poté struktura řídkosti matice A^TA a dále i tato matice včetně zaplnění. Obrázek také znázorňuje příslušný eliminační strom matice A^TA. Vidíme, že struktura řídkosti matice A^TA je **mnohem méně řídká**, než struktura řídkosti matice A.

Ještě pesimičtěji vypadá situace na Obrázku 10.1.9, který také přináší porovnání struktury řídkosti nesymetrického rozkladu (matice zaplnění $L+U$) a struktury řídkosti Choleského faktoru matice A^TA. Druhá z těchto struktur má mnohem větší zaplnění.

$$
\begin{pmatrix}
* & & & & & \\
 & * & & & * & \\
 & & & * & * & * \\
* & & & * & & * \\
 & * & * & & & \\
 & & * & & * &
\end{pmatrix}
\quad
\begin{pmatrix}
* & & & * & & * \\
 & * & * & & * & \\
 & * & * & & & * \\
* & & & * & * & * \\
 & * & & * & * & * \\
* & & * & * & * & *
\end{pmatrix}
\quad
\begin{pmatrix}
* & & & * & & * \\
 & * & * & & * & \\
 & * & * & & f & * \\
* & & & * & * & * \\
 & * & f & * & * & * \\
* & & * & * & * & *
\end{pmatrix}
\tag{10.18}
$$

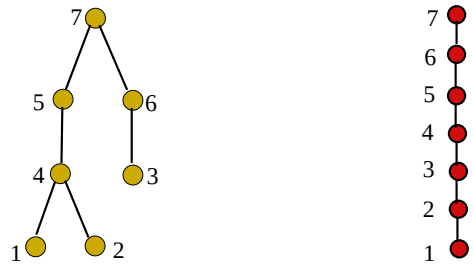


Obrázek 10.1.8: *Příklad struktury řídkosti matice A, struktury řídkosti matice A^TA a struktury řídkosti matice A^TA včetně zaplnění při jejím symetrickém rozkladu (nahoře). Eliminační strom matice A^TA (dole). Prvky zaplnění jsou označeny symbolem f.*

$$
\begin{pmatrix}
* & & & * & & & \\
* & * & & * & & & \\
 & & * & & & * & \\
* & * & & * & & & * \\
 & * & * & & * & * & * \\
 & & * & & & * & \\
* & & & * & * & & *
\end{pmatrix}
\qquad
\begin{pmatrix}
* & * & & * & * & & * \\
* & * & * & * & * & * & * \\
 & * & * & & * & * & * \\
* & * & & * & * & & * \\
* & * & * & * & * & * & * \\
 & * & * & & * & * & * \\
* & * & * & * & * & * & *
\end{pmatrix}
$$

$$
\begin{pmatrix}
* & & & * & & & \\
* & * & f & * & & & \\
 & & * & & & * & \\
* & * & & * & f & & * \\
 & * & * & f & * & * & * \\
 & & * & & & * & \\
* & & & * & * & f & *
\end{pmatrix}
\qquad
\begin{pmatrix}
* & * & & * & * & & * \\
* & * & * & * & * & * & * \\
 & * & * & f & * & * & * \\
* & * & f & * & * & f & * \\
* & * & * & * & * & * & * \\
 & * & * & f & * & * & * \\
* & * & * & * & * & * & *
\end{pmatrix}
$$

Obrázek 10.1.9: *Další příklad struktury řídkosti matice A (vlevo nahoře), struktury řídkosti matice A^TA (vpravo nahoře) a struktury jejich rozkladů: matice $L+U$ pro LU rozklad matice A (vlevo dole) a matice $L+L^T$ symetrického rozkladu matice A^TA (vpravo dole). Prvky zaplnění jsou označeny symbolem f.*



Obrázek 10.1.10: *Eliminační stromy rozkladů matic se strukturami řídkosti z Obrázku 10.1.9. Nesymetrický eliminační strom pro LU rozklad matice vlevo a symetrický pro symetrický rozklad matice vpravo.*

Obecně tedy sice vidíme, že hledání strukturálních vlastností i poměrně stabilního LU rozkladu s částečnou pivotací se dá převést na hledání strukturálních vlastností matic symetrických a pozitivně definitních. Efektivita tohoto postupu je ale velmi často příliš nízká kvůli menší řídkosti matice $A^T A$. Efektivnější varianta tohoto postupu navržená [77] také odděluje symbolickou a numerickou faktorizaci a je založena na explicitním provedení symbolické faktorizace, která simuluje kroky faktorizace a uvažuje přitom všechny možné řádkové permutace v částečné pivotaci. Výslednou strukturu pak určuje na základě jejich sjednocení. Obecně je ale v obou variantách postupu daň za částečnou pivotaci v tom, že symbolická faktorizace může být náročná, neboť není založena na efektivní grafové redukci.

10.1.4 LU rozklad a nesymetrický eliminační strom

Poslední způsob zachycení struktury řídkosti LU rozkladu, který zde zmíníme, je založen na pojmu **nesymetrického eliminačního stromu** [63]. Ten zobecňuje standardní eliminační strom zavedený výše pro symetrické a pozitivně definitní matice tak, že vytváří jednu stromovou strukturu tím, že uvažuje **struktury cest v obou grafech faktorů** $G(L)$ a $G(U)$ zároveň. Připomeňme, že eliminační strom pro symetrický rozklad je určen prostřednictvím zobrazení *parent* na základě faktoru L následujícím způsobem s tím, že je třeba dodefinovat jeho hodnotu v případě, že příslušná množina indexů je prázdná.

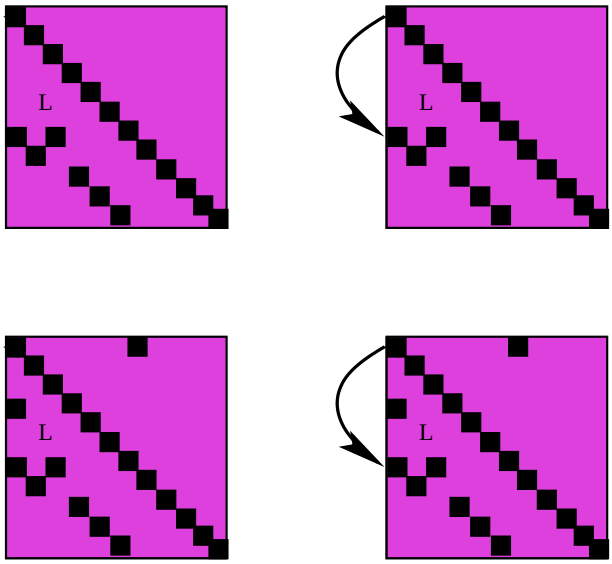
$$parent(k) = \min\{j \mid j > k \wedge j \xrightarrow{G(L)} k\}.$$

Pro symetrický rozklad lze tuto definici přepsat ekvivalentně následujícím způsobem

$$parent(k) = \min\{j \mid j > k \wedge j \xrightarrow{G(L)} k \xrightarrow{G(L^T)} j\}.$$

Nesymetrický eliminační strom je pak založena na definici zobrazení *parent*, která použije místo hran faktorů L a L^T Cholesk0ho rozkladu cesty ve faktorech L a U LU rozkladu.

$$parent(k) = \min\{j \mid j > k \wedge j \xRightarrow{G(L)} k \xRightarrow{G(U)} j\}. \tag{10.19}$$



Obrázek 10.1.11: *Obrázek znázorňující rozdíl mezi symetrickým a nesymetrickým eliminačním stromem. V druhém případě (dolní obrázky) je vidět, že prvek definující hranu eliminačního stromu nemusí být ten první poddiagonální ve faktoru L.*

Rozdíl mezi symetrickým a nesymetrickým eliminačním stromem je znázorněn na Obrázku 10.1.11. Je vidět, že prvek faktoru L, který definuje hranu nesymetrického eliminačního stromu, nemusí být nutně jeho prvním poddiaginálním prvkem.

Nesymetrický eliminační strom pro matici na Obrázku 10.1.12 je znázorněn na Obrázku 10.1.13. Vidíme například, že vrchol 6 je otcem vrcholů 2 a 5, protože ve faktorech L a U, jejichž odpovídají trojúhelníkovým částem matice z Obrázku 10.1.9 vpravo, existují cesty $6 \xrightarrow{G(L)} 2 \xrightarrow{G(U)} 5 \xrightarrow{G(U)} 6$ a $6 \xrightarrow{G(L)} 5 \xrightarrow{G(U)} 6$, které splňují podmínku minimality z definice nesymetrického eliminačního stromu podle (10.19). Analogicky i cesta $10 \xrightarrow{G(L)} 5 \xrightarrow{G(L)} 4 \xrightarrow{G(U)} 10$ implikuje, že vrchol 10 je otec vrcholu 4.

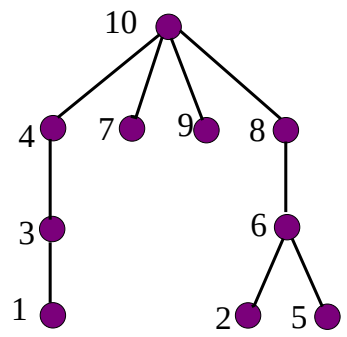
Vraťme se ještě k maticím z Obrázku 10.1.9. Na Obrázku 10.1.10 jsou znázorněny jejich eliminační stromy. Nesymetrický eliminační strom pro LU rozklad matic je vlevo a symetrický eliminační strom pro symetrický rozklad matice je vpravo. Stejně jako v případě eliminačního stromu symetrické matice tedy nepožadujeme souvislost nesymetrického eliminačního stromu. V grafové terminologii tento strom může být tedy také les.

To, že spolu souvisí cesty ve faktorech L a U LU rozkladu, které jsou použity v definici nesymetrického eliminačního stromu, a cesty v původní matici analogicky Choleskému rozkladu je obsahem následující Věty 10.1.14. Zde vyjadřujeme existenci cesty v dolním trojúhelníkovém faktoru L, ale analogické tvrzení platí i pro prvky faktoru U.

Věta 10.1.14 *Nechť $A = LU$ je LU faktorizace bez vyrušení nenulových prvků. Dosažitelnost mezi vrcholy j a $k, j > k$ v grafech $G(L)$ a $G(A)$, kde $V = \{1, \ldots, n\}$ splňuje následující*

$$
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array}\\
\begin{pmatrix}
* & & * & & & & & & & \\
* & * & & & * & & & * & & * \\
* & & * & * & & & & & & \\
 & & * & * & & & & & * & * \\
 & & & * & * & * & & * & & \\
 & * & & & & * & & & & \\
 & & & * & & & * & & & \\
* & * & & & & & & * & & \\
 & & & & & & & & * & * \\
 & & & & * & & * & & & *
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array}\\
\begin{pmatrix}
* & & * & & & & & & & \\
* & * & f & & * & & & * & & * \\
* & & * & * & & & & & & \\
 & & * & * & & & & & * & * \\
 & & & * & * & * & & * & f & f \\
 & * & f & f & f & * & & f & f & f \\
 & & & * & & & * & & f & f \\
* & * & f & f & f & f & & * & f & f \\
 & & & & & & & & * & * \\
 & & & & * & f & * & f & f & *
\end{pmatrix}
\end{array}
$$

Obrázek 10.1.12: *Struktura řídkosti nesymetrické matice bez zaplnění (vlevo) a včetně zaplnění (vpravo), na které budeme demonstrovat strukturální vlastnosti nesymetrického eliminačního stromu. Prvky zaplnění jsou označeny symbolem f.*



Obrázek 10.1.13: *Eliminační strom pro nesymetrickou matici z Obrázku 10.1.12.*

ekvivalenci:

$$j \overset{G(L)}{\Longrightarrow} k \iff j \underset{\{1,\ldots,j\}}{\overset{G(A)}{\Longrightarrow}} k, \tag{10.20}$$

kde všechny vrcholy na těchto cestách jsou nejvýše j.

Důkaz: Nechť platí $j \overset{G(L)}{\Longrightarrow} k$ pro dané vrcholy j, k, kde $j > k$. Pro každou hranu na této cestě platí podle Věty o zaplnění pro nesymetrický rozklad, že je ji možné nahradit cestou v $G(A)$, kde všechny vrcholy na této cestě jsou menší než oba krajní vrcholy. Propojením všech těchto cest získáme hledanou cestu v $G(A)$.

Opačně, nechť existuje výše uvedená cesta v $G(A)$. Nechť x je největší vrchol na této cestě různý od j. Pak cestu $j \underset{\{1,\ldots,j\}}{\overset{G(A)}{\Longrightarrow}} k$ můžeme rozložit na dvě cesty. Konkrétně na $j \overset{G(A)}{\Longrightarrow} x$ a $x \overset{G(A)}{\Longrightarrow} k$, kde druhá z těchto cest může být triviálně prázdná. Podle Věty 10.1.1 existuje hrana $j \overset{G(L)}{\longrightarrow} x$. Použitím indukčního argumentu pro indukci vztaženou na délku cesty na druhou z těchto cest víme, že existuje také cesta $j \overset{G(L)}{\Longrightarrow} x$. Věta je dokázána. ■

Následující Věta 10.1.15 je nesymetrickou analogií tvrzení Věty 8.2.1. Vidíme, že z formálního pohledu zde místo souvislosti hraje roli silná souvislost v grafu.

Věta 10.1.15 *Vrchol i je předek vrcholu j v nesymetrickém eliminačním stromu $T(A)$ grafu $G(A)$ právě tehdy, platí-li $i > j$ a oba vrcholy j a i patří do téže* **silně souvislé komponenty** *grafu $G(\{1, \ldots, i\})$.*

Důkaz: Je-li i předek vrcholu j, pak musí být $i > j$ a také existovat cyklus $i \overset{G(A)}{\Longrightarrow} j \overset{G(A)}{\Longrightarrow} i$, kde všechny vrcholy na těchto cestách jsou nejvýše i. Tento cyklus tedy leží nejen v $G(\{1, \ldots, i\})$, ale zároveň je celý ve stejné silně souvislé komponentě tohoto grafu. Opačně, nechť $i > j$ jsou vrcholy stejné souvislé komponenty $G(\{1, \ldots, i\})$. Pak musí v této komponentě být cyklus $i \overset{G(A)}{\Longrightarrow} j \overset{G(A)}{\Longrightarrow} i$ a všechny vrcholy na těchto cestách jsou nejvýše i. Předpokládejme nyní, že $\bar{j}$ je maximální vrchol takový, že $i > \bar{j}$, pro který $i \overset{G(A)}{\Longrightarrow} \bar{j} \overset{G(A)}{\Longrightarrow} i$, kde všechny vrcholy na těchto cestách jsou nejvýše i a přitom i není jeho předek. Musí platit, že rodič p vrcholu $\bar{j}$ v nesymetrickém eliminačním stromu je menší než i, protože ze všech vrcholů, pro které existuje výše uvedený cyklus jako pro i je zároveň minimální. Existuje tedy cyklus $p \overset{G(A)}{\Longrightarrow} \bar{j} \overset{G(A)}{\Longrightarrow} p$, kde všechny vrcholy na těchto cestách jsou nejvýše p. Propojením dvou výše uvedených cest získáme cestu $i \overset{G(A)}{\Longrightarrow} \bar{j} \overset{G(A)}{\Longrightarrow} p \overset{G(A)}{\Longrightarrow} \bar{j} \overset{G(A)}{\Longrightarrow} i$, tedy i $i \overset{G(A)}{\Longrightarrow} p \overset{G(A)}{\Longrightarrow} i$ se všemi vrcholy na těchto cestách nejvýše i. Ale i nemůže být předek ani vrcholu p, což je ve sporu s definicí vrcholu $\bar{j}$, protože $p > \bar{j}$. Proto musí platit, že z existence cesty $i \overset{G(A)}{\Longrightarrow} j \overset{G(A)}{\Longrightarrow} i$, kde všechny vrcholy jsou nejvýše i, plyne, že i je předek vrcholu j a věta je dokázána. ■

Zřejmým důsledkem Věty 10.1.15, který je analogický Důsledku 8.2.1, je následující tvrzení, které ukazuje, jak eliminační strom konstruovat na základě postupů pro hledání silných komponent orientovaného grafu.

Důsledek 10.1.1 *Vrchol p je rodič vrcholu k v nesymetrickém eliminačním stromu $T(A)$ právě tehdy, je-li $p > k$ nejmenší takový, že patří do stejné silně souvislé komponenty $G_p(A)$ grafu $G(A)$.*

Stejně jako v případě charakterizace řádků faktoru L v případě řídkého Choleského rozkladu, i v případě LU rozkladu je možné charakterizovat řádkové struktury faktoru L (a sloupcové struktury faktoru U) pomocí eliminačního stromu. Následující definice a věta, kterou uvádíme bez důkazu, k tomu poskytují návod.

Definice 10.1.3 *Řekneme, že podgraf $\bar{T}$ eliminačního stromu T je* **ořezaný eliminační strom** *(pruned forest podle [63]), jestliže pro jeho každý vrchol $v \in V(\bar{T})$ platí*

$$\text{bud'}\ parent(v) \in V(\bar{T}) \quad \text{nebo} \quad anc_T(v) \cap V(\bar{T}) = \emptyset.$$

Věta 10.1.16 *Struktury řídkosti řádků faktoru L a sloupců faktoru U LU rozkladu jsou ořezané eliminační stromy.*

Z této věty vidíme, že způsob ořezání nesymetrického eliminačního stromu je obecnější než v případě symetrické faktorizace právě proto, že v LU rozkladu hraje velkou roli souvislost dvou různých trojúhelníkových matic.

Nesymetrický eliminační strom je velmi mocná struktura, která umožňuje LU rozklad i v případě některých způsobů pivotace, jako jsou například diagonální nebo i částečná pivotace. K tomu je zapotřebí další podrobná diskuse a zavedení dalších speciálních druhů uspořádání. Tomuto se nebudeme v dalším textu věnovat a zájemce odkazujeme na sérii tří článků [63], [64] a [65].

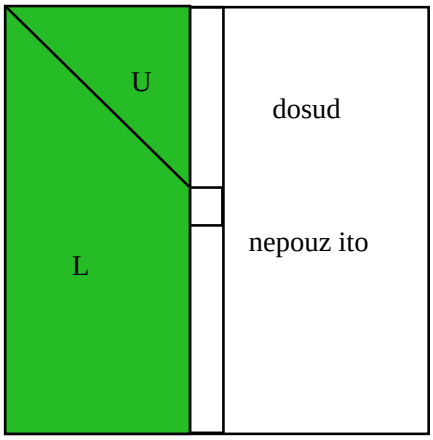
10.1.5 Souběžná symbolická a numerická faktorizace s částečnou pivotací

Velkou nevýhodou metod založených na nesymetrickém eliminačním stromu je omezená možnost výběru hlavního prvku (pivotace), chceme-li zachovat efektivitu LU rozkladu. V této sekci ukážeme jednu specifickou metodu a velmi průzračnou metodu, která plnou řádkovou pivotaci umožňuje. Navrhli ji v roce 1988 Gilbert a Peierls v roce 1988 [81] a jejím základem je sloupcový algoritmus LU rozkladu, který počítá v jednom kroku sloupec faktoru L i sloupec faktoru U. Konkrétně, pro každé j, $1 \leq j \leq n$ se nejprve spočítá nový sloupec faktoru U řešením soustavy s dolní trojúhelníkovou maticí tak, jak jsme o tom hovořili výše. Poté je nový sloupec faktoru L vyjádřen jako lineární kombinace předcházejících sloupců faktoru L. Tento postup umožňuje částečnou pivotaci. Z formálního pohledu tedy rozklad matice PA, kde P je permutační matice příslušné dimenze. Následující algoritmus tyto operace popisuje formálně.

Algoritmus 10.1.2 Sloupcový algoritmus LU faktorizace s částečnou pivotací.
Input: *Řídká matice $A = (a_1, \ldots, a_n)$.*
Output: *LU faktory $L = (l_1, \ldots, l_n)$ a $U = (u_1, \ldots, u_n)$ matice $PA = LU$. Permutační matice P je dána implicitně na základě permutací v kroku 4 algoritmu.*

1. **for** $j = 1 : n$ **do**
2. *Spočti* $u_{1:j-1,j}$ *ze vztahu* $L_{1:j-1,1:j-1}u_{1:j-1,j} = A_{1:j-1,j}$
3. *Polož* $b = A_{j:n,j} - L_{j:n,1:j-1}u_{1:j-1,j}$, $b \in \mathbb{R}^j$
4. *Najdi v* b *složku s maximální velikostí a permutuj* b *tak, aby byla tato složka* b_1.
5. *Polož* $U_{j,j} = b_1$
6. *Polož* $L_{j:n,j} = b_j/U_{jj}$
7. **end** j

Diagonální prvky faktoru L jsou samozřejmě jednotkové a není zapotřebí je ukládat. Permutace prvků vektoru b odpovídá částečné pivotaci a dohromady se tak algoritmus dá vyjádřit jako LU rozklad s částečnou pivotací. Schéma dat sloupcového algoritmu je znázorněno na následujícím obrázku, kde ale musím brát do úvahy, že řádky jsou postupně permutovány částečnou pivotací.



Klíčem k efektivní implementaci je rychlé nalezení těch dříve spočítaných sloupců faktoru L, které vystupují v aktualizaci v kroku 3 algoritmu, to jest struktura řídkosti u_j. Jak jsme uvedli výše v textu týkajícím se řádkového rozkladu, tuto strukturu lze nalézt hledáním množiny dosažitelnosti v orientovaném acyklickém grafu $G(L_{1:j-1,1:j-1})$ prohledáváním z počáteční množiny $A_{1:j-1,j}$. Aktualizace konkrétního sloupce se mohou pak aplikovat ve stejném pořadí. Následně je zřejmé, že počet symbolických operací nepřevýší řádově počet aritmetických operací LU rozkladu.

10.1.6 Nesymetrická multifrontální metoda

V případě LU rozkladu nesymetrických matic je několik možností, jak zkonstruovat nesymetrickou multifrontální metodu. Na základě symetrické a pozitivně definitní mutifrontální metody bychom rádi, aby analogicky platily následující dva vztahy:

- (a) Řádky, respektive sloupce nesymetrické frontální matice $\mathcal{F}_k$, odpovídají řádkům, kde jsou nenulové prvky na řádcích k-tého sloupce a L_{*k} faktoru L, respektive nenulovým prvkům ve sloupcích k-tého řádku U_{k*} faktoru U.

- (b) Každá aktualizační matice $\mathcal{U}_r$ může být vnořena pouze do jedné frontální matice.

Bohužel, tyto požadavky nelze současně splnit. V praxi tak existují různé přístupy. Obvyklý přístup, který splňuje podmínku (b), ale nikoli (a) používá strukturu symetrické matice $A + A^T$, pro kterou lze zkonstruovat grafové objekty stejně jako v případě matice symetrické a pozitivně definitní. Matici s případnými nulovými prvky v prvním řádku nebo sloupci lze pak rozložit na několik menších hustých matic. Druhou možností je předpokládat platnost (a) a rozložit aktualizační matici na několik matic. V každém případě, není-li rozkládaná matice silně regulární, pak je zapotřebí v jednotlivých krocích rozkladu hledat hlavní prvek, jak o tom budeme hovořit později.

Postup grafového modelu nesymetrické multifrontální metody založené na platnosti bodu (a) výše demonstrujeme na následujícím příkladě. Zdůrazněme, že frontální matice $\mathcal{F}_k$ je vždy určena řádky a sloupci, kde jsou, po řadě nenulové prvky L_{*k} a U_{k*}. Pro dotvoření frontální matice budeme pak používat pouze části aktualizačních matic. To formálně znázorníme rozdělením aktualizačních matic na části, ale nebudeme zde hovořit o formálním postupu takového rozdělení. Také zdůrazněme to, že se jedná **pouze o grafový model multifrontální metody**, protože v praxi nemáme zabezpečenu silnou regularitu matice a je třeba provádět výběr hlavního prvku, jak o tom budeme hovořit v dalším textu. Uvažujme následující matici, kde zároveň znázorňujeme zaplnění při standardním eliminačním pořadí.

$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{pmatrix}
* & & * & & & & & & & \\
* & * & f & & * & & & * & & * \\
* & & * & * & & & & & & \\
 & & * & * & & & & & * & * \\
 & & * & * & * & & & * & f & f \\
 & * & f & f & f & * & & f & f & f \\
 & & & * & & & * & & f & f \\
* & * & * & f & f & f & & * & f & f \\
 & & & & & & & & * & * \\
 & & & & * & f & * & f & f & *
\end{pmatrix}
\end{array}
$$

První frontální matice (určená zmíněnými řádkovými a sloupcovými indexy) a příslušná aktualizační matice jsou

$$
\mathcal{F}_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 8 \end{array}
\begin{array}{c} \begin{array}{cc} 1 & 3 \end{array} \\
\begin{pmatrix} * & * \\ * & \\ * & \\ * & \end{pmatrix} \end{array}, \quad
\mathcal{U}_1 = \begin{array}{c} \\ 2 \\ 3 \\ 8 \end{array}
\begin{array}{c} 3 \\ \begin{pmatrix} f \\ f \\ f \end{pmatrix} \end{array}.
$$

Pro druhou frontální matici můžeme pro její aktualizaci použít pouze část $\mathcal{U}_1$, jinak by nesplňovala podmínku (a). Tuto aktualizační matici tedy rozdělíme následovně.

$$
\mathcal{U}_1^2 = \begin{array}{c} \\ 2 \end{array}
\begin{array}{c} 3 \\ \begin{pmatrix} f \end{pmatrix} \end{array}, \; \mathcal{U}_1^3 = \begin{array}{c} \\ 3 \\ 8 \end{array}
\begin{array}{c} 3 \\ \begin{pmatrix} f \\ f \end{pmatrix} \end{array}, \; \mathcal{U}_1 = \mathcal{U}_1^2 \diamondsuit \mathcal{U}_1^3.
$$

Druhá frontální a aktualizační matice pak mohou být zapsány

$$\mathcal{F}_2 = \begin{array}{c} \\ 2 \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 2 & 3 & 5 & 8 & 10 \end{array} \\ \begin{pmatrix} * & f & * & * & * \\ * & & & & \\ * & & & & \end{pmatrix} \end{array} \equiv \begin{array}{c} \\ 2 \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 2 & 5 & 8 & 10 \end{array} \\ \begin{pmatrix} * & * & * & * \\ * & & & \\ * & & & \end{pmatrix} \end{array} \Diamond\!\!\!\!+ \mathcal{U}_1^2,\ \mathcal{U}_2 = \begin{array}{c} \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 3 & 5 & 8 & 10 \end{array} \\ \begin{pmatrix} f & f & f & f \\ f & f & f & f \end{pmatrix} \end{array}.$$

Třetí frontální matice je pak vytvořena následujícím způsobem

$$\mathcal{F}_3 = \begin{array}{c} \\ 3 \\ 4 \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 3 & 4 \end{array} \\ \begin{pmatrix} * & * \\ * & \\ f & \\ * & \end{pmatrix} \end{array} \equiv \begin{array}{c} \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 3 & 4 \end{array} \\ \begin{pmatrix} * & * \\ * & \end{pmatrix} \end{array} \Diamond\!\!\!\!+ \mathcal{U}_1^3 \Diamond\!\!\!\!+ \mathcal{U}_2^3,\ \mathcal{U}_3 = \begin{array}{c} \\ 4 \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{c} 4 \end{array} \\ \begin{pmatrix} f \\ f \\ f \end{pmatrix} \end{array}$$

pro následující rozklad aktualizační matice $\mathcal{U}_2$

$$\mathcal{U}_2 = \begin{array}{c} \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{c} 3 \end{array} \\ \begin{pmatrix} f \\ f \end{pmatrix} \end{array} \Diamond\!\!\!\!+ \begin{array}{c} \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{c} 5 \end{array} \\ \begin{pmatrix} f \\ f \end{pmatrix} \end{array} \Diamond\!\!\!\!+ \begin{array}{c} \\ 6 \\ 8 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 8 & 10 \end{array} \\ \begin{pmatrix} f & f \\ f & f \end{pmatrix} \end{array} \equiv \mathcal{U}_2^3 \Diamond\!\!\!\!+ \mathcal{U}_2^5 \Diamond\!\!\!\!+ \mathcal{U}_2^6.$$

Analogicky postupujeme i v následujících krocích. Existuje více možností, jak aktualizační matice rozložit, ale zde se jimi nebudeme podrobněji zabývat. I proto, že v praktických algoritmech musíme grafový model obvykle modifikovat.

Zdůrazněme nyní ještě souvislost postupu rozkladu v multifrontální metodě s nesymetrickým eliminačním stromem. Ta nám umožňuje efektivní implementaci a zobecňuje analogický vztah pro standardní symetrický eliminační strom. Konkrétně, Věta 10.1.17 ukazuje, že všechny prvky podmatice aktualizační matice $\mathcal{U}_k$ s indexy alespoň $parent(k)$ mohou být vždy přičteny pomocí operace $extend - add$ do frontální matice $\mathcal{U}_{parent(k)}$.

Věta 10.1.17 *Nechť $p = parent(k)$ v nesymetrickém eliminačním stromu $T(A)$. Za předpokladu nevyrušení platí*

$$\mathcal{S}(L_{*k})\ \{1, \ldots, p-1\} \subseteq \mathcal{S}(L_{*p}) \tag{10.21}$$

a také

$$\mathcal{S}(U_{k*})\ \{1, \ldots, p-1\} \subseteq \mathcal{S}(U_{p*}). \tag{10.22}$$

Důkaz: Podle definice otce v nesymetrickém eliminačním stromu dané matice A platí $p \overset{L}{\Rightarrow} k \overset{U}{\Rightarrow} p$. Je-li $u_{ij} \neq 0$ pro $i < j$, pak je násobek i-tého sloupce přidán k j-tému sloupci jako příspěvek příslušné eliminační matice $A^{(i)}$. Všechny nenulové prvky $\mathcal{S}(L_{*k})\ \{1, \ldots, p-1\}$ jsou tedy postupně předávány podle tohoto principu v krocích eliminace s vrcholy na cestě $k \overset{U}{\Rightarrow} p$ do eliminačních matic až je nakonec tato struktura řídkosti obsažena ve struktuře řídkosti $\mathcal{S}(L_{*p})$. Analogicky se dokáže druhé tvrzení na základě nenulových prvků faktoru L na cestě $p \overset{L}{\Rightarrow} k$. ■

10.2 Diagonální dominance a blokově trojúhelníkový tvar

V předcházející kapitole jsme probrali několik přístupů, které se zabývají hlavně symbolickou strukturou LU rozkladu. Většina z nich předpokládá, že matice je silně regulární, ale některé umožňují i částečnou pivotaci. Silná regularita rozkládané matice je v každém případě pravděpodobnější, když umožníme, aby její diagonální prvky patřily do její struktury řídkosti, to jest, aby byly nenulové, nebo dokonce, aby obsahovaly prvky s co největšími absolutními hodnotami. Varianty obou těchto úloh se dají definovat pomocí některých grafových modelů.

Uvažujme bipartitní grafový model zavedený v Podsekci 6.1.4, který umožní zachytit obecně nesymetrické permutace, které jsou potřeba k tomu, aby bylo možné permutací dosáhnout jejich změn z nulových na nenulové či naopak. K tomu je nesymetrická permutace je nutná, protože symetrická permutace matice není schopna odstranit nulové diagonální prvky matice. Na grafovém modelu je dokonce vidět i to, že na dosažení silné regularity stačí permutace jednostranná, řádková nebo sloupcová, což je v souladu s tím, že silnou regularitu regulární matice zabezpečí částečná pivotace. Detailnějším vlivem na stabilitu rozkladu se zde ale nebudeme zabývat.

Uvažujme Obrázek 10.2.14. Struktura řídkosti druhé zobrazené matice vznikne z první přehozením tří sloupců. Touto permutací sloupců se nám jeden diagonální prvek změnil na nulový. V následujícím odstavci uvidíme, jak tohoto faktu s úspěchem využít s přesně opačným úmyslem. A sice, budeme se snažit dostat na diagonální pozice matice co nejvíce nenulových prvků. Tím se zvýší **pravděpodobnost**, že rozkládaná matice je silně regulární.

10.2.1 Nesymetrické permutace pro získání nenulové diagonály matice nebo posílení diagonální dominance matice

Zaveďme nejprve pojem párování v obecném grafu.

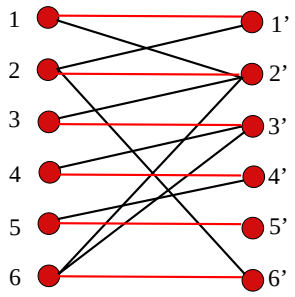
Definice 10.2.1 **Párováním** *prostého neorientovaného grafu* $G = (V, E)$ *se nazývá jeho podgraf* $M = (V_M, E_M)$ *indukovaný množinou vrcholově disjunktních hran* E_M, $E_M \subseteq E$.

Definice 10.2.2 *Párování* M *v grafu* G*, které má maximální počet hran* E_M *ze všech možných párování, se nazývá* **maximální párování** *grafu* G*. Párování* **maximální velikosti ve smyslu inkluze** *se nazývá takové párování* $M = (V_M, E_M)$*, pro které neexistuje párování s větším počtem hran než je* $|E_M|$ *a které obsahuje všechny hrany z* E_M*. Párování je* **perfektní***, má-li přesně* $|V|/2$ *hran.*

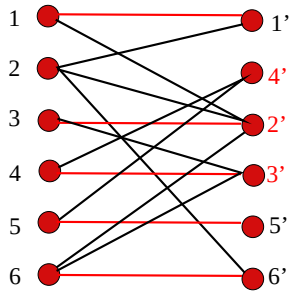
Na Obrázku 10.2.15 vidíme znázorněno maximální párování ve smyslu inkluze i maximální párování.

Zajímavá je souvislost mezi párováním v grafu, které je někdy zavedeno jen jako množina hraň a nikoli jako podgraf a speciální množinou vrcholů nazývanou pokrytí, kterou zavedeme v následující definici.

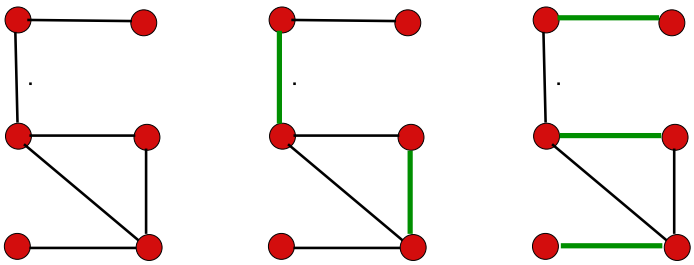
$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1' & 2' & 3' & 4' & 5' & 6' \end{array} \\
\begin{pmatrix}
* & * & & & & \\
* & * & & & & * \\
 & * & * & & & \\
 & & * & * & & \\
 & & & * & * & \\
 & * & * & & & *
\end{pmatrix}
\end{array}
$$



$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1' & 2' & 3' & 4' & 5' & 6' \end{array} \\
\begin{pmatrix}
* & & * & & & \\
* & & * & & & * \\
 & & * & * & & \\
 & * & & * & & \\
 & * & & & * & \\
 & & * & * & & *
\end{pmatrix}
\end{array}
$$



Obrázek 10.2.14: *Příklad změny v počtu nulových prvků na diagonále matice pomocí nesymetrické permutace matice.*



Obrázek 10.2.15: *Příklad neorientovaného grafu (vlevo), grafu s maximálním párováním ve smyslu inkluze (uprostřed) a maximálním párováním (vpravo). Hrany párování jsou zobrazeny silněji.*

Definice 10.2.3 *Jsou-li vrcholy i a j koncovými vrcholy nějaké hrany e neorientovaného grafu G, řekneme, že hrana e tyto vrcholy* **pokrývá**. *Také řekneme, že vrchol i pokrývá hranu e a vrchol j pokrývá hranu e. Množinu $K \subseteq V$ nazveme* **vrcholovým pokrytím** *grafu $G = (V, E)$, je-li každá jeho hrana pokryta alespoň jedním vrcholem z množiny K.* **Minimálním** *vrcholovým pokrytím množiny E* **ve smyslu inkluze** *se nazývá takové pokrytí $K \subseteq V$, jehož žádná vlastní podmnožina již není pokrytím E. Minimálním vrcholovým pokrytím množiny E, je takové její pokrytí že neexistuje její pokrytí množinou z V menší kardinality.*

V dalším textu se budeme soustředit na **párování v bipartitních grafech**. Především nás zajímají taková párování v bipartitních grafech, kde každý vrchol R respektive C je pokryt nějakou hranou párování. Taková párování budeme nazývat **řádkově** respektive **sloupcově perfektní**. Párování, která jsou řádkově i sloupcově perfektní budeme nazývat **perfektní párování.** Všechna perfektní párování bipartitního grafu jsou zřejmě maximální. Následující klasická věta teorie grafů ukazuje souvislost maximálního párování v bipartitním grafu s jeho **minimálním pokrytím**. Větu si zde dokážeme, i když není pro další text bezprostředně nutná.

Věta 10.2.1 *Königova věta: Velikost maximálního párování $\nu(G)$ v bipartitním grafu (R, C, E) je rovna velikosti minimálního pokrytí $\tau(G) = min_{X \subseteq C}(|C \setminus X| + |Adj(X)|)$ tohoto grafu.*

Důkaz: V důkaze postupujeme podle [136]. Zřejmě platí $\nu(G) \leq \tau(G)$ neboť každá hrana vrcholově disjunktních hran párování musí přispět alespoň jedním vrcholem do pokrytí G. Předpokládejme nyní, že G je nejmenší takový graf, pro který tvrzení neplatí, pro který je tedy

$$\nu(G) < \tau(G).$$

Tento graf je souvislý neboť jinak by nebyl minimální, protože nějaká jeho komponenta by musela také splňovat předpoklad. Dále, G má alespoň jeden vrchol stupně aspoň 3, protože pro cesty a cykly tvrzení věty zřejmě platí. Označme tento vrchol u a nechť v je jeden z jeho sousedů. Uvažujme nyní dva případy.

I. Nechť platí pro velikosti maximálního párování

$$\nu(G \setminus v) < \nu(G).$$

Z vlastnosti minimality musí mít tento zmenšený graf $G \setminus v$ pokrytí, které má velikost menší než $\nu(G)$. Označme si toto pokrytí W'. A potom je tedy $W' \cup \{v\}$ pokrytí o velikosti nejvýše $\nu(v)$. Tudíž $\tau(G) \leq \nu(G)$ a to je spor.

II. Nechť je tedy

$$\nu(G \setminus v) = \nu(G).$$

Pak tedy musí existovat maximální párování M grafu G, které nemá žádnou hranu incidentní vrcholu v. Nechť f je hrana $G \setminus M$ incidentní u a nikoli v. Taková hrana existuje, protože u má stupeň alespoň 3. Nechť dále W'' je pokrytí grafu $G \setminus f$ pro které je $|W''| = \nu(G)$. Z předpokladu minimality víme, že takové pokrytí musí existovat. Protože žádná hrana tohoto párování M neobsahuje v, tak ani W'' neobsahuje v. Proto musí obsahovat vrchol u a je tedy i pokrytím grafu G. Věta je dokázána. ■

Následující pozorování 10.2.1 ukazuje význam perfektního párování pro hledání nesymetrických permutací, které mohou umožnit umístění nenulových prvků na diagonálu matice.

Pozorování 10.2.1 *Existuje-li perfektní párování bipartitního grafového modelu čtvercové matice A $G(A) = (R, C, E)$, kde $|R| = |C|$ s množinou hran $E_M = \{(i_k, j_k) \mid k = 1, \ldots, n\}$, pak existují permutační matice P a Q příslušné dimenze takové, že diagonálními prvky matice PAQ jsou tvořeny množinou*

$$\{a_{i_k,j_k} \mid k = 1, \ldots, n\}.$$

Snadno nahlédneme, že permutace z tohoto pozorování může být jednostranná, tedy že stačí uvažovat permutovanou matici PA nebo AQ. V každém případě je permutace ale obecně nesymetrická. Demonstraci Pozorování 10.2.1 je možné vidět na Obrázku 10.2.14. Umístění nenulových prvků matice na její diagonálu neimplikuje nutně, že tato matice je silně regulární. Nicméně, v mnoha ohledech může být permutovaná matice mnohem vhodnější k rozkladu než původní matice. Problém nalezení této permutace se tedy dá převést na problém nalezení maximálního párování v bipartitním grafu. Následující dvě definice naznačují, jak maximální párování hledat. Totiž, převádějí tento problém na hledání zvětšujících cest v grafu G. Zde se budeme pro jednoduchost zabývat jen hledáním maximálního párování v bipartitním grafu, ale princip z dvou následujících definic je obecnější.

Definice 10.2.4 Střídavá cesta *grafu G s párováním $M = (V_M, E_M)$ je cesta, jejíž hrany jsou střídavě z M a z $G(V, E \setminus E(M))$ (střídavě patří nebo nepatří párování).*

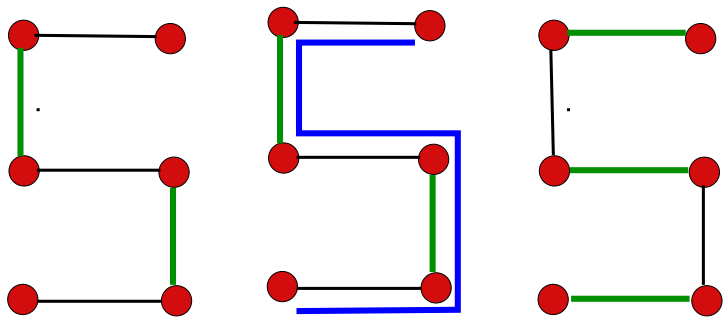
Definice 10.2.5 *Netriviální střídavá cesta (s alespoň jednou hranou), s oběma koncovými vrcholy v grafu $G \setminus M$ se nazývá* **zvětšující cesta**.

Praktická procedura zvětšování párování pomocí zvětšujících cest je znázorněna na následujícím obrázku. V praxi procedura ke zvětšení efektivity vytváří střídavý prohledávací strom podobně jako tomu je v prohledávání grafu do šířky, jak o tom budeme mluvit později. Existuje mnoho praktických algoritmů, viz například Gustavson (1976) či [55], kde je implementován algoritmus se složitostí $O(|E|\sqrt{n})$ algoritmu tak jak jej navrhli v roce 1973 Hopcroft a Karp [88]; viz také [49]. Umístění **libovolných** nenulových prvků na diagonálu nemusí být nejlepší volbou a proto existují procedury, které se snaží najít takové párování, aby byla permutovaná matice co nejvíc **diagonálně dominantní**. To vede na problém **váženého párování**, kterým se zde ale nebudeme zabývat. Později uvidíme, jak jsou maximální párování důležitá v přibližných rozkladech.

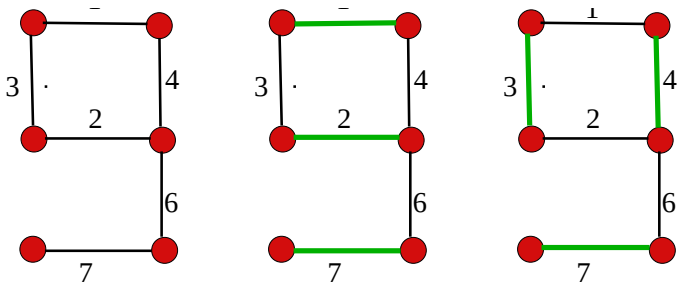
Velikost maximálního párování v bipartitním grafu se někdy nazývá **strukturní hodnost** matice (structural rank). To, že pro hodnost $rank(A)$ čtvercové matice A a pro strukturní hodnost $srank(A)$ čtvercové matice vždy platí

$$rank(A) \leq srank(A) \tag{10.23}$$

je snadno vidět například z definice determinantu matice A.



Obrázek 10.2.16: *Znázornění procedury hledání maximálního párování v bipartitním grafu pomocí zvětšujících cest.*



Obrázek 10.2.17: *Znázornění procedury hledání maximálního párování v bipartitním grafu pomocí zvětšujících cest.*

10.3 Maticové bloky a silná Hallova vlastnost

Praktické provedení LU rozkladu může značně těžit z faktu, že matice je rozložitelná. Jak je uvedeno v Definici 3.2.2, pro nesymetrickou matici to znamená, že matici lze permutovat symetrickou permutací na blokově trojúhelníkový tvar. K lepšímu popisu rozložitelnosti si zaveďme pojem **faktorgrafu**, což je graf odvozený z rozkladu množiny V pro nějaký daný graf (V, E). Jeho hrany určuje relace sousednosti mezi množinami vrcholů.

Definice 10.3.1 *Nechť $\mathcal{P} = \{P_1, \ldots, P_p\}$ je rozklad množiny vrcholů orientovaného grafu $G = (V, E)$ (viz Definice 2.1.2).* **Faktorgrafem** *grafu G vzhledem k rozkladu $\mathcal{P}$ je graf $G/\mathcal{P} = (\mathcal{P}, \mathcal{E}_\mathcal{P})$, kde $(P_i, P_j) \in \mathcal{E}_\mathcal{P}$ pro $1 \leq i, j \leq p$, $i \neq j$, právě tehdy, platí-li*

$$P_j \cap \{k \in V \mid (\exists i \in P_i)((i, k) \in E)\} \neq \emptyset.$$

Faktorgrafem neorientovaného grafu G budeme rozumět symetrizaci faktorgrafu jeho libovolné orientace. Zaveďme si následující definicí pojem kondenzace grafu, která umožňuje přehledně zachytit strukturu nerozložitelných podgrafů matice.

Definice 10.3.2 *Nechť $G = (V, E)$ je orientovaný graf a nechť $V_1, \ldots, V_k$ jsou množiny jeho vrcholů, které odpovídají jeho silným komponentám. Kondenzací grafu G nazveme orientovaný graf $G_C = (V', E')$, kde $V' = \{V_1, \ldots, V_k\}$ a kde*

$$E' = \{(V_i, V_j) \mid (\exists x \in V_i)(\exists y \in V_j)((x, y) \in E)\}.$$

Kondenzace je tedy speciální případ faktorgrafu, ve kterém je rozklad množiny vrcholů určen ekvivalencí silné souvislosti. Snadno se dokáže tvrzení následující věty, která vyjadřuje základní vlastnost kondenzace grafu.

Věta 10.3.1 *Kondenzace G_C grafu G neobsahuje žádný orientovaný uzavřený sled a je tedy orientovaný acyklický graf.*

Důkaz: Předpokládejme, že daná kondenzace takový uzavřený sled obsahuje. Jeho délka pro prosté grafy, které používáme v tomto textu a kde neuvažujeme smyčky, je nejméně 2. Existují tedy dva různé vrcholy tohoto sledu, ale by měly patřit do stejné souvislé komponenty grafu. To je spor s definicí kondenzace. ■

Důsledek 10.3.1 *Vrcholy kondenzace mohou být topologicky očíslovány.*

S pojmem kondenzace souvisí důležitá třída permutací, která je uvedena v následující větě.

Věta 10.3.2 *Pro matici A dimenze n existují permutační matice P příslušné dimenze a přirozené číslo $t \geq 1$ takové, že*

$$PAP^T = \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1t} \\ 0 & A_{22} & \ldots & A_{2t} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & A_{tt} \end{pmatrix}, \tag{10.24}$$

kde $A_{11}, A_{22}, \ldots, A_{tt}$ jsou čtvercové nerozložitelné matice. Matice $A_{11}, A_{22}, \ldots, A_{tt}$ jsou určeny jednoznačně až na symetrické permutace jejich řádků a sloupců, ale jejich pořadí na blokové diagonále matice není nutně jednoznačné.

Důkaz: Dokážeme pouze první část tvrzení a nikoli tvrzení o jednoznačnosti. Snadný důkaz celého tvrzení lze najít například v [24]. Nechť $V_1, \ldots V_t$ jsou množiny vrcholů silných komponent orientovaného grafu matice A. Kondenzace tohoto grafu je acyklický graf s topologickým očíslováním. V důkazu Věty 10.3.1 jsme ukázali, že v ní musí existovat zdroj. Vrcholy, které odpovídají komponentě zdroje očíslujeme jako první a uvažujeme podgraf kondenzace indukovanými zbylými vrcholy. Vrcholy zdroje tohoto podgrafu kondenzace očíslujeme následovně a tak postupujeme dále. Po konečném počtu kroků očíslujeme takto všechny vrcholy grafu. Výsledná matice v permutaci dané vytvořeným novým očíslováním má zřejmě bloky poddiagonální části nulové. ■

Tvar matice v (10.24) se nazývá **Frobeniova normální forma** matice a je tedy speciálním případem horního blokově trojúhelníkového tvaru zavedeného v Definici 3.2.1. Popis rozložitelnosti matice založený na silných komponentách úzce souvisí s následujícími dvěma vlastnostmi řídké matice. Tyto vlastnosti navíc dobře vyjadřují i vlastnosti řídkého QR rozkladu, o kterém budeme hovořit později.

Definice 10.3.3 *Mějme matici $A \in R^{m \times n}$, kde $m \geq n$. Řekneme, že A má* **Hallovu vlastnost** *jestliže každá množina jejích sloupců o velikosti k má nenulové prvky v alespoň k řádcích. Řekneme, že A má* **silnou Hallovu vlastnost** *jestliže každá množina jejích sloupců o velikosti $k, k = 1, \ldots, n-1$ má nenulové prvky v alespoň $k+1$ řádcích. Analogicky můžeme definovat i Hallovu a silnou Hallovu vlastnost i pro bipartitní graf, který modeluje strukturu nenulových prvků matice A.*

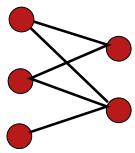
Následující tvrzení ukazuje, že Hallova vlastnost je nutnou podmínkou pro regularitu matice. Existence perfektního párování je podle následující Věty ekvivalentní tomu, že $srank(A) = n$ a to je nutnou podmínkou pro regularitu podle (10.23).

Věta 10.3.3 *Čtvercová matice $A \in R^{n \times n}$ má Hallovu vlastnost právě tehdy, když v jejím bipartitním grafovém modelu existuje perfektní párování.*

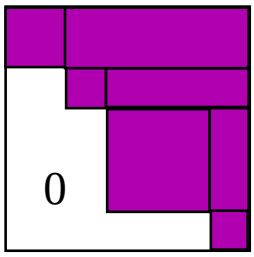
Hallovu vlastnost má například i diagonální matice nebo horní trojúhelníková matice s nenulovými diagonálními prvky. Příklad bipartitního grafu matice, která má silnou Hallovu vlastnost, je na Obrázku 10.3.18. Silná Hallova vlastnost požaduje po matici ještě další vlastnost, kterou popisuje Věta 10.3.4.

Věta 10.3.4 *Čtvercová matice má silnou Hallovu vlastnost právě tehdy, když její orientovaný grafový model je silně souvislý.*

Vidíme tedy, že nemá-li matice silnou Hallovu vlastnost, pak to podle Věty 10.3.4 znamená, že její orientovaný grafový model má více komponent. Matici lze tedy podle Věty 10.3.2 převést permutací na Frobeniovu normální formu. Schématicky je tvar Frobeniovy



Obrázek 10.3.18: *Příklad bipartitního grafu matice mající silnou Hallovu vlastnost.*



Obrázek 10.3.19: *Ukázka blokově trojúhelníkového tvaru.*

$$
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5\\ 6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array}\\
\begin{pmatrix}
* & * & & * & * & \\
 & * & & & * & \\
 & * & * & & * & * \\
 & * & & * & & \\
* & & & & * & \\
 & * & * & & & *
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c}
\\ 6\\ 3\\ 5\\ 4\\ 1\\ 2
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 6 & 3 & 5 & 4 & 1 & 2 \end{array}\\
\begin{pmatrix}
* & & * & & * & \\
* & * & * & & & * \\
 & & * & & * & \\
 & & & * & * & \\
 & & * & * & * & * \\
 & & * & & & *
\end{pmatrix}
\end{array}
$$

Obrázek 10.3.20: *Ukázka permutace matice na blokově trojúhelníkový tvar.*

normální formy znázorněn na Obrázku 10.3.19. Permutaci na tento tvar zachycujeme ještě na Obrázku 10.3.20

Shrňme si tedy, že algoritmus nalezení blokově trojúhelníkového tvaru může být založen na hledání silně souvislých komponent orientovaného grafového modelu. Získaný orientovaný faktorgraf, který se nazývá kondenzace, je acyklický a jeho vrcholy lze uspořádat tak, že výsledná matice je ve Frobeniově normální formě, která má blokově trojúhelníkový tvar. Navržení postupu hledání silných komponent grafu a této permutace přenecháváme jako cvičení (viz Tarjan (1972); maticová implementace Duff, Reid (1978); SCC algoritmus Kosaraju, Sharir (1981), Gabow (2000), viz také Corman, Leiserson, Rivest, Stein (2001) and Davis (2006)) Následující poznámka ukazuje, že permutace matice na blokově trojúhelníkový tvar zachová množinu diagonálních prvků, kterou jsme mohli získat v předcházejícím kroku nalezením párování v bipartitním grafu a příslušnou jednostrannou permutací.

Poznámka 10.3.1 *Přechod na Frobeniovu normální formu lze provést* **pouze symetrickými permutacemi**, *které zachovávají diagonální prvky.*

Blokově trojúhelníkový tvar matice implikuje, že v LU rozkladu takto uspořádané matice stačí rozkládat pouze diagonální bloky, což může výrazně zvýšit jeho efektivitu. Uvažujme soustavu rovnic s maticí z Věty 10.3.2 podle následujícího obrázku.

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1t} \\ 0 & A_{22} & \dots & A_{2t} \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & A_{tt} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_t \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_t \end{pmatrix} \tag{10.25}$$

Procedura řešení je v Algoritmu 10.3.1.

Algoritmus 10.3.1 Řešení soustavy rovnic s maticí v blokově trojúhelníkovém tvaru.
Input: *Soustava rovnic s maticí v blokově trojúhelníkovém tvaru.*
Output: *Řešení x.*

1. **for** $j = 1 : t$ **do**
2. Nalezni LU rozklad $A_{ii} = L_i U_i$
3. **end** j
4. **for** $j = t : 1$ **do**
5. $s_j = b_j$
6. **for** $i = j + 1 : t$ **do**
7. $s_j = s_j - A_{ji} x_i$
8. **end** i
9. $x_j = U_j^{-1} L_j^{-1} s_j$
10. **end** t

Souhrnně je možné říci, že procedury, které umísťují nenulové a případně co největší prvky matice na její diagonálu a permutují ji na blokové trojúhelníkový tvar jsou **základní procedury předzpracování** LU rozkladu řídkých matic.

11

Nalezení počátečního uspořádání matice

Počáteční uspořádání je poslední dosud blíže neprobraný krok Algoritmu 9.2.1. Toto uspořádání je naprosto podstatné pro zmenšení **velikosti zaplnění**, ale může plnit i další úkoly. Význam dobrého uspořádání pro Choleského faktorizaci jsme viděli na příkladu šípové matice na Obrázcích 9.1.1 a 9.1.2. Matice, která má své nenulové prvky uspořádány ve tvaru šípu, bude mít množinu hran grafu zaplnění stejnou jako původní množinu hran. Matice, kterou získáme její symetrickou permutací, indukovanou například přečíslováním vrcholů grafu $n, 1, \ldots, n-1$ má graf zaplnění úplně hustý.

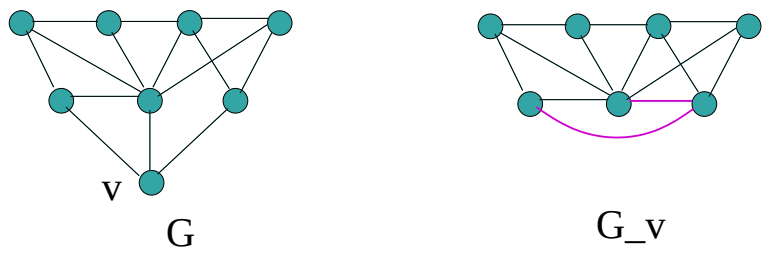
V následujících podkapitolách postupně probereme základní typy počátečního uspořádání. Algoritmy, které zde uvedeme, jsou z velké části pouze pouze **přibližnými algoritmy** neboli **heuristikami.** Většina z nich je totiž při snaze o splnění určitého kritéria optimality NP-těžká ve smyslu výše uvedeného popisu složitostí. V celé kapitole předpokládáme, že matice A pro kterou hledáme počáteční uspořádání je čtvercová a symetrická. To umožní uvažovat v celé kapitole model neorientovaného grafu. V případě hledání počátečního uspořádání pro nesymetrické matice a LU rozkladu je možné vyjít z grafu matice $A + A^T$.

11.1 Uspořádání minimálního stupně a zaplnění

Tato podkapitola je věnována uspořádáním, která **lokálně minimalizují** nějakou **algoritmicky snadno dostupnou** charakteristiku struktury řídkosti řídké matice, která je spojena s jejím zaplněním při rozkladu, bez ohledu na tvar struktury řídkosti matice. Jejich charakteristickým představitelem je uspořádání minimálního stupně, kde je ona snadno dostupná charakteristika stupeň vrcholu.

11.1.1 Základní algoritmus minimálního stupně

Algoritmus 11.1.1 je základním schématem uspořádání minimálního stupně. Výstupem z algoritmu je nové očíslování vrcholů grafu dané symetrické matice $G = (\{1, \ldots, n\}, E)$, které indukuje přeuspořádání této matice. Protože v něm tvoříme eliminační grafy nikoli ve standardním pořadí, jak jsme uváděli v textu věnovaném Gaussově eliminaci, budeme



Obrázek 11.1.1: *Ukázka eliminačního kroku v Algoritmu 11.1.1. Nalevo je původní graf, napravo je redukovaný graf G_v po eliminaci vrcholu v.*

potřebovat redukovaného eliminačního grafu po eliminaci obecného vrcholu. Konkrétně, graf, který vznikne eliminací vrcholu v z grafu G označíme G_v. V zápise tedy standardně vynecháváme index R.

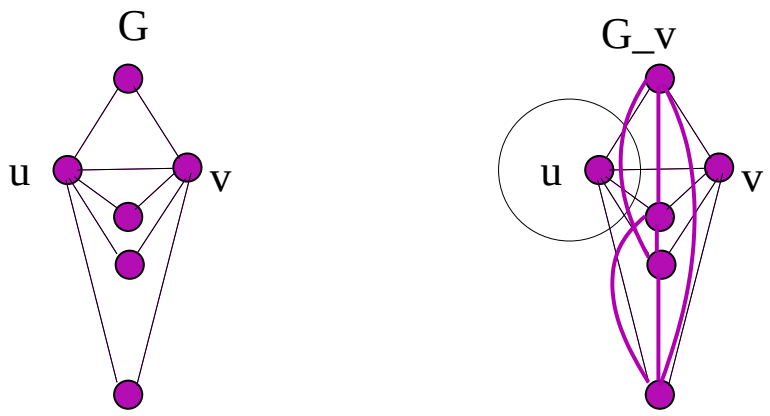
Algoritmus 11.1.1 Algoritmus minimálního stupně.
Input: *Neorientovaný graf $G = (\{1, \ldots, n\}, E)$ symetrické čtvercové matice.*
Output: *Nové uspořádání množiny vrcholů jako seznam $V' = (v_1, \ldots, v_n)$, které indukuje permutaci matice, pro kterou byl graf zkonstruován.*

1. *Polož $V' = \emptyset$*
2. **for** $i = 1 : n$ **do**
3. *Nalezni v_i takové, že $deg_G(v_i) = min_{v \in V(G)} deg_G(v)$*
4. *Vlož v_i na konec seznamu V'*
5. $G = G_v$
6. **end** i

V případě, kdy existuje více vrcholů splňujících podmínku minimality stupně v průběžném redukovaném grafu, Algoritmus 11.1.1 neurčuje vybíraný vrchol **jednoznačně**. Všechny vrcholy, které splňují podmínku minimality budeme nazývat **vrcholy minimálního stupně**. V Algoritmu není pravidlo, jak postupovat v případě, kdy existuje více vrcholů minimálního stupně. Existují dokonce příklady, kdy lze výběrem velmi negativně ovlivnit velikost zaplnění v grafu. Nicméně v konkrétních situacích je možné využít další informace o grafu úlohy. Například varianta algoritmu, kterou uvedeme později a kterou nazveme **násobný** algoritmus minimálního stupně, zmenšuje problémy s výběrem vrcholu z mnoha kandidátů se stejným nejmenším stupněm.

Příklad 11.1.1 *Jeden krok algoritmu minimálního stupně je znázorněn na Obrázku 11.1.1. Na levé straně je příklad grafu v nějakém kroku Algoritmu 11.1.1, kde vrchol minimálního stupně byl označen v, před jeho aktualizací. Na pravé straně je pak po aktualizaci.*

Ačkoli Algoritmus 11.1.1 vypadá jednoduše, velká časová náročnost je skryta v eliminačním kroku, který tvoří graf Schurova doplňku G_v. I proto budeme v dalším textu diskutovat možnosti, jak Algoritmus urychlit.



Obrázek 11.1.2: *Ukázka nerozlišitelnosti vrcholů* **u** *a* **v** *před a po eliminací vrcholu* **v**. *Nalevo je původní graf, napravo je graf po eliminaci vrcholu* **v** *doplněný o hrany zaplnění, kde jsme pro názornost nechali i hrany incidentní vrcholu* **v** *v* G.

11.1.2 Nerozlišitelnost a dominance

Aby počáteční uspořádání dobře plnilo svou funkci, je třeba, aby bylo efektivní a spotřebovalo pouze zlomek času potřebného k celé Choleského faktorizaci a mělo také rozumné paměťové nároky. Proto se nyní budeme algoritmem zabývat důkladněji a ukážeme, jak jej efektivně implementovat. Nejprve uvedeme definice některých situací, které se obecně v eliminačních grafech vyskytují a které se dají v implementacích využít.

Definice 11.1.1 *Dva různé vrcholy* **u** *a* **v** *grafu nazveme* **nerozlišitelné**, *jestliže platí*

$$Adj_G(u) \cup \{u\} = Adj_G(v) \cup \{v\}. \tag{11.1}$$

Příklad 11.1.2 *Nerozlišitelnost dvou různých vrcholů* **u** *a* **v** *před eliminací vrcholu* **v** *a po ní je znázorněna na Obrázku 11.1.2.*

Ukážeme si některé vlastnosti nerozlišitelných vrcholů.

Lemma 11.1.1 *Nechť* **u** *a* **v** *jsou nerozlišitelné vrcholy grafu* G. *Nechť dále* $y \in V(G)$, $y \neq u, v$. *Pak jsou* **u** *a* **v** *jsou také nerozlišitelné vrcholy grafu* G_y.

Důkaz: Nechť $u \notin adj_G(y)$. Pak platí také $v \notin adj_G(y)$ a relace nerozlišitelnosti vrcholů **u** a **v** zůstává v platnosti. Nechť nyní $u, v \in adj_G(y)$. Pak se množiny sousedů obou vrcholů u a v zmenší o jedničku, ale na jejich vzájemném vztahu podle Definice 11.1.1 se nic nezmění. ■

Lemma 11.1.2 *Nechť jsou vrcholy* **u** *a* **v** *nerozlišitelné vrcholy grafu* G. *Nechť dále* $y \equiv u$ *je vrchol minimálního stupně v* G. *Pak je vrchol* **v** *vrcholem minimálního stupně i v redukovaném eliminačním grafu* G_y.

Důkaz: Vrchol v bude mít stupeň v grafu G_y o jedničku menší než v grafu G, protože v něm nebude sousedem vrcholu u. Nerozlišitelné vrcholy jsou totiž vzájemně sousedními vrcholy. Sousedi vrcholů u a v se propojují do kliky a jejich stupeň přechodem k novému grafu tedy neklesne. U ostatních vrcholů zůstává stupeň stejný. Následně, vrchol v je vrcholem minimálního stupně v G_y. ■

Předcházející lemma jinými slovy říká, že **dva nerozlišitelné vrcholy mohou být eliminovány v algoritmu minimálního stupně v po sobě následujících krocích**. Definici nerozlišitelnosti a její důsledky pak snadno zobecníme na případ více vrcholů, jak uvádíme v následující poznámce.

Poznámka 11.1.1 *Skupina nerozlišitelných vrcholů v grafu G tvoří kliku, jejíž všechny vrcholy mají stejné množiny sousedů. Eliminací libovolného vrcholu této kliky zmenšíme stupeň všech zbývajících vrcholů kliky o jedničku a ostatní vrcholy této kliky zároveň zůstanou nerozlišitelné. Můžeme je tedy všechny eliminovat hned po sobě.*

Přirozeným důsledkem nerozlišitelnosti vrcholů je možnost použít pro eliminaci místo grafu **faktorgraf**, který je indukován rozkladem na množiny nerozlišitelných vrcholů, jinými slovy, je indukován **ekvivalencí nerozlišitelnosti**. Tyto množiny nerozlišitelných vrcholů lze na začátku algoritmu určit a v jeho průběhu aktualizovat. Poznamenejme, že důraz na efektivnost implementací obvykle neumožňuje nalézat vždy všechny nové skupiny nerozlišitelných vrcholů, což by bylo náročné, ale obvykle pouze některé z nich.

Velmi blízkým pojmem, který se dá v algoritmu minimálního stupně využít, je **dominance vrcholu nějakým jiným vrcholem**, ve smyslu minimálního stupně.

Definice 11.1.2 *Řekneme, že vrchol* **v** *je v grafu G dominován nějakým jiným vrcholem* **u**, *platí-li*

$$Adj_G(u) \cup \{u\} \subseteq Adj_G(v) \cup \{v\}. \tag{11.2}$$

Také s k tomuto pojmu uvedeme některá tvrzení, která ukazují cestu k zvýšení efektivnosti Algoritmu minimálního stupně. Předně je zřejmé, že pro vrcholy z Definice 11.1.2 platí

$$\deg_G(u) \leq deg_G(v).$$

Prakticky velmi dobře využitelné tvrzení je uvedeno jako Lemma 11.1.3.

Lemma 11.1.3 *Nechť u dominuje v v grafu G ve smyslu uvedené Definice. Nechť dále $y \neq u, v$ je vrchol minimálního stupně v G. Pak* **u** *dominuje* **v** *i v grafu G_y.*

Ukažme, jak prakticky využít tvrzení Lemmatu 11.1.3. Jak jsme viděli v Algoritmu 11.1.1, každý krok znamená změnu eliminačního grafu, která spočívá v odebrání hran spojených s eliminovaným vrcholem a propojení jeho sousedů. Pro tyto propojené sousedy musíme přepočítat jejich stupně, abychom mohli v dalším kroku rozhodnout o minimalitě dalšího eliminovaného vrcholu. Přepočítávání vrcholů při eliminaci vrcholu y můžeme stručně vyjádřit vztahy

$$v \notin Adj_G(y) \Rightarrow Adj_{G_y}(v) = Adj_G(v) \tag{11.3}$$

$$v \in Adj_G(y) \Rightarrow Adj_{G_y}(v) = (Adj_G(y) \cup Adj_G(v)) \setminus \{y\} \tag{11.4}$$

Máme-li situaci, kdy vrchol u dominuje vrchol v, pak do té doby, dokud u není eliminován, není zapotřebí si vrcholu v všímat, ve smyslu přepočítávání jeho stupně. Jak jsme uvedli výše, v praxi používáme rozklad indukovaný ekvivalencí nerozlišitelnosti na množině vrcholů grafu. Grafy, se kterými pracujeme, jsou tedy obecně faktorgrafy. Dominanci pak můžeme stejně jako nerozlišitelnost vztáhnout ke skupinám vrcholů, čili k vrcholům tohoto faktorgrafu. Počet kroků eliminačního procesu bude pak obecně menší než dimenze matice n a budeme jej značit N. Vzhledem k tomu, že počty vrcholů faktorgrafu upravujeme v jednotlivých krocích případným shlukováním nerozlišitelných vrcholů, není obvykle N známo dopředu.

11.1.3 Model eliminačního faktorgrafu

Samotný grafový model používaný v implementacích je obvykle dosti sofistikovaný. Důvodem je potenciálně velká paměťová náročnost úplných eliminačních faktorgrafů, do kterých vkládáme všechny nové hrany. To tedy znamená, že paměť potřebná pro jejich uložení, může pak být větší než paměť potřebná pro uložení hran původního grafu $G(A)$. Model **eliminačního faktorgrafu** definovaného níže umožňuje algoritmu minimálního stupně pracovat v paměťové složitosti $O(n + |E|)$. Tento pojem v praxi nahrazuje redukované eliminační grafy, pomocí kterých jsme definovali Algoritmus minimálního stupně.

Definice 11.1.3 Eliminační faktorgraf *Γ grafu $G = (V, E)$ je uspořádaná trojice $(\mathcal{S}, \mathcal{E}, E)$, kde $\mathcal{S} \cup \mathcal{E} = V, \mathcal{S} \cap \mathcal{E} = \emptyset$ a $E \subseteq \binom{\mathcal{S}}{2} \cup \binom{\mathcal{E}(\Gamma)}{2}$ je množina jeho hran.*

Na rozdíl od obecného faktorgrafu je tedy množina vrcholů V eliminačního faktorgrafu rozdělena na dvě disjuktní podmnožiny. Prvky $\mathcal{S}$ a $\mathcal{E}$ budeme nazývat opět vrcholy. Množinu sousedů vrcholu $r \in \mathcal{S} \cup \mathcal{E}$, které patří do $\mathcal{S}$, budeme značit $sadj(r)$. Množinu jeho sousedů vrcholu $r \in \mathcal{S} \cup \mathcal{E}$, které patří do $\mathcal{E}$ označíme $eadj(r)$. Postupně budeme vytvářet posloupnost eliminačních faktorgrafů $(\Gamma_0, \ldots, \Gamma_{N-1})$ následujícím způsobem. Nejprve položíme

$$\Gamma_0 = (\mathcal{S}_0, \mathcal{E}_0, E_0) \equiv (V(G), \emptyset, E(G)).$$

Eliminační faktorgraf $\Gamma_i = (\mathcal{S}_i, \mathcal{E}_i, E_i)$ pro $i = 1, \ldots, N-1$, zkonstruujeme ve dvou krocích. V **prvním kroku** položíme

$$\bar{\Gamma}_i = (\mathcal{S}_{i-1} \setminus \{i\}, \mathcal{E}_{i-1} \cup \{i\}, E_{i-1})$$

Druhým krokem je získání eliminačního faktorgrafu Γ_i z eliminačního faktorgrafu $\bar{\Gamma}_i$ tak, že všechny sousední vrcholy množiny $\mathcal{E}_i$ shlukneme do jednoho vrcholu faktorgrafu, najdeme dále nerozlišitelné vrcholy v $\mathcal{S}_i$ a tím se nám také může zmenšit počet vrcholů eliminačního faktorgrafu a vynecháme případné nadbytečné hrany a tím získáme množinu hran E_i.

Vidíme tedy, že množiny $\mathcal{E}_i$ slouží k zachycení eliminovaných vrcholů faktorgrafu. Ty by v redukovaném eliminačním grafu již nebyly. Také vidíme, že sousedy nějakého vrcholu z množiny $\mathcal{S}_i$ dosud neeliminovaných vrcholů našli jako sjednocení (i) jeho sousedů v této

množině a (i) množiny vrcholů, které jsou dosažitelné z $\mathcal{S}_i$ nějakého vrcholu cestou přes množinu vrcholů z $\mathcal{E}_i$.

Jinými slovy, sousedé vrcholu r z množiny $\mathcal{S}_i$ dosud neliminovaných vrcholů ve faktorgrafu Γ_i jsou určeny množinou dosažitelnosti $Reach_{\Gamma_i}(r)$ (viz Definici 5.1.4), která je dána výrazem

$$Reach_{\Gamma_i}(r) = sadj(r) \bigcup \left(\cup_{e \in eadj(r)} sadj(e) \right), \tag{11.5}$$

Pro zjednodušení jsme zde vynechali indexy faktorgrafu, ke kterému se množiny sousedů vztahují. Vidíme, že pojem eliminačního faktorgrafu je zobecněním eliminačního grafu, kde eliminaci vztahujeme zároveň na skupiny vrcholů a kde namísto struktury řídkosti Schurova doplňku hledáme sousedy pomocí cest přes skupiny eliminovaných vrcholů. Stejně jako v případě eliminace s nalézaným pořadím nových vrcholů faktorgrafu bude Γ_y označovat eliminační faktorgraf získaný z $\Gamma = (\mathcal{S}, \mathcal{E}, E)$ po eliminačním kroku s eliminovaným vrcholem faktorgrafu $y \in \mathcal{S}$.

Je zřejmé, že implementace algoritmu minimálního stupně pomocí modelu eliminačního faktorgrafu má výše zmíněnou rozumnou paměťovou složitost. Časovou náročnost silně redukuje používání supervrcholů, nalézání skupin nerozlišitelných vrcholů, redukce nadbytečných hran i využití dominance vrcholů, které vede k neúplné aktualizaci stupňů.

11.1.4 Násobný algoritmus minimálního stupně

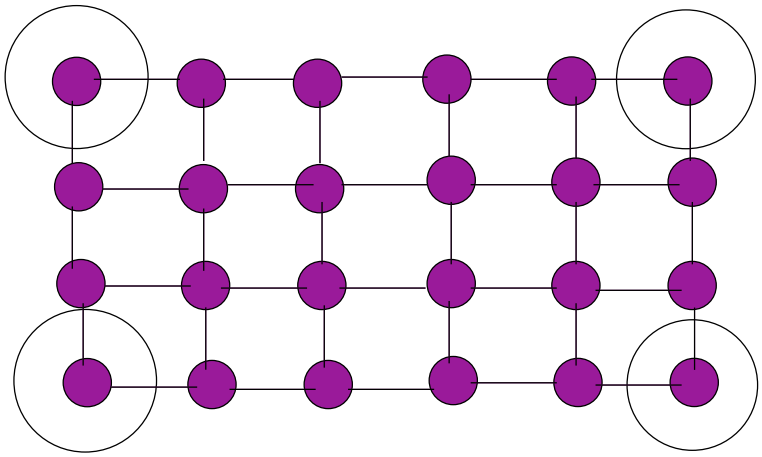
Významnou modifikací obecného schématu Algoritmu 11.1.1 je **násobná eliminace**. Schéma této modifikace je uvedeno jako Algoritmus 11.1.2.

Algoritmus 11.1.2 Algoritmus násobného minimálního stupně.
Input: *Neorientovaný graf* $G = (\{1, \ldots, n\}, E)$ *symetrické čtvercové matice,* $V' = \emptyset$.
Output: *Nové uspořádání množiny vrcholů jako seznam* $V' = (v_1, \ldots, v_n)$ *, které indukuje permutaci matice, pro kterou byl graf zkonstruován.*

1. *Polož* $V' = \emptyset$
2. **while** $G \neq \emptyset$ **do**
3. *Nalezni všechna* v'_j, $j = 1, \ldots, s$ *taková, že platí*
   $deg_G(v'_j) = min_{v \in V(G)} deg_G(v)$ *a zároveň* $adj(v'_j) \cap adj(v'_k)$ *pro* $j \neq k$
4. *Vlož všechna* v'_j, $j = 1, \ldots, s$ *na konec seznamu* V' *v libovolném pořadí*
5. **for** $j = 1 : s$ **do**
6. $G = G_{v_j}$
7. **end** i
8. **end while**

Příklad 11.1.3 *Násobný algoritmus minimálního stupně je znázorněn na Obrázku 11.1.3. Zakroužkováním jsou označeny všechny vrcholy, které budou najednou eliminovány v jeho prvním kroku.*

Násobná eliminace tedy eliminuje v každém kroku více vrcholů najednou. Její výhodou mimo jiné je to, že přepočítávání stupňů vrcholů se neprovádí tak často. V každém případě



Obrázek 11.1.3: *Ukázka vrcholů, které mohou být najednou eliminovány v prvním kroku Algoritmu 11.1.2.*

násobná eliminace mění pravidelný graf výrazně "pravidelnějším" způsobem, což budeme demonstrovat na grafu matice získané diskretizaci na strukturované síti. Pro takové sítě je známo, že algoritmus násobného minimálního stupně je v praxi velmi úspěšný v redukci zaplnění. Samozřejmě, za předpokladu efektivní implementace, založené na eliminačním faktorgrafu uvedené výše.

11.1.5 Algoritmus přibližně minimálního stupně

Dalším výrazným vylepšením celého schématu, které je blízké obecnému Algoritmu 11.1.1 spočívá v nepřesné aktualizaci stupňů vrcholu pouze s použitím **odhadu** na velikosti stupňů vrcholů. Pak je možné **aktualizaci** těchto přibližných stupňů provést velmi rychle. Popišme si tuto aktualizaci trochu podrobněji. Protože pracujeme s faktorgrafy, zaveďme si následující pojem **váhy** vrcholu.

Definice 11.1.4 *Pro prvek s množiny $\mathcal{S}$ eliminačního faktorgrafu definujeme jeho* **váhu** *$weight(s)$ jako počet vrcholů $v \in s$. Váhu $weight(e)$ prvku e množiny $\mathcal{E}$ eliminačního faktorgrafu Γ definujeme součtem*

$$\sum_{s \in sadj(e)} weight(s). \tag{11.6}$$

Ukažme nyní, jak získáme odhady na stupně vrcholů z $\mathcal{S}_i$ (to jsou vrcholy, co mají být ještě eliminovány) v nějakém eliminačním faktorgrafu Γ_i z hodnot pro Γ_{i-1} pro $i = 1, \ldots, N$. Triviálním horním odhadem na stupeň vrcholu $r \in \mathcal{S}_i$ je součet všech vah všech sousedů r z $\mathcal{S}_i$ i $\mathcal{E}_i$. Vzhledem k možným překryvům množin sousednosti vrcholů z $\mathcal{E}_i$, které sousedí s r, může ale být tento odhad daleko od skutečných stupňů. Proto při aktualizaci stupně vrcholu nesčítáme všechny váhy jeho sousedů, ale váhy sousedů z $\mathcal{E}_i$ upravujeme. Pro vrchol $e \in \mathcal{E}_i$ eliminovaný v i-tém kroku se jeho váha v eliminačním faktorgrafu Γ_i

získá pomocí množin dosažitelnosti v eliminačním faktorgrafu Γ_{i-1} následujícím způsobem

$$weight(e) = \sum_{r \in Reach_{\Gamma_{i-1}}(e)} weight(r). \tag{11.7}$$

Každý vrchol z $\mathcal{S}_i$ ve výše uvedené množině dosažitelnosti bude ale soused vrcholu e v Γ_i. Proto pro všechny vrcholy z $eadj(r)$ různé od e musíme do jejich stupňů započítat jen váhy vrcholů z $\mathcal{S}_i$ disjunktní s $Reach_{\Gamma_{i-1}}(e)$. K tomu slouží následující modifikace. Definujme si modifikovanou váhu sousedů $e \in \mathcal{E}_i$ vrcholu $s_i \in \mathcal{S}_i$, který byl v i-tém kroku algoritmu eliminován s pomocí následující rozdílové funkce

$$diff(e) = \begin{cases} weight(e) & pro\ e = s_i \\ weight(e) - \sum_{r \in reach_{\Gamma_{i-1}}(s_i) \cap sadj(e)} weight(r) & pro\ e \neq s_i. \end{cases}, \tag{11.8}$$

kde pro jednoduchost neindexujeme množiny sousednosti. Přibližný stupeň vrcholu $r \in reach_{\Gamma_{i-1}}(s_i)$ je pak definován výrazem

$$weight(s_i) + \sum_{s \in sadj(r)} weight(s) + \sum_{e \in eadj_{reach_{\Gamma_{i-1}}(s_i)}} diff(e). \tag{11.9}$$

Algoritmus 11.1.1, který používá místo přesných stupňů vrcholů jejich odhad podle vztahu (11.9), se nazývá algoritmus **přibližně minimálního stupně** (approximate minimum degree – AMD). Jeho použití vede v praxi k velmi dobrým výsledkům v minimalizaci zaplnění v grafu.

Postup algoritmů, kde se v jejich jednotlivých krocích minimalizují nebo odhadují stupně vrcholů není jediný možný, který je úspěšný. Jinou možností je minimalizovat **vznikající zaplnění**. Přibližné algoritmy tohoto typu hrají dnes v řídkých rozkladech významnou roli, byť jsou jejich implementace pomalejší než algoritmy ze skupiny minimálního stupně.

11.2 Uspořádání pásová a profilová

V této podkapitole se budeme zabývat uspořádáními, které opět minimalizují zaplnění pomocí **lokálního prohledávání**. Zároveň se ale snaží, aby struktura řídkosti matice měla některý ze speciálních tvarů, které si nyní zavedeme. Tyto tvary vycházejí z toho, že nenulové prvky matice jsou soustředěny na pozicích kolem její **hlavní diagonály** s případnými dalšími bloky nenulových prvků v jiných částech matice. V celé kapitole diskutujeme uspořádání čtvercové matice $A \in R^{n \times n}$, která má na hlavní diagonále nenulové prvky. Snaha soustředit prvky kolem hlavní diagonály byla v minulosti motivována jednoduchostí datových struktur implementace. Později se stala práce s obecnými datovými strukturami velmi efektivní i na moderních počítačových architekturách a význam uspořádání tohoto typu pro přímé metody se zúžil na specifické aplikace. Na druhé straně, v metodách řešení soustav rovnic, kde se s maticí opakovaně pracuje, jako jsou předpodmíněné iterační metody, soustředění prvků kolem hlavní diagonály může být výhodné z hlediska efektivního využití vyrovnávací paměti.

$$\begin{pmatrix} * & & & & & & & \\ * & * & * & & & & & \\ * & & * & & & * & & \\ & * & * & * & & & & \\ & & & * & * & * & & \\ & & & & * & * & & \\ & & & * & * & * & * & * \\ & & & & & * & * & * \end{pmatrix} \tag{11.12}$$

Obrázek 11.2.4: *Řídká matice, na které si demonstrujeme některé pojmy týkající se speciálních struktur řídkosti matic.*

11.2.1 Pásové, profilové a ovroubené matice

Pro přesnější charakterizaci matice s pozicemi nenulových prvků soustředěných kolem hlavní diagonály zaveďme následující pojmy. Pro i-tý řádek matice $A \in R^{n\times n}$, $i \in \{1,\dots,n\}$ označme $r_i(A)$ sloupcový index jeho prvního nenulového prvku:

$$r_i(A) = \min\{j|\ a_{ij} \neq 0\}. \tag{11.10}$$

Obdobně pro j-tý sloupec matice A, $j \in \{1,\dots,n\}$, označme $c_j(A)$ sloupcový index jeho prvního nenulového prvku

$$c_j(A) = \min\{k|\ a_{kj} \neq 0\}.$$

i-tou **šířkou dolního polopásu** $\beta_i^L(A)$ matice A pak nazveme rozdíl

$$\beta_i^L(A) = i - r_i(A)$$

a j-tou **výškou horního polopásu** $\beta_j^U(A)$ matice A nazveme rozdíl

$$\beta_j^U(A) = j - c_j(A)$$

Šířkou pásu (matice A) pak nazveme číslo $\beta(A)$ definované vztahem

$$\beta(A) = \max\{\beta_i^L(A)|\ 1 \le i \le n\} + \max\{\beta_i^U(A)|\ 1 \le i \le n\} + 1. \tag{11.11}$$

Definované veličiny si ozřejmíme v Příkladu 11.2.1 na matici z Obrázku 11.2.4

Příklad 11.2.1 *Pro matici A z Obrázku 11.2.4 platí $r_1 = 1$, $r_2 = 1$, $r_3 = 1$, $c_1 = 1$, $c_2 = 2$, $c_3 = 2$, $\beta_1^L(A) = 0$, $\beta_2^L(A) = 1$, $\beta_3^L(A) = 2$, $\beta_7^L(A) = 3$, $\beta_6^U(A) = 2$, $\beta_7^U(A) = 0$. Šířka pásu matice $\beta(A)$ má pak velikost 6.*

Neurčitý pojem okolí hlavní diagonály matice A můžeme tedy nahradit pojmem pásu $band(A)$ nebo profilu $env(A)$ jako množiny pozic podle následujících definic.

$$\begin{pmatrix} \circledast & \circledast & \circledast & \circledast & & & & \\ \circledast & \circledast & \circledast & \circledast & \circledast & & & \\ \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & & \\ \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & \\ & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast \\ & & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast \\ & & & \circledast & \circledast & \circledast & \circledast & \circledast \\ & & & & \circledast & \circledast & \circledast & \circledast \end{pmatrix} \tag{11.13}$$

Obrázek 11.2.5: *Řídká matice z Obrázku 11.12, na které si demonstrujeme pás matice. Do pásu matice spadají prvky označené symbolem* $\circledast$

$$\begin{pmatrix} \circledast & & & & & & & \\ \circledast & \circledast & \circledast & & & & & \\ \circledast & \circledast & \circledast & & \circledast & & & \\ & \circledast & \circledast & \circledast & \circledast & & & \\ & & & \circledast & \circledast & \circledast & & \\ & & & & \circledast & \circledast & & \\ & & & \circledast & \circledast & \circledast & \circledast & \circledast \\ & & & & & \circledast & \circledast & \circledast \end{pmatrix} \tag{11.14}$$

Obrázek 11.2.6: *Řídká matice z Obrázku 11.12, na které si demonstrujeme profil matice. Do profilu matice započítáváme prvky označené symbolem* $\circledast$

Definice 11.2.1 Pás *matice A, označovaný $band(A)$ je definován sjednocením*

$$band(A) = \{(i,j)|\ 0 \geq i - j \leq \beta(A)\} \cup \{(j,i)|\ 0 < j - i \leq \beta(A)\}.$$

Definice 11.2.2 Profil *matice $env(A)$ definujeme vztahem*

$$env(A) = \{(i,j)|\ 0 \geq i - j \leq \beta_i^L(A)\} \cup \{(j,i)|\ 0 < j - i \leq \beta_i^U(A)\}.$$

Na Obrázcích 11.13 a 11.14 znázorňujeme pás a profil matice zakroužkovanými hvězdičkami Vidíme, že profil může obsahovat výrazně méně prvků než pás matice a je tedy z tohoto pohledu efektivnější datovou strukturou.

Místo popisu profilu nebo pásu matice se někdy používá termínu **šířky polofronty**. Konkrétně, i-tou šířku dolní polofronty matice A pro $i \in \{1, \ldots, n\}$, kterou označíme $\omega_i^L(A)$, definujeme vztahem

$$\omega_i^L(A) = |\{k|k\ > i \wedge (\exists l \leq i)(a_{kl} \neq 0)\}|. \tag{11.15}$$

Její hodnota tedy označuje počet aktivních řádků pro i-tý sloupec, tedy takových řádků s indexem l alespoň i, které mají $r_l \leq i$, kde r_l je dáno vztahem (11.10). Analogicky pak

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

Obrázek 11.2.7: *Příklad matice pro demonstrování šířek dolního polopásu a dolní polofronty.*

definujeme j-tou výšku horní polofronty matice

$$\omega_j^U(A) = |\{k|k \ > i \wedge (\exists l \leq i)(a_{lk} \neq 0)\}|. \tag{11.16}$$

Definice 11.2.3 Frontu *matice A, označovanou $front(A)$, definujeme jako sjednocení*

$$front(A) = \{(i,j)|\ 0 \geq i - j \leq \omega_i^L(A)\} \cup \{(j,i)|\ 0 < j - i \leq \omega_i^U(A)\}.$$

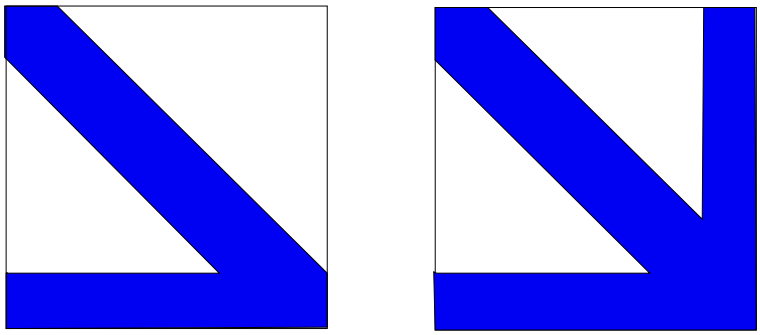
Příklad 11.2.2 *Rozdíl mezi šířkou dolního polopásu a šířkou dolní polofronty matice si ukážeme na Obrázku 11.2.7, kde je znázorněna řídká dolní trojúhelníková matice. Hodnoty ω_i^L pro $i = 1, \ldots, 5$ jsou, po řadě, $2, 1, 2, 1, 0$ a hodnoty β_i^L pro $i = 1, \ldots, 5$ jsou, po řadě, $0, 1, 0, 3, 2$.*

Má-li matice intuitivně malou velikost pásu či profilu, hovoříme o **matici pásové** nebo **profilové**. Zdůrazněme ale, že jde o terminologii utilitární, která je (analogicky k definici řídkosti) závislá na algoritmickém či obecně matematickém cíli práce s maticí.

Poznámka 11.2.1 *Řada zavedených pojmů se používá především pro matice se symetrickou strukturou řídkosti nenulových prvků, kdy se terminologie zjednoduší. V tomto případě budeme hovořit pouze o šířkách polopásu a šířkách polofront bez bližšího vymezení a v zavedených veličinách nebudeme používat horní indexy L a U. To nám také umožní zjednodušit některé předcházející definice. Například v definici pásu, profilu a fronty uvažujeme* **pozice v matici** *neorientované, například $\{i, j\}$ místo dvojice pozic (i, j) a (j, i).*

V některých případech se jsou oblíbené i matice, které mají nenulové prvky i v dalších, dobře charakterizovaných částech. Na Obrázku 11.2.8 je vlevo znázorněn tvar matice, který se nazývá **pásový s řádkovým ovroubením** a vpravo tvar **pásový s dvojitým (symetrickým) ovroubením.** Dále se ale takto zobecněným tvarům matic nebudeme věnovat.

V řadě praktických situací struktura řídkosti matice A sice nemá tvar, který se dá zachytit **malým** pásem nebo profilem, to jest pásem nebo profilem s "malou" velikostí $|band(A)|$ nebo $|env(A)|$, ale mohou existovat důvody, aby matice měla oblast nenulových prvků v okolí diagonály co nejkompaktnější. V těchto případech se matice může **transformovat** tak, že se velikost jejího pásu/profilu zmenší. Samotná transformace se provádí permutací matice.



Obrázek 11.2.8: *Tvary matice pásové s řádkovým ovroubením (vlevo) a pásové se symetrickým ovroubením. (vpravo)*

11.2.2 Ukládání pásových, profilových a blokově strukturovaných matic

Výše jsme hovořili o ukládání matic s obecnou strukturou řídkosti. V případě pásového či profilového tvaru je možné schéma uložení výrazně zjednodušit. Pro zachycení matice nám například u matice pásové stačí pole s numerickými hodnotami a číslo, které vyjadřuje šířku pásu matice. Indexy prvků nejsou třeba ukládat, protože jsou zřejmé. Na Obrázku 11.2.5 je zachycena matice a pásové schéma jejich uložení, které v něm uvádíme. Prvky pásových a profilových matic ale lze ukládat i úplně jinak, například po diagonálách do jednorozměrného nebo dvojrozměrného pole.

Je zřejmé, že při nevyrovnanosti dolních a horních šířek polopásů matice může být schéma uložení založené na pásovém paradigmatu neefektivní. Tomu může odpomoci schéma uložení, které používá pouze profilové informace. Při profilovém uložení matice je pak potřeba uschovat navíc (vzhledem k pásovému schématu uložení) vektory šířek horního a dolního polopásu. Pro ten účel vyhrazujeme pracovní vektory délky n, které po řadě značíme $\mathcal{IAL}$ a $\mathcal{IAU}$. Obrázek 11.2.10 tento formát uložení pro matici z Obrázku 11.2.9 znázorňuje.

11.2.3 Uspořádání do pásového nebo profilového tvaru

V případě pásového nebo profilového algoritmu je hlavním úkolem nalézt takové uspořádání, které minimalizuje velikost **pásu** nebo **profilu** matice ve smyslu počtu nenulových prvků uvnitř těchto struktur. To pak může také poskytnout malé zaplnění v Choleského faktoru. Jiným zmíněným možnostem, jako je minimalizace ovroubeného pásu či profilu se zde blíže nebudeme věnovat. Pro jednoduchost budeme zde vždy předpokládat, že je matice symetrická. Podstatným důvodem, proč jsou pás a profil matice výhodné z hlediska zaplnění a proč na uložení matice i v průběhu rozkladu stačí výše zmíněné struktury pro rozkládanou matici A, je vysvětlen tvrzením následující Věty 11.2.1.

Věta 11.2.1 *Mezi maticí symetrickou A a její maticí zaplnění platí vztah*

$$env(A) = env(F). \tag{11.17}$$

Matice A:

$$\begin{pmatrix} 1.1 & 2 & 3 & & \\ & 4 & 2 & & \\ & 3.1 & 5 & & \\ & & 6 & 7 & \\ & 2 & 0 & 3 & 8 \end{pmatrix}$$

Uložení matice A jako pásové ve formátu: dolní trojúhelník po sloupcích a horní trojúhelník po řádcích. Ukládá se vždy nejprve sloupec a pak řádek; diagonální prvek je připojen jen jednou a to ke sloupci. Dolní a horní šířky polopásů jsou, po řadě, 3 a 2.

	1	2	3	4	5	6	7	8	9	10	11
$\mathcal{A}$:	1.1	0	0	0	2	3	4	3.1	0	2	2
	0	5	6	0	0	0	7	3	0	8	

Obrázek 11.2.9: *Příklad uložení matice jako pásové v konkrétním popsaném formátu*

	1	2	3	4	5	6	7	8	9	10	11
$\mathcal{A}$:	1.1	0	0	0	2	3	4	3.1	0	2	2
	0	5	6	0	0	0	7	3	0	8	

$\mathcal{IAL}$:	0	0	1	0	3

$\mathcal{IAU}$:	2	1	0	0	0

Obrázek 11.2.10: *Příklad uložení matice A z Obrázku 11.2.5 jako profilové v popsaném formátu. Konkrétně: její dolní trojúhelník je uložen po řádcích a její horní trojúhelník je uložen po sloupcích. Ukládá se vždy nejprve řádek a pak sloupec; diagonální prvek je připojen jen jednou a to ke sloupci.*

Důkaz: Důkaz provedeme matematickou indukcí podle dimenze k podmatice A_{11} matice A. Budeme předpokládat, že numerické vlastnosti matice umožňují použít Choleského rozklad. Pro $k = 1$ a $k = 2$ je tvrzení zřejmé. Předpokládejme, že tvrzení věty platí pro všechny matice dimenze $k < n$ a ukážeme, že platí i pro matici A_{11} dimenze $k + 1$. Znázorněme A_{11} dimenze k společně s jejím Choleského faktorem jako ovroubenou matici následujícím způsobem

$$A = \begin{pmatrix} A_{11} & u \\ u^T & \alpha \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ v^T & \beta \end{pmatrix} \begin{pmatrix} L_{11}^T & v \\ 0 & \beta \end{pmatrix} \tag{11.18}$$

Porovnáním matice a jejího Choleského faktoru získáme $L_{11}v = u$. Pro komponenty vektoru u platí, že $u_j = 0$ pro $j < r_{k+1}(A)$ a také $u_{r_{k+1}} \neq 0$, kde $r_{k+1}(A)$ je definováno v (11.10) (využíváme symetrie matice). Proto i $v_j = 0$ pro $j < r_{k+1}(A)$ a $v_{r_{k+1}} \neq 0$. Indukční krok je tedy dokázán a Věta platí. ■

Základní myšlenkou přibližného postupu, jak najít nové očíslování řádků a sloupců takové, že matice bude mít po přeuspořádání malý pás a profil, je založeno na následujícím. V neorientovaném grafu symetrické matice budeme číslovat vrcholy tak, že pro každý z nich se snažíme **očíslovat co nejdříve** také jeho **sousedy**. Rozdíl hodnot nového očíslování nějakého vrcholu x a jeho souseda $y \in adj(x)$ je dolní mezí na velikost šířky polopásu matice. Postup, který z této úvahy vyplývá nese název CM algoritmus a je tak pojmenován podle svých autorů (Cuthill a McKee). Uvádíme jej v Algoritmu 11.2.1 a pro jednoduchost předpokládáme, že neorientovaný graf matice je souvislý. Jinak je třeba aplikovat Algoritmus na každou komponentu grafu samostatně.

Algoritmus 11.2.1 Algoritmus CM (Cuthill-McKee) pro minimalizaci šířky pásu a profilu matice.
Input: *Neorientovaný a souvislý graf $G = (\{1, \ldots, n\}, E)$ symetrické čtvercové matice.*
Output: *Nové uspořádání množiny vrcholů v seznamu $V' = (v_1, \ldots, v_n)$, které indukuje permutaci matice, pro kterou byl graf zkonstruován.*

1. *Nalezni počáteční vrchol r a polož $v_1 = r$*
2. *Vlož v_1 jako* **neoznačený prvek** *na konec seznamu V'*
3. **while** $|V'| \neq n$ **do**
4. *Nalezni první dosud neoznačený prvek v_j seznamu V'*
5. *Označ jej*
6. *Přidej všechny jeho sousedy, kteří nejsou dosud ve V' na konec tohoto seznamu.*
7. **end while**

Všimněme si, že Algoritmus 11.2.1, potřebuje ke své práci pouze graf $G(A)$ matice A. Jinými slovy, graf se v průběhu algoritmu není potřeba měnit tak jako tomu bylo u algoritmů skupiny minimálního stupně. To výrazně zjednodušuje implementaci a redukuje časovou složitost.

Neurčenou položkou v Algoritmu 11.2.1 je výběr počátečního vrcholu. Snadno nahlédneme, že vhodnými kandidáty pro tento výběr jsou takové vrcholy, pro které je nejkratší cesta k

nějakému dalšímu vrcholu grafu co nejdelší. To můžeme formálně popsat pomocí následujícího pojmu **excentricity**. Definujme excentricitu $\epsilon(u)$ vrcholu u grafu definujeme jako maximální délku $d(u,v)$ nejkratší cesty mezi vrcholem u a nějakým jiným vrcholem grafu v, jinými slovy

$$\epsilon(u) = \max_{v \in V} d(u,v) \tag{11.19}$$

Vhodnými kandidáty pro počáteční vrchol Algoritmu 11.2.1 budou tedy vrcholy s velkou excentricitou. Poznamenejme, že velikost maximální excentricity nějakého vrcholu grafu se nazývá jeho **průměr**.

Algoritmus nalezení vrcholů s maximální excentricitou, to jest těch, které realizují průměr grafu, je sice polynomiální, ale stejně příliš vysoká, protože pro jejich nalezení bychom museli počítat vzdálenosti mezi vrcholy. Proto se v praxi ujal algoritmus GPS, který je jednoduchým přibližným algoritmem pro nalezení vrcholů s dostatečnou excentricitou a je vlastně variantou prohledávání grafu do šířky. Dříve než jej uvedeme v následujícím Algoritmu 11.2.2 uveďme následující definici.

Definice 11.2.4 Strukturou úrovní *grafu $G = (V,E)$ nazveme rozklad $\mathcal{L} = (L_0, \ldots, L_\lambda)$ množiny vrcholů V, pro který platí $\lambda \in N$, $\lambda \geq 1$, $adj(L_i) \subseteq L_{i-1} \cup L_{i+1}$ pro $i = 1, \ldots, \lambda - 1$, $adj(L_0) \subseteq L_1$ a $adj(L_\lambda) \subseteq L_{\lambda-1}$.* **Šířkou** *struktury úrovní $\mathcal{L}$ nazveme číslo $w(\mathcal{L}) = \max_{0 \leq i \leq \lambda} |L_i|$*

Strukturu úrovní $\mathcal{L}(r)$ pro $r \in V$ můžeme vytvořit následujícím způsobem. Položme $L_0(r) = \{r\}$, $L_i(r) = adj(\bigcup_{k=0}^{i-1} L_k(r))$ pro $i = 1, \ldots, \lambda$, kde zřejmě $\lambda = \epsilon(r)$. Struktura úrovní je pak $\mathcal{L}(r) = (L_0(r), \ldots, L_\lambda(r))$.

Algoritmus 11.2.2 Algoritmus GPS (Gibbs-Poole-Stockmeyer) nalezení vhodného startovacího vrcholu pro Algoritmus 11.2.1.
Input: *Neorientovaný graf $G = (\{1, \ldots, n\}, E)$ symetrické čtvercové matice.*
Output: *Počáteční startovací vrchol r pro CM algoritmus.*

Zvol první aproximaci vrcholu r libovolně
do
  Vytvoř strukturu úrovní $\mathcal{L}(r) = (L_0(r), \ldots, L_{\lambda(r)}(r))$
  Uspořádej vrcholy v $x \in L_\lambda(r)$ podle vzrůstajícího stupně
  for all *$x \in L_\lambda(r)$ v takto získaném pořadí* **do**
    Vytvoř strukturu úrovní $\mathcal{L}(x)$.
    Zaznamenej délku této struktury $\lambda(x)$.
  end for all
  if *pro první x, které splňuje $|\mathcal{L}(x)| > |\mathcal{L}(r)|$* **then**
    polož $r = x$
  else
    exit do
  end if
end do

V praxi se používá více modifikací tohoto základního schématu. Jednoduchá varianta spočívá například v tom, že se v desátém kroku Algoritmu 11.2.2 vybere z poslední množiny struktury úrovní vrchol, který má nejmenší stupeň a pouze ten se použije na testování délky **jím generované struktury úrovní**. Uveďme si nyní některé teoretické vlastnosti související s uvedenými algoritmy. První z nich je jasné z definice nerozložitelnosti.

Lemma 11.2.1 *Pro nerozložitelnou symetrickou matici A (náš předpoklad) vždy platí*

$$r_i(A) < i,\ 1 < i \leq n.$$

Lemma 11.2.2 *Pro nerozložitelnou symetrickou matici A uspořádanou CM algoritmem platí*

$$r_i(A) \leq r_j(A),\ 1 < i < j \leq n.$$

Důkaz: Důkaz provedeme sporem. Uvažujme neorientovaný grafový model matice. Předpokládejme, že v matici, uspořádané CM algoritmem existuje sloupec k a dva řádky s indexy i, j pro které platí $i < j$ a přitom a také $r_i(A) > k$, $r_j(A) \leq k$. To ale znamená, že vrchol j je soused prvních k vrcholů, zatímco i není. To je ve sporu s CM algoritmem, který musel nejprve zařadit do seznamu všechny sousedy prvních k vrcholů dříve, než tam mohl být zařazen vrchol i. ■

V případě přeuspořádání matice, ke kterému vede očíslování CM algoritmem, neobsahuje profil Choleského faktoru žádné nulové prvky, jak ukazuje následující lemma.

Lemma 11.2.3 *Nechť A je symetrická nerozložitelná matice uspořádaná CM algoritmem, pro kterou platí $r_i(A) < i$ pro $1 < i \leq n$. Pak profil $env(F)$ neobsahuje* **nulové** *prvky.*

Důkaz: Důkaz provedeme matematickou indukcí podle dimenze n matice A. Pro $k = 1$ a $k = 2$ je tvrzení zřejmé. Předpokládejme, že tvrzení věty platí pro všechny matice dimenze k a ukážeme, že platí i pro matici A dimenze $k + 1 \leq n$. Znázorněme si podmatici matice A dimenze $k + 1$ společně s jejím Choleského faktorem následujícím způsobem

$$A = \begin{pmatrix} A_{11} & u \\ u^T & \alpha \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ v^T & \beta \end{pmatrix} \begin{pmatrix} L_{11}^T & v \\ 0 & \beta \end{pmatrix} \tag{11.20}$$

Porovnáním matice a jejího Choleského faktoru vidíme, že musí platit

$$L_{11}v = u.$$

Víme, že v $(k + 1)$-ním řádku platí $r_{n+1}(A) < n + 1$. Protože je matice uspořádána CM algoritmem a pro A_{11} platí indukční předpoklad, tak jsou podle Lemmatu 11.2.2 ve ve sloupci $r_{n+1}(A)$ matice L_{11} pouze nenulové prvky. To ale znamená, že jsou všechny vrcholy $r_{n+1}, \ldots, k$ orientovaného acyklického grafu L_{11} dosažitelné z vrcholu r_{n+1}, komponenty $v_{r_{n+1}}, \ldots, v_k$ jsou nenulové a indukční krok je dokázán. ■

Následující tvrzení (viz Theorem 4.2 v [109]) ukazuje vztah profilu a fronty matice očíslované algoritmem CM. Jak je vidět z důkazu, pro obecně uspořádanou matici tento vztah neplatí.

Lemma 11.2.4 *Nechť A je symetrická matice uspořádaná CM algoritmem. Pak platí*

$$front(A) \subseteq env(A). \tag{11.21}$$

Důkaz: Nechť $(i,j) \in front(A)$, $(i,j) \notin env(A)$. Bez újmy na obecnosti předpokládejme, že $i > j$, protože případ $i = j$ je jasný. Podle definice fronty existuje $k \geq i$ takové, že pro nějaké $l \leq j$ je $a_{kl} \neq 0$. Nemůže přitom platit $a_{ij'} \neq 0$ pro žádné $j' \leq j$, protože pak by platilo $(i,j) \in env(A)$. Pak tedy je $r_i(A) > j$. Zároveň ale $r_k(A) \leq j$, a dostáváme spor s tvrzením Lemmatu 11.2.2. Lemma je tak dokázána. ■

Uvažujme nyní následující Algoritmus 11.2.3, který budeme také nazývat algoritmem RCM (Reverse Cuthill-McKee). Uvidíme, že pro minimalizaci profilu matice je vhodnější než Algoritmus 11.2.1.

Algoritmus 11.2.3 Algoritmus RCM (reverse Cuthill-McKee) pro minimalizaci šířky pásu a profilu matice.
Input: *Neorientovaný graf $G = (\{1,\ldots,n\}, E)$ symetrické čtvercové matice.*
Output: *Nové uspořádání množiny vrcholů v seznamu V' , které indukuje permutaci matice, pro kterou byl graf zkonstruován.*

1. *Algoritmem 11.2.1 nalezni seznam $V'' = (v_1,\ldots,v_n)$*
2. *Seznam $V' = (v_n,\ldots,v_1)$ tvoří nové očíslování G*

Princip nového algoritmu je pouze v obrácení pořadí vrcholů z CM algoritmu. Následující tvrzení je zřejmé.

Lemma 11.2.5 *Nechť A je symetrická matice, která byla přeuspořádána CM algoritmem a $\tilde{A}$ je matice, kterou z ní získáme obrácením pořadí očíslování, to jest algoritmem RCM odvozeným z daného CM algoritmu. Pak platí*

$$env(\tilde{A}) = \{\{i,j\} \,|\, i \geq j \ \wedge \ \exists\, k \geq i \ with \ a_{kj} \neq 0\}.$$

Následující Lemma 11.2.6 ukazuje souvislost mezi frontou matice přeuspořádané algoritmem CM a profilem matice z algoritmu RCM.

Lemma 11.2.6 *Nechť A je symetrická matice, která byla přeuspořádána CM algoritmem a $\tilde{A}$ je matice, kterou z ní získáme obrácením pořadí očíslování, to jest algoritmem RCM. Pak platí*

$$\omega_{n-i+1}(\tilde{A}) \leq \beta_i(A).$$

Důkaz: Kdyby pro nějaké i platilo $\omega_{n-i+1}(\tilde{A}) > \beta_i(A)$, pak by muselo být nějaké j takové, že platí $j < r_i(A)$ a zároveň

$$\{i,j\} \in env(\tilde{A}) \equiv \{\{i,j\} \,|\, i \geq j \ \wedge \ \exists\, k \geq i \ with \ a_{kj} \neq 0\}.$$

To je spor s tvrzením Věty 11.2.2. ■

Zřejmým důsledkem předcházejících tvrzení je tedy platnost vztahu

$$env(\tilde{A}) \subseteq front(A) \subseteq env(A) \tag{11.22}$$

z čehož vidíme, že platí

Důsledek 11.2.1 *Nechť A je matice, která byla přeuspořádána CM algoritmem a $\tilde{A}$ je matice, kterou z ní získáme obrácením pořadí očíslování, to jest algoritmem RCM odvozeným z daného CM algoritmu. Pak platí*

$$|env(\tilde{A})| \leq |env(A)|.$$

Vidíme tedy, že pro minimalizaci profilu je obecně vhodnější Algoritmus 11.2.3 než Algoritmus 11.2.1. Jedná-li se pouze o minimalizaci pásu, snadno nahlédneme, že **šířka pásu** se obrácením pořadí vrcholů **nezmění** a je tedy jedno, který z nich použijeme. Podle Lemmatu 11.2.3 platí i to, že počet nenulových prvků ve faktoru matice A, získané přeuspořádáním CM algoritmem, je obecně alespoň takový, jako počet prvků ve faktoru matice, kterou získáme z A obrácením pořadí očíslování. V praxi je tento dodatek zajímavý ale jen v některých situacích, protože profilové schéma je většinou spojeno s explicitním ukládáním nenulových prvků profilu, kde už není potřeba rozlišovat, které z nich jsou opravdu nenulové a které jsou před rozkladem nulové.

Algoritmus CM (a tedy i RCM) jsou založeny na lokálním prohledávání. Velká snaha byla věnována nalezení nějaké globální korekce do relativně levného algoritmu. Velmi populárním přístupem v této kategorii metod je Sloanův algoritmus, ale jeho výklad jde už nad rámec tohoto textu.

11.3 Globální uspořádání

Diametrálně odlišnou strategií je hledání uspořádání, které berou do úvahy globální vlastnosti celé oblasti. Historicky prvním přístupem tohoto typu a nejdůležitějším generickým algoritmem pro uspořádání matic je postup, který zde nazveme **metodou vnořených řezů (nested dissection)**. Jeho základní princip můžeme demonstrovat na rozdělení grafu na dvě části pomocí vrcholového separátoru.

Definice 11.3.1 Vrcholovým separátorem *neorientovaného grafu $G = (V, E)$ nazveme množinu S jeho vrcholů takovou, že podgraf grafu G indukovaný množinou $V \setminus S$ má více komponent než G.*

Uvažujme nyní pro jednoduchost nerozložitelnou matici a podgraf $G(V \setminus S)$ jejího grafu G, který má právě dvě komponenty. Označme množiny vrcholů tyto dvou komponent symboly C_1 a C_2. Očíslujeme-li nyní nově množinu V tak, aby nejprve byly číslovány vrcholy množiny C_1, pak vrcholy množiny C_2 a nakonec vrcholy množiny S, pak má matice A po přeuspořádání indukovaném tímto novým očíslováním blokovou strukturu z Obrázku 11.3.11. Tato struktura se zřejmě **zachová** i při Choleského rozkladu. Bloková struktura příslušného Choleského faktoru je na Obrázku 11.3.12.

Nulové bloky v pozicích $(1, 2)$ a $(2, 1)$ rozkladu nejsou náhodné a plynou z vlastností separace množiny $V \setminus S$ množinou S na dvě části. Je tedy možné předpokládat určitou a jasně definovanou řídkost v Choleského faktoru navíc k tomu, že jednotlivé obecně nenulové bloky mohou být také řídké. Uspořádání metodou vnořených řezů získáme **rekurentní aplikací** tohoto postupu separace množiny pomocí vrcholového separátoru. Jinými

$$A = \begin{pmatrix} A_{11} & 0 & A_{31}^T \\ 0 & A_{22} & A_{32}^T \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \tag{11.23}$$

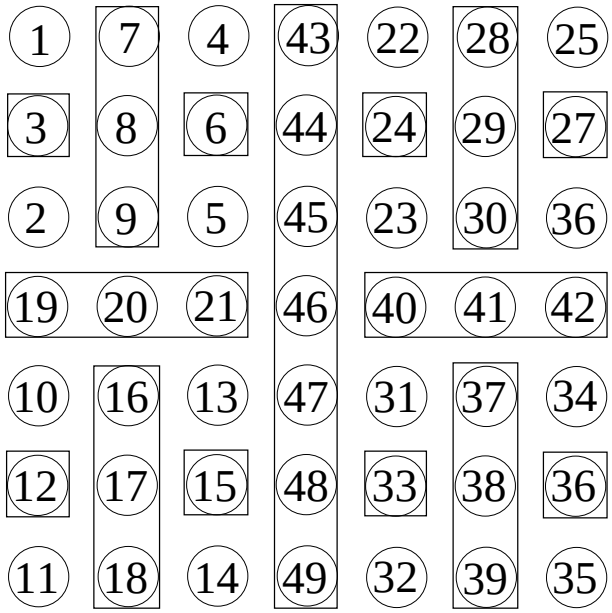
Obrázek 11.3.11: *Příklad matice A s blokovou strukturou indukovanou rozdělením její množiny vrcholů na tři části pomocí vrcholového separátoru S. Vrcholy množiny S jsou očíslovány jako poslední.*

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ 0 & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \tag{11.24}$$

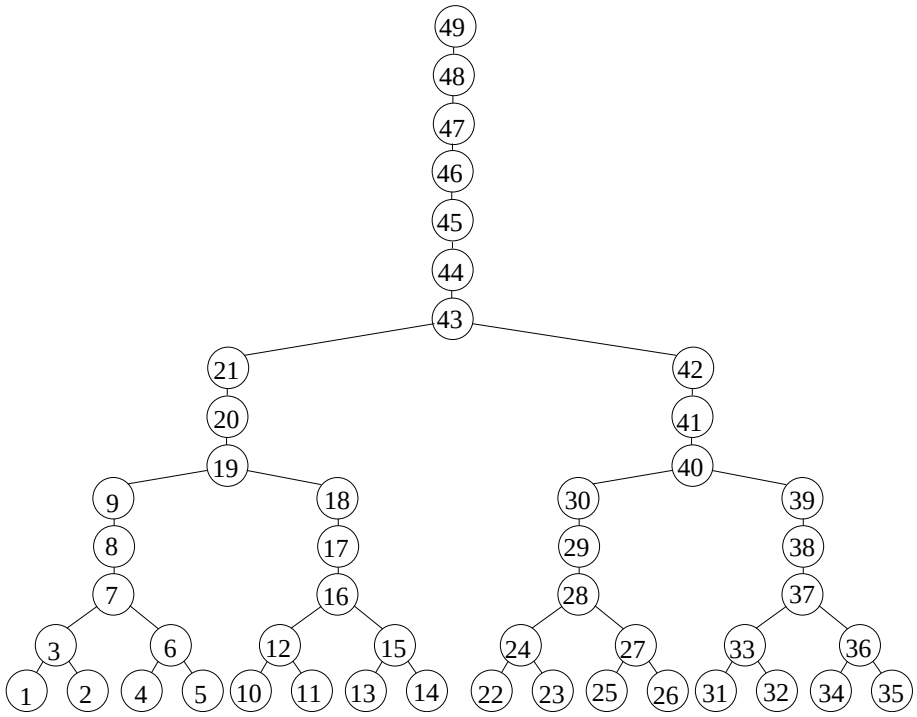
Obrázek 11.3.12: *Příklad Choleského faktoru L matice s blokovou strukturou indukovanou rozdělením její množiny vrcholů na tři části pomocí vrcholového separátoru S. Vrcholy množiny S jsou očíslovány jako poslední.*

slovy, hledání separátorů aplikujeme postupně na komponenty C_1 a C_2. Tím získáme separátory S_1 a S_2 a stejně postupujeme dále pro nově vytvořené komponenty. Separátory pak řadíme do výsledné permutace odzadu tak jak je postupně získáváme.

Řídkost i počet operací, které je potřeba provést při Choleského rozkladu takto přeuspořádané matice, je možné explicitně omezit, ale tyto vztahy zde nebudeme uvádět. Základní algoritmus metody vnořených řezů se ukázal velmi užitečným pro řešení soustav lineárních algebraických rovnic. Nicméně, zaznamenal také značný rozkvět od devadesátých let minulého století až dodnes jako obecná metoda, ve které se separátory hledají velmi důmyslně, pomocí technik nazývaných **dělení grafů** a **hypergrafů** Obrázky 11.3.13 a 11.3.14 zachycují, po řadě, očíslování vrcholů pravidelného grafu, který vznikl diskretizací pětibodovým schématem na strukturované síti ve dvou dimenzích (propojení vrcholů vertikálními a horizontálními spojnicemi odpovídá hranám grafu) a eliminační strom pro symetrickou a silně regulární matici této struktury.



Obrázek 11.3.13: *Příklad pravidelného grafu, který vznikl diskretizací ve dvou dimenzích a jeho očíslování metodou vnořených řezů.*



Obrázek 11.3.14: *Příklad eliminačního stromu pravidelného grafu, který vznikl diskretizací ve dvou dimenzích a jeho očíslováním metodou vnořených řezů.*

12

Stabilita řídkých rozkladů

Rozumná míra stability rozkladu v nepřesné aritmetice počítače je jedním ze základních požadavků na řešení soustav lineárních algebraických rovnic přímými metodami. Nedostatek stability se projeví především v malé přesností spočítaných faktorů a promítá se do jejich použití v substitučních krocích.

Nepřesná aritmetika počítače implikuje, že se rozklady v praxi počítají pouze přibližně, ale k dodatečným nepřesnostem v rozkladu mohou být ještě další motivace: zmenšení aritmetické náročnosti výpočtu, zmenšení strukturální náročnosti rozkladu řídké matice tím, že se zjednoduší použité datové struktury a/nebo zmenší vznikající zaplnění. Přibližný rozklad může také vzniknout, i když není prvotním cílem, dojde-li k problémům s existencí rozkladu dané matice na regulární faktory a rozkládaná matice pak musí být nějakým způsobem modifikována. Jiným důvodem pro počítání přibližného rozkladu může být zvýšení efektivity rozkladu či substitučních kroků na konkrétních počítačových architekturách. Problémy s existencí rozkladu přirozeně vznikají, protože v řadě případů je algoritmus rozkladu založen na dopředu obtížně ověřitelném předpokladu silné regularity. V případě , že se rozklady počítají pouze přibližně nad rámec nepřesností způsobených nepřesnou aritmetikou počítače, substituční kroky se obvykle aplikují v propojení s nějakou iterační metodou a stabilita rozkladů je tedy naprosto podstatná.

V případě přibližného rozkladu obvykle rozlišujeme dvojí druh přesnosti výpočtu. Za prvé, přesnost přibližného rozkladu, kterou můžeme měřit nějakou maticovou normou. Pro LU rozklad můžeme například tuto přesnost vyjádřit výrazem

$$\| A - LU \| . \tag{12.1}$$

A za druhé, přesnost substitučních kroků s faktory vyjádřenou pro LU rozklad například výrazem

$$\| I - U^{-1}L^{-1}A \| . \tag{12.2}$$

Specifickou třídu tvoří přibližné rozklady, které se používají pro transformaci iteračních metod jako jejich předpodmínění, jejichž výpočet by měl být velmi levný a o kterých budeme hovořit později. Je-li původní matice modifikována, obecnou metodou na získání řešení je zkombinovat získaný rozklad s iterační metodou. Touto iterační metodou může být iterační zpřesnění, ale může jí být i obecnější metoda, jak o tom budeme mluvit později.

12.1 Stabilita rozkladů

Zmiňme nejprve obecný výsledek, týkající se stability LU rozkladu v aritmetice s konečnou přesností. Jsou-li $\hat{L}$ a $\hat{U}$ vypočítané faktory LU rozkladu matice $A \in R^{n\times n}$ a značí-li ϵ strojovou přesnost použité aritmetiky počítače, pak pro matici E splňující

$$A + E = \hat{L}\hat{U}$$

platí

$$||E||_\infty \leq 2n\epsilon||\hat{L}||_\infty||\hat{U}||_\infty + \mathcal{O}(\epsilon^2). \tag{12.3}$$

Předpokládáme-li nyní platnost vztahu

$$n\epsilon \ll 1 \tag{12.4}$$

a je-li $\hat{x}$ vypočteným řešením soustavy s těmito faktory, pak je $\hat{x}$ také přesným řešením úlohy

$$(A + \Delta A)\hat{x} = b, \tag{12.5}$$

kde

$$||\Delta A||_\infty \leq 6n\epsilon||\hat{L}||_\infty||\hat{U}||_\infty + \mathcal{O}(\epsilon^2). \tag{12.6}$$

Analogické vztahy lze formulovat i v jiných maticových normách.

Problémem výše uvedených tvrzení z praktického pohledu je určení vztahu mezi velikostmi faktorů, danými například v (12.6) jako $||\hat{L}||_\infty$ a $||\hat{U}||_\infty$ na jedné straně a normou $||A||_\infty$. Pro určení tohoto vztahu jsou potřeba bližší detaily o LU rozkladu. V případě částečné pivotace, tedy rozkladu matice PA, kde P je řádková permutační matice určená tak, aby se maximalizovaly absolutní hodnoty pivotů, platí

$$||\hat{L}||_\infty \leq n \tag{12.7}$$

a stačí se omezit na zkoumání výrazu

$$\frac{||\hat{U}||_\infty}{||\hat{A}||_\infty}, \tag{12.8}$$

který se nazývá **růstový faktor** a je třeba, aby byl co nejmenší. Je-li růstový faktor malý, je LU rozklad s částečnou pivotací zpětně stabilní a v tomto smyslu hovoříme o jeho **podmíněné zpětné stabilitě**. V dalším textu budeme pro kompatibilitu s jinými výsledky růstový faktor pro LU rozklad s faktorem $\hat{U} = (\hat{u})_{ij}$ psát také ve tvaru

$$\rho_{growth} = max(\hat{u}_{ij})/max(a_{ij}) \tag{12.9}$$

pro příslušné indexy i a j.

Jiná situace je v Choleského rozkladu symetrické a pozitivně definitní matice v aritmetice s konečnou přesností.

Je-li $\hat{L}$ je vypočítaný Choleského faktor odmocninové Choleského rozkladu a platí-li $n^{3/2}\epsilon \ll 1$, pak pro matici splňující

$$A + E = \hat{L}\hat{L}^T$$

platí

$$||E||_F \leq \left(\frac{2n^{3/2}}{1-2n^{3/2}\epsilon}\right)\epsilon||A||_F + \mathcal{O}(\epsilon^2). \tag{12.10}$$

Vidíme, že v tomto případě je Choleského rozklad **bezpodmínečně zpětně stabilní**. Norma spočítaného Choleského faktoru $\hat{L}$ se v odhadu chybové matice E nevyskytuje a stabilitní výsledek je formulován explicitně pomocí vstupních dat.

V praktických rozkladech je problém, jak obecně heuristicky omezit normy spočítaných faktorů na pravé straně pro rozklady obecnější matic a obecnější rozklady, což někdy může být zabezpečeno **statickou permutací**, ale obecně musíme provádět **pivotaci** a přihlédnout k řídkosti matic.

12.2 Přeuspořádání, pivotace a jejich motivace v řídkých rozkladech

Uvažujme LU rozklad a rozlišme několik základních případů přeuspořádání/permutace, které budou vysvětleny v samostatných podsekcích.

12.2.1 Permutace v LU rozkladech

V přesné aritmetice vyžaduje LU rozklad silnou regularitu rozkládané a obecně nesymetrické matice. To je samozřejmě vlastnost, kterou v praxi nemusíme znát a priori. Prvním standardním krokem rozkladu, který zvyšuje šanci, že matice je silně regulární, je použít počáteční uspořádání, které permutuje nenulové prvky matice, případně její prvky o co největší absolutní hodnotě, na hlavní diagonálu matice. To ale nemusí stačit, protože LU rozklad je, na rozdíl od Choleského rozkladu, pouze **podmíněně zpětně stabilní**. V praxi to znamená, že je zapotřebí maximálně omezit normy spočítaných faktorů $\hat{L}$ a $\hat{U}$ pivotací.

Uvažujme, bez újmy na obecnosti, podmaticový LU rozklad, tedy variantu generického algoritmu kij nebo kji, kde rozkladem postupně měníme aktivní podmatice $A_R^{(i)}$ a eliminační matice $A^{(i)}$ pro $i = 0, \ldots, n-1$. Pivotace znamená, že před provedením eliminačního kroku na pozici $(1,1)$ matice $A_R^{(i)}$ permutujeme do této diagonální pozice vybraný hlavní prvek neboli pivot. Podívejme se na pivotaci nejprve pod zorným úhlem zlepšení stability rozkladu. V obecně nesymetrickém případě může být hlavní prvek vybírán z libovolné pozice aktivní části matice. To se dá formálně vyjádřit obecně nesymetrickými permutacemi, které rozkládanou matici A transformují na PAQ, kde P a Q tyto permutace vyjadřují. Tuto pivotaci, kde pivot je prvek s největší absolutní hodnotou z celé aktivní části matice, nazýváme na rozdíl od částečné pivotace **úplnou pivotací**. V tomto případě je růstový faktor omezen podle vztahu [161], [162]

$$\rho_{growth} \leq \sqrt{n2\ 3^{1/2}4^{1/3}\ldots n^{1/(n-1)}}\max_{i,j}|a_{ij}|. \tag{12.11}$$

Permutování řádků a sloupců v průběhu algoritmu ale klade velké nároky na flexibilitu datových struktur použitých v řídkém rozkladu, a tak úplná pivotace bývá omezena ještě

více pouze na nějakou část aktivní podmatice (Schurova doplňku). V případě částečné pivotace, která může být implementována mnohem efektivněji, je omezení růstového faktoru mnohem slabší a dáno výrazem

$$\rho_{growth} \leq 2^{n-1} \max_{i,j} |a_{ij}|. \tag{12.12}$$

Navíc lze ukázat, že takového růstu lze dosáhnout.

12.2.1.1 Pivotace v řídkém LU rozkladu

V případě řídkého rozkladu je nutné pivotaci propojit v nějakém **společném kritériu** s permutacemi, které alespoň přibližně minimalizují zaplnění. Historicky první návrh takového kritéria pochází od Markowitze [116], který stabilitu neuvažuje vůbec a předpokládá úplnou pivotaci pro ddosažení maximální řídkosti faktorů. Hlavní element v k-tém kroku rozkladu je určen, že zaplnění generované v tomto kroku je co možná nejmenší. Konkrétně, označíme-li v matici o dimenzi k počty prvků na řádcích r_i, $i = 1, \ldots, k$ a počty prvků ve sloupcích c_i, $i = 1, \ldots, k$, pak je možné míru zaplnění přibližně posuzovat podle velikosti **Markowitzova čísla**

$$M_{ij} = (r_i - 1)(c_j - 1),\ i, j = 1, \ldots, k \tag{12.13}$$

prvků v příslušné aktivní podmatici. Toto číslo vlastně reprezentuje **nesymetrickou variantu algoritmu minimálního stupně**. Algoritmus minimálního stupně pro symetrické a pozitivně definitní matice byl však navržen později, konkrétně Tinneyem and Walkerem v roce 1967 [154]. Uvažujme příklad matice na následujícím obrázku.

$$\begin{pmatrix} 1 & 2 & 3 & & 4 \\ & 5 & & 6 & \\ 7 & & 8 & & 9 \\ & 10 & 11 & 12 & \\ 13 & 14 & & & 15 \end{pmatrix}$$

Nenulové prvky s nejmenším Markowitzovým číslem jsou na pozicích $(2, 4)$ a $(4, 4)$. Pivotace, která se snaží vybrat prvek s co největší absolutní hodnotou, pak z nich zvolí prvek na pozici $(4, 4)$. Tímto způsobem jsou heuristická kritéria založená na velikosti zaplnění a potenciální velikosti růstového faktoru zkombinována. Takovýto postup je ale implementačně velmi náročný. Navíc se v praxi může stát, že prvek, který implikuje co nejmenší zaplnění v nějakém kroku rozkladu vede k velkému růstovému norem počítaných faktorů. Jak jsme zmínili výše, praktické procedury obvykle **neprohledávají celou aktivní matici**, ale pouze několik jejích řádků a sloupců. Mezi nimi pak hledají prvek s nejmenším Markowitzovým číslem, jako navrhl například Duff a Reid v implementacích MA28 a MA48 [133] a Zlatev (1980) v Y12M, viz publikaci [166]. Amestoy, Li a Ng navrhli v roce 2002 diagonální Markowitzovu pivotaci, kde je pivotace omezena na prvky hlavní diagonály matice, tedy na symetrickou permutaci.

Analogickou konkrétní snahou o kompromis mezi uvedenými protichůdnými požadavky je **prahová pivotace**, která se dá dobře skloubit s lokálním prohledáváním. Uvažujme LU rozklad z částečnou pivotací

$$LU = PA, \tag{12.14}$$

kde P je permutační matice. Prahová pivotace znamená, že pivot zvolený v aktivní podmatici $A_R^{(i)}$ pro $k = 1, \ldots, n-1$ prvek $a_{ij}^{(k)}$ splňuje vztah

$$|a_{ij}^{(k)}| \geq \mu \; max_l |a_{lj}^{(k)}| \tag{12.15}$$

pro nějaký daný parametr $0 < \mu \leq 1$ a zároveň maximalizuje mezi takovými prvky Markowitzovo číslo. Je možné ukázat, že lokální růst prvků v eliminačních maticích je omezen pro prvky aktivních matic vztahem

$$max_i |a_{ij}^{(k+1)}| \leq (1 + 1/\mu) max_i |a_{ij}^{(k)}|.$$

V předcházejícím textu jsme zmínili dva obecné přístupy k LU rozkladu, které umožňují jednoduché zapojení částečné pivotace nebo její prahové varianty. První z nich je založen na sloupcovém eliminačním stromu pro matici $A^T A$. Výše zmíněnou nevýhodou je potenciálně velké zaplnění, viz George, Ng (1986), které je možné částečně (ale jen částečně) eliminovat postupem navrženým Georgem a Ng (1987) []. Matici $A^T A$ je možné používat pouze implicitně, viz například Davis, Gilbert, Larimore, Ng (2000) []. Druhým postupem je algoritmus souběžné symbolické a numerické faktorizace uvedený zde jako Algoritmus 10.1.2.

Částečnou i omezenou úplnou pivotaci je možné do algoritmů rozkladu zahrnout ještě dalšími technikami, jako je například **odložení** pivotace. Tento přístup vědomě omezuje hledání hlavního prvku pouze v části aktivní podmatice s tím, že se odložené pivoty odpovídající řádkům případně i sloupcům permutovaným na pozice s vyššími indexy vyřeší později. To je přirozené v multifrontální metodě a jejích nesymetrických variantách. Poslední strategií, kterou zde zmíníme, je změna rozkládané matice namísto permutací, které mohou být velmi drahým postupem. Konkrétně, diagonální prvky aktivních matic, které jsou velmi malé, mohou být modifikovány například přičtením výrazu

$$||A||\sqrt{\epsilon}, \tag{12.16}$$

viz [43]. Místo původní matice je tedy rozkládána matice modifikovaná. Modifikace slouží k tomu, aby rozkládaná matice byla dostatečně silně regulární.

13

Přímé metody řešení soustav lineárních rovnic s řídkou, symetrickou a obecně indefinitní maticí

V této kapitole stručně zmíníme přímé metody pro řešení soustav se symetrickými a obecně indefinitními maticemi. Ačkoli bezodmocninovou Choleského faktorizaci je možné použít i pro rozklad indefinitních matic, standardní rozklad nemusí být stabilní nebo nemusí dokonce existovat při omezení přeuspořádání na **symetrické permutace**. Příkladem matice, pro kterou takový Choleského rozklad neexistuje ani pro žádnou její symetrickou permutaci je uveden níže

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

Rozklad následující symetrické matice

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

je pro malou hodnotu ϵ nestabilní, protože vede vede k velkému růstu ve faktorech.

13.1 Symetrický indefinitní rozklad

Rozklad obecně symetrické matice vede ke speciálnímu typu rozkladu s blokovou diagonální maticí, která má bloky velikosti 1×1 a 2×2. Tento rozklad budeme nazývat **symetrickým indefinitním rozkladem**. Rozklad první z uvedených matic můžeme pak psát ve tvaru

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

Pivotace v symetrickém indefinitním rozkladu se obvykle provádí dynamicky. V tomto textu uvedeme několik procedur s různou složitostí i různými zárukami na stabilitu rozkladu. V každém případě je v teoretických odhadech na tyto záruky i závislost na příslušném růstovém faktoru. První metodou pivotace, která najde výše zmíněné bloky LBL^T rozkladu je **úplná diagonální pivotace**. Ta je založena na tom, že v aktivní matici nejprve najde diagonální prvek maximální velikosti a_{rr} a mimodiagonální prvek maximální velikosti a_{pq}. Poté, jestliže $a_{rr} \geq \sigma a_{pq}$, vybere a_{rr} jako 1×1 pivot, jinak vybere jednoznačně určený 2×2 pivot s mimodiagonálním prvkem a_{pq}. Teoretickými odhady, které se týkají růstového faktoru a volbou parametru α se zde nebudeme zabývat. Důvod, proč je v případě malých diagonálních prvků někdy výhodné použít 2×2 pivot snadno vidíme z následujícího vztahu pro inverzi bloku 2×2, který uvádíme pro obecnější nesymetrický pivot (a kde předpokládáme, že tato inverze existuje).

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = 1/(ad - bc) \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Výběr diagonálního pivota s velkou absolutní hodnotou a výběr 2×2 pivota s mimodiagonálním prvkem s velkou absolutní hodnotou jsou tedy komplementární. Krok úplné diagonální pivotace v symetrickém indefinitním rozkladu znázorníme následujícím algoritmem.

Algoritmus 13.1.1 Krok úplné pivotace podle Bunche a Parletta (1971)

1. *Polož* $\alpha = (1 + \sqrt{17})/8 \approx 0.64$
2. *Nalezni* a_{kk}*: diagonální prvek maximální velikosti*
3. *Nalezni* a_{ij}*: mimodiagonální prvek maximální velikosti* $(i < j)$
4. **if** $|a_{kk}| \geq \alpha |a_{ij}|$ **then**
5. *použij* a_{kk} *jako* 1×1 *pivot (***hotovo** *pro* $a_{kk} = 0$*)*
6. **else**
7. *použij* $\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix}$ *jako blokový* 2×2 *pivot*
8. **end if**

Algoritmus, který provádí úplnou diagonální pivotaci je velmi stabilní, ale jeho implementace může být pro rozklad řídké matice velmi neefektivní ze stejného důvodu jako je obtížné provést úplnou pivotaci v řídkém LU rozkladu. Problém efektivního postupu v případě řídkých matic řeší do značné míry pivotace podle Bunche a Kafmannové, jejíž jeden krok je uveden v následujícím algoritmu.

Algoritmus 13.1.2 Krok úplné pivotace podle Bunche a Kaufmannové (1977)

1. $\alpha = (1 + \sqrt{17})/8 \approx 0.64$
2. $i = 1$ *(možná varianta, kde* i *splňuje* $|a_{ii}| \geq \alpha |a_{kk}|$ *mezi všemi* k
3. *Nalezni* $j \neq i$ *takové, že* $a_{ji} = \max\{|a_{ki}|, k \neq i\} =: \lambda$
4. **if** $|a_{ii}| \geq \alpha\lambda$ **then**

5. *použij* a_{ii} *jako* 1×1 *pivot*
6. **else**
7. $\sigma = \max\{|a_{kj}|, k \neq j\}$
8. **if** $|a_{ii}|\sigma \geq \alpha\lambda^2$ **then**
9. *use* a_{ii} *jako* 1×1 *pivot*
10. **else if** $|a_{jj}| \geq \alpha\sigma$ **then**
11. *použij* a_{jj} *as a* 1×1 *pivot*
12. **else**
13. *použij* $\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix}$ *jako blokový* 2×2 *pivot*
14. **end if**
15. **end if**

Schématicky si můžeme znázornit, které prvky jsou v jednom kroku Algoritmu 13.1.2 testovány.

$$\begin{pmatrix} d & . & . & \lambda & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \lambda & . & . & c & . & \sigma & . \\ . & . & . & . & . & . & . \\ . & . & . & \sigma & . & . & . \\ . & . & . & . & . & . & . \end{pmatrix}$$

Růstový faktor je v tomto algoritmu omezen s asymptotickým faktorem $(2.57)^{n-1}$, což je ještě o něco větší růst než v případě standardní částečné pivotace LU rozkladu, ale pro většinu praktických úloh je tato pivotace dobrým řešením. Prahová modifikace pivotace [110] zavádí další parametr, který vyvažuje řídkost a stabilitu analogicky k výše zmíněné prahové pivotaci v LU rozkladu. Zvolíme-li tento parametr τ z $(0, 1)$, pak můžeme výše uvedený BK rozklad modifikovat tak, že v následujícím schématu platí $|f| \geq \tau|\lambda|$

$$\begin{pmatrix} d & . & f & \lambda & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ f & . & e & . & . & \sigma & . \\ \lambda & . & . & . & . & . & . \\ . & . & \gamma & . & . & . & . \\ . & . & . & . & . & . & . \end{pmatrix}$$

Potřebnost omezit prostor, ve kterém se hledá pivot, například u indefinitních variant multifrontální metody, vedl k dalším modifikacím pravidel pivotace, jako jsou například v následujícím Algoritmu 13.1.3 [111].

Algoritmus 13.1.3 Pivotační pravidla prahové pivotace
1. **if** $|d| \geq \alpha|\lambda|$ *použij d jako* 1×1 *pivot*
2. **if** $|d\gamma| \geq \alpha|\lambda|^2$*: použij d jako* 1×1 *pivot*

3. **if** $|e| \geq \alpha|\gamma|$: *použij* e *jako* 1×1 *pivot*
4. **else** *5. použij* $\begin{pmatrix} d & f \\ f & e \end{pmatrix}$ *jako blokový* 2×2 *pivot*

Zde platí, že optimální velikost α závisí také na prahu τ a může být spočítána prostřednictvím jednoduché algebraické rovnice.

Později bylo ukázáno [11], že faktor L nemusí být omezený a na vině je právě blokový charakter rozkladu, který může být prováděn různými způsoby. Zatímco v případě diagonální pivotace platí pro nějakou konstantu c vztah $||L|| \leq c$, Algoritmus 13.1.2 může být modifikován, aby platilo $||L||||D||||L^T|| \leq c||A||$. Jiným způsobem, jak se vyhnout problémům s růstem prvků v trojúhelníkovém faktoru je následující postup.

Algoritmus 13.1.4 Rook pivoting: Duff, Reid (1982,1983); Ashcraft et al. (1998)
1. $\alpha = (1 + \sqrt{17})/8 \approx 0.64$
2. $i = 1$ *nebo* i *je index diagonálního prvku s maximální velikostí (bounded BK / fast BP)*
3. Nalezni $j \neq i$ *tak, že* $a_{ji} = \max\{|a_{pi}|, p \neq i\}$
4. **if** $|a_{ii}| \geq \alpha|a_{ji}|$ *a* $a_{ii} \neq 0$ **then**
5. *použij* a_{ii} *jako* 1×1 *pivot*
6. **else**
7. **repeat**
8. *Najdi* $k \neq j$ *tak, že* $|a_{kj}| = \max\{|a_{pj}|, p \neq j\}$
9. **if** $|a_{jj}|\sigma \geq \alpha|a_{kj}|$ *a* $a_{jj} \neq 0$ **then**
10. *použij* a_{jj} *jako* 1×1 *pivot*
11. **else if** $|a_{ij}| = |a_{kj}|$ **then**
12. *použij* $\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix}$ *jako blokový* 2×2 *pivot*
13. **else**
14. *Polož* $i = j$ *and* $j = k$
15. **end**
16. **until** *je nalezen pivot nebo byly navštíveny všechny sloupce*
16. **end if**

Poslední dva algoritmy, které uvedeme, hledají pivot ve velmi omezeném prostoru. Prvním z nich je Bunchova tridiagonální pivotace s modifikací z roky 1999, která umožňuje zvýšit pravděpodobnost výběru 1×1 pivota. Druhý z těchto algoritmů se dá použít bez nutnosti znát matici dopředu, což může být výhodné například při řešení soustav rovnic v rámci Lanczosovy metody.

Algoritmus 13.1.5 Stabilní tridiagonální pivotace: Bunchova metoda
1. $\alpha = (\sqrt{5} - 1)/2 \approx 0.62$
2. Najdi σ: *prvek maximální velikosti v* A
3. **if** $|a_{11}|\sigma \geq \alpha|a_{21}|^2$ **then**
4. *použij* a_{11} *jako* 1×1 *pivot*
5. **else**

6. *použij* $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ *jako blokový* 2×2 *pivot*
7. **end**
end

Algoritmus 13.1.6 Pivotace Bunch-Marcia
1. $\alpha = (\sqrt{5} - 1)/2 \approx 0.62$
2. $\Delta = a_{11}a_{12} - a_{21}^2$
3. **if** $|\Delta| \leq \alpha |a_{11}a_{32}|$ *nebo* $|a_{21}\Delta| \leq \alpha |a_{11}^2 a_{32}|$ *nebo* $|a_{11}a_{22}| \geq \alpha a_{21}^2$ **then**
4. *použij* a_{11} *jako* 1×1 *pivot*
5. **else**
6. *použij* $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ *jako blokový* 2×2 *pivot*
7. **end if**

I v případě symetrických indefinitních systémů je vhodné permutovat na hlavní diagonálu nenulové prvky s co největší absolutní hodnotou. Analogicky k LU rozkladu existují i statické grafové algoritmy, které berou do úvahy diagonální prvky i možné bloky velikosti 2×2 a priori. Ty mohou být založeny na rozkladu permutací na cykly, viz [48], [53], [54].

14

Řídká QR faktorizace

Unitární transformace představují další důležitou komponentu numerické lineární algebry a v mnoha případech jsou důležité také jako procedury, které umožňují efektivní řešení soustav lineárních algebraických rovnic. V tomto textu nás budou zajímat tyto transformace aplikované na reálné matice, které poskytnou QR rozklad ve tvaru

$$A = QR \tag{14.1}$$

Budeme se přitom soustředit pouze na QR rozklad získaný prostřednictvím Householderových reflexí a Givensových rotací bez toho, abychom diskutovali jejich použití například pro řešení problému nejmenších čtverců nebo jejich stabilitu. Hlavní motivací je ukázat strukturální odlišnost těchto ortogonálních transformací při QR rozkladu řídkých matic. Nebudeme přitom zmiňovat některé praktické otázky související se stabilitou těchto reflexí.

14.1 Householderovy reflexe a husté matice

Householderova reflexe je určena elementární maticí reflexe

$$H = I - 2\,vv^T,$$

kde $I \in R^{n\times n}$ je jednotková matice a $v \in R^n$, $||v|| = 1$. Snadno nahlédneme, že Householderova reflexe je symetrická a ortogonální. Její použití pro QR rozklad je založeno na tom, že aplikace na nenulový vektor $x \in R^n$ poskytne

$$Hx = (\pm||x||_2, 0, \ldots, 0)^T \tag{14.2}$$

QR rozklad matice $A \in \mathbb{R}^{n\times n}$ pomocí Householderových reflexí je založen na jejich postupné aplikaci na sekvenci Schurových doplňků, kde příslešné vektory x jsou zvoleny jako jejich první sloupce. Standardním doplněním těchto reflexí na dimenzi n diagonálními jedničkami získáme QR rozklad $A = QR$, pro který můžeme formálně napsat

$$Q = H_1 H_2 \ldots H_n,\ \ R = Q^T A.$$

Uvažujme aplikaci Householderovy reflexe $H = I - 2vv^T A$ na nějakou matici $A \in \mathbb{R}^{n\times n}$, kde v je její normalizovaný první sloupec a kde

$$Hv = (\pm||v||_2, 0, \ldots, 0)^T \tag{14.3}$$

Pak platí:

- $v_i = 0 \Rightarrow (HA)_{i*} = A_{i*}$
- $v_i \neq 0 \Rightarrow$ Struktura řídkosti $(HA)_{i*}$ je dána sjednocením všech struktur řídkosti A_{j*} takových, že $v_j \neq 0$.

Na rozdíl od LU rozkladu může být faktor R QR rozkladu výrazně hustší než faktor U LU rozkladu téže matice. Související vztah je popsán následující větou, která ukazuje, že R faktor se vztahuje k rozkladu matice soustavy normálních rovnic.

Věta 14.1.1 *Jestliže $A^T A$ je pozitivně definitní s Choleského faktorizací $A^T A = LL^T$ a QR rozkladem $A = QR$, pak $L = R^T$.*

QR faktorizace má kvůli svému charakteru ještě další významnou makroskopickou odlišnost of LU rozkladu.

Věta 14.1.2 *Nechť A má silnou Hallovu vlastnost. Pak je struktura řídkosti R rovna struktuře symetrického faktoru L^T rozkladu $A^T A = LL^T$. V opačném případě je struktura řídkosti L^T obecně nadmnožinou struktury řídkosti R.*

14.2 Givensovy rotace

Jiným postupem pro získání QR rozkladu je použití Givensových rotací. Ty jsou v maticové formě reprezentovány maticí G, která vznikne z jednotkové matice nahrazením jejích čtyř prvků obecnějšími hodnotami. Schématicky

$$G = \begin{array}{c} \\ \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} & & & i & & & & j & & & \end{array} \\ \begin{pmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \cos\phi & & & & -\sin\phi & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & \sin\phi & & & & \cos\phi & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{pmatrix} \end{array} \tag{14.4}$$

Výsledek aplikace na Gx Givensovy rotace na vektor $x = \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix}$ je

$$
G_{ij}(\phi)x = \begin{array}{c} \\ \\ \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \\ \end{array}
\begin{array}{c}
\begin{array}{ccccccccccccc} & & & & i & & & & j & & & & \end{array} \\
\begin{pmatrix}
1 & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & \\
 & & & & \cos\phi & & & & -\sin\phi & & & & \\
 & & & & & 1 & & & & & & & \\
 & & & & & & \ddots & & & & & & \\
 & & & & & & & 1 & & & & & \\
 & & & & \sin\phi & & & & \cos\phi & & & & \\
 & & & & & & & & & 1 & & & \\
 & & & & & & & & & & \ddots & & \\
 & & & & & & & & & & & & 1
\end{pmatrix}
\end{array}
\begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix}
$$

$$
= \begin{array}{c} \\ \\ \\ i \\ \\ j \\ \\ \\ \end{array}
\begin{pmatrix}
\xi_1 \\
\vdots \\
\xi_i \cos\phi - \xi_j \sin\phi \\
\vdots \\
\xi_i \sin\phi + \xi_j \cos\phi \\
\vdots \\
\xi_n
\end{pmatrix}
$$

V QR rozkladu jsou Givensovy rotace jsou voleny tak, aby nulovaly konkrétní prvky v rozkládané matici. Jejich výhodou je být, že je možné nulovat nenulové prvky v matici velmi specificky. Na druhou stranu, princip vzniku zaplnění v matici je analogický jako u Householderových reflexí. Konkrétně aplikace Givensovy rotace G na matici A ve tvaru GA implikuje, že struktury řídkosti A_{i*} a A_{j*} jsou sjednoceny na obou těchto řádcích.

15

Software pro přímé metody řešení řídkých soustav lineárních algebraických rovnic

V této kapitole stručně (někdy i heslovitě) popíšeme některý software pro přímé řešení soustav. Náš výběr je dán především snahou o demonstraci některých výše zmíněných inovativních prvků teorie řešení i algoritmů. O vlastnostech kódů budeme informovat pouze genericky s tím, že u mnohých z nich pokračuje vývoj, který dodává stále nové funkční a implementační možnosti. U publikací citujeme obvykle původní zdroje a nikoli často velké množství publikací prezentujících další vylepšení. Samozřejmě, že existuje mnoho dalších programů a nové stále vznikají (WSMP, Pardiso, Taucs, DSCPACK, KLU, NSPIV, atd.). Nicméně, náš úkol zde je hlavně demonstrovat, že popsané techniky našly od počátku své implementace a že i u soudobých aktualizací zmíněného software existuje stále více variant přístupu k řešení, které mohou používat výše uvedené stavební kameny. Mnohem obsáhlejší seznam najdeme například v [40] i s odkazem na pravidelně inovovaný seznam volně dostupných kódů.

15.1 Yale Sparse Package I. - symmetric codes: 1982

Jedním z prvních kódů moderního Choleského rozkladu, byl tento software založený na publikaci autorů Eisenstat, Gursky, Schulz, Sherman z roku 19áě [62]. Základní datovou strukturou programu je CSR (komprimovaná řídká struktura po řádcích: IA, JA, A). Počáteční uspořádání je založeno na algoritmu minimálního stupně. Postup předpokládá tři fáze rozkladu: symbolickou faktorizaci, numerickou faktorizaci a substituční kroky. Symbolická faktorizace je efektivní a je založena na míšení struktur řídkosti řádků přičemž ale velikost faktoru není spočítána dopředu. V naší klasifikaci lze postup v tomto software charakterizovat jako (transponovaný) sloupcový algoritmus bez použití bloků. Z předchůdců v použití CSR schématu nebo jeho analogií v software jmenujme například A. Changa, 1969, který oznámil software pro analýzu elektrických rozvodných sítí [30] a samotný program používal symbolickou a numerickou faktorizaci odděleně. Viz také návrh Curtise a Reida z roku 1971 s popisem dvou algoritmických postupů [37].

15.2 Yale Sparse Package II. - nonsymmetric codes: 1977

Algoritmus je založen na LDU rozkladu bez využití bloků [61]. Oproti CSR/CSC uložení řídkých matic pro matici soustavy a oba faktory se používá sofistikovanější komprimovaná struktura s využitím opakování určitých struktur matici. Uspořádání je pro větší stabilitu a minimalizaci zaplnění LDU rozkladu naplánované dopředu, ačkoli, jak víme, stabilita rozkladu může být velkým problémem. Poté je implementace bezprostředním rozšířením postupu pro symetrický rozklad.

15.3 SPARSPAK, 1981

Obsažná knihovna na řešení soustav se symetrickými a pozitivně definitními maticemi, později i lineární problém nejmenších čtverců. Základní literaturou je publikace autorů A. George a J.W.-H. Liu [75] v níž jsou kromě jiného popsány programy pro mnohé komponenty kódu popsané v jazyce Fortran. Počáteční uspořádání je založeno na násobném algoritmu minimálního stupně s případným počítání externího stupně Symbolická faktorizace je založena na eliminačním stromu (implicitně použitém). Velmi efektivní počítání v programu SPARSPAK, které ale bylo později nahraženo blokovými kódy a mnohými vylepšeními, které můžeme souhrnně nalézt v následujícím software.

15.4 Blkchol, 1997

Tento software v jazyku Fortran, který implementuje supervrcholový sloupcový Choleského rozklad, je učebnicovým příkladem použití mnohých komponent tohoto rozkladu diskutovaných výše. Byl vytvořen začátkem devadesátých let minulého století autory E. Ng a B. Peytonem [123], [124].

15.5 Y12M, 1981

Vlivný program, který už má jako jiné další především symbolický význam a který se zaměřil na řešení obecně nesymetrických soustav lineárních algebraických rovnic podmaticovou metodou. Stabilita rozkladu je podporována třemi základními pivotačními strategiemi zobecňujícími Markowitzův návrh na nesymetrické permutace a kombinujícími jej s kritérii na velikost prvku. Nově jsou zde navržena kritéria pro odvrhování prvků, která vedou k nepřesnému rozkladu a použití iteračního zpřesnění. Omezená prahová pivotace má svůj původ mimo jiné zde. Dynamické datové struktury, které umožňují vkládání zaplnění jsou podrobně popsané v [128] a navazují na struktury použité v programech z HSL, jako je MA28, zmíněné níže. Základní popis programu je v manuálu [167], viz také knihy [128] a [166] a citace v nich obsažené.

15.6 Nesymetrické kódy z Harwell Subroutine Library (HSL) jako MA28, 1983; MA48, 1996

Velká skupina programů vyvinutá v Harwell Laboratory (Rutherford Appleton Laboratory) zaměřených na řešení obecně nesymetrických soustav podmaticovými metodami s dynamickou pivotací. Kód MA48 je založen na blokovém rozkladu a vnitřně pracuje s technikou pivotace podle Gilberta a Peierlse [81]. Dynamické datové struktury zmíněné u Y12M mají svůj původ zde.

15.7 Symetrické a indefinitní multifrontální kódy z HSL od MA27, 1983 až po MA57, 2004 a HSL_MA97, 2013

Další skupina programů vyvinutá v Harwell Laboratory (Rutherford Appleton Laboratory) zaměřená především na řešení soustav se symetrickou a indefinitní matici, tedy umožňující také pivotaci s bloky 2×2. V jejich rámci vznikla první multifrontální implementace. Tak jako v jiných HSL programech jsou fáze symbolické a numerické faktorizace jinak přeskupeny a přejmenovány. Postupný vývoj byl i vývojem počátečních uspořádání se zahrnutím předpokládaných pivotů velikosti 2×2 a sofistikovaných pivotací s rozlišením tříd 2×2 pivotů, které například umožnily efektivnější řešení sedlobodových problémů.

15.8 MA41, 1990

Významným mezníkem v řešení nesymetrických soustav je kód MA41, což je nesymetrický multifrontální kód P. Amestoye a I. Duffa, kde je ale multifrontální metoda založena na struktuře řídkosti matice $A + A^T$ a pivotace využívá odsun pivotů. Kromě samotného kódu v knihovně HSL patří mezi základní publikace [4], [5]. Řídící strukturou je varianta eliminačního stromu, kde jsou bloky tvořeny i přirozenou amalgamací jeho vrcholů. Počáteční uspořádání je založeno na algoritmu AMD. Algoritmus je přirozeně paralelní využívající stromovový algoritmus i paralelní zpracování bloků. Vývoje tohoto kódu se zahrnutím velkého množství dalších postupů pro různé typy soustav vedl k programu MUMPS.

15.9 MUMPS, since 90s

Jednou z hlavních možností současnosti pro řešení soustav lineárních algebraických rovnic přímými řídkými metodami je mocný kód MUMPS. Jeho vývoj je možné vysledovat z následujících publikací, které odděluje propast času [7], [6]. Kód zahrnuje řešení soustav se symetrickou a pozitivně definitní maticí, obecně indefinitní i nesymetrickou maticí, zahrnuje mnoho alternativních metod na počáteční uspořádání i pivotace, je-li potřeba. Základní technikou je multifrontální metoda. Má ale mnoho dalších funkcionalit jako jsou

analýza chyby, počítání prvků inverzní matice nebo determinantu a umožňuje obecné formátové vstupy.

15.10 UMFPACK, Davis, since 2004

Základem software je plně nesymetrická multifrontální metoda [39], [38].

15.11 CHOLMOD, Davis, 2004

Jedním z nejrychlejších kódu současnosti pro řešení soustav se symetrickou a pozitivně definitní maticí je supervrcholová řídká Choleského faktorizace CHOLMOD, která umožňuje i dynamickou tvorbu supervrcholů a je plně založena na BLAS3 aritmetice; viz [31] Specifickým rysem tohoto programu je možnost aktualizovat tento rozklad při různých jednoduchých změnách dané matice.

15.12 SuiteSparse

Integrujícím kódem, který zahrnuje nejen nové verze UMFPACKu a CHOLMODu, ale i mnoho dalších funkcionalit včetně různých možností paralelizace výpočtů je SuiteSparse. Informace o tomto software je možné najít v řadě publikací T. Davise.

15.13 SuperLU, since 1999

SuperLU představuje alternativní pohled na rozklad tím, že vyšel jako nesymetrické zobecnění blokových sloupcových přístupů [44] a zahrnul nejen obecně nesymetrické počáteční uspořádání i částečnou pivotaci, nesymetrické supervrcholy, symetrické ořezávání pro rychlou symbolickou faktorizaci [103], [102]. Novým rysem kódu byly techniky nepřesného výpočtu diagonální modifikací doplněné iteračním zpřesněním. Vývoj tohoto kódu šel dále k výkonným paralelním implementacím a zahrnutím nových technik včetně datové řídkosti.

16

Přibližné maticové rozklady, štěpení a předpodmiňování

V této kapitole budeme hovořit o přibližných rozkladech matic. Velmi speciální maticové rozklady představují také jejich štěpení. Aplikace štěpení a aproximací rozkladů je obvykle úzce spjata s **iteračními metodami** (iteračními procesy) procesem zvaným **předpodmiňování** iteračních metod. Zde budeme diskutovat nejen interpretaci maticově založeného předpodmiňování v rámci stacionárních iteračních metod pro řešení soustav lineárních algebraických rovnic, ale probereme jeho souvislost s iteračními metodami založenými na krylovovských prostorech. Proto se nevyhneme jednoduchému přiblížení iteračních metod.

Jejich základem je vytváření **iterační posloupnosti** $x_0, x_1, \ldots$ do té doby, než získáme **přijatelné** řešení. Vyhodnocení, zda řešení je přijatelné a kdy tedy posloupnost ukončit, je problém, kterým se zde nebudeme zabývat. Je-li nějaký konkrétní člen iterační posloupnosti x, pak následující člen budeme často pro zjednodušení zápisu označovat x^+.

16.1 Stacionární iterační metody

Obecně, **jednokrokovou lineární stacionární iterační** metodou nazveme proces, ve kterém je vztah mezi dvěma po sobě následujícími členy $x, x^+ \in R^n$ vyjádřen

$$x^+ = Sx + M^{-1}b, \tag{16.1}$$

kde $S, M \in R^{n\times n}$ a kde matice M je regulární. Matice S se nazývá **iterační matice**. Takové iterační metody pro řešení soustav lineárních algebraických rovnic budeme dále zjednodušeně nazývat stacionární iterační metody.

16.1.1 Konzistence

Důležitou vlastností stacionární iterační metody (16.1) je její **konzistence**, vyjádřenou vztahem

$$x^* = Sx^* + M^{-1}Ax^*,$$

ze kterého plyne

$$S = I - M^{-1}A,$$

kde x^* je řešení soustavy rovnic $Ax = b$. Tento vztah můžeme přepsat do obecně používaného tvaru

$$x^+ = x - M^{-1}Ax + M^{-1}b \equiv (I - M^{-1}A)x + M^{-1}b. \tag{16.2}$$

Jiným vyjádřením konzistentní stacionární iterační metody je použít následující tvar **chybové korekce**

$$M(x - x^+) = Ax - b. \tag{16.3}$$

Různými volbami regulární matice M získáme různé iterační metody. Jednoduchou variantou je odvození matice M přímo z dané matice A. Volba matice M vztahem

$$A = M - R \equiv M - (M - A) \tag{16.4}$$

pro nějakou matici $R \in R^n$ se nazývá volba **štěpením** matice A. Nejjednodušší štěpení, které dále zmíníme, vybírají M ve tvaru diagonální či trojúhelníkové části matice A. Speciální iterační metoda pro $M = I$ se nazývá **metoda prosté iterace**.

Z motivačního pohledu je vhodné, aby matice M nějakým způsobem a "dostatečně dobře" aproximovala matici A, ale vlastností, které musí splňovat dobrá iterační metoda v praxi je víc a dále je zmíníme podrobněji. Matici M budeme nazývat **matice předpodmínění** metody prosté iterace a celý proces transformace nazveme **předpodmíněním** metody prosté iterace.

16.1.2 Konvergence

Stacionární iterační metodu pro řešení soustavy lineárních algebraických rovnic

$$Ax = b,\ A \in R^{n\times n},\ x \in R^n,\ b \in R^n \tag{16.5}$$

nazveme **konvergentní**, jestliže vytvářená posloupnost konverguje k řešení x^* této soustavy nezávisle na volbě počátečního přiblížení x_0. Připomeňme si nyní pojem spektrálního poloměru matice. **Spektrální poloměr** matice $S \in R^{n\times n}$ je dán vztahem

$$\lim_{k\to\infty} \| S^k \|^{1/k}, \tag{16.6}$$

ale dá se také vyjádřit pomocí pomocí jejích vlastních čísel výrazem

$$\rho(S) = \max\{|\lambda_i| \mid \lambda \in s(A)\}, \tag{16.7}$$

kde $s(A)$ označuje spektrum matice A. Známý výsledek je uveden v následující větě

Věta 16.1.1 *Stacionární iterační metoda (16.1) s iterační maticí S je konvergentní právě tehdy, platí-li*

$$\rho(S) < 1,$$

kde $\rho(S)$ je spektrální poloměr matice S.

16.1.3 Předpodmínění jako obecná transformace

Pojem předpodmínění můžeme zavést obecněji než transformaci metody prosté iterace. Uvažujme obecnou soustavu lineárních algebraických rovnic

$$Ax = b \tag{16.8}$$

a regulární matici M. Transformujme tuto soustavu podle vztahu

$$M^{-1}Ax = M^{-1}b. \tag{16.9}$$

Tato transformace ale bezprostředně souvisí s následující stacionární iterační metodou, u které předpokládáme její konvergenci k řešení soustavy (16.8). Doplněním přiblížení $x, x^+ \in R^n$ do rovnosti následujícím způsobem dostaneme iterační metodu, která je zjevně konzistentní

$$x^+ + M^{-1}Ax = x + M^{-1}b. \tag{16.10}$$

Tento vztah můžeme přepsat na tvar

$$x^+ = (I - M^{-1}A)x + M^{-1}b, \tag{16.11}$$

který je ekvivalentní (16.2).

To ukazuje význam předpodmínění jako obecnější algebraické transformace použitelné i mimo stacionární iterační metody. Matice předpodmínění může být nejen zvolena pomocí nějakého štěpení matice A, ale může reprezentovat **maticovou** nebo i obecnější lineární či nelineární transformaci soustavy. Transformovaná soustava ve tvaru (16.9) může pak být řešena i obecnějším iteračním postupem, například **nestacionární** krylovovskou metodou. Dále budeme tedy hovořit o obecném **předpodmínění iteračních metod** bez dalších přívlastků. Volná souvislost názvu této transformace s **podmíněností** soustavy lineárních rovnic vyplyne z následujícího textu.

16.1.4 Předpodmínění v rámci iteračních metod

Chápeme-li předpodmínění jako transformaci soustavy, máme dvě základní možnosti jak tuto transformaci včlenit do řešení soustavy. První možností je soustavu **přímo transformovat** na tvar (16.9) a teprve potom řešit. Problémem tohoto postupu i když neuvažujeme ztrátu přesnosti danou tímto přednásobením, je možný velký počet operací pro výpočet maticového výrazu

$$M^{-1}A. \tag{16.12}$$

Souvisejícím problémem je potenciálně velké zaplnění v matici $M^{-1}A$, je-li A rozsáhlá a řídká. Proto je takový postup vhodný jen ve speciálních případech.

Mnohem praktičtějším postupem je aplikace matici M^{-1} na matici A pouze **implicitně**. Pro naprostou většinu iteračních metod tento postup znamená, že se v každém iteračním kroku jejich algoritmů aplikuje matice M^{-1} nebo její transpozice na jeden nebo více vektorů ve formě násobení vektoru maticí. Z praktického pohledu pak není potřeba konstruovat žádnou další maticovou strukturu, kde by byly problémy s velikostí zaplnění.

16.2 Obecné vlastnosti předpodmínění

Bez podrobné analýzy toho, jak řešení soustavy závisí na konkrétní volbě předpodmínění M, nejde činit exaktní závěry o kvalitě jeho volby. Taková analýza musí nejenom zahrnovat konvergenční vlastnosti použité iterační metody, ale zároveň zahrnovat i implementaci a spolupráce s konkrétní počítačovou architekturou. Přesto je možné zhruba odhadnout, jaké vlastnosti jsou obecně vhodné pro kvalitní maticové předpodmínění M.

16.2.1 Kvalita předpodmínění

Matice M musí být především kvalitní aproximace matice A. Tuto kvalitu můžeme hodnotit různým způsobem.

- Za prvé, matice M a A by měly být **blízké**. Tuto blízkost můžeme hodnotit funkcionálem vzdáleností těchto matic v nějaké normě. Tato vzdálenost zapsaná jako

  $$\| M - A \|$$

  by měla být malá.

- Za druhé, i matice I a $M^{-1}A$ by měly být **blízké**. Tuto blízkost můžeme vyjádřit pomocí normy iterační matice

  $$\| I - M^{-1}A \| .$$

  Význam a odlišnost tohoto kritéria od předchozího je vidět, když si uvědomíme, že v reálné implementaci iteračního řešení transformované soustavy skutečně můžeme aplikovat matici $M^{-1}A$.

- Za třetí, je-li matice M dána součinem dvou nebo více faktorů, například ve tvaru $M = M_1M_2$, pak aplikace těchto faktorů v rámci iterační metody by měla být **stabilní**. Jsou-li tyto faktory dány rozkladem, pak by tento rozklad měl být stabilní. Stabilitu rozkladu lze v iterační metodě různými způsoby odhadovat.

- Za čtvrté, aplikace inverze matice M by měla mít **nízkou výpočetní složitost** za předpokladu, že v konkrétním algoritmu budeme počítat přímo matici M^{-1}. Předpodmínění založená na výpočtu M^{-1} budeme nazývat **inverzní předpodmínění**. Ostatní předpodmínění budeme nazývat **přímá předpodmínění**.

- Za páté, předpodmínění by mělo být vhodné pro konkrétní cílovou **počítačovou architekturu**, která může být i vektorová, paralelní nebo masívně paralelní. To může znamenat, že by předpodmínění mělo mít vhodnou blokovou strukturu nebo být snadno paralelizovatelné.

16.2.2 Jednostranná a oboustranná předpodmínění

Uvažujeme-li maticové předpodmínění jako transformaci, pak je více možností jak ji aplikovat. Konkrétně, transformaci pomocí M^{-1}, ať je tato matic uložena explicitně nebo

pouze implicitně, můžeme aplikovat na matici soustavy nejen, jako v (16.9), ale třeba násobením z pravé strany nebo z obou stran, předpokládáme-li, že matici M lze vyjádřit ve tvaru součinu dvou či více faktorů. Tři základní způsoby transformace zapsané jako

$$\begin{aligned} M^{-1}Ax &= M^{-1}b \\ AM^{-1}y &= b,\ x = My \\ M_1^{-1}AM_2^{-1}y &= M_1^{-1}b,\ x = M_2y,\ M = M_1M_2 \end{aligned}$$

nazýváme, po řadě, předpodmínění **zleva**, **zprava** a **oboustranné předpodmínění.** Bez důkladné analýzy nelze říci, která z variant je v konkrétním případě výhodnější. Následující věta jen dokresluje složitost situace.

Věta 16.2.1 *Nechť ϵ a Δ jsou kladná čísla. Pak platí, že pro každé $n \geq 2$ existují regulární matice $A \in R^n$ a $X \in R^n$ takové, že $XA - I$ má absolutní hodnoty všech svých prvků menší než ϵ a $AX - I$ má absolutní hodnoty všech svých prvků větší než Δ [118].*

Připomeňme ještě, že pro posuzování předpodmiňování v konkrétním případě nelze oddělit jeho numerické vlastnosti od implementace předpodmíněné iterační metody a vlastností cílové počítačové architektury.

16.2.3 Předpodmiňování soustav se symetrickou a pozitivně definitní maticí

Je-li matice A symetrická a pozitivně definitní, pak oboustranně předpodmíněná soustava

$$L_M^{-1}AL_M^{-T}y = L_M^{-1}b,\ \ x = L_M^Ty \tag{16.13}$$

založený na rozkladu $M = L_ML_M^T$ má opět symetrickou a pozitivně definitní matici $L_M^{-1}AL_M^{-T}$ a na její řešení je možné použít vhodnou iterační metodu. Tou je například **metoda sdružených gradientů.** Stále ale máme na výběr více ekvivalentních možností matematické formulace předpodmíněné iterační metody, jak naznačuje následující Věta 16.2.2.

Věta 16.2.2 *Uvažujme řešení soustavy $Ax = b$ a symetrickou a pozitivně definitní matici předpodmínění M. Pak*

- *$M^{-1}A$ je samosdružená ve skalárním součinu $(.,.)_M = (M.,.)$.*
- *AM^{-1} je samosdružená ve skalárním součinu $(.,.)_{M^{-1}} = (M^{-1}.,.)$.*

Důkaz: Samosdruženost příslušných maticových operátorů vyplývá z následujícího přepisu.

$$\begin{aligned} (M^{-1}Ax, y)_M &= (Ax, y) \\ &= (x, Ay) \\ &= (x, MM^{-1}Ay) \\ &= (Mx, M^{-1}Ay) \\ &= (x, M^{-1}Ay)_M \end{aligned}$$

$$(AM^{-1}x, y)_{M^{-1}} = (AM^{-1}x, M^{-1}y) = (M^{-1}x, AM^{-1}y) = (x, AM^{-1}y)_{M^{-1}} \tag{16.14}$$

■

Důsledek této Věty je následující.

Důsledek 16.2.1 *Metoda sdružených gradientů soustavy rovnic předpodmíněné* **zleva** *a založené na skalárním součinu* $(.,.)_M$*, metoda sdružených gradientů soustavy rovnic předpodmíněné* **zprava** *založená na skalárním součinu* $(.,.)_{M^{-1}}$ *a metoda sdružených gradientů se standardním skalárním součinem oboustranně předpodmíněná podle (16.13) poskytují v přesné aritmetice stejnou posloupnost iterací.*

Matematická ekvivalence však není vše. Výběr varianty metody konkrétního předpodmínění v rámci metody sdružených gradientů je pak v mnohém dán například tím, jaké jsou mezivýsledky metody, které se používají k monitorování konvergence předpodmíněné iterační metody, ale i samotnou implementací, kterou se mohou jednotlivé varianty lišit.

16.3 Předpodmínění metody prosté iterace

V této sekci ukážeme, jak se základní stacionární iterační metody, které lze chápat jako metodu prosté iterace s předpodmíněním založeným na štěpení matice, od sebe vzájemně liší tím, co od nich můžeme očekávat v praxi. Tedy především jak se liší konvergencí i teoretickými zárukami konvergence. Zapišme výše uvedenou metodu prosté iterace, kde $M = I$, ve tvaru

$$x^+ = (I - A)x + b \tag{16.15}$$

a uvažujme některá její předpodmínění vzniklé nahrazením jednotkové matice regulární maticí M ve vztahu (16.15). Vzniklé metody zde uveďme ve formě iteračních metod a u uvádění jejich vlastností zůstaneme u reálných matic, ačkoli některá zobecnění do obecnějšího komplexního oboru jsou přímočará.

16.3.1 Richardsonova metoda

Richardsonova metoda je určená volbou

$$M = (1/\theta)\, I, \tag{16.16}$$

kde $\theta \in \mathbf{R}$ je skalární parametr. Následující věta je základním výsledkem, týkajícím se její konvergence ve speciální případě

Věta 16.3.1 *Pro matici A s kladnými vlastními čísly metoda a reálným parametrem θ konverguje Richardsonova metoda pro libovolné počáteční přiblížení x_0 právě tehdy, platí-li*

$$0 < \theta < 1/\lambda_{max}(A), \tag{16.17}$$

kde $\lambda_{max}(A)$ je maximální vlastní číslo matice A.

Důkaz: Vztah (16.17) implikuje

$$-1 < 1 - \theta\lambda_{max}(A) \leq 1 - \theta\lambda_{min}(A) < 1 \tag{16.18}$$

a proto také

$$\rho(I - M^{-1}A) = \rho(I - \theta A) < 1 \tag{16.19}$$

a jedna implikace je dokázána. Konverguje-li metoda, pak

$$1 - \theta\lambda_{max}(A) \leq |1 - \theta\lambda_{max}(A)| \leq \rho(I - \theta A) < 1. \tag{16.20}$$

Protože matice A má kladná vlastní čísla, musí platit $\theta > 0$. Na druhé straně, protože

$$1 - \theta\lambda_{max}(A) \geq -|1 - \theta\lambda_{max}(A)| \geq -\rho(I - \theta A) > -1, \tag{16.21}$$

pak musí být i

$$\theta\lambda_{max}(A) > 0 \tag{16.22}$$

a důkaz je hotov. ■

Rychlost konvergence Richardsonovy iterační metody je dána jejím poloměrem konvergence určeným následující větou.

Věta 16.3.2 *Nechť má matice A pouze reálná vlastní čísla a uvažujme reálný parametr θ. Pak její poloměr konvergence je dán vztahem*

$$\rho(I - \theta A) = \max(|1 - \theta\lambda_{min}(A)|, |1 - \theta\lambda_{max}(A)|). \tag{16.23}$$

Důkaz: Tvrzení věty plyne z toho, že vlastní čísla iterační matice jsou dána výrazem

$$1 - \theta\lambda,\ \lambda \in \sigma(A) \tag{16.24}$$

a z faktu, že funkce $|1-\theta x|$ nemá lokální maxima a poloměr konvergence je dán hodnotou v některé ze svých extremálních hodnot. ■

Snadno nahlédneme, že optimální rychlost konvergence získáme pro parametr minimalizující poloměr konvergence, tedy minimalizující výraz

$$\max(|1 - \theta\lambda_{min}(A)|, |1 - \theta\lambda_{max}(A)|), \tag{16.25}$$

což vede k porovnání

$$\theta\lambda_{max}(A) - 1 = 1 - \theta\lambda_{min}(A) \tag{16.26}$$

a tedy pro

$$\theta = \frac{2}{\lambda_{min}(A) + \lambda_{max}(A)}. \tag{16.27}$$

16.3.2 Jacobiho metoda

Další velmi používanou metodou je Jacobiho metoda založená na štěpení $A = M - N$, kde

$$M = D_A.$$

D_A zde standardně označuje diagonální část matice A. Definujme nyní **ostře diagonálně dominantní** matici.

Definice 16.3.1 *Řekneme, že matice $A \in \mathbb{R}^{n\times n}$ je* **ostře diagonálně dominantní**, *platí-li pro prvky na jejích řádcích vztah*

$$\sum_{j=1, j\neq i}^{n} |\, a_{ij}| < |\, a_{ii}|,\ i = 1, \ldots, n \tag{16.28}$$

Platí-li pro alespoň jeden řádek rovnost a musíme-li tedy ve vztahu (16.28) použít neostré nerovnosti, hovoříme o **diagonálně dominantní** *matici.*

Následující dva konvergenční výsledky charakterizují nejzákladnější vlastnosti Jacobiho metody.

Věta 16.3.3 *Je-li matice $A \in \mathbb{R}^{n\times n}$ ostře diagonálně dominantní, Jacobiho metoda konverguje pro libovolné počáteční přiblížení x_0.*

Důkaz: Pro ostře diagonálně dominantní matici máme podle definice

$$\sum_{j=1, j\neq i}^{n} |a_{ij}| < |a_{ii}|,\ i = 1, \ldots, n$$

tedy

$$\sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1,\ i = 1, \ldots, n \tag{16.29}$$

a tedy

$$\max_{i=1,\ldots,n} \sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1, \tag{16.30}$$

což je postačující podmínka pro konvergenci v maximové maticové normě. ■

Věta 16.3.4 *Je-li matice $A \in \mathbb{R}^{n\times n}$ symetrická s diagonálou D_A a kladnými diagonálními prvky, pak Jacobiho metoda konverguje pro libovolné počáteční přiblížení x_0 právě tehdy, jsou-li A a $2D_A - A$ pozitivně definitní.*

Důkaz: Nejprve si všimněme, že matice $2D_A - A$ se liší od matice A jen změnou znamének u mimodiagonálních prvků. Nechť $D_1 = diag(\sqrt{a_{11}}, \ldots, \sqrt{a_{nn}})$, to jest $D = D_1^2$. Pak dostaneme podobnostní transformaci

$$I - D^{-1}A = D^{-1}(L + U) = D_1^{-1}[D_1^{-1}(L + U)D_1^{-1}]D_1. \tag{16.31}$$

Jacobiho metoda tedy konverguje právě tehdy, je-li $\rho(D_1^{-1}(L+U)D_1^{-1}) < 1$.

Matice $D_1^{-1}(L+U)D_1^{-1} \equiv I - D_1^{-1}AD_1^{-1}$ je symetrická. A protože symetrická matice má všechna vlastní čísla reálná, pak Jakobiho metoda konverguje právě tehdy, jsou-li tato vlastní čísla z otevřeného intervalu $(-1, 1)$. To platí právě tehdy, mají-li obě matice $M_1 = I + D_1^{-1}(L+U)D_1^{-1}$ a $M_2 = I - D_1^{-1}(L+U)D_1^{-1}$ všechna vlastní čísla kladná. Protože jsou obě tyto matice také symetrické, toto platí právě tehdy, když jsou obě pozitivně definitní. Protože ale

$$M_1 = I + D_1^{-1}(L+U)D_1^{-1} = D_1^{-1}(D_1^2 + L + U)D_1^{-1} = D_1^{-1}(2D - A)D_1^{-1} \tag{16.32}$$

a také

$$M_2 = I - D_1^{-1}(L+U)D_1^{-1} = D_1^{-1}(D_1^2 - L - U)D_1^{-1} = D_1^{-1}AD_1^{-1}. \tag{16.33}$$

Tvrzení je tak dokázáno, protože je-li obecně matice B pozitivně definitní a T regulární, pak je T^TBT také pozitivně definitní. ■

Jacobiho metoda bývá někdy velmi užitečně zobecněna doplněním skalárního parametru θ na tvar matice M

$$M = (1/\theta)\, D_A.$$

16.3.3 Gauss-Seidelova metoda

Uvažujme matici A rozloženou štěpením následujícím (konvenčním) způsobem na diagonální část D_A, striktní dolní trojúhelníkovou část L_A a striktní horní trojúhelníkovou část U_A

$$A \;=\; D_A - L_A - U_A.$$

Gauss-Seidelova metoda je dána volbou

$$M = D_A - L_A, \tag{16.34}$$

jejíž inverze se dá aplikovat jako substituce (přímý chod). Na sekvenčních počítačích je tato substituce efektivní. Iterační matice Gauss-Seidelovy metody se dá potom zapsat ve tvaru $(D_A - L_A)^{-1}U$, neboť platí

$$\begin{aligned} I - M^{-1}A &= I - M^{-1}(D_A - L_A - U_A) \\ &= I - (D_A - L_A)^{-1}(D_A - L_A) + (D_A - L_A)^{-1}U_A \\ &= (D_A - L_A)^{-1}U_A. \end{aligned}$$

První konvergenční výsledek pro Gauss-Seidelovu je analogií tvrzení pro Jacobiho metodu.

Věta 16.3.5 *Je-li matice $A \in \mathbb{R}^{n\times n}$ ostře diagonálně dominantní, Gauss-Seidelova metoda konverguje pro libovolné počáteční přiblížení x_0.*

Důkaz: Pro konvergenci stačí, aby platilo $\|(D_A - L_A)^{-1}U_A\|_\infty < 1$. Tato maticová norma je generována vektorovou maximovou normou a proto musí existovat vektor u takový, že platí

$$\|(D_A - L_A)^{-1}U_A\|_\infty = \|(D_A - L_A)^{-1}U_A\, u\|_\infty,\ \|u\|_\infty = 1. \tag{16.35}$$

Označme $v = (D_A - L_A)^{-1}U_A u$. Pro složky vektoru v, kde $\|v\|_\infty = \|(D_A - L_A)^{-1}U_A\, u\|_\infty$ platí $|v_i| \le \|v\|_\infty$. Nechť $|v_s| = \|v\|_\infty$. Napišme nyní vztah mezi vektory u a v

$$(D_A - L_A)v = U\, u.$$

to jest pro s-tý řádek

$$a_{s1}v_1 + \ldots + a_{ss}v_s = -a_{s,s+1}u_{s+1} - \ldots - a_{sn}u_n.$$

Odtud

$$v_s = -\sum_{i=1}^{s-1} \frac{a_{si}}{a_{ss}} v_i - \sum_{i=s+1}^{n} \frac{a_{si}}{a_{ss}} u_i$$

Označme

$$a = \sum_{i=1}^{s-1} \left|\frac{a_{si}}{a_{ss}}\right|, \quad b = \sum_{i=s+1}^{n} \left|\frac{a_{si}}{a_{ss}}\right|.$$

Neboť matice A je ostře diagonálně dominantní, pak platí $a + b < 1$. Dále

$$\|v\|_\infty = |v_s| \le a\|v\|_\infty + b\|u\|_\infty = a\|v\|_\infty + b$$

a z toho plyne

$$\|(D_A - L_A)^{-1}U_A\|_\infty = \|v\|_\infty \le \frac{b}{1-a} < 1, \tag{16.36}$$

jak jsme měli dokázat. ■

Další výsledek se týká symetrických a pozitivně definitních matic a ukazuje, že Gauss=Seidelova metoda již odpovídá relativně silně předpodmíněné metodě prosté iterace.

Věta 16.3.6 *Je-li matice $A \in \mathbb{R}^{n\times n}$ symetrická a pozitivně definitní, Gauss-Seidelova metoda konverguje pro libovolné počáteční přiblížení x_0.*

Důkaz: Uvažujme danou matici ve tvaru standardního štěpení $A = D_A - L_A - U_A,\ L_A = U_A^T$ a ukažme, že $\rho(D_A - U_A^T)^{-1}U_A < 1$. Nechť λ je vlastní číslo matice $(D_A - U_A^T)^{-1}U_A$ a u vlastní vektor odpovídající tomuto vlastnímu číslu. Note that both the eigenvalue and eigenvector can be complex. Pak platí

$$(D_A - U_A^T)^{-1}U_A u = \lambda u \Rightarrow U_A u = \lambda D_A u - \lambda U_A^T u = \lambda(D_A - U_A^T - U_A)u + \lambda U_A u = \lambda A u + \lambda U_A u.$$

Pro následující skalární součin s vektorem u pak dostaneme

$$(U_A u, u) = \lambda(Au, u) + \lambda\,(U_A u, u).$$

Označme $(Au, u) = p$. Z pozitivní definitnosti matice A víme, že $p > 0$. Uvažujme dále označení $(U_A u, u) = a + ib$, kde p, a, b jsou reálná čísla. Pak můžeme psát

$$a + ib = \lambda\, p + \lambda\,(a + ib) \Rightarrow \lambda = \frac{a + ib}{p + a + ib}.$$

Dále uvidíme, že $p + a$ je nenulové a zlomek je tedy korektně definován. Vynásobením tohoto zlomku číslem $\bar{\lambda}$ získáme

$$|\lambda|^2 = \frac{a^2 + b^2}{(p + a)^2 + b^2}.$$

Předcházející vztah také můžeme vidět z následujícího odvození

$$\left|\frac{a + ib}{p + a + ib}\right|^2 = \left|\frac{\rho_1}{\rho_2} e^{i(\phi_1 - \phi_2)}\right|^2 = \left|\frac{\rho_1}{\rho_2}\right|^2 = \frac{|\rho_1|^2}{|\rho_2|^2} = \frac{|a + ib|^2}{|p + a + ib|^2} = \frac{a^2 + b^2}{(p + a)^2 + b^2}.$$

Neboť

$$(U_A^T u, u) = (u, U_A u) = \overline{(U_A u, u)} = a - ib,$$

pak

$$p = (D_A u, u) - (U_A u, u) - (U^T u, u) = (D_A u, u) - (a + ib + a - ib) = (D_A u, u) - 2a.$$

$(D_A u, u)$ je podle předpokladu kladné, neboť pozitivně definitní matice A musí mít kladné diagonální prvky a také

$$(D_A u, u) = \sum_{i=1}^{n} a_{ii} u_i^* u_i = \sum_{i=1}^{n} a_{ii} |u_i|^2.$$

Dále

$$(p + a)^2 = p(p + 2a) + a^2 = p\,(D_A u, u) + a^2.$$

Všimněme si, že $p(D_A u, u)$ musí být kladné, neboť je součinem kladných čísel. Dostáváme tedy

$$|\lambda|^2 = \frac{a^2 + b^2}{p\,(D_A u, u) + a^2 + b^2}$$

a neboť jsme neměli omezení na výběr vlastního čísla, tvrzení věty je dokázáno. Důkaz také ukazuje, že ani předpoklad, že matice A je obecně komplexním, by jej nijak výrazně nezkomplikoval. ■

16.3.4 SOR (successive over-relaxation)

Tato metoda je reprezentant metod, založených na následujícím parametrizovaném štěpení.

$$\begin{aligned} \omega A &= (D_A - \omega L_A) + (\omega - 1) D_A - \omega U_A \\ 0 &< \omega < 2 \end{aligned}$$

V tomto případě je možné psát

$$M = \omega^{-1}(D_A - \omega L_A),\ S = (D_A - \omega L_A)^{-1}[(1 - \omega) D_A + \omega U_A].$$

Ke konvergenci této metody a jiných ještě složitějších variant metod založených na štěpení, často s mnoha parametry, existuje rozsáhlá teorie. V následujícím textu ukázku jednoho konvergenčního výsledku uvedeme. Definujme si nejprve slabě cyklickou matici indexu $p > 1$.

Definice 16.3.2 *Řekneme, že matice $A \in \mathbf{R}^{n\times n}$ je slabě cyklická s celočíselným indexem $p > 1$ právě tehdy, existuje-li permutační matice $P \in \mathbf{R}^{n\times n}$ taková, že matice PAP^T má netriviální blokový tvar*

$$\begin{pmatrix} 0 & 0 & \dots & 0 & \dots & A_{1p} \\ A_{21} & 0 & \dots & 0 & \dots & 0 \\ 0 & A_{32} & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & A_{p,p-1} & \dots & 0 \end{pmatrix} \tag{16.37}$$

se čtvercovými diagonálními bloky.

Vlastnost slabé cykličnosti omezuje množinu vlastních čísel následujícím způsobem.

Věta 16.3.7 *Má-li slabě cyklická matice s indexem 2 vlastní číslo λ, pak má i vlastní číslo $-\lambda$.*

Definice 16.3.3 *Řekneme, že matice $A \in \mathbf{R}^{n\times n}$ je dvoucyklická, je-li její Jacobiova matice $D_A^{-1}(L_A + U_A)$ slabě cyklická s indexem 2. Dvoucyklická matice se nazývá shodně uspořádaná, jestliže vlastní čísla matice*

$$\alpha D_A^{-1} L_A + \alpha^{-1} D_A^{-1} U_A$$

nezávisí na volbě $\alpha \neq 0$.

S pomocí uvedených definic nyní vyslovíme bez důkazu větu o konvergenci metody SOR.

Věta 16.3.8 *Předpokládejme, že parametr SOR metody ω je nenulový a matice $A \in \mathbf{R}^{n\times n}$ je dvoucyklická, shodně uspořádaná a pozitivně definitní. Pak metoda SOR konverguje pro libovolné počáteční přiblížení x_0 právě tehdy, je-li $\omega \in (0, 2)$.*

Kolem této metody vznikla dále celá teorie, která například diskutuje volbu ω. Nicméně, hlavním důvodem, proč jsme zde tento text uvedli je, abychom viděli, že k docílení rychlé konvergence metody je obvykle apriorní znalost velmi speciálních vlastností matice A. Bez ohledu na krásu teorie, často takovou znalost nemáme.

16.3.5 SSOR (symmetric successive over-relaxation)

Stejně jako Gauss-Seidelova metoda, tak i metoda SOR poskytují obecně nesymetrickou matici M. Existuje symetrická varianta metody SOR označovaná jako SSOR, stejně tak jako existuje symetrická varianta Gauss-Seidelovy metody. Metoda SSOR je dána maticí předpodmínění

$$M = \omega^{-1}(D_A - \omega L_A) D_A^{-1} (D_A - \omega U_A).$$

K metodám SOR a SSOR se ještě později vrátíme s tím, že ukážeme, jak koncept speciálních matic umožnil jednodušší formulaci vlastností iteračních metod.

16.3.6 Stacionární iterační metody a další rozvoj předpodmínění

Jak bylo popsáno, že záruky konvergence pro předpodmínění založená na stacionárních iteračních metodách jsou založeny na dobře definovaném vztahu mezi maticemi M a A. Takto zkonstruovaná předpodmínění lze samozřejmě aplikovat jako transformace i mimo koncept stacionárních iterací v rámci krylovovských metod aplikovaných na soustavu (16.9). Lze tedy například předpodmínit metodu sdružených gradientů (kde je předpokladem symetrie a pozitivní definitnost matice A) maticí D_A z Jacobiho metody. Postupem času přestala ale předpodmínění konstruovaných na základě štěpení matice A stačit jak z hlediska teoretického popisu, tak i z hlediska efektivity.

Další rozvoj předpodmíněných iteračních metod pro řešení složitějších úloh postupoval v zásadě dvojím způsobem. **První možností** bylo omezit teoretickými vlastnostmi samotnou matici A. Jinými slovy, bylo třeba předpokládat, že matice A má velmi specifické vlastnosti, pro které ji budeme stručně nazývat **speciální matice**. Cesta tímto směrem je například požadavek ostré diagonální dominance matice A. Příklad mnohem sofistikovanějších, ale i speciálnějších požadavků jsme viděli v případě metody SOR. Je třeba ale dodat, že řada požadavků na takto formálně vyjádřenou speciálnost matice byla historicky motivována skutečně řešenými problémy. Pak bylo ovšem silnou motivací včlenit požadavky na matici do nějakého rámce daného základními etalony speciálních matic.

Druhou možností bylo získat matici M nikoli jen štěpením, ale použít k tomu složitější, ale **dobře definovaný proces** vycházející z matice A. I v tomto případě je možné získat teoretické odůvodnění, byť ne třeba velmi obecné. Z praktického pohledu lze ale získat velmi efektivní předpodmínění. Obě zmíněné možnosti hrály svou roli v historii vývoje moderních předpodmínění, vzájemně se prolínaly a oba tyto způsoby v následujícím textu zmíníme.

16.4 Předpodmínění a speciální matice

V řadě praktických případů lze získat speciální matice přímo z dobře definovaných numerických metod pro řešení parciálních diferenciálních rovnic. Jedním ze základních etalonů speciálních matic je **M-matice**. Existuje celá řada různých způsobů, jak ji definovat a zde si vybíráme jeden z nich.

Věta 16.4.1 *Matici A nazveme* **regulární M-maticí**, *je-li matice $A + D$ regulární pro každou nezápornou diagonální matici D a zároveň jsou všechny mimodiagonální prvky matice A nekladné. Tato poslední podmínka se někdy nazývá, že matice A je* **Z-matice**.

Jinou definicí, která je atraktivní v souvislosti s konvergencí stacionárních iteračních metod je

Věta 16.4.2 *Matici A nazveme* **regulární M-maticí**, *platí-li pro její prvky $a_{ij} \leq 0$, je regulární a platí $A^{-1} \geq 0$.*

Když budeme v dalším textu hovořit o M-maticích, budeme vždy předpokládat jejich regularitu a zkracovat tak jejich plné označení. Příklad M-matice je na následujícím obrázku

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

Zaveďme dále podmínky, které rozšiřují pojem ostré diagonální dominance a navažme na ně pojem H-matice, který použijeme k formálnějšímu a jednoduššímu popisu konvergence předpodmínění založených na stacionárních iteračních metodách.

Věta 16.4.3 *Řekneme, že matice A je* **zobecněně diagonálně dominantní***, je-li matice AD diagonálně dominantní pro nějakou diagonální matici D s kladnými prvky; matice A je* **nerozložitelně diagonálně dominantní***, je-li diagonálně dominantní a zároveň nerozložitelná (neexistuje její permutace na netriviální blokově trojúhelníkový tvar).*

Věta 16.4.4 *Řekneme, že matice A je* **H-matice** *jestliže matice $B = |D_A| - |A - D_A|$ je M-matice. Výraz $|X|$ zde vyjadřuje matici s prvky z absolutních hodnot prvků matice X.*

Souvislost pojmů z těchto definic vystihuje následující věta.

Věta 16.4.5 *Matice ostře diagonálně dominantní, zobecněně diagonálně dominantní a nerozložitelně diagonálně dominantní jsou zároveň H-maticemi.*

Zavedené speciální matice nám umožňují jednoduše formulovat konvergenční výsledky, z nichž některé jsme již uvedli dříve a kde při konvergenci uvažujme libovolné počáteční přiblížení x_0.

- Je-li A M-matice, pak Jacobiho, Gauss-Seidelova a také SOR metoda pro $0 \leq \omega < 1$ jsou konvergentní.
- Jacobiho a Gauss-Seidelova metoda konvergují pro ostře diagonálně dominantní matice.
- Jacobiho a Gauss-Seidelova metoda konvergují pro nerozložitelně diagonálně dominantní matice. Totéž platí pro SOR s parametrem $0 < \omega \leq 1$
- Je-li A H-matice, pak SOR konverguje pro parametr
$$0 < \omega < \frac{2}{1 + \rho(|(I - (D_A - \omega L_A)^{-1} A)|)}.$$
- SOR obecně nekonverguje pro $\omega \leq 0$ nebo $\omega \geq 2$ [122].
- Pro A s pozitivní maticí D_A metoda SOR konverguje pro $0 < \omega < 2$ právě tehdy, je-li A symetrická a pozitivně definitní [129], [132].

- Je-li A H-matice, pak SSOR konverguje pro parametr

$$0 < \omega < \frac{2}{1 + \rho(|(I - M^{-1}A)|)},$$

kde $(I - M^{-1}A$ je iterační matice metody SSOR [2].

- Pro symetrickou a pozitivně definitní maticí A metoda SSOR pro $0 < \omega < 2$ konverguje.

16.5 Předpodmínění a jednoduché maticové procesy

Odvozování matice předpodmínění štěpením A i za předpokladu, že matice je velmi speciální, má svá silná omezení. Hlavní směr, který zde budeme dále popisovat, je založen naší druhé možnosti a tedy na konstrukci matice M z matice A pomocí dobře definovaného procesu. Tento postup byl v zásadě i postupem historického vývoje směřujícího přes iterační metody s velmi speciálními požadavky na matici A, jako jsme uvedli u metody SOR, konsolidací pro "univerzálnější" speciální matice směrem k tomu, co budeme nazývat neúplné rozklady.

16.5.1 Prostá substituce a její zobecnění

V následujícím textu ukážeme, jak se předpodmínění ve tvaru rozkladu pro obecnější matice dá odvodit analogií k případu řešení soustavy s tridiagonální maticí.

16.5.1.1 Soustavy s tridiagonální maticí

Uvažujme následující metodu pro řešení soustav rovnic s tridiagonální maticí, kde předpokládáme, že pro stabilitu rozkladu není potřeba pivotace. To je zřejmé v případě, kdy je matice soustavy symetrická a pozitivně definitní, ale často lze samozřejmě rozkládat bez pivotace i matice obecnější. Tuto metodu pro soustavy s tridiagonální matice budeme nazývat metodou **prosté substituce**, ale byla zavedena i pod názvem faktorizační metoda (Tichonov, Nikolaev) nebo **sweeping method** (Ilin). Zapišme tedy matici A soustavy rovnic s tridiagonální maticí

$$Ax = f \tag{16.38}$$

s následujícím označením jejích prvků

$$A = \begin{pmatrix} c_1 & -b_1 & & & & & & \\ -a_2 & c_2 & -b_2 & & & & & \\ & -a_3 & c_3 & -b_3 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & -a_i & c_i & -b_i & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & -a_{n-1} & c_{n-1} & -b_{n-1} \\ & & & & & & -a_n & c_n \end{pmatrix} \tag{16.39}$$

Jedna z možností, jak najít řešení této soustavy je postupně vyjadřovat proměnné $x_1, x_2, \dots x_n$ dosazováním z první rovnice do druhé, z druhé do třetí atd. Jinými slovy, postupovat tak, jak by postupovala i Gaussova eliminace. Zaveďme tedy nejprve označení

$$\beta_1 = b_1/c_1,\ z_1 = f_1/c_1 \tag{16.40}$$

a přepišme první dvě rovnice soustavy následovně.

$$\begin{aligned} x_1 - \beta_1 x_2 &= z_1 \\ -a_2 x_1 + c_2 x_2 - b_2 x_3 &= f_2 \end{aligned}$$

Přičtením a_2 násobku první z těchto rovnic ke druhé získáme rovnici

$$(c_2 - a_2\beta_1)x_2 - b_2 x_3 = f_2 + a_2 z_1 \tag{16.41}$$

a formálně tuto rovnici zapíšeme následujícím způsobem

$$x_2 - \beta_2 x_3 = z_2, \tag{16.42}$$

kde jsme definovali β_2 a z_2 vztahy

$$\beta_2 = \frac{b_2}{c_2 - \beta_1 a_2},\ z_2 = \frac{f_2 + a_2 z_1}{c_2 - \beta_1 a_2}. \tag{16.43}$$

Opakujme tento substituční krok. Nejprve k získané rovnici přibereme třetí rovnici původní soustavy, následně ji převedeme na analogický tvar, pro který definujeme hodnoty β_3 a z_3 a takto pokračujme dále. V obecném kroku tak získáme vztahy

$$x_i - \beta_i x_{i+1} = z_i,\ \beta_i = \frac{b_i}{c_i - \beta_{i-1} a_i},\ z_i = \frac{f_i + a_i z_{i-1}}{c_i - \beta_{i-1} a_i},\ i = 2, \dots, n-1 \tag{16.44}$$

K poslední z takto získaných rovnic přidáme poslední rovnic původní soustavy a dostáváme vztahy

$$\begin{aligned} x_{n-1} - \beta_{n-1} x_n &= z_{n-1} \\ -a_n x_{n-1} + c_n x_n &= f_n. \end{aligned}$$

Z nich vyjádříme hodnoty x_n a x_{n-1} přičtením a_n násobku první z nich k druhé rovnici a při definování skaláru z_n

$$x_n = z_n,\ x_{n-1} = \beta_{n-1} x_n + z_{n-1}, z_n = \frac{f_n + a_n z_{n-1}}{c_n - \beta_{n-1} a_n} \tag{16.45}$$

Hodnoty řešení pak vyjádříme následujícím rekurentním vztahem, kde potřebné hodnoty β_i a z_i jsou již známy.

$$x_i = \beta_i x_{i+1} + z_i,\ i = n-1, \dots, 1. \tag{16.46}$$

Tento odvozený vztah je tedy dán dvojí závislostí jednostrannou rekurencí mezi komponentami řešení dané soustavy s tridiagonální maticí. Nejprve vyjadřujeme postupně

vzájemnou závislost hodnot z_i, $i = 1, \dots n$ na předcházejících hodnotách. Potom jsou v opačném pořadí počítány hodnoty x_i, $i = n, \dots 1$.

Vyjádříme-li vztahy mezi použitými veličinami a koeficienty matice A pro $i = 1, \dots, n$ následovně se zavedením komponent vektoru δ

$$\begin{aligned} \delta_i &= c_i - a_i\beta_{i-1}, \\ \beta_i &= b_i/\delta_i \\ f_i &= -a_i z_{i-1} + \delta_i z_i, \end{aligned}$$

pro $z_0 = 0,\ \beta_0 = 0,\ x_{n+1} = 0$, vidíme že rekurentní výpočet můžeme interpretovat jako následující LU rozklad matice A

$$A = LU \equiv \begin{pmatrix} \delta_1 & & & & & \\ -a_2 & \delta_2 & & & & \\ & \ddots & \ddots & & & \\ & & -e_i & \delta_i & & \\ & & & \ddots & \ddots & \\ & & & & -e_n & \delta_n \end{pmatrix} \begin{pmatrix} 1 & -\beta_1 & & & & \\ & \ddots & \ddots & & & \\ & & 1 & -\beta_i & & \\ & & & \ddots & \ddots & \\ & & & & 1 & \beta_{n-1} \\ & & & & & 1 \end{pmatrix} \tag{16.47}$$

a získání řešení jako substituční kroky

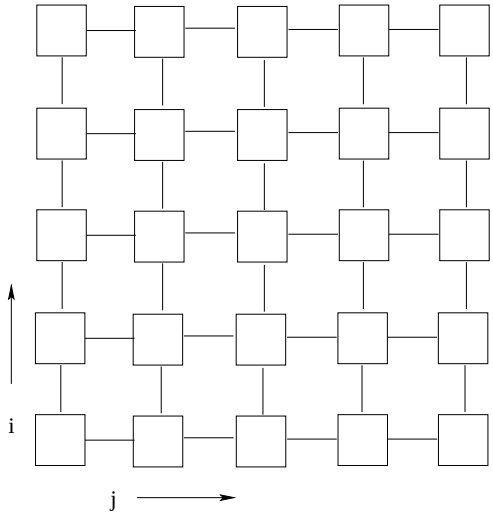
$$\begin{aligned} Lz &= f \\ Ux &= z. \end{aligned}$$

Vztahy pro prvky znázorněných matic L a U získáme porovnáním levých a pravých stran vztahů

$$a_{ij} = L_{i*}U_{*j},\ i, j = 1, \dots, n. \tag{16.48}$$

16.5.1.2 Zobecnění na soustavy z pětibodového (2D) síťového schématu

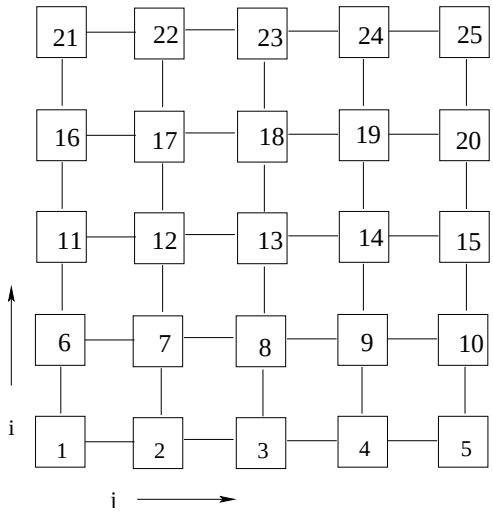
Uvažujme soustavu rovnic, která vznikla diskretizací parciální diferenciální rovnice na pravidelné dvojrozměrné síti s diskretizačním parametrem h. Ten je dán vzdálenostmi mezi body sítě ve směru souřadných os. Její prvky a stejně tak i prvky hledaného vektoru řešení budeme označovat dvěma indexy, které označují jejich polohu v dvojrozměrné síti. Pro jednoduchost předpokládáme, že síť je pravidelná čtvercová s oběma síťovými dimenzemi stejnými a rovnými k. Matice tedy bude mít dimenzi $n = k^2$. Taková síť pro $k = 5$ je znázorněna na následujícím obrázku.



Vrcholy sítě odpovídají ve standardních neorientovaných nebo orientovaných grafových modelech řádkům a sloupcům matice a můžeme je tedy očíslovat také pomocí bijekce β mezi vrcholy $(i, j) \in \{1, \ldots, k\} \times \{1, \ldots, k\}$ sítě a indexy řádků/sloupců danou vztahem

$$\beta : (i, j) \longleftrightarrow i + i + nx * (j - 1). \tag{16.49}$$

Takovému očíslování matice říkáme **přirozené** očíslování a očíslování znázorníme na následujícím obrázku.



Je-li diskretizovaný operátor negativní Laplaceův operátor, pak pětibodové síťové schéma jeho jednoduché diskretizace je dáno vztahem pro hodnoty řešení

$$x_{i,j} \approx \frac{4x_{i,j} - x_{i-1,j} - x_{i+1,j} - x_{i,j-1} - x_{i,j+1}}{h^2},$$

které v přirozeném uspořádání poskytne matici se strukturou řídkosti jako je na následujícím obrázku.

V obecnějším případě můžeme například vztah odpovídající řešení ve vrcholu sítě, tedy jeden řádek matice, zapsat

$$\begin{pmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{pmatrix} \tag{16.50}$$

Obrázek 16.5.1: *Matice diskretizované Poissonovy rovnice ve 2D s přirozeným očíslováním vrcholů sítě.*

$$-a_{i,j}x_{i-1,j} - b_{i,j}x_{i,j-1} + e_{i,j}x_{i,j} - c_{i,j}x_{i+1,j} - d_{i,j}x_{i,j+1} = f_{i,j} \tag{16.51}$$

a celá matice A má následující strukturu

$$\begin{pmatrix} e_{1,1} & -c_{1,1} & & -d_{1,1} & & & & & & & \\ -a_{2,1} & e_{2,1} & -c_{2,1} & & -d_{2,1} & & & & & & \\ & \ddots & \ddots & \ddots & & \ddots & & & & & \\ -b_{1,2} & & -a_{1,2} & e_{1,2} & -c_{1,2} & & -d_{1,2} & & & & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots & & & \\ & & -b_{j,k} & & -a_{j,k} & e_{j,k} & -c_{j,k} & & -d_{j,k} & & \\ & & & \ddots & & \ddots & \ddots & \ddots & & \ddots & \\ & & & & \ddots & & \ddots & \ddots & \ddots & & -d_{k,k-1} \\ & & & & & \ddots & & \ddots & \ddots & \ddots & \\ & & & & & & \ddots & & \ddots & \ddots & -c_{k-1,k} \\ & & & & & & & -b_{k,k} & & -a_{k,k} & e_{k,k} \end{pmatrix}$$

Analogicky k metodě prosté substituce s tridiagonální maticí předpokládejme, že komponenty řešení soustavy

$$Ax = f$$

pro i, $j = 1, \ldots, k$ se dají vyjádřit závislostmi

$$x_{i,j} = \xi_{ij}x_{i+1,j} + \delta_{i,j}x_{i,j+1} + z_{i,j}, \tag{16.52}$$

a

$$z_{i,j} = \alpha_{i,j}z_{i-1,j} + \beta_{i,j}z_{i,j-1} + \phi_{i,j}, \tag{16.53}$$

kde $z_{i,j}$, $\xi_{i,j}$, $\delta_{i,j}$ jsou komponenty pomocných vektorů z, ξ, $\delta \in R^{k\times k}$ a

$$\alpha_{i,j}, \beta_{i,j}, \xi_{i,j}, \delta_{i,j}, \gamma_{i,j} \tag{16.54}$$

jsou neznámé koeficienty. Vyjádříme-li z druhé z těchto závislostí $z_{i,k}$ a dosadíme do první z nich, dostaneme

$$-\alpha_{i,j}x_{i-1,j} - \beta_{i,j}x_{i,j-1} + (1 - a_{i,j}\xi_{i-1,j} - b_{i,j}\delta_{i,j-1})x_{i,j} - \xi_{i,j}x_{i+1,j} - \delta_{i,j}x_{i,j+1} = \phi_{i,j}. \tag{16.55}$$

K tomu, aby tato rovnice byla ekvivalentní vztahu (16.51), musí platit úměra mezi neznámými koeficienty

$$\alpha_{i,j} = a_{i,j}/\gamma_{i,j},\ \beta_{i,j} = b_{i,j}/\gamma_{i,j},\ \phi_{i,j} = f_{i,j}/\gamma_{i,j}, \xi_{i,j} = c_{i,j}/\gamma_{i,j},\ \delta_{i,j} = d_{i,j}/\gamma_{i,j} \tag{16.56}$$

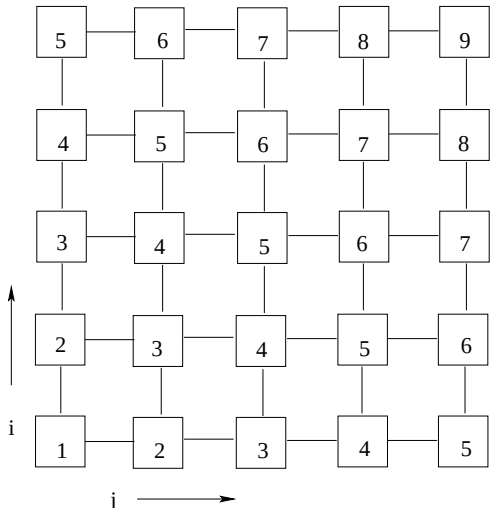
pro

$$\gamma_{i,j} = e_{i,j} - a_{i,j}\xi_{i-1,j} - b_{i,j}\delta_{i,j-1}. \tag{16.57}$$

Dosazením vztahu (16.53) do (16.55) získáme rovnost

$$z_{i,j} = \alpha_{i,j}z_{i-1,j} + \beta_{i,j}z_{i,j-1} + \alpha_{i,j}\delta_{i-1,j}x_{i-1,j+1} + \beta_{i,j}\xi_{i,j-1}x_{i+1,j-1} + \phi_{i,j}, \tag{16.58}$$

pomocí které můžeme postupně počítat hodnoty $z_{i,j}$ postupně pro $i, j = 1, \ldots, k$, známe-li příslušné hodnoty vektoru x. Závislost ve výpočtu demonstrujeme na následujícím obrázku sítě šipkami postupně od jejího levého dolního rohu. Hodnoty v bodech sítě označené stejným číslem můžeme počítat zároveň.



Známe-li všechny komponenty z, pro výpočet řešení x v opačném směru použijeme (16.53). Protože ale hodnoty x na počátku neznáme, pak odvozené vztahy

$$\begin{aligned} z_{i,j} &= \alpha_{i,j}z_{i-1,j} + \beta_{i,j}z_{i,j-1} + \alpha_{i,j}\delta_{i-1,j}x_{i-1,j+1} + \beta_{i,j}\xi_{i,j-1}x_{i+1,j-1} + \phi_{i,j};\ i,\ j = 1, \ldots, k \\ x_{i,j} &= \xi_{ij}x_{i+1,j} + \delta_{i,j}x_{i,j+1} + z_{i,j};\quad i,\ j = k, \ldots, 1 \end{aligned} \tag{16.59}$$

definují stacionární iterační metodu. Přitom pro indexy (i, j) mimo síť položíme

$$a_{i,j} = b_{i,j} = c_{i,j} = d_{i,j} = f_{i,j} = 0,\ e_{i,j} = 1. \tag{16.60}$$

a celý výpočet startujeme z nějakého počátečního přiblížení x_0. Mimo síť tak budou vektory z a x mít nulové hodnoty. Buleev v původních článcích diskutoval také konvergenci iteračního schématu a předpokládal diagonální dominanci soustavy. Později rozšířil tento metodu na komplikovanější diskretizované soustavy. Především ale už v základní formulaci [27] navrhl modifikaci, která může být obecně efektivnější, kde je konvergence metody delikátnějším problémem a o které budeme mluvit později.

Stejně jako v případě tridiagonální matice se dá tento výpočet **v rámci jedné iterace** interpretovat pomocí LU rozkladu, ale už tento rozklad není přesný, protože prvky odpovídající zaplnění jsou zanedbány. Takovému rozkladu budeme dále říkat **neúplný** a tento postup byl historicky prvním způsobem zavedení neúplného rozkladu, který pro diskretizace eliptických parciálních diferenciálních rovnic navrhl Buleev v roce 1959 [27], [28]. Konkrétně se dá ukázat, že metoda, kterou jsme tímto způsobem získali, je stacionární iterační metoda s maticí předpodmínění M ve tvaru

$$(G - L_A)G^{-1}(G - U_A) \tag{16.61}$$

pro standardní štěpení matice

$$A = D_A - L_A - U_A, \tag{16.62}$$

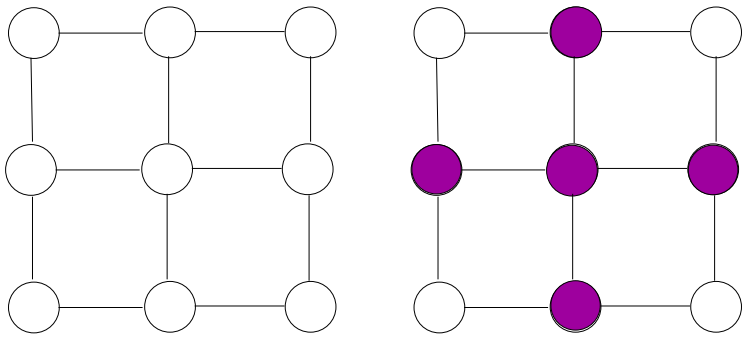
která se zapíše ve standardním tvaru

$$x^+ = (I - M^{-1}A)x + M^{-1}b, \tag{16.63}$$

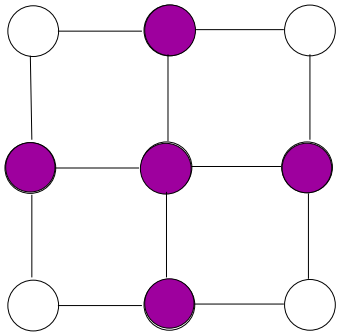
a kde matice G je diagonální matice, která obsahuje prvky $\gamma_{i,j}$.

16.5.2 Od rekurzí k hvězdám a maticovým zápisům předpodmínění

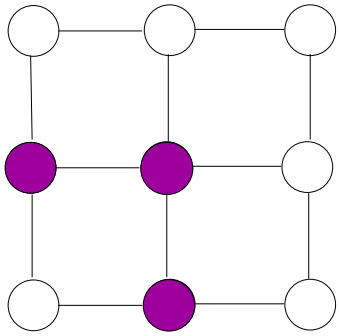
Znázorněme si graficky závislost hodnoty řešení x na pozici odpovídající nějakému vrcholu sítě daným indexy i, j pro i, $j = 1, \ldots, k$. Toto znázornění budeme nazývat **hvězdou**. Hodnoty, na kterých závisí hodnota řešení uprostřed hvězdy, jsou na našich obrázcích označeny tmavě.



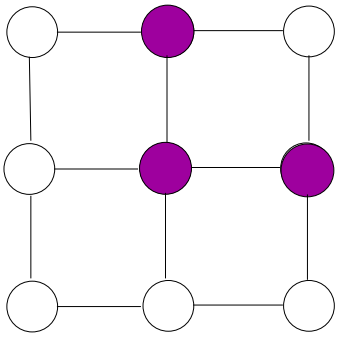
Obrázek napravo znázorňuje závislost výpočtu hodnoty hledané 2D funkce na pozici (i, j) sítě pro $i, j = 1, \ldots, k$ s diskretizačním parametrem h na hodnotách funkce v bodech $\{(i-1, j), (i, j), (i+1, j), (i, j-1), (i, j+1)\}$, které jsou podle naší konvence tmavě označeny. Následující obrázek znázorňuje postupně hvězdu pro matici A původního operátoru, prvky z její dolní trojúhelníkové části $D_A - L_A$ a prvky z její horní trojúhelníkové části $D_A - U_A$.



hvězda pro A

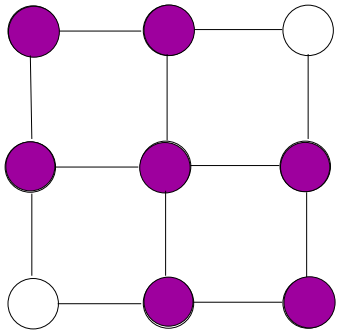


hvězda pro $D_A - L_A$



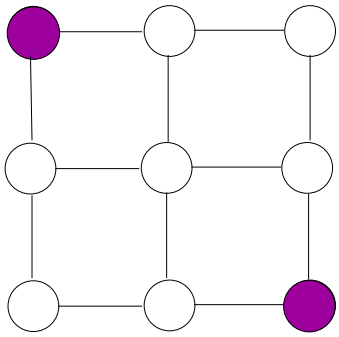
hvězda pro $D_A - U_A$

Hvězdy pro $D_A - L_A$ a $D_A - U_A$ tedy odpovídají, po řadě, dolní a horní trojúhelníkové matici. Kdyby tyto hvězdy odpovídaly **rozkladu** of A a napsali jejich součin symbolicky, dostaneme následující hvězdu.



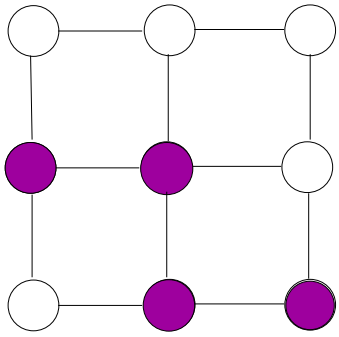
hvězda pro $(D_A - L_A)(D_A - U_A)$

Vidíme tedy, že hvězdy pro $D_A - L_A$ a $D_A - U_A$ **nevystihují** strukturu rozkladu matice A neboť rozdíl těchto hvězd je



rozdílová hvězda pro $A - (D_A - L_A)(D_A - U_A)$

Pro rozklad $A = LU$ musí být hvězdy rozšířeny tak, jak znázorňujeme pro L níže.



rozšířená hvězda pro L

Definice operací nad hvězdami má dlouhou tradici. Bylo používáno již v prvních publikacích, které hledaly neúplné rozklady a nejen pro pětibodová diskretizační schémata. Její pokračování, které je v některých aplikačních komunitách velmi zažité je také jeden ze zdrojů moderního popisu neúplných rozkladů v maticovém tvaru. Dva přístupy, na jejichž základě vznikly moderní neúplné rozklady zároveň ukazují změnu paradigmatu na přelomu padesátých a šedesátých let dvacátého století spojovanou se jménem Alstona Householdera, kde se namísto často nepřehledného popisu rozkladů po prvcích (prvkového formalismu) přešlo k elegantnímu maticovému zápisu, který pak zjednodušil další rozvoj.

16.5.3 Ke složitějším rozkladům v rámci diskretizací PDR

Uvažujme následující obecný vztah mezi maticí předpodmínění M, maticí A a chybovou maticí R ve tvaru

$$M - A = E. \tag{16.64}$$

Jak jsme viděli, důležitou motivací počátečního výzkumu algebraických předpodmínění bylo řešení soustav rovnic s maticemi z diskretizací diferenciálních rovnic. Jedním směrem k složitějším a efektivnějším rozkladům bylo sledovat asymptotickou závislost chyby v rozkladech na parametru nebo parametrech diskretizace a snažit se ji zmenšit v praktických situacích. Často i na základě teoretických úvah. Platí-li, že prvky chybové matice asymptoticky splňují pro pravidelnou síť s parametrem diskretizace h vztah

$$(E)_{ij} = O(h), \tag{16.65}$$

kde uvažujeme rovnici vzniklou z diskretizace na pravidelné síti s jedním diskretizačním parametrem h, hovoříme o neúplném **rozkladu (faktorizaci) prvního řádu**. Takovou modifikaci zavedl Buleev v [27] proto, aby zvýšil rychlost konvergence metody v případech, kdy metoda konvergovala. Tuto modifikaci získal přičtením výrazu

$$-\theta_{ij}(a_{ij}\delta_{i-1,j} + b_{ij}\xi_{i,j-1})x_{ij}, \tag{16.66}$$

k oběma stranám vztahu (**??**) s volným parametrem $\theta_{ij},\ 0 \le \theta_{ij} \le 1$ pro $i, j = 1, \ldots, k$. Ukázalo se, že v mnoha případech byl optimální výběr parametru $\theta_{ij} = 1 - O(h)$. Uvažujeme-li Předpodmínění ve tvaru přibližného rozkladu $A \approx LU$ matice A Obecně můžeme modifikovanou předpodmíněnou metodu popsat standardním způsobem, kde předpodmínění je založeno na **přibližném** rozkladu matice

$$A + B$$

kde matice B odpovídá modifikaci. V případě návrhu Buleeva je tedy

$$B_{ij} = a_{ij}\delta_{i-1,j}(u_{i-1,j+1} - \theta_{ij}u_{ij}) + b_{ij}\gamma_{i,j-1}(u_{i+1,j-1} - \theta_{ij}u_{ij}).$$

Postupně bylo v rámci předpodmínění více takových typů modifikací, které byly specifické pro soustavy vzniklé diskretizací parciálních diferenciálních rovnic.

Podle Saylora [142] nazvěme **silně implicitní** metodou každou takovou metodu předpodmínění, kde struktura řídkosti $\mathcal{S}(LU)$ přibližného LU rozkladu modifikované matice $A + B$ je shodná se strukturou řídkosti původní matice A. Jinými slovy, protože rozklad matice A obvykle generuje zaplnění, pak modifikace B musí kompenzovat toto zaplnění numerickými hodnotami ostatních prvků rozkladu. Významnou silně implicitní metodu vyjádřenou maticí modifikační maticí B navrhl v roce 1968 Stone a výslednou metodu nazval SIP (Strongly Implicit Procedure). Tato metoda pro pětibodovou diskretizaci aproximuje hodnoty $x_{i-1,j+1}$ a $x_{i+1,j-1}$ následujícím způsobem.

$$x_{i-1,j+1} \approx x_{i-1,j} + x_{i,j+1} - x_{i,j} \quad x_{i+1,j-1} \approx x_{i+1,j} + x_{i,j-1} - x_{i,j} \tag{16.67}$$

a tuto aproximaci lze ve formě výše zmíněné kompenzace přímo aplikovat do iteračního schématu (16.60). Potom se dá ukázat, že $(M - A)_{i,j} \approx O(h^2)$ a tato vlastnost řadí SIP mezi neúplné faktorizace **druhého řádu**. Formálně se dá struktura této metody pro pětibodovou diskretizaci znázornit volbou matice B ve tvaru dvou vedlejších diagonál nad rámec těch, které jsou ve struktuře matice A z pětibodového schématu.

Přirozenou otázku, zdali je možné iterační metodu symetrizovat a zároveň udržet efektivnost SIP pro diskretizační úlohy, na které ji lze aplikovat, zodpověděl negativně Saylor v [142], protože symetrizací získá tato metoda druhého řádu jiné a nevhodné numerické vlastnosti.

V následující sekci ukážeme ještě jiný postup, který vedl k třídě předpodmínění ve formě neúplných rozkladů a který byl založen, spíše než na zobecňování rekurzívních schémat, na explicitním pozorování, jak jsou nenulové prvky rozkladu reprezentovány ve schématech podobných pětibodovému síťovému schématu této sekce.

16.6 Lepší teoretické porozumění procesům

Souběžně s vývojem předpodmínění založených na jednoduchých procesech, tedy především na nějaké formě neúplného rozkladu, se vyvíjelo teoretické poznání, které se týkalo vlastností předpodmínění a předpodmíněných iteračních metod. Jedním cílem bylo získat odhad čísla podmíněnosti

$$M^{-1/2}AM^{-1/2} \tag{16.68}$$

předpodmíněné matice. V případě předpodmiňování stacionárních iteračních metod je tak možné získat dobrou představu o konvergenci metody i její rychlosti. Na druhé straně, při předpodmiňování krylovovských iterační metod takový odhad konvergenci nemusí vystihovat a ani zmenšování čísla podmíněnosti předpodmíněné matice nemusí být klíčem k dosažení lepší konvergence.

Jedny z prvních teoretických výsledků tohoto typu se týkaly předpodmiňování diagonálními maticemi.

16.6.1 Diagonální předpodmínění

Forsythe a Straus ukázali v roce 1955 [71], že Jacobiho předpodmínění, tedy volba

$$M = A_D, \tag{16.69}$$

je optimální nebo blízké optimálnímu v minimalizaci čísla podmíněnosti mezi všemi speciálními diagonálními maticemi s **vlastností A**, definovanou níže.

Definice 16.6.1 *Řekneme, že matice A má* **vlastnost A***, je-li symetrická a dá-li se symetricky permutovat na blokový tvar*

$$\begin{pmatrix} D_1 & L_A^T \\ L_A & D_2 \end{pmatrix}, \tag{16.70}$$

kde matice D_1 a D_2 jsou netriviální diagonální matice.

Věta 16.6.1 *([71]; viz také [87], Věta 4.12.8.) Má-li symetrická a pozitivně definitní matice A vlastnost A, pak platí*

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) = min_{\{D \mid D_{ij}=0 \ pro \ i \neq j\}} \kappa(D^{-1/2} A D^{-1/2}), \tag{16.71}$$

kde D_A je diagonální část matice A.

Později, v roce 1969, dokázal podobný výsledek bez předpokladu vlastnosti A, van der Sluis [156]

Věta 16.6.2 *([156]) Je-li A symetrická a pozitivně definitní, pak platí*

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) \leq p \, min_{\{D \mid D_{ij}=0 \ pro \ i \neq j\}} \kappa(D^{-1/2} A D^{-1/2}), \tag{16.72}$$

kde $D_A = diag(A)$ a kde p označuje maximum z počtu nenulových prvků řádků matice A.

16.6.2 K obecnějším základům předpodmínění a obecnějším modifikacím

Jiný pohled, který může naznačit kvalitu předpodmínění v případě problémů z diskretizací parciálních diferenciálních rovnic je příznivá asymptotická závislost čísla podmíněnosti předpodmíněné matice

$$M^{-1/2} A M^{-1/2}$$

na diskretizačním parametru h. Zatímco číslo podmíněnosti matice A u řady těchto soustav, pro které byla vylepšení navržena, má typicky (nepůjdeme zde do detailů) asymptotickou velikost

$$O(h^{-2}),$$

ukázalo se, že je možné zkonstruovat předpodmínění, kde platí

$$\kappa(M^{-1/2} A M^{-1/2}) = O(h^{-1}) \tag{16.73}$$

a kde je počet iterací stacionární iterační metody úměrný $O(h^{-1})$. Takové metody byly postupně, především po roce 1950, viz ale také například [134], navrženy v řadě publikací, přičemž se obrovský pokrok dosáhl v obecnosti diskretizovaných problémů, které tak mohly být řešeny v počtu iterací úměrném $O(h^{-1})$ [73], [148], [135], [164], [165]. Ve velmi speciálních případech byla možná ještě i rychlejší lineární konvergence [146]. Často ovšem za cenu závislosti na parametrech, které se v praxi mohly obtížně hledat. A často také modifikací schématu, kde se uvažovalo s iteracemi tvaru

$$x+ = (A+B)^{-1}(x - \omega(Ax - b)), \tag{16.74}$$

kde se parametr ω mohl měnit v každé iteraci.

Využití větší rychlosti konvergence v praxi ale znamenalo také takto řešit obecnější úlohy. A to přesto, že další odvozené výsledky se stejně týkaly diskretizací s konkrétními okrajovými podmínkami a jiné okrajové podmínky je mohly změnit. Přelomový výsledek tohoto typu v obecné formulaci předpodmínění ve tvaru $A + B$, kde je předpodmíněná stacionární iterační metoda vyjádřena ve tvaru (16.74) byl uveden v práci [57]. Autoři zde ukazují pro řešení diskretizovaného a poměrně obecného eliptického problému zmíněnou závislost čísla podmíněnosti předpodmíněné úlohy ve tvaru $O(h^{-1})$ a odvozují konvergenční výsledky i pro obecně nehladké koeficienty rovnice a pro konstantní i proměnný akcelerační parametr ω. Konkrétně bylo ukázáno, že pro A, M, N pozitivně definitní a specifickou faktorizaci prvního řádu s $A = M - N$ kde $N = O(h)$ existují pozitivní konstanty m_1 a m_2 takové, že platí

$$\frac{\langle Ax, x\rangle}{\langle Mx, x\rangle} \in \langle m_1, m_2/h\rangle, \tag{16.75}$$

což implikuje $\kappa(M^{-1/2}AM^{-1/2}) = O(h^{-1})$. Podstatnou součástí těchto postupů byla nutnost perturbovat diagonální prvky rozkladu hodnotou, která například pro problém s čistě Dirichletovými okrajovými podmínkami měla velikost

$$c_0 h^2, \tag{16.76}$$

kde c_0 je konstanta nezávislá na parametru diskretizace h. Pro neúplné faktorizace matic z diskretizací diferenciálních operátorů pak tento typ výsledku byl rozšířen na nejrůznější typy rovnic a okrajových podmínek. Další zobecnění těchto postupů je možné nalézt například v publikaci [56]. Zobecněná metoda SSOR s malou citlivostí na použitém relaxačním parametru se objevila například v práci O. Axelssona [12].

Modifikace diagonálních prvků rozkladu, která se objevila už v práci Buleeva a postupně v mnoha zde citovaných i necitovaných pracích se postupně přenesla do čistě algebraických procedur nalezení neúplné faktorizace, kde místo o diskretizované parciální diferenciální rovnici a okrajových podmínkách hovoříme spíše o matici soustavy.

16.7 Maticově vyjádřený neúplný rozklad

Základní vztah stojící v pozadí vzniku a počátečního rozvoje neúplných rozkladů lze vyjádřit schématem

násobení hvězd $\leftrightarrow$ lokální interpolace na sítích $\leftrightarrow$ eliminační proces

Uvažujme neúplný rozklad v maticovém zápisu

$$A = LU - E,$$

kde $E = (e_{ij})$ představuje chybovou matici a LU tedy značí **přesný rozklad** matice $A + E$. Odlišme tento zápis od značení v (16.74), které vzniklo v prostředí diskretizací parciálních diferenciálních rovnic a kde matice $A+B$ mohla být obecně rozkládána pouze přibližně. Algoritmus neúplného LU rozkladu pak můžeme v případě, že všechny operace lze provést, vyjádřit následovně

Algoritmus 16.7.1 Neúplný LU rozklad (ILU)
Input: *Matice $A \in R^{n\times n}$, struktura řídkosti $\mathcal{S}$.*
Output: *Faktory rozkladu $L \in R^{n\times n}$ s jednotkovou diagonálou a $U \in R^{n\times n}$, kde $A \approx LU$.*

```
for k = 1 : n do
    for i = k+1, ... n  taková, že (i,k) ∈ S
        l_ik = a_ik / a_kk
        for j = k+1, ..., n  taková, že (i,j) ∈ S  a  (k,j) ∈ S
            a_ij = a_ij - l_ik a_kj
        end do
    end do
    for j = k, ..., n do
    u_kj = a_kj
    end do
end do
```

Všimněme si, že prvky $a_{kk}, k = 1, \ldots, n$ jsou zároveň nenulové prvky matice D LDU rozkladu, který bychom získali, kdybychom navíc modifikovali jeho horní trojúhelníkový faktor vynásobením $D^{-1}U$. První aproximační vlastností neúplného rozkladu je v následujícím lemmatu. Tato vlastnost by byla snad i překvapující, kdybychom chápali neúplný rozklad jen jako mechanický postup daný Algoritmem 16.7.1.

Lemma 16.7.1 *Pro neúplný LU rozklad matice A daný strukturou řídkosti $\mathcal{S}$, kde $A = LU - E$ platí pro prvky chybové matice E následující implikace*

$$(i,j) \in \mathcal{S} \Rightarrow e_{ij} = 0. \tag{16.77}$$

Důkaz: Uvažujme pozici $(i,j) \in \mathcal{S}$ a předpokládejme, že $i < j$. Příslušný prvek neúplného LU rozkladu je podle algoritmu roven

$$l_{ij} = \left(a_{ij} - \sum_{k<j,(i,k)\in\mathcal{S}\wedge(k,j)\in\mathcal{S}} l_{ik}u_{kj} \right) / u_{jj}. \tag{16.78}$$

Znásobením i-tého řádku matice L a j-tého sloupce matice U dostaneme a_{ij} and toto násobení je dáno přesně sumou v (16.78) s přičtením dodatečného členu $l_{ij}u_{jj}$. Pro $i \geq j$ je důkaz analogický. ■

Poznámka 16.7.1 *Triviálně, obecně neplatí, že prvek na pozici (i, j) struktury řídkosti je totožný s příslušným prvkem matice $L + U - I$ úplného LU rozkladu.*

Různé volby struktury řídkosti $\mathcal{S}$ a různé algoritmy její konstrukce určují základní varianty neúplné faktorizace. Samotná struktura řídkosti může, ale i nemusí být známa a priori, tedy před provedením rozkladu. Rozklad, který budeme označovat $ILU(0)$ je dán následující volbou struktury řídkosti

$$\mathcal{S}(L + U) = \mathcal{S}(A). \tag{16.79}$$

Poznámka 16.7.2 *Samozřejmě, pro existenci neúplného LU rozkladu, tedy aby Algoritmus 16.7.1 dokončil výpočet, je nutnou podmínkou platnost vztahu*

$$\{(1, 1), \ldots, (n, n)\} \subseteq \mathcal{S}(L + U). \tag{16.80}$$

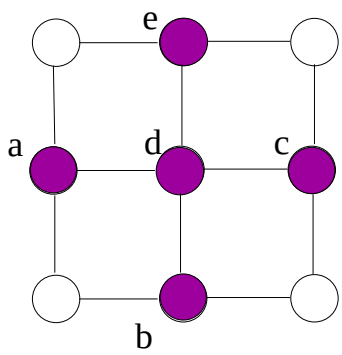
Podmínka (16.80) ale není postačující. a jedním z problémů Algoritmu 16.7.1 je, že se jej nemusí povést dopočítat a rozklad tedy neexistuje. Na rozdíl od úplného rozkladu nestačí pro jeho existenci ani předpoklad silné regularity matice. Garance **existence rozkladu** *je důležitá motivace pro volbu $\mathcal{S}$, ale v obecném případě je obtížné ji zabezpečit a priori.*

Poznámka 16.7.3 *Příkladem neúplného rozkladu bez zaplnění je dříve uvedená metoda podle Buleeva, kde rekurence popsané výše v prvkovém formalismu počítaly hodnoty, které se dají interpretovat jako prvky faktorů rozkladu, pouze na pozicích původních prvků matice. Pozdější silně implicitní metody pak používají obvykle jako strukturu řídkosti rozkladu nějakou nadmnožinu struktury řídkosti matice A.*

Neúplný rozklad ILU(0) v případě matic z pětibodového síťového schématu podle hvězdicového schématu zobrazeného níže můžeme zapsat pomocí matic ze štěpení původní matice

$$LU = (D - L_A)D^{-1}(D - U_A)$$

kde $A = D_A - L_A - U_A$ a matice D je diagonální matice, obecně různá od D_A. Soustavy s maticemi, které toto splňují, je možné efektivně předpodmínit s využitím speciálního triku, který se v maticové formě nazývá Eisenstatův trik a uvedeme jej níže.



Obecnější neúplné LU rozklady budeme standardně značit ILU s případným parametrem v závorce. Jedná-li se o neúplný Choleského rozklad symetrické a pozitivně definitní matice, budeme používat značení IC s případným parametrem. K parametrům, které se používají v tomto značení, se vrátíme později.

Poznámka 16.7.4 *V případě neúplného Choleského rozkladu budeme na rozdíl od přímých metod vždy vyžadovat kladnost diagonálních prvků, abychom mohli použít rozklad korektně jako předpodmínění, kde je pozitivní definitnost předpodmínění velmi často vyžadována. Takový požadavek je obvyklý nejenom pro předpodmiňování iteračníéch metod při řešení soustav se symetrickými a pozitivně definitními maticemi, ale i při řešení soustav se symetrickými a indefinitními maticemi.*

Jedním z mezníků ve vývoji neúplných rozkladů byl příspěvek Meijerinka a van der Vorsta [117], ve kterém tito autoři ukázali, že pro M-matice a zvolenou strukturu řídkosti $\mathcal{S}(\mathcal{L}+\mathcal{U})$, která obsahuje diagonální pozice (splňující tedy podmínku (16.80)), neúplný rozklad existuje. Jejich důkaz je založen na dvou tvrzeních, která zde uvedeme bez důkazu. Prvním z nich je generičnost vlastnosti M-matice v průběhu rozkladu v následujícím smyslu

Věta 16.7.1 *[69] Je-li matice A M-matice, pak je M-matice i Schurův doplněk vzhledem k jejímu prvku a_{11}.*

Druhé z těchto tvrzení ukazuje, že vlastnost M-matice se zachovává při určité množině modifikací jejích prvků.

Věta 16.7.2 *Je-li M-matice A modifikovaná na matici B tak, že platí vztahy*

$$\begin{aligned} a_{ij} \le b_{ij} \le 0 \qquad & i \ne j, \quad i,j = 1,\dots,n \\ 0 < a_{ii} \le b_{ii} \qquad & i = 1,\dots,n, \end{aligned} \tag{16.81}$$

pak B bude také M-maticí.

Meijerink a van der Vorst [117] pak ukázali, že tvorba Schurových doplňků a případné vynechání prvků na pozicích, které nepatří do zvolené struktury řídkosti $\mathcal{S}(\mathcal{L}+\mathcal{U})$, vede právě k modifikaci podle (16.82) a tak se v postupně tvořených Schurových doplňcích zachovává vlastnost M-matice. V konečném důsledku to také znamená, že neúplný LU rozklad s danou obecnou strukturou řídkosti, kde je podmínkou jen zahrnutí diagonálních pozic podle (16.80), existuje. Mimo jiné tak existuje i neúplný Choleského rozklad symetrické a pozitivně definitní M-matice. Meijerink a van der Vorst také ukázali, že takový rozklad definuje **regulární štěpení** matice A, což implikuje konvergenci stacionární iterační metody určenou tímto štěpením. Konkrétně, ohromný vliv na další rozvoj předpodmiňování mělo použití neúplného Choleského rozkladu k předpodmiňování metody sdružených gradientů, jehož existence tak byla v řadě praktických případů zaručena. Později byla existence neúplného rozkladu ukázána i pro regulární H-matice [160], [114].

Na druhé straně, ačkoli teoretické záruky na neúplný rozklad v případě M-matic a H-matic vypadají velmi lákavě a výsledný neúplný rozklad je v určitém smyslu ještě stabilnější než rozklad úplný [117], je nutné si uvědomit, že tyto modely založené na speciálních maticích jsou velmi hrubé a chybová matice měřená v nějaké normě může být obrovská. Proto je nutné stát i o sofistikovanější přístupy byť ne s takovými teoretickými zárukami na existenci a další potřebné vlastnosti neúplného rozkladu.

16.7.1 Volba struktury řídkosti a základní teoretické možnosti neúplných rozkladů

Aniž bychom podrobněji popisovali návrh struktury řídkosti $\mathcal{S}$, rozlišme dvě základní možnosti její volby. Obecně můžeme tuto volbu popsat jako **odvrhování** nenulových prvků faktorů rozkladu, čili určení těch prvků rozkladu, které nepatří do $\mathcal{S}(L+U)$. Termínem odvrhování tedy popisujeme situaci, kdy spočtené nenulové zaplnění na nějaké pozici faktoru/faktorů nahrazujeme nulou a pozici nepočítáme do struktury řídkosti LU rozkladu. Můžeme tedy uvažovat následující dvě možnosti.

- Odvrhování prvků **podle pozice** v $\{1,\dots,n\}\times\{1,\dots,n\}$. Struktura řídkosti $\mathcal{S}(L+U)$ je v tomto případě často určena **a priori** vymezením konkrétních pozic pro odvrhování.
- Odvrhování **podle významu** prvků ve faktorech L, U pro rozklad. Význam prvků je často hodnocen na základě jejich velikosti (absolutní hodnoty), ale existují i sofistikovanější přístupy. V tomto případě je $\mathcal{S}(L+U)$ je často určováno **dynamicky** v průběhu algoritmu.

V praxi existuje více různých přístupů kombinujících tyto varianty. Poznamenejme, že základní neúplný rozklad $ILU(0)$ nemusí zlepšit ve funkci předpodmínění zlepšit konvergenci iterační metody. Rychlost konvergence se často vztahuje k číslu podmíněnosti matice $M^{-1/2}AM^{-1/2}$, což je sice velmi používaná, ale zároveň hrubá pomůcka, která nemusí konvergenci postihovat. Pro diskuse na toto téma odkazujeme k publikaci [104]. Ale ILU(0) nemusí zmenšovat ani číslo podmíněnosti a často se hledají jeho modifikace, které číslo podmíněnosti ve speciálních případech zlepšují a zároveň se v praxi vykazují rychlejší konvergenci.

16.7.2 Modifikované rozklady neúplné MIC, MILU a jejich zobecnění

Standardním **modifikovaným neúplným rozkladem** je nazýván neúplný rozklad, který k předepsané struktuře pro $L+U$ splňuje navíc následující **kritérium řádkového součtu** pro prvky **součinu** faktorů $M = LU$.

$$\sum_j m_{ij} = \sum_j a_{ij}, \quad i = 1,\dots,n.$$

Termín "modifikovaný" budeme tedy chápat jako technický termín vyjadřující tento konkrétní postup. Idea modifikace postupně vznikla v pracích [57], ale navazuje již na dříve zmíněné postupy Buleeva [27] či Vargy [159].

Poznámka 16.7.5 *Stejně jako u výše uvedeného neúplného rozkladu platí $e_{ij} = 0$ pro mimodiagonální prvky chybové matice $E = LU - A$. Pro diagonální prvky E to neplatí.*

Algoritmus modifikovaného rozkladu lze zapsat ve tvaru. Snadno nahlédneme, že kroky 6 a 7 zabezpečují, že bude splněna podmínka řádového součtu a zároveň budou ostatní mimodiagonální prvky spočítány (v přesné aritmetice) bez chyby.

Algoritmus 16.7.2 Modifikovaný neúplný LU rozklad (MILU)
Input: *Matice* $A \in R^{n\times n}$, *struktura řídkosti* $\mathcal{S}$.
Output: *Faktory rozkladu* $L \in R^{n\times n}$ *s jednotkovou diagonálou a* $U \in R^{n\times n}$, *kde* $A \approx LU$.

```
for k = 1 : n do
    for i = k+1, ... n taková, že (i,k) ∈ S
        l_ik = a_ik / a_kk
        for j = k+1, ..., n taková, že (k,j) ∈ S
            if (i,j) ∈ S a_ij = a_ij − l_ik a_kj
            if (i,j) ∉ S a_ii = a_ii − l_ik a_kj
        end do
    end do
    for j = k, ..., n do
        u_kj = a_kj
    end do
end do
```

Modifikace zachováním řádkového součtu prvků ve faktorech neúplné faktorizace je příkladem, kdy se v řešení **modelových úloh** často sníží počet iterací předpodmíněné iterační metody. Souvisejícím faktem může být, že spolu s dalším technickým předpokladem, kterým je vnesení určité nepřesnosti do řádkového součtu, je možné v modelových úlohách asymptoticky snížit velikost čísla podmíněnosti předpodmíněné matice $\kappa(M^{-1/2}AM^{-1/2})$. To teď postupně probereme. Analogicky jako v neúplném nemodifikovaném rozkladu výše, budeme rozklad daný Algoritmem 16.7.2 s volbou

$$\mathcal{S}(L+U) = \mathcal{S}(A) \tag{16.82}$$

označovat $MILU(0)$. V případě, kdy použijeme v tomto algoritmu Choleského rozklad jako modifikaci pro symetrické a pozitivně definitní matice, budeme hovořit o MIC(0).

Uvažujme nyní rozklad symetrické a pozitivně definitní matice A z modelového problému z pětibodového hvězdicového schématu. Rozepišme vztah pro chybovou matici jako

$$M = LD^{-1}L^T = A + E = A + R + \bar{D},$$

kde matice R má v i-tém řádku obecně nenulové prvky na třech pozicích, pro které platí

$$r_{i,i} = -r_{i,i-m+1} - r_{i,i+m-1}$$

a prvky matice $\bar{D}$ reprezentují diagonální modifikaci a pro jejichž velikost platí

$$\bar{D} = \xi h^2 D_A \tag{16.83}$$

pro parametr diskretizace h, konstantu ξ nezávislou na h a D_A představující diagonální část matice A s prvky

$$\alpha_i, \quad i = 1, \dots n. \tag{16.84}$$

V následujícím budeme používat následující značení prvků na diagonálách symetrické a pozitivně definitní matice z pětibodové diskretizace podle následujícího obrázku.

$$A = \begin{pmatrix}
\alpha_1 & -\beta_1 & & -\gamma_1 & & & & & & & \\
-\beta_1 & \ddots & \ddots & & \ddots & & & & & & \\
 & \ddots & \ddots & \ddots & & \ddots & & & & & \\
-\gamma_1 & & \ddots & \ddots & \ddots & & \ddots & & & & \\
 & \ddots & & \ddots & \ddots & \ddots & & \ddots & & & \\
 & & -\gamma_{i-m} & & -\beta_{i-1} & \alpha_i & -\beta_i & & -\gamma_i & & \\
 & & & \ddots & & \ddots & \ddots & \ddots & & \ddots & \\
 & & & & \ddots & & \ddots & \ddots & \ddots & & -\gamma_{n-m} \\
 & & & & & \ddots & & \ddots & \ddots & \ddots & \\
 & & & & & & \ddots & & \ddots & \ddots & -\beta_{n-1} \\
 & & & & & & & -\gamma_{n-m} & & -\beta_{n-1} & \alpha_n
\end{pmatrix} \tag{16.85}$$

$$L = \begin{pmatrix}
a_1 & & & & & & \\
b_1 & \ddots & & & & & \\
 & \ddots & \ddots & & & & \\
c_1 & & \ddots & \ddots & & & \\
 & \ddots & & \ddots & \ddots & & \\
 & & c_{i-m} & & b_{i-1} & a_i & \\
 & & & \ddots & & \ddots & \ddots \\
 & & & & c_{n-m} & & b_{n-1} \quad a_n
\end{pmatrix} \qquad L^T = \begin{pmatrix}
a_1 & b_1 & & c_1 & & & \\
 & \ddots & \ddots & & \ddots & & \\
 & & a_i & b_i & & c_i & \\
 & & & \ddots & \ddots & & \ddots \\
 & & & & \ddots & \ddots & & c_{n-m} \\
 & & & & & \ddots & \ddots & \\
 & & & & & & \ddots & b_n \\
 & & & & & & & a_n
\end{pmatrix}$$

Diagonální kompenzaci dodržení nepřesných řádkových součtů tedy můžeme zapsat

$$r_{i,i} = -r_{i,i-m+1} - r_{i,i+m-1}, \tag{16.86}$$

kde hodnoty chybové matice s indexy mimo rozmezí $1, \ldots, n$ pokládáme za nulové. Vztahy mezi nenulovými hodnotami prvků matic A, L a L^T můžeme zapsat ve tvaru

$$\begin{aligned}
a_i^2 &= \alpha_i(1+\delta) - r_i - r_{i-m+1} - b_{i-1}^2 - c_{i-m}^2 \\
b_i &= -\beta_i / a_i \\
c_i &= -\gamma_i / a_i \\
r_i &= b_{i-1} c_{i-1} \\
\delta &= \xi h^2, \xi > 0
\end{aligned} \tag{16.87}$$

Zmiňme zde také, že neúplný rozklad MIC(0) lze zapsat ve tvaru

$$(D - L_A) D^{-1} (D - U_A). \tag{16.88}$$

V tomto vyjádření jsou prvky diagonální matice D rovny

$$a_i^2, \quad i = 1, \ldots n \tag{16.89}$$

a tak s nimi budeme také dále pracovat.

Následující obrázek ukazuje strukturu neúplného rozkladu $MIC(0)$ pro modelový Poissonův problém (kde je matice symetrická a pozitivně definitní). Hvězdičky odpovídají zaplnění, které je kompenzováno na diagonále dodržováním nepřesných řádkových součtů, kde nepřesnost je dána prvky matice $\bar{D}$.

$$A = \begin{pmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & * & -1 & & & & \\ & -1 & 4 & & * & -1 & & & \\ -1 & * & & 4 & -1 & & -1 & & \\ & -1 & * & -1 & 4 & -1 & * & -1 & \\ & & -1 & & -1 & 4 & & * & -1 \\ & & & -1 & * & & 4 & -1 & \\ & & & & -1 & * & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{pmatrix}$$

Pro Poissonův modelový problém v tomto označení máme $m = 3$ a platí

$$\begin{aligned} \alpha_i &= 4, \quad i = 1, \ldots, n \\ \beta_i &= 1, \quad i = 1, \ldots, n-1; \quad i \neq m, 2m, \ldots, (n/m-1)m \\ \beta_i &= 0, \quad i = m, 2m, \ldots, (n/m-1)m \\ \gamma_i &= 1, \quad i = 1, \ldots, n-m; \end{aligned}$$

V následujícím textu ukážeme, že v případě diagonální modifikace a tento modelový problém lze očekávat asymptotické snížení velikosti čísla podmíněnosti předpodmíněné matice. Ačkoli v tomto případě a v řadě jiných případů bylo ukázáno, že diagonální kompenzace není pro odvozené teoretické výsledky potřeba, budeme s ní zde počítat. I proto, že budeme poté s diagonální kompenzací modifikovaných rozkladů počítat v obecném případě jako s praktickým nástrojem algoritmů.

Lemma 16.7.2 *Pro předpodmínění MIC(0) aplikované na modelový problém s maticí (16.85) existuje konstanta $\xi_1 > 0$ nezávislá na diskretizačním parametru h taková, že*

$$r_i \leq 1/(2(1 + \xi_1 h).$$

Důkaz: Uvažujme výše uvedené vztahy mezi hodnotami modelového problému, kde

$$\alpha_i = 4, \beta_i \leq 1, \gamma_i \leq 1. \tag{16.90}$$

Platí-li $\xi = 0$ snadno dokážeme vztah

$$a_i^2 \geq 2, \quad , i = 1, \ldots, n \tag{16.91}$$

matematickou indukcí neboť pro indukční předpoklad

$$a_i^2 \geq 2, \; i = 1, \ldots, p-1 \tag{16.92}$$

dostaneme v indukčním kroku

$$a_p^2 = 4 - b_{p-1}(b_{p-1} + c_{p-1}) - c_{p-m}(c_{p-m} + b_{p-m}) \geq 4 - 2/a_{p-1}^2 - 2/a_{p-m}^2 \geq 4 - 1 - 1 = 2. \tag{16.93}$$

Uvažujme nyní $\xi > 0$ a ukažme, že

$$a_i^2 \geq 2(1 + \xi_1 h), \quad , i = 1, \ldots, n \tag{16.94}$$

Analogicky, z indukčního předpokladu pro $i = 1, \ldots, p-1$ můžeme psát

$$\begin{aligned}(1 + \xi_1 h)a_p^2 &\geq (1 + \xi_1 h)\left(4(1 + \xi h^2) - \frac{1}{2(1 + \xi_1 h)} - \frac{1}{2(1 + \xi_1 h)} - \frac{1}{2(1 + \xi_1 h)} - \frac{1}{2(1 + \xi_1 h)}\right) \\ &= 4(1 + \xi h^2 + 4\xi_1 h + 4\xi\xi_1 h^3) - 2 = \\ &= 2 + 4\xi h^2 + 4\xi_1 h + 4\xi\xi_1 h^3.\end{aligned}$$

Položením $\xi_1 = \sqrt{2c}$ dostaneme

$$(1 + \xi_1 h)a_p^2 \geq 2 + 4\xi_1 h + 2\xi_1^2 h^2 + 2\xi_1^3 h^3 \geq 2(1 + \xi_1 h)(1 + \xi_1 h). \tag{16.95}$$

Protože ale

$$r_i = b_{i-1}c_{i-1}, \tag{16.96}$$

pak také

$$r_i = 1/a_{i-1}^2 \leq 1/(2(1 + \xi_1 h)) \tag{16.97}$$

a konstanta ξ_1 taková, že platí tvrzení lemmatu tedy existuje. ■

Následující lemma zjednodušuje výraz pro hodnotu kvadratické formy s chybovou maticí R.

Lemma 16.7.3 *Pro $x \in \mathbf{R}^{n\times n}$ platí $(Rx, x) = -\sum_i r_{i,i+m-1}(x_{i+m-1} - x_i)^2$*

Důkaz: Podle definice matice R a s využitím toho, že matice R má na každém řádku pouze tři nenulové prvky dostáváme

$$(Rx, x) = -\sum_i r_{i,i}x_i^2 + 2\sum_i r_{i,i+m-1}x_{i+m-1}x_i \tag{16.98}$$

Přitom jsme využili faktu symetrie, neboť dvojnásobek druhé sumy je dán součtem

$$\sum_i r_{i,i+m-1}x_{i+m-1}x_i + \sum_i r_{i+m-1,i}x_i x_{i+m-1}. \tag{16.99}$$

Pravidlo nulového součtu modifikací na řádku implikuje přepis (16.98) na

$$(Rx, x) = -\sum_i (r_{i,i-m+1} + r_{i,i+m-1})x_i^2 + 2\sum_i r_{i,i+m-1}x_{i+m-1}x_i \tag{16.100}$$

S využitím obecného vztahu

$$2x_{i+m-1}x_i = x_{i+m-1}^2 + x_i^2 - (x_{i+m-1} - x_i)^2$$

přepíšeme druhý člen výrazu (16.100) a získáme vztah

$$\begin{aligned}(Rx,x) &= -\sum_i (r_{i,i+m-1}+r_{i,i-m+1})x_i^2 + \\ &+ \sum_i (r_{i,i+m-1}x_i^2 + r_{i,i+m-1}x_{i+m-1}^2) - \sum_i r_{i,i+m-1}(x_{i+m-1}-x_i)^2.\end{aligned}$$

První členy prvních dvou sum se vzájemně vyruší. Totéž ale platí i pro druhé členy prvních dvou sum, jak názorně vidíme, když změníme pořadí u jedné řady sčítanců a napíšeme

$$\sum_{i=m}^{n} r_{i,i-m+1}x_i^2 = \sum_{i=1}^{n-m+1} r_{i,i-m+1}x_{i+m-1}^2.$$

Výsledný vztah pro (Rx,x) lemma dokazuje. ■

Lemma 16.7.4 $-(Rx,x) \le 1/(1+\xi_1 h)(Ax,x)$

Důkaz: Dosazením odhadu pro r_i z Lemmatu 16.7.2 do vztahu z Lemmatu lm:mic02 získáme

$$-(Rx,x) = \sum_i r_{i,i+m-1}(x_{i+m-1}-x_i)^2 \le \sum_i 1/(2(1+\xi_1 h))(x_{i+m-1}-x_i)^2. \quad (16.101)$$

Použitím následující nerovnosti platné pro reálná čísla a, b, e

$$(a-b)^2 \le 2(a-e)^2 + 2(e-b)^2 \quad (16.102)$$

dostaneme

$$-(Rx,x) \le \sum_i 1/(1+\xi_1 h)[(x_{i+m-1}-x_{i-1})^2+(x_{i-1}-x_i)^2] \le 1/(1+\xi_1 h)(Ax,x), \quad (16.103)$$

kde jsme využili platnosti vztahu

$$(Ax,x) \ge \sum_i \left((x_i-x_{i+1})^2 + (x_i-x_{i+m})^2\right) \quad (16.104)$$

Nerovnost v tomto posledním vztahu a nikoli rovnost je dána tím, že jedna z vedlejších diagonál v matici našeho modelového problému z diskretizace na čtvercové síti obsahuje také nulové prvky. ■

Výsledný asymptotický vztah pro číslo podmíněnosti předpodmíněné matice je důsledkem předcházejících odvození

Důsledek 16.7.1 *Pro číslo podmíněnosti daného modelového problému platí následující asymptotický vztah* $\kappa(M^{-1/2}AM^{-1/2}) = O(h^{-1})$

Důkaz: Upravujme formálně podíl kvadratických forem pro $x \in \mathbf{R}^{n\times n}$

$$\begin{aligned}(Ax,x)/(Mx,x) &= \frac{(Ax,x)}{(Ax,x)+(Rx,x)+(\bar{D}x,x)} \leq \frac{1}{1+(Rx,x)/(Ax,x)+(\bar{D}x,x)/(Ax,x)} \\ &\leq \frac{1}{1+(Rx,x)/(Ax,x)} \leq \frac{1}{1-1/(1+\xi_1 h)} = 1+\frac{1}{\xi_1 h}.\end{aligned}$$

Neboť se nejmenší vlastní číslo matice A modelového problému se dá zapsat jako $\xi_0 h^2$ pro konstantu ξ_0 nezávislou na h, pak dostaneme

$$\begin{aligned}(Ax,x)/(Mx,x) &\geq \frac{1/(Ax,x)}{1+(\bar{D}x,x)/(Ax,x)} = \frac{1}{1+\xi h^2(x,x)/(Ax,x)} \\ &\geq \frac{1}{1+c/\xi_0}\end{aligned}$$

z čehož plyne výsledek. V této poslední úpravě jsme využili faktu, že $(Rx,x) \leq 0$ jak je vidět pro náš modelový problém například z Lemmatu 16.7.3 a kladnosti prvků R z vyjádření (16.88). ■

Analýza asymptotického čísla podmíněnosti může být rozšířena na obecnější modifikované rozklady i obecnější problémy, které zahrnují například smíšené okrajové podmínky úlohy [85], [86]. V mnoha modelových případech není nutná ani diagonální perturbace zahrnutá v matici $\bar{D}$, viz například souhrn přístupů v [163].

Problém existence je subtilnější u modifikovaných rozkladů než u obecných neúplných rozkladů. Obecně MILU nemusí existovat ani pro M-matice. U MIC varianty, která předpokládá matici symetrickou a pozitině definitní, je rozhodující i další vlastnost, a vytvoření rozkladu, který má na diagonále kladné prvky. V následující definici zavedeme relevantní pojem.

Definice 16.7.1 *Řekneme, že neúplný LU rozklad je stabilní právě tehdy, jsou-li diagonální prvky faktoru U kladné.*

Následující věta dává postačující podmínku stability varianty modifikovaného neúplného rozkladu.

Věta 16.7.3 *Je-li symetrická a pozitivně definitní matice A ostře diagonálně dominantní, pak její modifikovaný rozklad pro danou strukturu řídkosti rozkladu existuje a je stabilní s diagonální perturbací i bez ní.*

Poznamenejme, že se často předpokládá pouze slabá diagonální dominance, ale kladná diagonální perturbace. Důkaz této věty je možné najít v [86]. Dále je známo, že metoda MILU(0) pro symetrickou a pozitivní matici je ekvivalentní výše citovanému zobecněnému SSOR předpodmínění [12] s optimálním parametrem ω.

Různé výsledky bez perturbací byly dokázány později pomocí kombinatorických technik. Zobecnění modifikací je možné požadavkem na obecnou diagonální kompenzaci rozkladu s parametrem $c > 0$, pro který platí $Ac > 0$ (zobecněné kritérium řádkového součtu). Modifikovaný rozklad s kompenzací pak splňuje vztah

$$\sum_j m_{ij} = \sum_j c_j a_{ij},\ r_{ij} = 0 \ \text{ pro } \ (i,j) \in \mathcal{S} \quad \forall i$$

Obecně je vidět, že modifikace může v řadě problémů vzniklých z diskretizací parciálních diferenciálních rovnic může přinést větší efektivnost, ale za cenu slabších teoretických výsledků i pro modelové problémy. Jazýčkem na vahách volby metody může pak být perturbace diagonálních prvků. Axelson a Lindskog [15], [16] toho využili k zavedení relaxovaného rozkladu, který modifikuje diagonální prvky pouze přibližně a odečítá tak od diagonálního prvku a_{ii} hodnotu

$$\omega \sum_i r_{ij},$$

kde parametr ω je téměř roven jedné. Fourierovou analýzou modelového problému se dá ukázat, že modifikace odpovídá matici $\bar{D}$, kterou jsme použili v důkazu asymptotické rychlosti konvergence perturbovaného modifikovaného rozkladu. Tento postup motivuje statickou nebo i dynamickou volbu ω modifikace při řešení obecných problémů [66], [67], [157], [125]. V souvislosti s konkrétními obecnějšími diskretizovanými problémy se zobecnil i pojem stability tak, aby zahrnul to, že diagonální prvky rozkladu například nedegenerují se zjemňováním sítě [26], [25].

16.7.3 Parametrizace neúplných rozkladů a odvrhování prvků faktorů po úrovních: ILU(k)

Neúplné rozklady mohou být kromě algoritmů popsány různým způsobem jako maticové transformace. Uvažujme jeden krok rozkladu matice $A \in R^{n\times n}$, kde $a \in R$, $r^T \in R^n, c \in R^n, S \in R^{n-1\times n-1}$ a zapišme jej v podmaticové formulaci, kde S označuje příslušný Schurův doplněk.

$$A = \begin{pmatrix} a & r^T \\ c & S \end{pmatrix} = \begin{pmatrix} I & \\ c\,a^{-1} & I \end{pmatrix} \begin{pmatrix} a & r^T \\ & S - c\,a^{-1}\,r^T \end{pmatrix}$$

Krok přibližného rozkladu pak můžeme znázornit

$$A = \begin{pmatrix} a & r^T \\ c & S \end{pmatrix} = \begin{pmatrix} I & \\ \hat{c}\,a^{-1} & I \end{pmatrix} \begin{pmatrix} a & \hat{r}^T \\ & S - \hat{c}\,a^{-1}\,\hat{r}^T \end{pmatrix}$$

Zaveďme nyní následující ohodnocení všech prvků matice skalární hodnotou, kterou budeme nazývat **úroveň** prvku funkcí **level**.

- Inicializace úrovní před rozkladem:

$$level_{ij}^{(0)} = \begin{cases} 0 & \text{if } a_{ij} \neq 0 \text{ or } i = j \\ +\infty & \text{otherwise} \end{cases}$$

- Definice úrovně prvku v průběhu LU rozkladu: Pro prvek matice na pozici (i, j) v kroku k podmaticového algoritmu definujeme

$$level_{ij}^{(k+1)} = min(level_{ij}^{(k)}, level_{ik}^{(k)} + level_{kj}^{(k)} + 1)$$

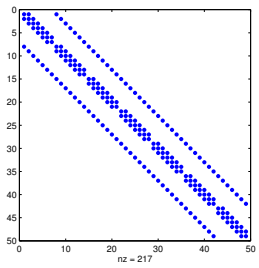
Takto je tedy úroveň definována pro všechny pozice v $\{1, \ldots, m\} \times \{1, \ldots, n\}$.

Neúplný rozklad založený na konceptu úrovně budeme značit $ILU(p)$ v případě neúplného LU rozkladu a $IC(p)$ v případě neúplného Choleského rozkladu. $p \geq 0$ zde představuje zvolený celočíselný parametr. Principem postupu je aktualizace prvků pouze na těch pozicích z $\{1, \ldots, m\} \times \{1, \ldots, n\}$, které mají úroveň nejvýše rovnu danému celočíselnému parametru $p \geq 0$. Tímto způsobem omezujeme je omezeno zaplnění ve faktorech rozkladu. Myšlenka, která motivuje tuto dynamickou definici struktury řídkosti $\mathcal{S}$ je, že prvky s vyšší úrovní, které odpovídají delším **cestám zaplnění**, přispívají k výslednému rozkladu méně. To je skutečný fakt, který nastává v některých praktických modelových situacích, kde zaplnění s vysokou úrovní danou funkcí level, mají malou velikost (absolutní hodnotu). Algoritmus neúplného LU rozkladu založeného na konceptu úrovní pak můžeme lze vyjádřit následovně

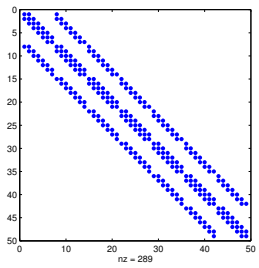
Algoritmus 16.7.3 Neúplný LU rozklad založený na úrovních (ILU(p))
Input: *Matice* $A \in R^{n \times n}$, *inicializované úrovně level,* $p \geq 0$.
Output: *Faktory rozkladu* $L \in R^{n \times n}$ *s jednotkovou diagonálou a* $U \in R^{n \times n}$, *kde* $A \approx LU$.

1. **for** $i = 1 : n$ **do**
2. **for** $k = 1, \ldots i-1$ *taková, že* $level(i,k) \leq p$
3. $l_{ik} = a_{ik}/a_{kk}$
4. **for** $j = k+1, \ldots, n$
5. $a_{ij} = a_{ij} - l_{ik}a_{kj}$
6. $level(i,j) = \min level(i,j), level(i,k) + level(k,j) + 1$
7. **end do**
8. **end do**
9. **for** $j = 1, \ldots n$
10. **if** $level(i,j) > p$ *polož* $a_{ij} = 0$
11. **end do**
12. **for** $j = k, \ldots, n$ **do**
13. $u_{kj} = a_{kj}$
14. **end do**
15. **end do**

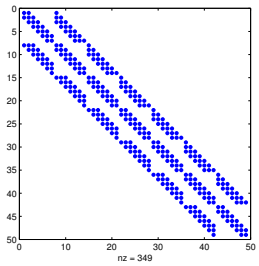
Algoritmus pracuje po řádcích a to je také důvod, proč nulujeme prvky s úrovní přesahující p teprve po vytvoření konkrétního řádku. V průběhu cyklu pro jeho tvorbu totiž ještě nemusíme vědět, jestli nějaká přípustná cesta zaplnění existuje nebo ne. Následující sada obrázků ukazuje strukturu zaplnění pro $ILU(p)$ s postupně rostoucím parametrem p a naznačuje problém, který může v praxi nastat u tohoto typu volby struktury řídkosti neúplných faktorů. První čtyři obrázky se týkají modelového problému a zbylé dva obrázky znázorňují strukturu neúplného rozkladu založeného na úrovních pro matici s komplikovanější strukturou řídkosti.



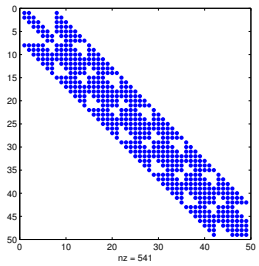
ILU(0)



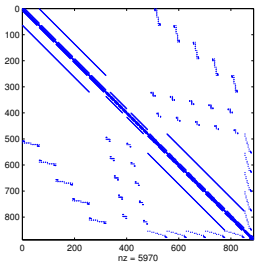
ILU(1)



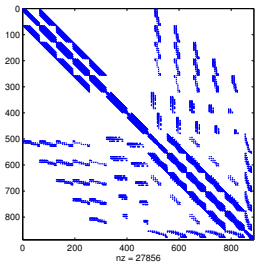
ILU(2)



ILU(4)



ILU(0)



ILU(3)

Konkrétně, u tohoto typu volby struktury řídkosti může dojít k tomu, že velmi rychle vzrůstá velikost zaplnění pro rostoucí p.

16.7.4 Odvrhování prvků menšího významu

Hlavním problémem takto koncipovaného způsobu odvrhování je odhad, které prvky mají pro rozklad malý význam. Jak bylo naznačeno výše, obvykle se za takové prvky považují ty, které jsou jako prvky faktoru nebo faktorů v absolutní hodnotě menší, než nějaký pevně daný parametr $\tau > 0$ (**absolutní odvrhování**), nebo, které jsou v absolutní hodnotě menší vzhledem k jiným prvkům, například maximální absolutní hodnotě prvků ze Schurova doplňku (**relativní odvrhování**).

Ačkoli se relativní odvrhování zdá vhodnější, nemusí tomu tak v praxi vždy být. Procedura, která takové kritérium využívá, nemusí detekovat velký **růstový faktor roz-**

kladu a spočítané faktory mohou být nestabilní. Naproti tomu algoritmus rozkladu s absolutním odvrhováním pro škálovanou matici, takovou situaci snadno pozná podle zmenšování počtu odvrhovaných prvků a může na tento fakt adaptivně reagovat v průběhu rozkladu. Existuje dokonce celá řada adaptivních technik, které parametr τ mění v průběhu rozkladu z podobných důvodů, ale třeba i kvůli udržení předem vymezené velikosti paměti pro neúplné faktory.

16.7.5 Existence a stabilita neúplného rozkladu

Je-li matice A symetrická a pozitivně definitní, její **úplný Choleského rozklad** je **zpětně stabilní**. Počítáme-li ale nějaký její **neúplný** rozklad, nemusí ani silná regularita matice A stačit pro jeho existenci či stabilita tohoto rozkladu může být výrazně porušena, jak jsme již zmínili. Neexistence neúplného Choleského rozkladu znamená, že v jeho průběhu je nějaký spočítaný diagonální prvek nekladný. Pro neúplný LU rozklad (ILU) to znamená, že je na diagonála nula a záporný diagonální prvek nevadí. V obou případech ale i spočítaný diagonální prvek blízký nule znamená potenciální **nestabilitu** rozkladu. Následující příklad ukazuje neúplný Choleského rozklad (ve tvaru LDL^T) symetrické a pozitivně definitní matice. Poznamenejme, že matice z tohoto příkladu není M-matice, protože pak by rozklad musel existovat, jak jsme uvedli výše.

$$A = \begin{pmatrix} 3 & -2 & & 2 \\ -2 & 3 & -2 & \\ & -2 & 3 & -2 \\ 2 & & -2 & 3 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & & & \\ -2/3 & 1 & & \\ & -1.2 & 1 & \\ 2/3 & 0.8 & -2/3 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 3 & & & \\ & 5/3 & & \\ & & 0.6 & \\ & & & 1/3 \end{pmatrix}$$

Po třech krocích rozkladu bez zaplnění získáme následující neúplné faktory.

$$L = \begin{pmatrix} 1 & & & \\ -2/3 & 1 & & \\ & -1.2 & 1 & \\ 2/3 & & -10/3 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 3 & & & \\ & 5/3 & & \\ & & 0.6 & \\ & & & -5 \end{pmatrix}$$

Neúplný Choleského rozklad tedy neexistuje neboť došlo ke **zlomu** tím, že na diagonále vzniknul záporný prvek. Heuristická minimalizace pravděpodobnosti, že nastane taková situace může být založena například na tom, že se před rozkladem hledá nějaký kompromis mezi permutací minimalizující zaplnění a permutací, která přesune na diagonálu matice prvky s co největší absolutní hodnotou. Takovou permutaci lze nalézt například a priori aplikací kombinatorických technik založených na variantách hledání párování v grafovém modelu matice. Druhou možností je hledat takovou permutaci dynamicky a použít například diagonální pivotaci. To znamená, že v k-tém kroku rozkladu vybíráme hlavní prvek ze zbývajících $n-k-1$ diagonálních prvků. Ve výběru opět můžeme kombinovat

kritéria, aby tento hlavní prvek byl dostatečně velký v absolutní hodnotě a zároveň nedocházelo k přílišnému zaplnění matice. Ani v tomto případě nemusí ale neúplný rozklad (symetricky permutované) matice existovat. Neúplné faktory ale mohou trpět i dalšími nešvary. Faktory mohou být nestabilní z důvodu existence více menších hlavních prvků (pivotů) a přitom tento problém nemusí být algoritmem detekován. Dalším problémem může být nestabilita substitučních kroků dlouhými rekurencemi, které vytvářely neúplné faktory [66], [67], [33]. V praxi tak vznikla celá řada specifických modifikací algoritmů neúplných rozkladů (nezaměňujme takové modifikace s technickým termínem modifikace zavedeným výše). Několik takových možností zmiňujeme v následujícím výčtu:

- **Změna hodnoty diagonálního prvku** v případě jeho malé absolutní hodnoty (neúplný LU rozklad) nebo malé hodnoty (neúplný Choleského rozklad) [96]. Tato změna někdy může pomoci, ale často je neúčinná, protože když jsou problémy algoritmem detekovány, může už být pozdě na nápravu.

- **Sofistikované diagonální modifikace v průběhu rozkladu.** Takové modifikace mohou brát například do úvahy poměr mezi absolutními hodnotami diagonálních a mimodiagonálních prvků v průběhu rozkladu. Takový postup, který byl svého času populární v optimalizačních aplikacích, může vyústit ve velmi nepřesný neúplný rozklad nebo poskytnout rozklad, které žádnou modifikaci nepotřeboval.

- **A priori posunuté rozklady** [114] rozkládají matici $A+\alpha I$ nebo $A+\alpha D_A$, kde α je skalární parametr. Dostatečně velké kladné α, které zabezpečuje existenci rozkladu, vždy existuje, ale problémem může takový parametr, aby chybová matice E, která vznikne na základě a priorního posunu i neúplnosti rozkladu, nebyla příliš veliká a rozklad se tedy blížil rozkladu původní matice.

- **Kompenzované redukce**, jejichž varianta byla zmíněna výše, jsou například neúplné rozklady M-matice blízké matici A. Pro ty pak neúplný rozklad existuje. Jejich konstrukce je například založena na tom, že hodnoty kladných mimodiagonálních prvků matice A jsou přičteny k diagonále matice, čímž se zachovají řádkové součty mezi původní a modifikovanou maticí. Tento přístup je robustní hlavně pro ty matice, které jsou samy blízké M-maticím; viz [14] nebo podobný přístup v [59].

- **Lokální modifikace podle Ajize a Jenningse**. Tato strategie [92], [93], [1] nabízí dynamický způsob odvrhování, kde odvrhované prvky nejsou vybrány ze struktury řídkosti původní matice, ale jsou zvoleny na základě hodnot v Schurově doplňku neúplného rozkladu. Schéma lze znázornit následovně

$$
\begin{array}{c c}
 & \begin{array}{ccccc} & \mathrm{i} & & \mathrm{j} & \end{array} \\
\begin{array}{c} \\ \mathrm{i} \\ \\ \mathrm{j} \\ \\ \end{array} &
\begin{pmatrix}
\ddots & & & & \\
 & \gamma\,|a_{ij}| & & -a_{ij} & \\
 & & \ddots & & \\
 & -a_{ij} & & |a_{ij}|/\gamma & \\
 & & & & \ddots
\end{pmatrix},
\end{array}
$$

kde nenulové hodnoty v této matici odpovídají prvkům, kterými modifikujeme Schurův doplněk, nechceme-li připustit zaplnění na jeho pozici (i, j) (provádíme odvrhnutí na této pozici). Parametr γ je často pro jednoduchost zvolen jednotkový. Strategie je robustní z hlediska existence, protože taková modifikace znamená přičítání pozitivně semidefinitní matice k matici A. Byla-li původní matice regulární, pak tedy rozklad zřejmě existuje. Blokové varianty tohoto postupu, kde je vidět ještě další zvýšení robustnosti rozkladu, je možné najít například v [21]. Stejně jako v podobných situacích, rozklad může být efektivní, není-li rozkládaná matice příliš vzdálená od matice A.

- Pozitivně semidefinitní modifikace podle [155] je dalším možným postupem. Uvažujme nejprve krok úplného LU rozkladu v následujícím podmaticovém zápisu zapsaným stejně jako výše

$$A = \begin{pmatrix} a & r^T \\ c & S \end{pmatrix} = \begin{pmatrix} I & \\ c\,a^{-1} & I \end{pmatrix} \begin{pmatrix} a & \\ & S - c\,a^{-1}\,r^T \end{pmatrix} \begin{pmatrix} 1 & a^{-1}\,r^T \\ & I \end{pmatrix}$$

Nyní rozložme obecně řídké vektory c a r na dvě části s různými komponentami a označme je následovně.

$$r = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}, \qquad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

Schurův doplněk

$$S - c\,a^{-1}\,r^T$$

úplného rozkladu pak budeme modifikovat na tvar

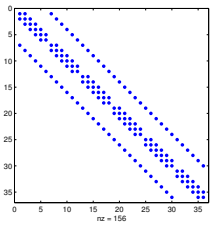
$$S - c\,a^{-1}\,r^T + R = S - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}^T + \begin{pmatrix} 0 & 0 \\ 0 & c_2\,r_2^T \end{pmatrix} \tag{16.105}$$

Existuje více způsobů jak rozložit sloupcový a řádkový vektor vystupující v konkrétním kroku rozkladu. Populární volbou je takový rozklad, kde v r_1 a c_1 jsou prvky s obecně většími absolutními hodnotami než v zbylých částech, tedy než, po řadě, v r_2 a c_2. Tato varianta rozkladu velmi efektivní, ačkoli implementace není přímočará a je zapotřebí ji obvykle ještě dále modifikovat, aby byla efektivní.

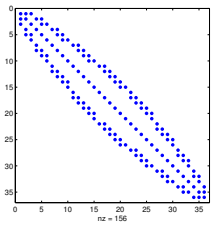
16.7.6 Neúplné rozklady a přeuspořádání

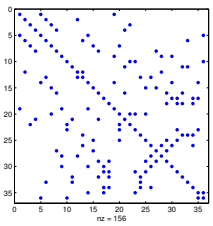
V případě úplného rozkladu symetrické a pozitivně definitní matice Choleského rozkladem je obvyklé volit počáteční uspořádání statickou symetrickou permutací, která **minimalizuje zaplnění v matici**, jak bylo diskutováno výše. V průběhu řídké Choleského faktorizace se pak obvykle používají další permutace motivované například vhodným topologickým očíslováním eliminačního stromu, vytvářením supervrcholů, ale i paralelním zpracováním, o kterém jsme zatím nemluvili. Tyto další permutace už často nemění velikost zaplnění. V případě neúplných rozkladů je volba vhodného uspořádání, ať už

počátečního nebo vytvářeného v průběhu rozkladu, obtížnější, protože vliv uspořádání na různé aspekty chodu předpodmíněných iteračních metod je méně předvídatelný. Následující obrázky demonstrují graficky několik typů uspořádání, které byly vyvinuty především pro použití v úplných rozkladech přímých metod a které byly také zkoumány v souvislosti s neúplnými rozklady.

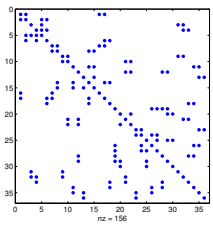


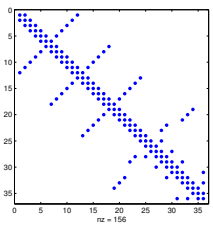
$\longleftarrow$ přirozené
rcm $\longrightarrow$



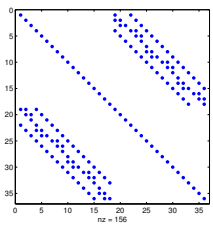


$\longleftarrow$ mmd
nd $\longrightarrow$





$\longleftarrow$ spirální
r/b $\longrightarrow$



Sumarizujme nyní několik základních faktů o použití uspořádání pro neúplné faktorizace, přičemž budeme také hovořit o způsobech uspořádání, které vysvětlíme teprve později.

- Uspořádání pro neúplný Choleského rozklad [52].
  - **Přirozené uspořádání** a **permutace nenulových prvků matice směrem k diagonále** pro matice z diskretizací na strukturovaných sítích je obecně dobrou volbou pro neúplné rozklady, kde připouštíme malé zaplnění. Přirozeným uspořádáním je přitom systematické očíslování vrcholů sítě po řádcích nebo sloupcích.
  - Uspořádání typu nejmenšího stupně a nejmenšího zaplnění (MD/MF) a **červeno-černé** uspořádání v kombinaci s malým dovoleným zaplněním není obvykle vhodné.
  - Většina uspořádání s cílem zvýšit **paralelismus** není také obvykle velmi efektivní.
  - Neúplné rozklady, které jsou opravdu efektivní, je vhodné založit nejen na struktuře zaplnění, ale i na významu prvků ve faktorech ve formě jejich velikosti.

- Uspořádání pro LU rozklad obecně nesymetrických matic
  - **Permutace nenulových prvků matice směrem k diagonále** bývá velmi účinná. Ta může zároveň rozklad **stabilizovat** a posílit **datovou lokalitu**.
  - Pro jednoduché neúplné rozklady z diskretizací na strukturovaných sítích není přirozené uspořádání obvykle příliš vhodné.
- Další možnosti jsou například
  - Uspořádání, které heuristicky minimalizuje nějakou normu chybové matice E [41]
  - Pivotace v rozkladech [138], [23] sice obvykle zvyšuje složitost, ale může se vyplatit. Poznamenejme, že pivotace v neúplných rozkladech může být na sekvenční počítačové architektuře často mnohem levnější než pivotace v úplných rozkladech.
  - A priorní uspořádání založené na posilování diagonály technikami maximálního párování a váženého maximálního párování [121], [50], [51].

16.7.7 Poznámky k implementaci neúplného rozkladu

Všechny uvedené algoritmy neúplného rozkladu představují tento rozklad z matematického pohledu. K tomu, aby byla implementace efektivní je zapotřebí navrhnout také vhodné datové struktury a práci s nimi. Nejmarkantněji je tento problém vidět u rozkladu, kde odvrujeme prvky podle významnosti nebo u ILU(p), kde dokonce hovoříme explicitně o n^2 pozicích, pro které se vyčísluje a případně aktualizuje hodnota jejich úrovně. Přesto i v tomto případě existují implementace, které se vyhnou paměťově náročným postupům. Existuje například algoritmus, jak spočítat výslednou strukturu ILU(p) dopředu, a to dokonce paralelně [91]. V tomto textu se nebudeme implementačními postupy zblízka zabývat, ale jejich roli připomeneme v cvičeních k tomuto textu.

16.7.8 Kombinovat předpodmínění s maticí soustavy nebo ne?

Vraťme se nyní k otázce, zdali zkombinovat matici A a matici/matice vyjadřující předpodmínění do jedné matice. Výše jsme zmínili, že ve většině situací je předpodmínění aplikováno nezávisle na maticovém násobení jako samostatný krok v rámci iterační metody. Nicméně, existují případy, kdy je vhodné předpodmínění zkombinovat s maticovým násobením z hlediska efektivity. Uvažujme matici

$$A = D_A - L_A - U_A,$$

kde D_A je diagonální část matice A, L_A je ostře dolní trojúhelníková a U_A je ostře horní trojúhelníková část matice. Následující postup se nazývá **Eisenstatův trik** [60] a dá se použít na předpodmiňování soustav s maticí A, kde se předpodmínění M dá zapsat ve tvaru

$$M = (D - L_A)D^{-1}(D - U_A) \equiv (D - L_A)[D^{-1}(D - U_A)] \equiv M_1 M_2, \tag{16.106}$$

kde D je diagonální matice, obecně různá od D_A. Tím tento typ předpodmiňování pokrývá neúplné rozklady z některých jednoduchých diskretizačních modelů jak jsme o nich hovořili výše. Uvažujme oboustranné předpodmínění, zapsané jako

$$(I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1}.$$

Do tohoto tvaru přepíšeme předpodmínění aplikované jako $M = M_1 M_2$ oboustranně na matici A podle (16.13) následovně

$$\begin{aligned}(D - L_A)^{-1}A[D^{-1}(D - U_A)]^{-1} &= [(D - L_A)^{-1}D]D^{-1}A[D^{-1}(D - U_A)]^{-1} \\ &= [D^{-1}(D - L_A)]^{-1}D^{-1}A(I - D^{-1}U_A)^{-1} \\ &= (I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1}\end{aligned}$$

pro

$$\tilde{L}_A = D^{-1}L_A, \tilde{U}_A = D^{-1}U_A, \ \tilde{A} = D^{-1}A.$$

Oboustranně předpodmíněnou soustavu soustavu můžeme pak zapsat ve tvaru

$$(I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1}y \equiv (I - D^{-1}L_A)^{-1}D^{-1}A(I - D^{-1}U_A)^{-1}y = (I - D^{-1}L_A)^{-1}D^{-1}b. \tag{16.107}$$

Vyjádřeme nyní matici $\hat{A} \equiv (I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1}$ transformované soustavy následovně

$$\begin{aligned}\hat{A} &= (I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1} = (I - \tilde{L}_A)^{-1}(D^{-1}D_A - \tilde{L}_A - \tilde{U}_A)(I - \tilde{U}_A)^{-1} \\ &= (I - \tilde{L}_A)^{-1}(I - \tilde{L}_A + D^{-1}D_A + I - \tilde{U}_A - 2I)(I - \tilde{U}_A)^{-1} \\ &= (I - \tilde{L}_A)^{-1}[(I - \tilde{L}_A)(I - \tilde{U}_A)^{-1} + (D^{-1}D_A - 2I)(I - \tilde{U}_A)^{-1} + I] \\ &= (I - \tilde{U}_A)^{-1} + (I - \tilde{L}_A)^{-1}(D^{-1}D_A - 2I)(I - \tilde{U}_A)^{-1} + I.\end{aligned}$$

Pak může být aplikace matice $\hat{A}z$ na vektor z spočtena použitím vztahů

$$\begin{aligned}\tilde{z} &= (I - \tilde{U}_A)^{-1}z \\ \tilde{\tilde{z}} &= (I - \tilde{L}_A)^{-1}(z + (D^{-1}D_A - 2I)\tilde{z}) \\ (I - \tilde{L}_A)^{-1}\tilde{A}(I - \tilde{U}_A)^{-1}z &= \tilde{z} + \tilde{\tilde{z}}\end{aligned}$$

Snadno nahlédneme, že složitost aplikace předpodmíněné matice je založena na složitosti dvou substitucí s trojúhelníkovými maticemi, násobení diagonální maticí a součtu vektorů. To není o mnoho více než složitost samotných substitucí. Jinými slovy, zkombinováním předpodmínění a maticového násobení získáme efektivnější proceduru. Připomeňme ale, že podstatný je speciální tvar předpodmínění podle (16.106), protože tento trik není obecný.

16.8 Předpodmínění a paralelní výpočty

V této části budeme diskutovat aspekty předpodmiňování při výpočtech na paralelních počítačích, tedy stručně paralelního předpodmiňování. Sumarizujme si nejprve některé aspekty paralelního předpodmiňování.

- Konstrukce předpodmínění musí být často přeformulována tak, aby byla efektivní z pohledu počítačové architektury. Ačkoli je konstrukce neúplných rozkladů obvykle velmi levná, cenou za ni je jejich častá neefektivita ve smyslu konvergence.

- Časté je snížení efektivity neúplných rozkladů jako předpodmínění v rámci moderních počítačových architektur z důvodu menšího výskytu hustých bloků, které by mohly využívat efektivní BLAS3 aritmetiku.

- Tato neefektivita je ale často kompenzována větším potenciálem pro využití stromového paralelismu, který jde nad rámec paralelismu dovoleného eliminačním stromem.

- Velkou výzvou pro využití paralelních počítačových architektur jsou **substituční kroky**. Ty mohou být pro řidší neúplné faktory v tomto případě někdy efektivnější než jejich analogie pro úplné rozklady využitím techniky **substituce po úrovních** (level scheduling). Efektivita substitucí je v případě neúplných rozkladů použitých jako předpodmínění velmi kritická, protože jsou v rámci iteračních metod aplikovány **opakovaně**. Jednou či víckrát v každé iteraci.

V následujícím textu rozlišíme několik hlavních směrů, které se historicky uvažovaly pro vývoj vhodných předpodmínění založených na neúplných rozkladech pro moderní počítačové architektury. Některé z těchto směrů se týkají pouze efektivní paralelní konstrukce neúplných rozkladů, jiné z nich se zaměřují na jejich aplikaci v rámci iterační metody nebo na obě dvě oblasti zároveň: konstrukci i aplikaci.

- Specifické aproximační techniky nebo modifikace rozkladů.

- Specifická **uspořádání**.

- Nalezení a využití bloků rozkládané matice.

- Aposteriorní uspořádání pro efektivní maticové operace s faktory.

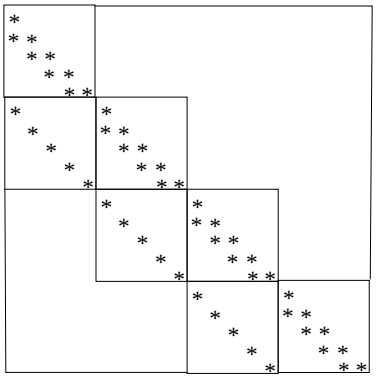
- Přímá aproximace maticové inverze $M^{-1} \approx A^{-1}$ (inverzní neúplné rozklady).

16.8.1 Specifické aproximační techniky nebo modifikace rozkladů

Standardní přímé neúplné rozklady jako jsou ILU/MILU/IC/MIC mohou být dále modifikovány tak, aby byly na paralelních počítačích efektivnější. Některé z těchto postupů mají za sebou dlouhý historický vývoj. Velmi často přitom byly modifikace původně formulovány ve formě **diskretizačních schémat** namísto v maticovém tvaru. Často také předpokládaly modelový problém s jednoduchou strukturu rozkládané matice. V následujícím textu budeme takové přístupy ilustrovat několika příklady navržených postupů.

16.8.1.1 Částečná vektorizace (partial vectorization)

Částečná vektorizace je strategie, která využívá vektorizační potenciál matic z diskretizací parciálních diferenciálních rovnic na strukturovaných sítích. Neúplný rozklad často omezuje nové zaplnění na pozice původní struktury řídkosti z hlavní diagonály a z několika vedlejších diagonál a případně pozice několika dalších vektorů. Rozklad i jeho aplikace ve formě substitucí mohou být částečně **vektorizovány** algoritmem, který pracuje s vektory zachycujícími diagonály matic. Příklad matice z pětibodového diskretizačního schématu je znázorněn níže.



Problémem postupu je speciálnost struktury řídkosti matice, ktrerou je třeba uvažovat.

16.8.1.2 Aposteriorní sparsifikace (forced aposteriori annihilation)

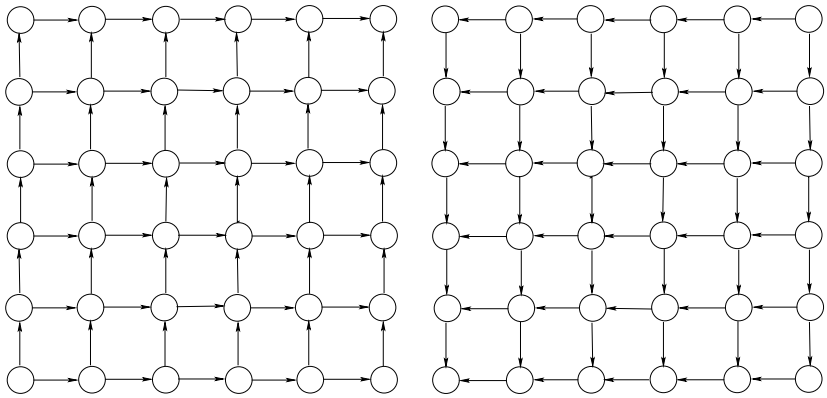
Jiným způsobem, jak alespoň částečně paralelizovat výpočet a substituční kroky je modifikace faktorů předepsáním struktury řídkosti $\mathcal{S}$ tak, že se odvrhnou prvky neúplných faktorů, které zabraňují efektivní vektorizaci nabo paralelizaci algoritmu a tím umožní efektivnější použití více procesorů pro výpočet. Máme-li k dispozici již spočítaný neúplný rozklad, můžeme jej modifikovat **aposteriorní sparsifikací** trojúhelníkových faktorů [145]. Pro znázornění uvažujme následující schéma. Na levé straně obrázku je původní spočítaný bidiagonální faktor. Sparsifikovaný bidiagonální faktor je znázorněn na pravé straně tohoto obrázku.

$$
\begin{pmatrix}
* & & & & & & & & & & & \\
* & * & & & & & & & & & & \\
 & * & * & & & & & & & & & \\
 & & * & * & & & & & & & & \\
 & & & * & * & & & & & & & \\
 & & & & * & * & & & & & & \\
 & & & & & * & * & & & & & \\
 & & & & & & * & * & & & & \\
 & & & & & & & * & * & & & \\
 & & & & & & & & * & * & & \\
 & & & & & & & & & * & * & \\
 & & & & & & & & & & * & *
\end{pmatrix}
\rightarrow
\begin{pmatrix}
* & & & & & & & & & & & \\
* & * & & & & & & & & & & \\
 & * & * & & & & & & & & & \\
 & & & & & & & & & & & \\
 & & & * & & & & & & & & \\
 & & & * & * & & & & & & & \\
 & & & & * & * & & & & & & \\
 & & & & & * & * & & & & & \\
 & & & & & & & & & & & \\
 & & & & & & & & * & & & \\
 & & & & & & & & * & * & & \\
 & & & & & & & & & & * & *
\end{pmatrix}
$$

Tento postup má obvykle dvě zásadní vady. Za prvé, nalézt takové prvky faktorů, které mají být tímto postupem odvrhnuty a zároveň příliš neohrozily kvalitu rozkladu, může být velmi obtížné. Sparsifikace, která je založena pouze na struktuře faktorů a nikoli na numerických hodnotách může totiž výrazně **zpomalit** konvergenci předpodmíněné iterační metody.

16.8.1.3 Paralelní zpracování po úrovních (wavefront processing).

Paralelní zpracování po úrovních je technika navržená pro rozklad matic na strukturovaných sítích stejně jako řada dalších a zde diskutovaných postupů. Potenciál pro paralelizaci je zde velmi podobný technikám pro paralelní implementace stacionárních iteračních metod, které ztotožňují komunikační síť s diskretizační sítí a pracují s jemnozrným paralelismem. Konkrétní strategie pro neúplné rozklady matic z pětibodových diskretizací dvojrozměrných oblastí na strukturovaných sítích je znázorněna na obrázku níže. Šipky ukazují **závislost** ve výpočtu numerických hodnot. Právě fakt, že faktorizace je neúplná ale hraje pozitivní roli v efektivitě postupu.



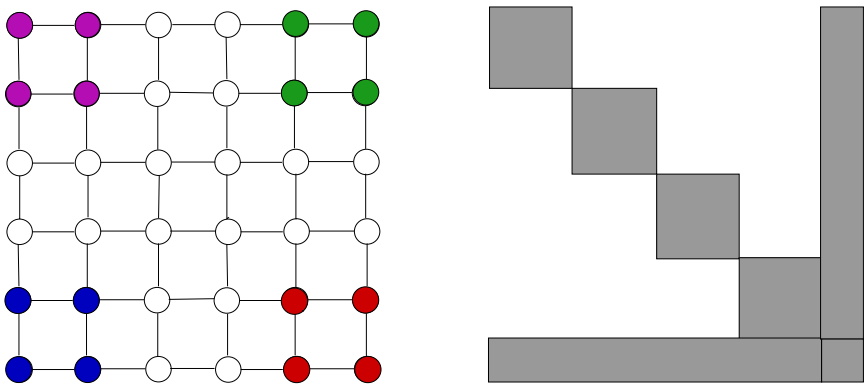
Toto schéma může být zobecněno například na rozklad matic ze sedmibodové diskretizace na strukturovaných sítích ve 3D. V tomto případě se postup tradičně nazývá **nadrovinový přístup**. Další zobecnění zahrnují například předpodmiňování na diskretizace nestrukturovaných sítí, kde se ale závislosti mezi numerickými hodnotami obvykle obtížněji detekují a postup může být nepříliš efektivní.

16.8.1.4 Přetočený rozklad a jeho zobecnění na souběžný rozklad více oblastí.

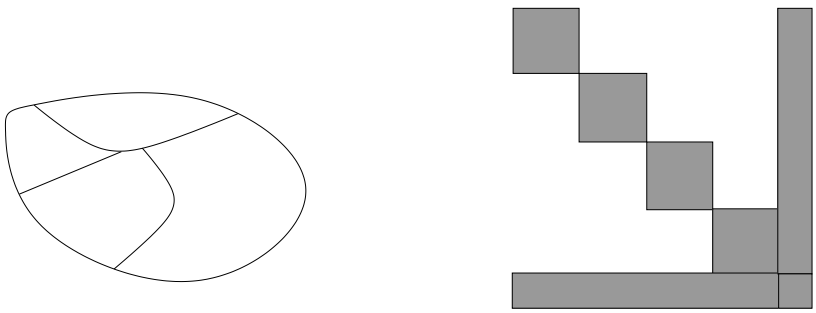
Zajímavý způsob rozkladu nazývaným **přetočený rozklad** (twisted factorization) kde výsledné faktory kombinují horní a dolní trojúhelníkové části byl zaveden v souvislosti s paralelním počítáním. Tento rozklad je charakterizován paralelním rozkládáním matice z obou konců najednou, tedy zároveň podle vzrůstajícího indexu a zároveň podle klesajícího indexu. Struktura řídkosti rozkládané matice je pak reprezentována následujícím zobrazeným způsobem [17], [158] a samotné vyjádření algoritmu rozkladu není obtížné.

$$\begin{pmatrix} * & & & & & & & \\ * & * & & & & & & \\ & * & * & & & & & \\ & & * & * & & & & \\ & & & * & * & * & & \\ & & & & & * & * & \\ & & & & & & * & * \\ & & & & & & & * \end{pmatrix} \qquad \begin{pmatrix} * & * & & & & & & \\ & * & * & & & & & \\ & & * & * & & & & \\ & & & * & * & & & \\ & & & & * & & & \\ & & & & * & * & & \\ & & & & & * & * & \\ & & & & & & * & * \end{pmatrix}$$

Paralelismus získaný tímto způsobem je pouze dvojnásobný, ale může být ještě zvýšen několika způsoby. Za prvé, přetočený rozklad může být prováděn **rekurzívně**, což ale samozřejmě komplikuje implementaci založenou na hierarchickém přístupu. Za druhé, postup může být zobecněn na matice z diskretizací na jednoduchých strukturovaných 2D/3D sítích **pro paralelním rozklad směrem od rohů sítě** nebo i na matice obecnější. Uspořádání, které je použito pro rozklad začínající simultánně z rohů sítě pravidelné 2D sítě je znázorněno na následujícím obrázku.



Zobecněné schéma, které používá více **oblastí** a vyhýbá se tak i pojmu rohů, které jsou přirozené pouze pro strukturované sítě je na následujícím obrázku. Algoritmus pak může být efektivnější, ale jeho implementace je samozřejmě mnohem náročnější.



Takovéto obecné schéma je užitečné pro matice z diskretizací obecných oblastí nebo pro matice, které jsou dané pouze algebraicky. Algoritmus obvykle předpokládá předzpracování

sofistikovaným dělením grafu. Paralelní výpočetní proudy provádějící rozklad jsou na rozhraní propojeny řešením problému, který rozklad finalizuje. Takový problém není obvykle obtížné odvodit, ale zde se tomu nebudeme věnovat. Iterační i přímé metody založené na obecné technice **rozkladu na oblasti (domain decomposition)** mají silné teoretické základy a také nabízejí standardizované postupy jak řešit problém na rozhraní. Podobně se tedy i oblast neúplných faktorizací vedoucích k paralelním algoritmům se musí vypořádat s řešením problémů na rozhraní [131], [151], [19], ale bližší zkoumání je už za hranicí tohoto textu.

16.8.2 Specifická uspořádání pro neúplné rozklady

Řada způsobů, jak zvýšit efektivitu výpočtů předpodmíněných iteračních metod na paralelních počítačích je založena na nalezení vhodného uspořádání, které možnost souběžného zpracování v neúplném rozkladu a případně i jeho aplikaci posílí. Zde zmíníme způsob, který je založený na **červeno-černém** uspořádání pro dosti speciální matice a který se dá ale také zobecnit. Další podobné a často velmi sofistikované způsoby uspořádání zde nebudeme diskutovat.

Zdůrazněme ale jeden fakt spojený s neúplnými rozklady přeuspořádaných matic. Rychlost konvergence předpodmíněných iteračních metod je obvykle novým uspořádáním ovlivněna. Konkrétně, uspořádání, která jsou založena na tom, že pro zvýšení paralelismu omezují globální propojení uvnitř matice, tak jak tomu je například u metody vnořených řezů, ale je tomu tak i u červeno-černého uspořádání, mohou silně zpomalit konvergenci předpodmíněné iterační metody [95].

16.8.2.1 Červeno-černá uspořádání a jejich víceúrovňová rozšíření

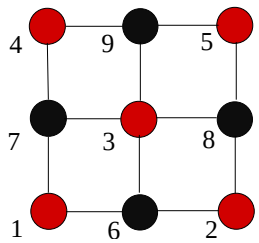
Tato uspořádání podporují paralelní zpracování podobným způsobem jako metoda cyklické redukce. Populární metodou tohoto druhu je červeno-černé uspořádání jehož motivací byly jednoduché diskretizace na strukturovaných sítích. Uvažujme diskretizovanou 2D Poissonovu rovnici znázorněnou na Obrázku 16.50.

16.8.2.1.1 Červeno-černé uspořádání Červeno-černé uspořádání je uspořádání vrcholů sítě, respektive, řádků a sloupců odpovídající matice takové, že symetricky permutovaná matice může být zapsána v blokovém tvaru (16.108).

$$P^T A P = P^T (D_A - L_A - U_A) P = \begin{array}{c} \\ \text{červené vrcholy} \\ \text{černé vrcholy} \end{array} \begin{array}{c} \begin{array}{cc} \text{červené vrcholy} & \text{černé vrcholy} \end{array} \\ \begin{pmatrix} D_1 & -\tilde{U}_A \\ -\tilde{L}_A & D_2 \end{pmatrix} \end{array} \tag{16.108}$$

kde D_1 a D_2 jsou diagonální matice. Pro symetrickou matici přitom máme samozřejmě $L_A = U_A^T$. Následující obrázek ukazuje síť s vrcholy, jejichž obarvení motivuje název tohoto uspořádání.

Červeno-černé uspořádání existuje právě tehdy, je-li neorientovaný grafový model rozkladu **bipartitní**. První **blokový** krok tohoto rozkladu využívající faktu, že blokový pivot



$$\begin{pmatrix} 4 & & & & & -1 & -1 & & \\ & 4 & & & & -1 & -1 & & \\ & & 4 & & & -1 & -1 & -1 & -1 \\ & & & 4 & & & -1 & & -1 \\ & & & & 4 & & & -1 & -1 \\ -1 & -1 & -1 & & & 4 & & & \\ -1 & & -1 & -1 & & & 4 & & \\ & -1 & -1 & & -1 & & & 4 & \\ & & -1 & -1 & -1 & & & & 4 \end{pmatrix}$$

Obrázek 16.8.2: *Červeno-černé uspořádání a matice soustavy jsou uspořádány tak, že vrcholy obarvené červenou barvou jsou zařazeny jako první.*

označený jako D_1 je diagonální lze zapsat

$$A = P \begin{pmatrix} D_1 & -\tilde{U}_A \\ -\tilde{L}_A & D_2 \end{pmatrix} P^T = P \begin{pmatrix} I & \\ -\tilde{L}_A D_1^{-1} & I \end{pmatrix} \begin{pmatrix} D_1 & \\ & D_2 - \tilde{L}_A D_1^{-1} \tilde{U}_A \end{pmatrix} \begin{pmatrix} I & -D_1^{-1}\tilde{U}_A \\ & I \end{pmatrix} P^T. \tag{16.109}$$

Částečná eliminace řádků a sloupců, která odpovídá prvnímu blokovému kroku rozkladu, je tedy založena na jednoduchém **škálování mimodiagonálního bloku** maticí D_1, což je operace dobře paralelizovatelná. Je-li matice D_2 také diagonální, zjednodušuje to dále výpočet Schurova doplňku

$$S = D_2 - \tilde{L}_A D_1^{-1} \tilde{U}_A. \tag{16.110}$$

V případě naší matice z 2D Poissonovy rovnice může být matice A považována za **blokově tridiagonální** a tato vlastnost se přenáší do Schurova doplňku. V případě úplného rozkladu se rekurzívní aplikace takového postupu nazývá **cyklická redukce**.

16.8.2.1.2 Červeno-černé uspořádání a problémy se stabilitou neúplného rozkladu Tak jak jsem už obecněji naznačili, červeno-černé uspořádání může **zefektivnit rozklad** na paralelních počítačových architekturách, ale může zároveň negativně ovlivnit rychlost konvergence předpodmíněné iterační metody. Ale nejenom to, uspořádání může ovlivnit **existenci rozkladu** v některých neúplných rozkladech (určených, například předpokládanou strukturou řídkosti a modifikacemi). Abychom to ukázali, uvažujme jednoduchý modifikovaný rozklad MIC(0) [58], který nepřipouští žádné zaplnění, ale modifikuje diagonálu ve Schurově doplňku tak, aby zachovával řádkové součty.

Diskretizační hvězdu černého vrcholu (i, i) v 2D Laplaceově operátoru pro diagonální prvek a_{ii} znázorníme následovně. Pro jednoduchost předpokládejme, že vrchol na pozici (i, i) je vnitřním vrcholem tak, že na všech stranách je mezi sousedícím červeným vrcholem a hranicí oblasti ještě alespoň jeden černý vrchol.

$$\begin{matrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & -1 & \cdot & \cdot \\ \cdot & -1 & 4 & -1 & \cdot \\ \cdot & \cdot & -1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{matrix} \tag{16.111}$$

Uvažujme nyní eliminaci **červených** vrcholů (řádků/sloupců), které jsou v červeno-černém uspořádání očíslovány dříve než černé vrcholy. Pak platí:

- Na pozici (i, i) je černý vrchol modifikován v MIC(0) sousedními červenými vrcholy následovně

$$a_{ii} = a_{ii} - \sum_{j_i=1}^{4} \frac{1}{a_{j_i j_i}} = 4 - 4 \times \frac{1}{4} = 3, \; j_i \text{ je soused i v síti.} \tag{16.112}$$

- Každý z těchto **čtyřech** červených sousedů generuje tři prvky zaplnění mezi těmito vrcholy navzájem, protože se eliminací propojí do kliky. Příslušné hodnoty zaplnění jsou v $MIC(0)$ ale odvrženy a tedy jsou jejich hodnoty odečteny od diagonálního prvku. To můžeme zapsat

$$a_{ii} = a_{ii} - \sum_{j_i=1}^{4} 3 \times \frac{1}{a_{j_i j_i}} = 3 - 3 = 0, \; j_i \text{ je soused i v síti.} \tag{16.113}$$

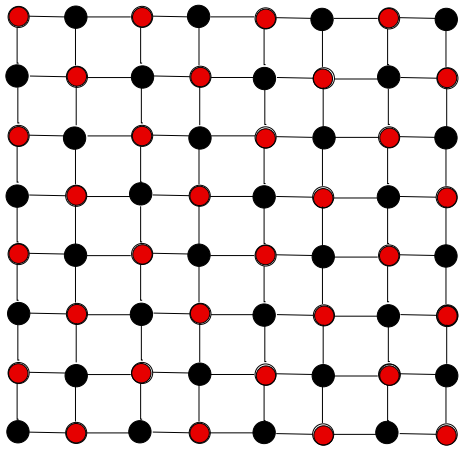
Rozklad MIC(0) tedy s maticí z modelového příkladu a červeno-černým uspořádáním bez dalších opravných modifikací **nebude existovat**. V tomto případě je vidět, že honba za zvýšeným paralelismem **může být někdy kontraproduktivní**. Z obecného pohledu je že červeno-černé uspořádání kombinované s agresívním odvrhováním prvků faktorů příkladem výrazného narušení **lokálních propojení** ve faktorech tím, že

- se nepřirozeně oddělují vrcholy, které byly v původní síti topologicky blízko a přitom
- jsou kompenzovány pouze ideou zachování řádkových součtů.

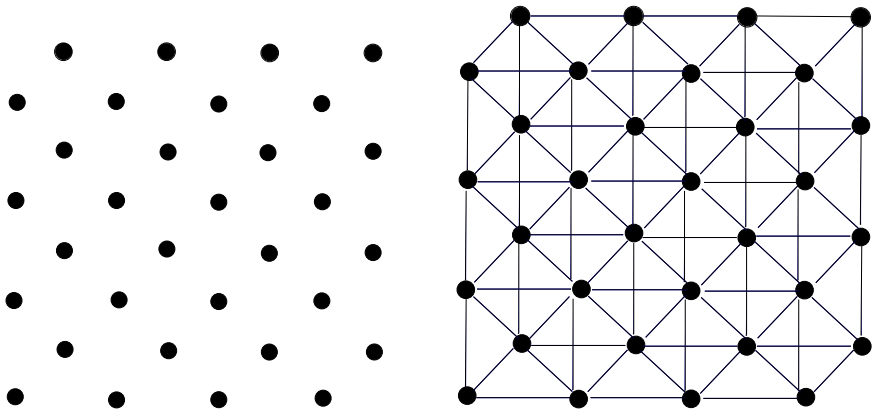
Podobná situace může nastat i u jiných modifikovaných neúplných rozkladů.

16.8.2.1.3 Rekurzívní uspořádání červeno-černého typu **Rekurzívní aplikace** uspořádání založených na červeno-černém očíslování může mít pozitivní vliv na velikost čísla podmíněnosti předpodmíněné soustavy. Přesvědčivě se to dá ukázat v případě řešení úloh z modelových problémů. Takové zmenšení se ale nemusí jednoduše transformovat do rychlosti konvergence metody sdružených gradientů, protože závislost její konvergence na spektru je složitější, ale stejně může být takové zmenšení někdy výhodou. Existuje více podobných rekurzívních postupů aplikace lišících se obvykle přípustným zaplněním na jednotlivých úrovních. Některé z nich nyní schématicky uvedeme.

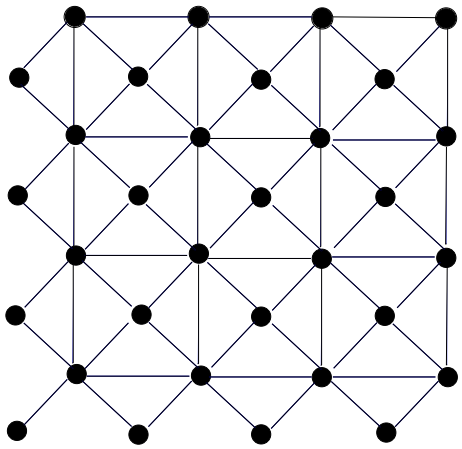
Uvažujme červeno-černou síť na na následujícím obrázku.



Na dalším obrázku vidíme dva extrémní případy. Vlevo černé vrcholy, které nebyly v dané matici propojeny a které odpovídají řádkům a sloupcům Schurova doplňku po jednom blokovém kroku rozkladu, kdybychom **odvrhli všechno zaplnění**. Napravo jsou znázorněny černé vrcholy se zaplněním, které by vzniklo v případě **úplného rozkladu**.



Nyní ukážeme dvě možnosti, které připustí v neúplném rozkladu pouze část zaplnění. Na následujícím obrázku je situace navržená v publikaci Brand (1992), viz také Brand, Heinemann (1989).



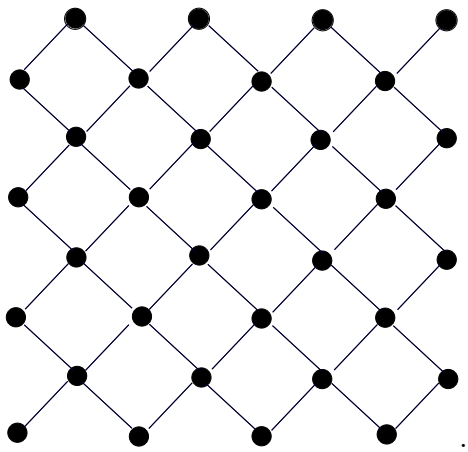
V tomto případě je možné ukázat, že číslo podmíněnosti symetricky předpodmíněné matice, tedy matice

$$M^{-1/2}AM^{-1/2}, \tag{16.114}$$

s použitím modifikovaného neúplného rozkladu pro modelovou úlohu s takto definovanou strukturou řídkosti je při rekurzívní aplikaci rovno asymptoticky

$$O(h^{-1/2}). \tag{16.115}$$

Jiná struktura přípustného zaplnění v MIC (Ciarlet, Jr. (1992)) je znázorněná na obrázku



V tomto případě je při rekurzívní aplikaci číslo podmíněnosti

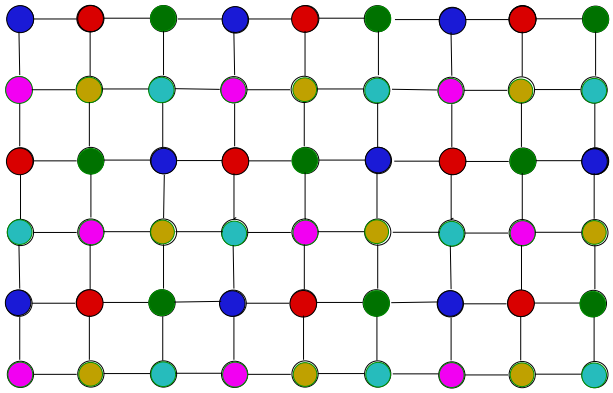
$$O(h^{-1/3}). \tag{16.116}$$

V obou situacích se uvažuje **modifikovaná** neúplná faktorizace. Experimenty naznačují, že podobné asymptotické snížení čísla podmíněnosti předpodmíněné matice může platit i pro obecnější problémy a i na diskretizacích z nestrukturovaných sítí s příslušnými algoritmickými modifikacemi.

16.8.2.1.4 Vícebarevná uspořádání Vícebarevná uspořádání zobecňují červeno-černé uspořádání. Vycházejí z **korektního** obarvení vrcholů grafu více než dvěma barvami, tak, že žádné vrcholy stejné barvy nejsou spojeny hranou. Dva důvody proč jimi nahradit červeno-černá uspořádání jsou

- **problémy se stabilitou** při použití červeno-černého uspořádání,
- existence uspořádání s $k > 2$ barvami pro grafové modely mnohem obecnějších matic a
- prokazatelně lepší konvergenční vlastnosti předpodmíněné iterační metody při použití více barev, protože vícebarevná uspořádání neruší tolik lokálních vztahů mezi vrcholy, které byly v grafovém smyslu blízko, viz Doi, 1991; Doi, Lichnewsky, 1991; Doi, Hoshi, 1992; Wang, Hwang, 1995.

Příkladem obarvení grafového modelu s použitím více barev je na následujícím obrázku.



Vícebarevné uspořádání implikuje takovou blokovou strukturu matice, kde diagonální bloky jsou tvořeny **diagonálními maticemi.** Je tedy zřejmé, že použití více barev omezuje využitelný paralelismus v konstrukci neúplného rozkladu.

16.8.2.1.5 Dynamické konstrukce diagonálních pivotů Princip vícebarevného uspořádání můžeme dále zobecnit a propojit s konstrukcí neúplného rozkladu. Vícebarevné uspořádání z předchozího textu přímo vytváří přeuspořádanou matici s diagonálními bloky o dimenzích rovných počtům vrcholů stejné barvy a krok rozkladu je od tohoto přeuspořádání oddělen a může být proveden více způsoby. Z hlediska rozkladu můžeme postupovat i dynamicky následujícím způsobem střídáním kroků blokové pivotace a kroků rozkladu, konkrétně střídáním následujících dvou kroků:

- Najít co největší **nezávislou** množinu V_I vrcholů grafu,
- provést krok rozkladu s blokovým pivotem určeným V_I.

Pro hledání nezávislé množiny v grafu existuje celá řada způsobů. Některé z nich mohou i formálně využívat technik barvení grafu, ale podrobnější popis jde nad rámec tohoto textu. Formálně můžeme jeden krok takového LU rozkladu symetricky permutované matice A s diagonálním blokem D zapsat ve tvaru

$$P^T A P = \begin{pmatrix} D & -\hat{U}_A \\ -\hat{L}_A & C \end{pmatrix} = \begin{pmatrix} D_1 & \\ -\hat{L}_A & I \end{pmatrix} \begin{pmatrix} I & \\ & C - \hat{L}_A\, D^{-1}\, \hat{U}_A \end{pmatrix} \begin{pmatrix} D_2 & -\hat{U}_A \\ & I \end{pmatrix}, \quad (16.117)$$

kde $D = D_1\, D_2$.

Tento postup se může implementovat standardně v cyklu a nebo i rekurzívně. V tomto druhém případě často hovoříme o víceúrovňové konstrukci neúplných rozkladů. Takové algoritmy kromě možnosti přímočarého využití paralelismu mohou vykazovat ještě další výhody. Například implicitní uchovávání faktorů. Například ve víceúrovňovém algoritmu uchováváme matici L_A a nikoliv její škálovanou variantu $L_A D_1^{-1}$ jak je obvyklé ve standardní submaticové variantě rozkladu. To je obzvláště vhodné, je-li D_1 nikoli jen diagonální, ale blokově diagonální v mírně zobecněném schématu tohoto typu.

Víceúrovňové algoritmy neúplných rozkladů mohou být nejenom efektivnější na paralelních počítačových architekturách kvůli transparentnější konstrukci Schurových doplňků, ale i z důvodu lepšího využití vyrovnávací paměti počítače tím, že přirozeně formují data do bloků. Mnoho souvisejících výzkumných směrů je možné nalézt v publikacích van der Ploeg, 1994 (vnořené ILU rozklady); Botta, van der Ploeg, Wubs, 1996 (ILU pro matice z vnořených sítí nazvané NGILU), zobecnění NGILU nazvané MRILU se snahou nalézt silně diagonálně dominantní bloky, ARMS Saad, Suchomel, 2001, sofistikovaný kód ILUPACK Bölhoffer, 2004 apod.

16.8.3 Nalezení a využití bloků rozkládané matice.

Bude doplněno později.

16.8.4 Aposteriorní uspořádání pro efektivní maticové operace s faktory.

Existuje více způsobů jak zefektivnit maticové operace na více typech počítačových architektur. Jednou z možností je po výpočtu faktorů neúplného rozkladu změnit formát matice tak, aby byl vhodný pro operace jako jsou maticová násobení na vektorové architektuře nebo na počítačích s distribuovaným paralelismem. V následujícím uvedeme jako příklad dvě takové možnosti formátu, na který se musí po spočítání rozkladu matice nebo i faktory rozkladu převést.

16.8.4.0.1 Maticové formáty podporující vektorizaci Příkladem takového formátu je JAD (jagged diagonal) a jeho modifikace podle Melhem, 1988; Anderson, 1988; Paolini, Di Brozolo, 1989. Konstrukce takového schématu se dá v obecném tvaru zapsat jako

- **Komprimace** struktur řádků,
- **jejich uspořádání** podle počtu jejich nenulových prvků a
- uvažováním matice jako **množiny sloupců**, pro které je jejich součin s vektory dobře vektorizovatelný.

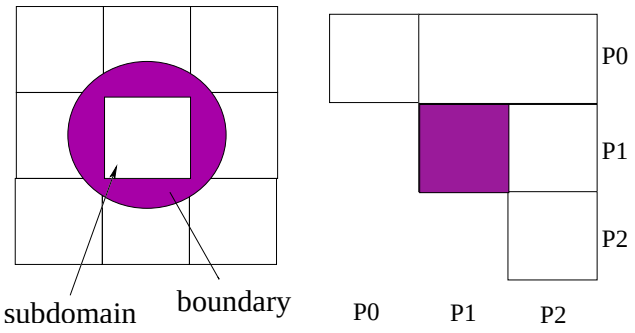
Schématicky tento postup můžeme znázornit následovně:

$$\begin{pmatrix} & & a & \\ & b & & c \\ d & & e & \\ f & & g & h \end{pmatrix} \rightarrow \begin{pmatrix} a & & \\ b & c & \\ d & e & \\ f & g & h \end{pmatrix} \rightarrow \begin{pmatrix} f & g & h \\ b & c & \\ d & e & \\ a & & \end{pmatrix}$$

Samozřejmě, existují sofistikovanější varianty tohoto schématu, viz Heroux, Vu, Yang, 1991. Vzhledem k rychle se vyvíjejícím počítačovým architekturám, schéma zde ukazujeme jako demonstraci myšlenek a nikoli jako hotový koncept.

16.8.4.0.2 Distribuovaná násobení matice a vektoru Obecná generická myšlenka pro násobení vektoru maticí je spojena s **distribuovaným paralelismem**. Uvažujme schéma této operace, kde je zapotřebí **překryv komunikace a výpočtu** napsanou jako posloupnost následujících kroků.

- **Inicializace odesílání** and **přijímání** informací o hraničních vrcholech,
- **lokální násobení**,
- **příjem** hraničních informací,
- **zakončení** výpočtu.



Připomeňme, že v obecném případě je samotné rozdělení matice/grafového modelu samostatnou úlohou nazývanou v nejjednodušší podobě **dělení grafu**.

16.8.4.0.3 Zefektivnění substitučních kroků Substituce s trojúhelníkovými faktory mohou být efektivnější **plánováním po úrovních** založeným na hledání paralelismu s využitím modelu orientovaného acyklického grafu diskutované výše. V případě neúplných rozkladů je šance získat malý počet **úrovní** a tím i významnou možnost paralelního zpracování velmi velká.

16.8.5 Přímá aproximace maticové inverze $M^{-1} \approx A^{-1}$ (inverzní neúplné rozklady)

Aproximace maticové inverze umožňuje nahradit substituční kroky, které jsou přes možnost plánování po úrovních na paralelních počítačových architekturách obvykle neefektivní.

Existuje ale ještě další důvod, proč aproximace matice A^{-1} může být výhodná. **Přímé** neúplné LU rozklady, Choleského rozklady reprezentují matici A lokálně, neboť nenulové prvky faktorů M odpovídají lokálnímu zaplnění danému **aproximací cest zaplnění**. Obecně, nenulové prvky ve faktorech aproximace A^{-1} odpovídají **cestám** v tomto grafu a tato globální informace se přenáší do M^{-1}. Neúplnost zkracuje **boldf průměrnou délku** cest zaplnění v rozkladu. Analogické závěry platí i pro nefaktorizované aproximace matice A^{-1}.

Následně, aproximace inverze matice může poskytnout efektivní předpodmínění pro řešení některých obtížných problémů nebo sloužit jako stavební prvek blokového předpodmínění, přestože z praktického pohledu udržení dostatečně řídkosti v matice M^{-1} může

být obtížné. Inverze matice se silně souvislým neorientovaným, ale i orientovaným grafovým modelem je totiž za předpokladu o nevyrušení úplně **hustá** matice a odvrhování v neúplném rozkladu musí být obvykle agresivní, aby výsledná aproximace maticové inverze byla dostatečně řídká.

Existuje mnoho způsobů jak aproximovat maticovou inverzi. Aproximace mohou být nefaktorizované nebo ve formě neúplných faktorů, ve formě polynomu či aplikovány jako iterační proces a v následujícím textu zmíníme několik základních přístupů. První z nich je založen na **minimalizaci** normy výrazu, který měří kvalitu aproximace rozdílem matice $M^{-1}A$ a jednotkové matice v nějaké normě. Další přístupy jsou založeny na přímém výpočtu **rozkladu matice** A^{-1}, často bez toho, abychom počítali nejprve Choleského nebo LU rozklad matice A a ten teprve přibližně invertovali. Další možností je iterační výpočet aproximace matice A^{-1} nebo reprezentace maticové inverze polynomem. Na závěr uvedeme přístupy, které mohou být v užitečné ve specifických aplikacích.

16.8.5.1 Předpodmínění přes minimalizaci funkcionálu ve Frobeniově normě

Uvažujme následující proceduru pro matici $A \in R^{n\times n}$ s obecně nesymetrickou, ale pozitivně definitní maticí $W \in R^{n\times n}$. Přibližná inverze matice $G \in R^{n\times n}$, kde neúplnost předepíšeme ve formě **předepsané struktury řídkosti** $\mathcal{S}$, je řešením následujícího optimalizačního problému.

$$\text{minimize } F_W(G,A) = \|I - GA\|_W^2 = tr\left[(I-GA)W(I-GA)^T\right]$$

Pro $W = I$ dostáváme řešením tohoto problému přibližnou inverzi **ve smyslu nejmenších čtverců** (least-squares approximate inverse, AI) složenou z řešení n problémů nejmenších čtverců.

$$\text{Minimize } F_I(G,A) = \|I - GA\|_F^2 = \sum_{i=1}^{n} \|e_i^T - \tilde{g}_i^T A\|_2^2, \tag{16.118}$$

kde

$$\tilde{g}_i^T, i = 1, \dots n$$

jsou řádky matice G. Analogicky můžeme formulovat i aproximaci inverze matice po sloupcích, uvažujeme-li minimalizační problém

$$\text{Minimize } F_{II}(G,A) = \|I - AG\|_F^2 = \sum_{i=1}^{n} \|e_i - Ag_i\|_2^2 \tag{16.119}$$

pro sloupce $g_i, i = 1, \dots n$ matice G.

Vyjádřeme řešení minimalizačního problému (16.118) jiným způsobem. Pozitivní definitnost matice W implikuje, že funkcionál $F_W(G,A)$ je nezáporný a jeho minima splňují vztah

$$(GAWA^T)_{ij} = (WA^T)_{ij},\ (i,j) \in \mathcal{S}. \tag{16.120}$$

To můžeme vidět z následujících úprav (platí $tr(AB) = \sum_i \sum_j a_{ij} b_{ji}$)

$$\begin{aligned} F_W(G) &= tr\left[(I - GA)W(I - GA)^T\right] \\ &= tr(W) - tr(GAW) - tr(WA^TG^T) + tr(GAWA^TG^T) \\ &= tr(W) - \sum_{i,j} g_{ij}\left[(AW)_{ji} + (WA^T)_{ij}\right] + tr(GAWA^TG^T) \end{aligned}$$

Použitím

$$\frac{\partial F_W(G)}{\partial g_{ij}} = 0,\ (i,j) \in \mathcal{S} \tag{16.121}$$

dostaneme

$$-(AW)_{ji} - (WA^T)_{ij} + (AWA^TG^T)_{ji} + (GAWA^T)_{ij},\ (i,j) \in \mathcal{S}, \tag{16.122}$$

což implikuje

$$(GAWA^T)_{ij} = (WA^T)_{ij},\ (i,j) \in \mathcal{S}. \tag{16.123}$$

Je-li matice A je symetrická a pozitivně definitní, pak metodu založenou řešení soustavy (16.120) s volbou $W = A^{-1}$ nazýváme **přímou blokovou metodou** (direct block method, DB). To tuto soustavu zjednodušuje na tvar

$$[GA]_{ij} = \delta_{ij},\ (i,j) \in \ \mathcal{S}.$$

Podotkněme, že problém kompatibility struktury řídkosti $\mathcal{S}$ a obecně soustavy (16.120) obvykle vede k dalším aproximacím v průběhu výpočtu.

Speciální tvar výpočtu výpočtu přibližné inverze založeného na minimalizační formulaci byl navržen v publikaci [20]. Sofistikovanější postup nazvaný SPAI dynamicky mění předepsanou strukturu řídkosti v iteračním cyklu [36], [84]. Tento postup řeší problémy nejmenších čtverců pro jednotlivé řádky nebo sloupce výše uvedenou minimalizací. Algoritmus, který vychází z (16.119) schématicky popíšeme v následujících krocích.

Algoritmus 16.8.1 SPAI algoritmus pro přibližnou inverzi matice
Input: *Matice* $A \in R^{n\times n}$.
Output: *Přibližná aproximace* $G = (g_1, \ldots, g_n)$ *matice* A^{-1}.

1. *Volba jednoduché počáteční struktury řídkosti* $\mathcal{S}$
2. **do until** *splnění zastavovacího kritéria*
3. *Paralelní řešení problémů* $\min \|e_i - Ag_i\|_2^2$, $i = 1, \ldots, n$ *s danou strukturou řídkosti.*
4. *Výpočet komponent rezídua* $\|Ag_i - e_i\|$
5. *Rozšíření struktur sloupců v* $\mathcal{S}$ *o pozice největších komponent rezídua*

ž. **end do until**

Jednoduchou strukturou řídkosti $\mathcal{S}$ může být například struktura diagonální. Problémy nejmenších čtverců počítané paralelně v kroku 3 algoritmu mají řádky určené sloupcovými komponentami těchto sloupců. Zastavovací kritérium obvykle zahrnuje nějaké vyhodnocení norem rezíduí počítaných sloupců matice G. Pozdější vylepšení zahrnovala například

přesnější výpočet rezídua [82] nebo volbu kvalitnější **počáteční struktury řídkosti** [90], [89], [32]. Přístup, který takto počítá jednotlivé sloupce je sice **procedurálně paralelní**, ale někdy může být obtížné distribuovat matici A tak, že všechny procesory mají efektivně k dispozici data potřebná pro dynamické změny struktury řídkosti $\mathcal{S}$.

Specifický postup založený na minimalizaci ve Frobeniově normě byl navržen pro výpočet **přibližné inverze Choleského faktoru** G_L symetrické a pozitivně definitní matice. Zvolme omezení ve formě předepsané struktury řídkosti Choleského faktoru $\mathcal{S}_{\mathcal{L}}$.

$$\bar{Z} = \arg\min_{G_L \in \mathcal{S}} F_I({G_L}^T, L) = \arg\min_{G_L \in \mathcal{S}_{\mathcal{L}}} \|I - {G_L}^T L\|_F^2, \text{ kde } A = LL^T.$$

Aplikací (16.123) na tento minimalizační problém (při značení $A \to L$) s $W = I$ dostaneme soustavu

$$(G_L L L^T)_{ij} = (L^T)_{ij},\ (i,j) \in \mathcal{S}_{\mathcal{L}}, \tag{16.124}$$

což dává při substituci A do tohoto vztahu

$$(G_L A)_{ij} = (L^T)_{ij},\ (i,j) \in \mathcal{S}_{\mathcal{L}}. \tag{16.125}$$

Všimněme si, že struktura řídkosti $\mathcal{S}_{\mathcal{L}}$ je **dolní trojúhelníková** a matice L^T je horní trojúhelníková. Známe-li diagonální prvky faktoru L, dostaneme G_L z rozpisu

$$(G_L A)_{ij} = \begin{cases} (L^T)_{ij} & i = j, \\ 0 & i \neq j. \end{cases} \tag{16.126}$$

Neznáme-li diagonální prvky faktoru, což je typická situace, tento postup spočítá obecně jinou aproximaci inverze Choleského faktoru $\hat{G}_L$ ze vztahu

$$(\hat{G}_L A)_{ij} = \delta_{ij}, (i,j) \in \mathcal{S}_{\mathcal{L}} \tag{16.127}$$

a pak položí $G_L = D\hat{G}_L$ pro diagonální matici D splňující

$$(D\hat{G}_L A \hat{G}_L^T D)_{ii} = 1, \quad , i = 1, \ldots, n. \tag{16.128}$$

V řadě případů je tento postup velmi efektivní a dá se rozšířit i pro aproximaci faktorů rozkladu nesymetrické matice A. Zde se ale může narazit na další problémy obecně spjaté s nesymetrickými maticemi.

16.8.5.2 Iterační výpočet sloupců aproximace inverze matice

Postupem, který nabízí efektivní paralelizaci, je použít jednoduchou (například stacionární) **iterační metodu** pro výpočet jednotlivých sloupců c_i maticové inverze řešením soustav tvaru

$$Ac_i = e_i.$$

To bylo navrženo v publikaci Chow and Saad, 1994. Výsledné faktory ale nemusí poskytnout efektivní předpodmínění. Proceduru je možné modifikovat způsobem podobně jako je z formálního pohledu Gauss-Seidelova iterační metoda modifikací Jacobiho metody pro řešení soustav lineárních algebraických rovnic tak, že již vypočítané sloupce jsou použity ve výpočtu dalších sloupců. Tím na jedné straně omezujeme potenciální paralelismus, ale tento přístup někdy vede k výpočtu lepších aproximací.

16.8.5.3 Faktorizované inverze počítané bikonjugací

Uvažujme opět aproximaci inverze matice ve tvaru neúplného rozkladu. Obecně máme dvě možnosti jak tento rozklad spočítat. První možností je použít neúplnou LU faktorizaci

$$A \approx LU \tag{16.129}$$

a potom získané trojúhelníkové faktory **invertovat**. Invertovat je můžeme přesně nebo i jen přibližně. Jsou-li neúplné faktory L a U řídké, pak i jejich **úplné inverze** mohou být ještě řídké a jejich přesná inverze může být efektivní. Obecně tedy pak můžeme psát předpodmínění ve tvaru

$$M^{-1} = U^{-1}L^{-1} \tag{16.130}$$

a i takto lze někdy získat kvalitní předpodmínění, viz [3].

Jinou možností je počítat přibližné faktory bez toho, aby se napřed získaly aproximace přímých faktorů L a U matice A. V následujícím textu popíšeme základní postup bez toho, abychom specifikovali, které prvky odvrhujeme, kde je možná celá řada variant.

Je-li A silně regulární, pak existují její jednoznačně určené faktory L, D a U takové, že L je dolní trojúhelníková matice s jednotkovou diagonálou, U je horní trojúhelníková matice s jednotkovou diagonálou a D je diagonální matice. Pak lze pro $Z = L^{-T}$ a $W = U^{-1}$ psát

$$A^{-1} = WD^{-1}Z^T \quad \Rightarrow \quad A = Z^{-T}DW^{-1}, \tag{16.131}$$

což lze přepsat na vztah

$$Z^TAW = D. \tag{16.132}$$

Předpokládejme nyní, že A je symetrická a pozitivně definitní. Pak jsou sloupce matic Z a $W \equiv Z$ vzájemně ortogonální v následujícím skalárním součinu

$$\langle \cdot\,,\,\cdot \rangle_A \tag{16.133}$$

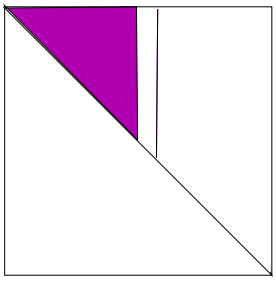
s maticí A. Algoritmus, kterým lze Z spočítat je **Gram-Schmidtova ortogonalizace** v tomto skalárním součinu. Tato procedura se někdy nazývá A-orthogonalizace a její přímočaré zobecnění na nesymetrický případ LU rozkladu (s menšími teoretickými zárukami) se také nazývá **bikonjugace**. První algoritmy, které počítaly inverzní faktory tímto způsobem byly vyvíjeny od poloviny dvacátého století [120], [72]. Podobně jako pro Gram-Schmidtův proces ve standardním skalárním součinu existuje i zde více algoritmických způsobů výpočtu, které se mohou drasticky lišit po numerické stránce. První algoritmické schéma je následující.

Algoritmus 16.8.2 Inverzní rozklad A-orthogonalizací
Input: *Řídká SPD matice $A \in R^{n\times n}$.*
Output: *Jednotková horní trojúhelníková matice Z, pro kterou platí $A^{-1} = ZD^{-1}Z^T$.*

1. **for** $i = 1 : n$ **do**
2. $\quad z_i = e_i - \sum_{k=1}^{i-1} z_k \frac{e_i^T A z_k}{z_k^T A z_k}$
3. $\quad d_i = z_i^T A z_i$

4. **end** i
5. Polož $Z = [z_1, \ldots, z_n]$

Pořadí operací v Algoritmu 16.8.2 je znázorněno na následujícím obrázku. V i-tém kroku algoritmu je spočítán sloupec z_i matice Z a diagonální prvek d_i matice D. Pro výpočet těchto prvků se použije přístup k vybarvené části matice. Tento způsob výpočtu faktorů nazýváme **sloupcový** (levostranný) algoritmus.



Z

Snadno nahlédneme, že platí

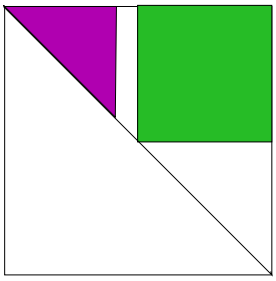
Lemma 16.8.1 *Sloupce faktoru Z z Algoritmu 16.8.2 splňují*

$$d_i = e_k^T A z_k \equiv z_k^T A z_k, 1 \leq k \leq n. \tag{16.134}$$

Tento způsob se nazývá **stabilizovaný** výpočet diagonálních prvků. Jeho název je motivován platností vztahu

$$z_k^T A z_k > 0 \tag{16.135}$$

pro libovolné odvrhování mimodiagonálních prvků matice Z pro A symetrickou a pozitivně definitní. Diagonální prvky v Algoritmu 16.8.2 mohou být ale spočítány i jinak. Matematicky ekvivalentní výpočet $d_i = e_i^T A z_i$, který je v přesné aritmetice ekvivalentní stabilizovanému výpočtu nemusí vztah (16.135) obecně splňovat, ale v případě speciálních matic jako jsou M-matice a H-matice jej splňuje. Neúplná A-orthogonalizace jako postup ke konstrukci předpodmínění se nazývá AINV [22]. Jiné schéma AINV algoritmu výpočtu, které nazveme jeho **submaticovou** (pravostrannou) variantou, uvádíme schématicky níže. Zeleně vybarvená oblast odpovídá oblasti, kde jsou prvky Z aktualizovány, ale nejsou ještě dopočteny. Fialová oblast je už definitivně spočítána a používá se k postupnému vyjadřování sloupců Z a diagonálních prvků tvořících matici D.



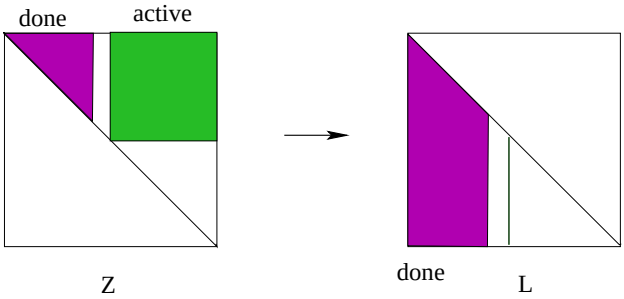
Z

AINV procedura může být použita nejenom samostatně pro výpočet předpodmínění ve faktorizovaném tvaru, ale může případně sloužit i jako pomocný algoritmus k aproximaci bloků v blokových rozkladech, viz [13], [34].

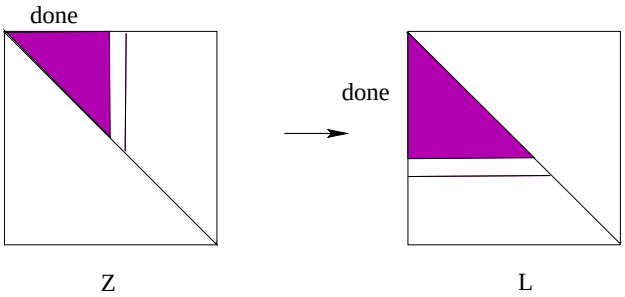
16.8.5.3.1 Vedlejší efekt A-orthogonalizace: RIF Výše uvedená neúplná faktorizace AINV může být také chápána jako postup, kterým jsou spočítány přibližné Choleského faktory jiným způsobem než standardním rozkladem odvozeným z Gaussovy eliminace. Připomeňme, že v případě úplného rozkladu jsou Choleského faktory (faktory LDL^T rozkladu) jednoznačně určeny. Kroky konstrukce jsou následující:

- Nalezni rozklad $Z^TAZ = D$, kde Z je jednotková horní trojúhelníková matice a D je diagonální.
- Faktor L Choleského rozkladu $A = LDL^T$ je v přesné aritmetice dán vztahem $L = AZD^{-1}$ a může být snadno vypočítán z mezivýsledků inverzního rozkladu AINV jako multiplikátory lineárních kombinací pro sloupce matice Z. Tento postup, kde se výpočet Z a L prolíná, se někdy nazývá robustní neúplný rozklad (RIF).

Následující dva obrázky znázorňují datové toky při výpočtu faktoru RIF, ale do detailů se zde nebudeme pouštět.



Pravostranný algoritmus



Levostranný algoritmus

Choleského rozklad získaný RIF algoritmem neposkytuje příliš mnoho paralelismu při výpočtu, ale může pomoci řešit jinak obtížně řešitelné soustavy lineárních algebraických rovnic.

16.8.5.3.2 Jiné způsoby získání aproximace matice A^{-1} Zajímavé schéma výpočtu inverze, které je možné použít pro výpočet neúplných faktorů a které založené na metodě ovroubení [141] je založené na následující ekvivalenci. Předpokládáme, že matice A je symetrická, a pro jednoduchost, že je i pozitivně definitní.

$$\begin{pmatrix} Z^T & \\ -y^T & 1 \end{pmatrix} \begin{pmatrix} A & v \\ v^T & \alpha \end{pmatrix} \begin{pmatrix} Z & -y \\ & 1 \end{pmatrix} = \begin{pmatrix} D & \\ & \delta \end{pmatrix}.$$

Roznásobením získáme maticovou rovnost

$$\begin{pmatrix} Z^T AZ & -Z^T Ay + Z^T v \\ -y^T AZ + v^T Z & y^T Ay - v^T y - y^T v + \alpha \end{pmatrix} = \begin{pmatrix} D & \\ & \delta \end{pmatrix}.$$

Rozepsáním rovností získáme vztahy

$$\begin{aligned} Z^T Ay &= Z^T v \\ \delta &= y^T Ay - 2\, v^T y + \alpha \end{aligned}$$

Po inicializaci faktoru Z jednoprvkovou jednotkovou maticí a položením $D_{11} = a_{11}$ se ovroubením postupně počítají vektory y a skaláry δ. Samotná implementace ve formě rozkladu je poměrně sofistikovaná. Důvodem je, že konstruujeme řídkou matici Z postupným přidáváním sloupců, ale ve výpočtu zároveň násobíme řídké vektory její transpozicí, kde by bylo vhodné mít i Z^T uloženou po sloupcích.

Existují i další postupy jak získat nebo aplikovat přibližnou inverzi a výhodně využít možnost paralelního výpočtu. Ty jsou založeny například na Sherman-Morrison-Woodburyho aktualizační formuli.

16.8.5.4 Globální maticové iterace

Nerozložená inverze matice G matice A může být iteračně aproximována pomocí Schulzových iterací [144] založených na následujícím schématu

$$G_{i+1} = G_i(2I - AG_i),\ i = 1, \ldots$$

Algoritmus je odvozen z Newtonovy-Raphsonovy iterace, která v jednorozměrném případě hledá skalár p, pro který je hodnota dané funkce f rovna nule. Odvození algoritmu si nyní ukážeme.

Uvažujme rovnici tečny pro funkci f v bodě p_n v následující obecné formě

$$y = f'(p_n)p_n + b. \tag{16.136}$$

Tečna prochází bodem $(p_n, f(p_n))$, což můžeme zapsat

$$f(p_n) = f'(p_n)p_n + b. \tag{16.137}$$

To dává

$$b = f(p_n) - f'(p_n)p_n \tag{16.138}$$

a tedy máme funkci v obecném bodě x

$$y = f'(x)x + f(p_n) - f'(p_n)p_n. \tag{16.139}$$

Předpokládejme **nulový bod funkce v** p_{n+1} a dostáváme

$$0 = f'(p_{n+1})p_{n+1} + f(p_n) - f'(p_n)p_n \tag{16.140}$$

a tedy

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}. \tag{16.141}$$

Uvažujme nyní funkci inverze

$$f(x) = 1/x - a$$

a máme pak

$$p_{n+1} = p_n - \frac{1/p_n - a}{-1/p_n^2} = ap_n^2 = 2p_n - ap_n^2. \tag{16.142}$$

Tento vztah implikuje maticové zobecnění Newton-Raphsonových iterací nazvané Schulzovou iterací.

16.8.6 Polynomiální předpodmínění

Myšlenka užitečnosti polynomiálního předpodmínění je založena na faktu, že aplikace předpodmínění v iteračních metodách se dá provést často velmi dobře paralelizovatelnými součiny matice-vektor. Uvažujme předpodmínění soustavy rovnic v následujícím tvaru

$$M^{-1}Ax = M^{-1}b. \tag{16.143}$$

Jednou z možností, jak vybrat předpodmínění, je uvažovat vyjádření přibližné inverze ve formě maticového polynomu $s(A)$ stupně k. Přirozenou motivací k této úvaze je fakt, že inverze matice může být vyjádřena pomocí charakteristického polynomu $p(\lambda) = det(A - \lambda I)$ matice A. Cayley-Hamiltonova věta implikuje vztah

$$p_n(A) \equiv \sum_{j=0}^{n} \beta_j A^j = 0. \tag{16.144}$$

Pro regulární matici A máme $\beta_0 = (-1)^n \det(A) \neq 0$ a po vynásobení maticí A^{-1} získáme

$$A^{-1} = -\frac{1}{\beta_0} \sum_{j=1}^{n} \beta_j A^{j-1}. \tag{16.145}$$

Předpodmínění pak může být vyjádřeno zkrácením charakteristického polynomu, tedy omezením na několik jeho členů

$$M^{-1} = s(A) = \sum_{j=0}^{k} c_j A^k. \tag{16.146}$$

Myšlenka polynomiálního předpodmínění byla poprvé zmíněna v publikaci [29], kde byla předpodmiňována Richardsonova iterační metoda. Další vývoj byl navázán v rámci rozvoje vektorových a obecně paralelních výpočtů viz [149], [150], neboť aplikace polynomu v rámci iteračních metod je bohatá na **násobení vektorů maticemi**, což jsou obvykle dobře vektorizovatelné nebo paralelizovatelné ioperace. Neboť A a $s(A)$ vzájemně komutují, pak aplikace polynomu může využívat Hornerova schématu nebo jeho paralelnějších verzí, jejichž vývoj byl iniciován v publikaci [68].

Většina praktických polynomiálních předpodmínění je spojena se symetrickými maticemi, především s těmi, co jsou také pozitivně definitní. Ačkoli polynomiální předpodmínění pro nesymetrické matice mohou být také konstruována, získat s nimi efektivní předpodmíněné iterační metody může být mnohem obtížnější [112], [113], [143].

16.8.6.1 Polynomiální předpodmínění a metoda sdružených gradientů

Metoda sdružených gradientů pro řešení soustav lineárních algebraických rovnic se symetrickými a pozitivně definitníémi maticemi hledá ve svém $k+1$-ním kroku aproximaci řešení x_{k+1} ve tvaru

$$x_{k+1} = x_0 + \mathcal{P}_k(A)r_0,\ k = 0, \ldots. \tag{16.147}$$

Polynom $\mathcal{P}_k(A)$ je přitom zvolen tak, že A-norma chyby daná vztahem

$$||x_{k+1} - x^*||_A = \sqrt{(x_{k+1} - x^*)^T A(x_{k+1} - x^*)} \tag{16.148}$$

je minimální mezi všemi polynomy stupně nejvýše k pro které $\mathcal{P}_k(0) = 1$. Důvody, proč polynomiální předpodmínění této metody může být užitečné, i když je to jen transformace iterací **dalším polynomem** nad rámec polynomu metody jsou například následující.

- **Počet iterací** může být menší, přestože celková složitost vzroste. V některých případech je minimalizace počtu iterací velmi důležitá, například je-li matice A dána pouze implicitně v bezmaticových (**matrix-free**) implementacích.
- Souvisejícím cílem je, že polynomiální předpodmínění může **zmenšit počet skalárních součinů**, které mohou být problematické na paralelních počítačových architekturách.
- Polynomiální předpodmínění obvykle potřebuje **mnohem méně paměti** než jiná předpodmínění a jeho implementace je velmi **přímočará**.

Podobné důvody motivují i polynomiální předpodmínění jiných Krylovovských metod.

16.8.6.2 Předpodmínění Neumannovými řadami

Neumannovou řadou matice $G \in R^{n\times n}$ nazveme řadu

$$\sum_{j=0}^{+\infty} G^j. \tag{16.149}$$

Platí následující věta

Věta 16.8.1 *Neumannova řada matice $G \in R^{n\times n}$ konverguje právě tehdy, platí-li*

$$\rho(G) \equiv \max\{|\lambda_1|, \ldots, |\lambda_n|\} < 1.$$

V tomto případě také platí

$$(I - G)^{-1} = \sum_{j=0}^{+\infty} G^j. \tag{16.150}$$

Poznamenejme, že $\rho(G) < 1$ je splněno, jestliže nějaká multiplikativní norma $|||G|||$ matice G je menší než 1. Uvažujme nyní následující štěpení matice A dané regulární maticí M_1.

$$\omega A = M_1 - R, \tag{16.151}$$

kde $M_1, R \in R^{n\times n}$ a ω je zatím neurčený nenulový skalární parametr. Pak můžeme psát

$$\omega A = M_1(I - M_1^{-1}R) = M_1(I - G), \tag{16.152}$$

kde

$$G = I - \omega M_1^{-1}A.$$

Obvykle lze snadno najít takový parametr $\omega > 0$, aby

$$\rho(G) = \rho(I - \omega M_1^{-1}A) < 1.$$

Pak ale můžeme psát

$$A^{-1} = \omega\,(I - G)^{-1}M_1^{-1} = \omega\left(\sum_{j=0}^{+\infty} G^j\right) M_1^{-1} \tag{16.153}$$

Předpodmínění, které aproximuje A^{-1} pak získáme tak, že uvažujeme pouze **konečný počet** $k+1$ členů tohoto rozvoje inverze matice A. Inverzní předpodmínění M^{-1} je pak vyjádřeno **zkrácenou** Neumannovou řadou. Praktické rady k použití tohoto předpodmínění je možné najít například v [119].

$$M^{-1} = \omega\left(\sum_{j=0}^{k} G^j\right) M_1^{-1}. \tag{16.154}$$

Matice M takto definována je hledanou maticí vyjadřující předpodmínění, přičemž máme rovnou aproximovánu její inverzi. Z toho vztahu zároveň dostaneme

$$M^{-1}A = \left(\sum_{j=0}^{k} G^j\right) \omega\, M_1^{-1}A = \left(\sum_{j=0}^{k} G^j\right)(I - G) = (I - G^{k+1}). \tag{16.155}$$

Tímto způsobem jsme vyjádřili **rozdíl** mezi $M^{-1}A$ a jednotkovou maticí, který, jak je vidět se zmenšuje se vzrůstajícím počtem členů k.

Polynomiální předpodmínění ve tvaru zkrácené Neumannovy řady pro symetrickou a pozitivně definitní matici A můžeme aplikovat zleva. Víme, že je-li M_1 symetrická a pozitivně definitní, pak se dá totiž ukázat, že $M^{-1}A$ je samosdružená vzhledem ke skalárnímu

součinu $(.\,,\,.)_{M_1}$. Pak jsou například volby $M_1 = I$ nebo $M_1 = D_A$ nebo i M_1 rovné blokově diagonální části matice A v rámci metody sdružených gradientů korektní.

Zobecnění tohoto předpodmínění z článku Dubois, Greenbaum and Rodrigue (1979) další parametrizací zkrácené Neumannovy řady pro $(I-G)^{-1}$ lze nalázt v publikaci Johnson, Micchelli and Paul, 1983. Konkrétně zde byla navržena aproximace

$$I + \gamma_1 G + \gamma_2 G^2 + \ldots \gamma_k G^k \tag{16.156}$$

a dodatečné stupně volnosti byly použity k zlepšení jejích vlastností.

16.8.6.3 Předpodmínění založené na Čebyševových polynomech

Uvažujme následující optimalizační problém pro symetrickou a pozitivně definitní matici A

$$||I - s(A)A|| = \max_{\lambda \in \sigma(A)} |1 - \lambda s(\lambda)|, \tag{16.157}$$

kde s je polynom nezáporného stupně nejvýše **stupně** $k-1$. Tato úloha je příliš obtížná, ale může být relaxována (nahrazena jednodušší úlohou) hledáním polynomu, který tuto maximalizaci uvažuje na nějaké množině E, která zahrnuje spektrum matice A formulovanou jako

$$\max_{\lambda \in E} |1 - \lambda s(\lambda)|, \text{ E zahrnuje spektrum matice } A \tag{16.158}$$

mezi všemi polynomy s nejvýše daného stupně $k-1$, $k \geq 1$, tedy $s \in P_{k-1}$. Je-li A symetrická a pozitivně definitní, množinou E je interval v $\mathbb{R}^+$. Označíme-li tento interval

$$[a,\, b], \tag{16.159}$$

pak se problém redukuje na hledání polynomu s, který minimalizuje výsledné maximum a splňuje tedy

$$s = \min_{p \in P_{k-1}} \max_{\lambda \in [a,\, b]} |1 - \lambda p(\lambda)|. \tag{16.160}$$

Je známo, že řešení tohoto problému může být vyjádřeno pomocí **škálovaných a posunutých Čebyševových polynomů** prvního druhu

$$T_0(\lambda), T_1(\lambda), \ldots. \tag{16.161}$$

V případě matice A symetrické a pozitivně definitní můžeme tyto polynomy explicitně zkonstruovat následujícím způsobem, kde δ a θ jsou rovny, po řadě, hodnotám $(a+b)/2$ and $(b-1)/2$.

$$\begin{aligned}
\sigma_0 &= 1,\, \sigma_1 = \theta/\delta,\, \sigma_{k+1} = 2\theta/\delta\sigma_k - \sigma_{k-1} \\
T_0(\lambda) &= 1/\theta,\, T_1(\lambda) = (4\theta - 2\lambda)/(2\theta^2 - \delta^2) \\
T_k(\lambda) &= \frac{2\sigma_k}{\delta\sigma_{k+1}} + \frac{2\sigma_k(\theta - \lambda)}{\sigma_{k+1}\delta} T_{k-1}(\lambda) - \frac{\sigma_{k-1}}{\sigma_{k+1}} T_{k-2}(\lambda)
\end{aligned}$$

Zvolíme-li $\lambda_1 = a, \lambda_n = b$ pak lze ukázat [94], že předpodmíněná matice

$$s(A)A \tag{16.162}$$

má minimální číslo podmíněnosti mezi všemi takovými maticemi, kde polynom $s(A)$ má předepsaný stupeň nejvýše $k-1$. Čebyševovo polynomiální předpodmínění lze jednoduše použít v rámci metody sdružených gradientů transformací rezíduí r_i vztahem

$$r_i = T_i(A)r_0,\ i = 1, \ldots, n. \tag{16.163}$$

Je-li A symetrická a indefinitní, i pak lze navrhnout polynomiální předpodmínění. Uvažujme takovou matici se spektrem uvnitř sjednocení dvou intervalů

$$[a,\, b] \cup [c,\, d],\ -\infty < a \leq b < 0 < c \leq d < +\infty, \tag{16.164}$$

stejné délky. To jest, když

$$b - a = d - c. \tag{16.165}$$

Řešení odpovídajícího **polynomiálního minimaxového problému** lze vyjádřit explicitně pomocí de Boor-Riceových polynomů [42] následujícím způsobem

$$P_m(\lambda) = \frac{1}{\lambda}\left(1 - \frac{T_k(\Phi(\lambda))}{T_k(\Phi(0))}\right),\ \Phi(\lambda) = 1 + \frac{2(\lambda - b)(\lambda - c)}{ad - bc}. \tag{16.166}$$

Stupeň polynomu v polynomiálním předpodmínění, které monotónně zobrazuje oba intervaly na interval $[-1,\, 1]$ je pak

$$k = \lfloor m/2 \rfloor. \tag{16.167}$$

Poznamenejme, že m je vždy sudé.

Nejsou-li intervaly stejné délky, jeden z nich může být prodloužen, aby tomu tak bylo. V tomto případě ale výsledné předpodmínění může být nepříliš efektivní. Nicméně, existují algoritmy, které počítají polynomiální předpodmínění i v případě intervalů nestejné délky iteračně (Remezův algoritmus, [9]). Jinou možností pro polynomiální předpodmínění v tomto případě je použít Grcarovy polynomy nebo **dvojúrovňové polynomy** [83], [9], [8], [74], které transformují spektrum předpodmíněné matice na hodnoty v blízkosti hodnot -1 a 1 na reálné ose. Předpodmínění pomoví dvojúrovňových polynomů pak často funguje lépe než když intervaly obsahující spektrum **nemají stejnou délku.** Poslední možností, kterou zmíníme, jsou polynomiální předpodmínění adaptivně aktualizovaná podle toho, jak se vylepšují odhady spektra matice [70], [126].

Podobně jako v případě metody sdružených gradientů transformace spektra taková, že jsou vlastní čísla v klastrech, **nemusí implikovat rychlejší konvergenci metody** neboť její charakterizace je složitější. Pro kvalitu předpodmíněných iteračních metod je i v tomto případě důležitý vliv aritmetiky konečné přesnosti na výpočet [104].

16.8.6.4 Předpodmínění založené na polynomech nejmenších čtverců

Kvalita polynomiálních předpodmínění založených na Čebyševových polynomech pro matice symetrické a pozitivně definitní silně závisí na zvolených/odhadnutých intervalech pro spektrum matice. Přímočaré použití Geršgorinovy věty nemusí přitom stačit a existují další a sofistikovanější metody ke zlepšení konvergence metod polynomiálním předpodmíněním.

Jeden z takových návrhů je založen na polynomech nejmenších čtverců [94]. Uvažujme následující skalární součin dvou funkcí p a q na reálné ose

$$\langle p, q\rangle = \int_a^b p(\lambda)q(\lambda)w(\lambda)d\lambda, \tag{16.168}$$

kde $w(\lambda)$ je nezáporná **váhová funkce na** (a, b)**.** Odpovídající normu

$$||p||_w = \int_a^b |p(\lambda)|^2 w(\lambda)d\lambda, \tag{16.169}$$

budeme nazývat w-normou. Budeme hledat předpodmínění $s(A)$ ve tvary polynomu matice A na intervalu reálné osy, který obsahuje spektrum matice. Speciálně, polynom s bude řešením minimalizačního problému

$$\min_{p\in P_{k-1}} ||1 - \lambda p(\lambda)||_w. \tag{16.170}$$

Předpokládejme, že matice A je symetrická a pozitivně definitní. Zvolíme-li například váhovou funkci $w \equiv 1$ (Legendreova váhová funkce) nebo funkci

$$w(\lambda) = (b - \lambda)^\alpha(\lambda - a)^\alpha, \alpha > 0, \beta \geq -\frac{1}{2}, \tag{16.171}$$

(Jacobiho váhová funkce), polynom $s(A)$ lze explicitně vyjádřit. Jestliže navíc

$$\alpha - 1 \geq \beta \geq -\frac{1}{2}, \tag{16.172}$$

pak má $s(A)A$ všechna vlastní čísla reálná a větší než nula a předpodmínění se dá aplikovat v rámci metody sdružených gradientů.

Několik prvních polynomů nejmenších čtverců $s_k(\lambda)$ stupně nejvýše k je for $k = 1, 2, 3$, $\alpha = 1/2. \beta = -1/2$.

$$\begin{aligned} s_0(\lambda) &= \frac{4}{3} \\ s_1(\lambda) &= 4 - \frac{16}{5}\lambda \\ s_2(\lambda) &= \frac{2}{7}(28 - 56\lambda + 32\lambda^2)). \end{aligned}$$

Odvození polynomů může být také založeno na vztahu takzvaných jádrových polynomů aplikovaných na reziduální polynom

$$R_k(\lambda) = 1 - \lambda s_k(\lambda) \tag{16.173}$$

nebo použitím tříčlenné polynomiální rekurence v případě některých váhových funkcí. Oba dva způsoby je možné najít v publikaci [150]. Jiným způsobem odvození je explicitní řešení soustavy normálních rovnic

$$\langle 1 - \lambda s_k(\lambda), \lambda Q_j(\lambda)\rangle_w,\ j = 0, \ldots, k - 1, \tag{16.174}$$

kde Q_j je libovolná báze prostoru polynomů stupně nejvýše k. V případě, kdy je matice A symetrická a pozitivně definitní je možné odvodit polynomy nejmenších čtverců aproximací sjednocení dvou intervalů v nichž se nachází spektrum matice a modifikovat váhovou funkci. Publikace [137] navrhuje použití následující váhové funkce

$$w(\lambda) = \begin{cases} w_1 & for\ \lambda \in [a,\, b] \\ w_2 & \lambda \in [c,\, d] \\ 0 & otherwise \end{cases} . \tag{16.175}$$

16.8.7 Další předpodmínění pro paralelní počítačové architektury

Zde budou zmíněny n2kter0 další druhy takových předpodmínění.

16.8.7.1 Předpodmínění po elementech

Element je tradičně nazývána matice A_e, která je určena svými řádkovými a sloupcovými indexy a platí-li

$$A = \sum_{e=1}^{n_e} A_e. \tag{16.176}$$

Abychom sečetli příspěvky of jednotlivých elementů, lze pracovat s jednotlivými elementy s lokálními indexy a použít operaci **extend-add** k sečtení těchto příspěvků matice A. Elementy přirozeně vznikají například jako lokální matice v metodě konečných prvků.

Jedna z možností jak předpodmínit soustavy rovnic s maticemi vyjádřenými jako součet elementů je navrhnout předpodmínění také jako součet, ve formě předpodmínění jednotlivých elementů. To samozřejmě umožní efektivní paralelismus. Nejjednodušším návrhem je vybrat předpodmínění jako **součet diagonálních matic**

$$M_e = \sum_{e=1}^{n_e} diag(A_e). \tag{16.177}$$

Sofistikovanější přístup navržený v publikaci Hughes, Levit, Winget, 1983 a formulovaný pro matici, která je symetricky škálovaná svou diagonálou (Jacobiho škálování) definuje předpodmínění po elementech následujícím způsobem:

$$M = \sqrt{W}\Pi_{e=1}^{n_e} L_e \Pi_{e=1}^{n_e} D_e \Pi_{e=n_e}^{1} L_e^T \sqrt{W}, \tag{16.178}$$

kde

$$W = diag(A),\ I + \sqrt{W}^{-1}(A_e - diag(A_e)\sqrt{W}^{-1} = L_e D_e L_e^T,\ e = 1, \ldots, n_e \tag{16.179}$$

Jiný postup byl navržen Gustafssonem a Linskogem, 1986. Zde se navrhuje položit

$$M = \sum_{e=1}^{n_e} (\hat{L}_e + D_e) \left(\sum_{e=1}^{n_e} D_e) \right)^{-1} \sum_{e=1}^{n_e} (\hat{L}_e^T + D_e) \tag{16.180}$$

pro

$$A_e = (L_e + D_e)D_e^+(L_e^T + D_e),\ e = 1, \ldots, n_e, \tag{16.181}$$

kde D_e^+ je pseudoinverze matice D_e.

17

Elementární operace s řídkými maticemi a vektory

V této kapitole probereme některé základní algebraické operace, kde vystupují řídké matice a které jsou někdy užitečné jako výpočetní bloky v složitějších operacích s řídkými maticemi. Konkrétně si uvedeme, jak přistupovat k násobení vektoru nebo matice maticí.

17.1 Násobení hustého vektoru řídkou maticí

Uvažujme operaci $y = Ax$, kde $A \in R^{m \times n}$ je řídká matice uložená v CSR nebo CSC formátu a $x \in R^n$ je hustý vektor. Tato operace je jednou ze základních komponent velkého množství iteračních metod řešení soustav lineárních rovnic. Předpokládáme, že matice je uložena v CSR nebo CSC formátu. Pro řídkou matici A v CSR formátu je postup zde uveden jako Algoritmus 17.1.1.

Algoritmus 17.1.1 Násobení $y = Ax$ hustého vektoru x řídkou maticí A v CSR formátu
Input: *Řídká matice $A \in R^{m \times n}$ uložená v CSR formátu v polích (ia,ja,aa), hustý vektor $x \in R^n$.*
Output: *Hustý vektor $y \in R^m$.*

1. **for** $i = 1 : m$ **do**
2. $\quad y(i) = 0$
3. $\quad$ **for** $j = ia(i) : ia(i+1) - 1$ **do**
4. $\quad\quad y(i) = y(i) + aa(ja(j)) * x(ja(j))$
5. $\quad$ **end** j
6. **end** i

Snadno nahlédneme, že složitost tohoto algoritmu je $O(m+|A|)$. V případě, že je řídká matice v CSC formátu, postup o stejné složitosti je uveden jako Algoritmus 17.1.2

Algoritmus 17.1.2 Násobení $y = Ax$ hustého vektoru x řídkou maticí A v CSC formátu
Input: *Řídká matice $A \in R^{m \times n}$ uložená v CSC formátu v polích (ia,ja,aa), hustý vektor $x \in R^n$.*
Output: *Hustý vektor $y \in R^m$.*

1. **for** $i = 1 : m$ **do**
2. $\quad y(i) = 0$
3. **end** i
4. **for** $i = 1 : m$ **do**
5. $\quad$ **for** $j = ia(i) : ia(i+1) - 1$ **do**
6. $\quad\quad y(ja(j)) = y(ja(j)) + aa(j) * x(i)$
7. $\quad$ **end** j
8. **end** i

17.2 Násobení řídkého vektoru řídkou maticí

V této sekci ukážeme, jak násobit řídký vektor řídkou maticí, což je stále ještě velmi jednoduchá operace. Oba algoritmy, které uvedeme, používají vektory délky řádkové nebo sloupcové dimenze matice, což může být v některých situacích, například v paralelních implementacích, nevhodné, ale lze to za cenu mírnějšího navýšení složitosti obvykle obejít.

Algoritmus 17.2.1 Násobení $y = Ax$ řídkého vektoru x řídkou maticí A v CSR formátu
Input: *Řídká matice $A \in R^{m \times n}$ uložená v CSR formátu v polích (ia,ja,aa), řídký vektor $x \in R^n$ o nx nenulových komponentách uložený v polích (jx,ax).*
Output: *Řídký vektor $y \in R^m$ o ny nenulových komponentách uložený v polích (jy,ay).*

1. **for** $i = 1 : n$ **do**
2. $\quad wn01(i) = 0$
3. **end** i
4. $ny = 0$
5. **for** $i = 1 : nx$ **do**
6. $\quad wn01(jx(i)) = i$
7. **end** i
8. **for** $i = 1 : m$ **do**
9. $\quad value = 0$
10. $\quad$ **for** $j = ia(i) : ia(i+1) - 1$ **do**
11. $\quad\quad$ **if** $wn01(ja(j)) \neq 0$ **then**
12. $\quad\quad\quad value = value + aa(ja(j)) * ax(wn01(ja(j)))$
13. $\quad\quad$ **end if**
14. $\quad$ **end** j
15. $\quad$ **if** $value \neq 0$ **then**
16. $\quad\quad ny = ny + 1$

```
17.         jy(ny) = i
18.         ay(ny) = value
19.     end if
20. end i
```

Je-li matice A uložena v CSC formátu, základní jednoduché schéma, které pro uložení výsledku ve vektoru y nejprve použije vektor dimenze m a poté jej zkomprimuje, je uvedeno jako Algoritmus 17.2.2.

Algoritmus 17.2.2 Násobení $y = Ax$ řídkého vektoru x řídkou maticí A v CSC formátu
Input: *Řídká matice $A \in R^{m\times n}$ uložená v CSC formátu v polích (ia,ja,aa), řídký vektor $x \in R^n$ o nx nenulových komponentách uložený v polích (jx,ax).*
Output: *Řídký vektor $y \in R^m$ o ny nenulových komponentách uložený v polích (jy,ay).*

```
1. for i = 1 : nx do
2.     for j = ia(jx(i)) : ia(jx(i) + 1) − 1 do
3.         y(ja(j)) = y(ja(j)) + aa(j) * x(i)
4.     end j
5. end i
6. ny = 0
7. for i = 1 : m do
8.     if y(i) ≠ 0 then
9.         ny = ny + 1
10.         jy(ny) = i
11.         ay(ny) = y(ja(j))
12.     end if
13. end i
```

17.3 Násobení řídké matice řídkou maticí

Násobení řídké matice jinou řídkou maticí je mírně důmyslnější, než byly postupy v předcházejícím textu pro násobení vektorů. Kdybychom uvažovali násobení řídkých matic jen jako n opakování násobení řídkého vektoru maticí, byla by výsledná složitost příliš vysoká. Zde zmíníme dva možné přístupy k násobení řídkých matic.

Zapišme si operaci $C = AB$ s obecně řídkými maticemi, kde $A \in R^{m\times p}$, $B \in R^{p\times n}$ a $C \in R^{m\times n}$. Pro efektivní násobení řídkých matic je zapotřebí předpokládat konkrétní způsob uložení těchto matic tak, aby operace mohla být efektivní. Libovolná kombinace formátů uložení matic A a B nemusí toto splňovat. Budeme-li, například, přímočaře počítat součin matic podle vzorce

$$c_{ij} = A_{i*}B_{*j},\ i, j \in \{1, \ldots, n\}, \tag{17.1}$$

bez toho, abychom dopředu věděli, které prvky matice $C = (c_{ij})_{i,j\in\{1,\dots,n\}}$ jsou nenulové, počet operací bude nutně řádu $\Theta(n^2)$. Obvykle musíme provést nejdříve celou operaci symbolicky, to jest, nejprve se zjistí velikost $|C|$ (počet jejích nenulových prvků), aby mohl být pro C vyhrazen (alokován) paměťový prostor a potom se provede vlastní násobení s numerickými hodnotami.

Předpokládejme nejprve, že obě matice A a B jsou uloženy po řádcích v CSR formátech. V tomto případě je možné násobit matice efektivně a postup je uveden v Algoritmu 17.3.1. Výsledná matice C je získána také po řádcích.

Algoritmus 17.3.1 Násobení řídkých matic $C = AB$
Input: *Řídké matice $A \in R^{m\times p}$, $B \in R^{p\times n}$ v CSR formátu: (ia,ja,aa), (ib,jb,ab).*
Output: *Řídká matice $C \in R^{m\times n}$ v CSR formátu: (ic,jc,ac).*

```
1. for i = 1, n
1.     wn01(i) = 0
1. end i
1. indc = 1
1. ic(1) = indc
1. for i = 1 : m
1.     ind2 = 0
1.     for j = ia(i), ia(i + 1) − 1
1.         temp = aa(j)
1.         kstrt = ib(ja(j))
1.         kstop = ib(ja(j) + 1) − 1
1.         for k = kstrt : kstop
1.             if wn01(jb(k)) == 0 then
1.                 ind2 = ind2 + 1
1.                 wn02(ind2) = jb(k)
1.                 wr02(ind2) = temp ∗ ab(k)
1.                 wn01(jb(k)) = ind2
1.             else
1.                 wr02(wn01(jb(k))) = wr02(wn01(jb(k))) + temp ∗ ab(k)
1.             end if
1.         end k
1.     end
1.     for j = 1 : ind2
1.         jc(indc) = wn02(j)
1.         wn01(wn02(j)) = 0
1.         ac(indc) = wr02(j)
1.         indc = indc + 1
1.     end j
1.     ic(i + 1) = indc
1. end i
```

17.3.1 A uložena po sloupcích a B, C uloženy po řádcích

I v tomto případě je také možné matice efektivně násobit. Procedura, kterou zde uvedeme je založena na tom, že matici B, která je uložena po sloupcích budeme procházet s použitím pomocných polí opět po řádcích.

Algoritmus 17.3.2 Násobení řídkých matic $C = AB$
Input: *Řídké matice* $A \in R^{m\times p}$, $B \in R^{p\times n}$.
Output: *Řídká matice* $C \in R^{m\times n}$.
Datové struktury: B, C *jsou uloženy v CSR formátu: (ib,jb,ab), (ic,jc,ac), A je uložena v CSC formátu (ia,ja,aa).*

```
for i = 1 : max(p, n) do
    wn01(i) = 0
    head(i) = 0
end i
for i = 1 : p do
    first(i) = ia(i)
end i
for i = 1 : p do
    if first(i) < ia(i + 1) then
        k = ja(first(i))
        link(i) = head(k)
        head(k) = i
    end if
end
indc = 1
ic(1) = indc
for i = 1 : m do
    newj = head(i)
    ind2 = 0
    j = newj
    while j ≠ 0 do
        newj = link(j)
        jfirst = first(j)
        first(j) = jfirst + 1
        if jfirst + 1 < ia(j + 1) then
            if head(ja(jfirst + 1)) == 0 then
                link(j) = 0
            else if head(ja(jfirst + 1)) ≠ 0 then
                link(j) = head(ja(jfirst + 1))
            end if
            head(ja(jfirst + 1)) = j
        end if
        temp = aa(jfirst)
        for k = ib(j) : ib(j + 1) − 1
            if wn01(jb(k)) == 0 then
```

```
36.                 ind2 = ind2 + 1
37.                 wn02(ind2) = jb(k)
38.                 wr02(ind2) = temp * ab(k)
39.                 wn01(jb(k)) = ind2
40.             else
41.                 wr02(wn01(jb(k))) = wr02(wn01(jb(k))) + temp * ab(k)
42.             end if
43.         end k
44.         j = newj
45.     end while
46.     for j = 1 : ind2
47.         jc(indc) = wn02(j)
48.         ac(indc) = wr02(j)
49.         wn01(wn02(j)) = 0
50.         indc = indc + 1
51.     end j
52.     ic(i + 1) = indc
53. end i
```

17.4 Nalezení blokové struktury v řídké matici

Další důležitou základní operací s řídkými maticemi je hledání hustých podstruktur uvnitř jejich struktury. Tyto podstruktury pak mohou hrát roli bloků v blokových algoritmech rozkladu, a proto budeme dále hovořit o hledání bloků v matici.

17.4.1 Hledání blokové struktury pomocí grafové komprese založené na strukturách sousednosti

Následující postup hledání bloků v řídké symetrické matici $A \in R^{n \times n}$ je založen na přístupu, který navrhl Ashcraft [10]. Předpokládáme, že struktura řídkosti matice je reprezentována neorientovaným grafovým modelem. V Definici 17.4.1 zavedeme pojem nerozlišitelnosti vrcholů.

Definice 17.4.1 *Řekneme, že dva vrcholy u a v neorientovaného grafu $G = (V, E)$ jsou* **nerozlišitelné**, *jestliže $adj(u) = adj(v)$. Množinu vzájemně nerozlišitelných vrcholů nazveme nerozlišitelnou množinou. Je-li $U \subseteq V$ nerozlišitelnou množinou, řekneme, že U je maximální nerozlišitelná množina, jestliže množina $U \cup \{w\}$ není nerozlišitelná pro žádné $w \in V \setminus U$.*

Poznámka 17.4.1 *Nerozlišitelnost vrcholů reprezentuje relaci ekvivalence a maximální množiny nerozlišitelných vrcholů jsou tak třídy ekvivalence podle této relace. To umožní nerozlišitelné množiny vrcholů reprezentovat řádkové a sloupcové vrcholy bloků v odpovídající matici.*

Je také zřejmé, že každé dva nerozlišitelné vrcholy jsou zároveň vrcholy sousedními. Obecněji, množina nerozlišitelných vrcholů je jednak propojena klikou a všechny vrcholy z této množiny mají zároveň stejné sousední vrcholy. Následující algoritmus pro nalezení maximálních množin nerozlišitelných vrcholů je uveden jako Algoritmus 17.4.1. Předpokládá obecně symetrickou matici. Algoritmus nezohledňuje nenulovost diagonálních prvků.

Algoritmus 17.4.1 Hledání maximálních množin nerozlišitelných vrcholů v neorientovaném grafu $G(A)$ symetrické matice $A \in R^{n \times n}$
Input: *Symetrická matice A a její grafový model $G(A) = (V, E)$.*
Output: *Množiny maximálních nerozlišitelných vrcholů.*

1. **for** $i = 1 : nx$ **do**
2. $\quad chksum = 0$
3. $\quad$ **for** $j \in adj_{G(A)}(i)$ **do**
4. $\quad\quad chksum = chksum + j$
5. $\quad$ **end** j
6. **end** i
7. *Setřiď $j \in V$ podle hodnoty chksum*
8. **for** $i = 1 : n$ **do**
9. $\quad$ **for** j *taková, že* $chksum(j) == chksum(i)$ **do**
10. $\quad\quad$ **if** $|adj(i)| == |adj(j)|$ **then**
11. $\quad\quad\quad$ **if** $adj(i) == adj(j)$ **then**
12. $\quad\quad\quad\quad$ *Zařaď j do stejné maximální množiny nerozlišitelných vrcholů jako i*
13. $\quad\quad\quad$ **end if**
14. $\quad\quad$ **end if**
15. $\quad$ **end if**
16. **end** i

Setřídění vrcholů na řádku 7 tohoto algoritmu je nápověda k efektivní implementaci s nízkou praktickou složitostí, ačkoli omezení na složitost algoritmu je $O(|E| + |V| log(|V|))$. Pro některé aplikace není nutné a zároveň může být z hlediska efektivity výpočtů výhodné, aby maximální množiny nerozlišitelných vrcholů byly určeny pouze přibližně. Toho se může docílit tím, že i rovnost v podmínkách na řádcích 9-11 Algoritmu 17.4.1 je vyhodnocována pouze přibližně. Podrobnější postup v tomto ohledu je už nad rámec tohoto textu. Jiný způsob nalezení skupin vrcholů, které představují husté podstruktury v řídké matici je uveden v následující podsekci.

17.4.2 Hledání blokové struktury pomocí grafové komprese založené na průnicích řádkových struktur

V této podkapitole budeme předpokládat, že matice, ve které hledáme bloky, je obecně nesymetrická čtvercová matice $A \in R^{n \times n}$. Místo maximálních množin nerozlišitelných vrcholů, který vede k symetrické permutaci matice, je možné uvažovat o seskupení řádků a sloupců do řádkových a sloupcových množin nezávisle na sobě, ale zde zmíníme jen

jednodušší myšlenku, která vede k rozdělení množiny vrcholů na skupiny, tedy také k symetrické permutaci. Algoritmus je přibližný a tyto skupiny tedy představují pouze aproximaci maximálních množin nerozlišitelných vrcholů. Myšlenka v pozadí [140] je ta, že do stejné skupiny vložíme všechny vrcholy takové, že jejich řádky mají velký průnik struktur. Předpokládejme, že všechny nenulové prvky matice A mají hodnotu 1, pak je průnik řádkových struktur řádků A_{i*} a A_{j*} dán součinem $A_{i*}A_{j*}^T$. Obecněji, součinem $A_{i*}A^T$ je vektor, ve kterém jeho největší hodnoty odpovídají řádkům, jejichž indexy jsou nejvhodnějšími kandidáty na to, aby byly ve stejné skupině jako je vrchol i. Výsledná implementace algoritmu, který takto roztřídí vrcholy do skupin, může být tedy založena na násobení dvou řídkých matic v CSR formátu tak, jak je popsáno výše.

17.5 Transponování matice

Bude doplněno později.

Literatura

[1] M. A. Ajiz and A. Jennings. A robust incomplete Choleski-conjugate gradient algorithm. *International J. of Numerical Methods in Engineering*, 20(5):949–966, 1984.

[2] G. Alefeld and R. S. Varga. Zur Konvergenz des symmetrischen Relaxationsverfahrens. *Numer. Math.*, 25(3):291–295, 1975/76.

[3] F. Alvarado and H. Dag. Sparsified and incomplete sparse factored inverse preconditioners. In *the 1992 Copper Mountain Conference on Iterative Methods, Vol. I (April 9–14, 1992)*, 1992.

[4] P. R. Amestoy. *Factorization of Large Sparse Matrices Based on a Multifrontal Approach in a of Multiprocessor Environment.* PhD thesis, CERFACS, Toulouse, 1991. (Report TH-PA-91-2).

[5] P. R. Amestoy and I. S. Duff. Memory allocation issues in sparse multiprocessor multifrontal methods. *Int. J. of Supercomputer Appl.*, 7:64–82, 1993.

[6] Patrick R. Amestoy, Alfredo Buttari, Jean-Yves L'Excellent, and Theo Mary. Performance and scalability of the block low-rank multifrontal factorization on multicore architectures. *ACM Trans. Math. Software*, 45(1):Art. 2, 26, 2019.

[7] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41, 2001.

[8] S. F. Ashby. Minimax polynomial preconditioning for Hermitian linear systems. *SIAM J. on Matrix Analysis and Applications*, 12(4):766–789, 1991.

[9] Steven Flynn Ashby. *Polynomial preconditioning for conjugate gradient methods.* ProQuest LLC, Ann Arbor, MI, 1988. Thesis (Ph.D.)–University of Illinois at Urbana-Champaign.

[10] C. Ashcraft. Compressed graphs and the minimum degree algorithm. *SIAM J. on Scientific Computing*, 16:1404–1411, 1995.

[11] C. Ashcraft, R. G. Grimes, and J. G. Lewis. Accurate symmetric indefinite linear equation solvers. *SIAM J. on Matrix Analysis and Applications*, 20(2):513–561, 1999.

[12] O. Axelsson. A generalized SSOR method. *Nordisk Tidskr. Informationsbehandling (BIT)*, 12:443–467, 1972.

[13] O. Axelsson, S. Brinkkemper, and V. P. Il′ in. On some versions of incomplete block-matrix factorization iterative methods. *Linear Algebra Appl.*, 58:3–15, 1984.

[14] O. Axelsson and L. Y. Kolotilina. Diagonally compensated reduction and related preconditioning methods. *Numerical Linear Algebra with Applications*, 1(2):155–177, 1994.

[15] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numerische Mathematik*, 48(5):479–498, 1986.

[16] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[17] Ivo Babuška. Numerical stability in problems of linear algebra. *SIAM J. Numer. Anal.*, 9:53–77, 1972.

[18] R. E. Bank and R. K. Smith. General sparse elimination requires no permanent integer storage. *SIAM J. Sci. and Stat. Comput.*, 8(4):574–584, 1987.

[19] T. J. Barth, T. F. Chan, and W. Tang. A parallel non-overlapping domain decomposition algorithm for compressible fluid flow problems on triangulated domains. In J. Mandel, C. Farhat, and X.-C. Cai, editors, *Tenth International Conference on Domain Decomposition Methods*, pages 23–41. AMS, Contemporary Mathematics 218, 1998.

[20] M. W. Benson. *Iterative Solution of Large Scale Iterative Systems.* PhD thesis, M.Sc. Thesis, 1973.

[21] M. Benzi, R. Kouhia, and M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Comput. Methods Appl. Mech. Engrg.*, 190(49-50):6533–6554, 2001.

[22] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. on Scientific Computing*, 17(5):1135–1149, 1996.

[23] M. Bollhöfer. A robust and efficient *ILU* that incorporates the growth of the inverse triangular factors. *SIAM J. on Scientific Computing*, 25(1):86–103, 2003.

[24] R. A. Brualdi and H. J. Ryser. *Combinatorial Matrix Theory.* Cambridge Univ. Press, 1991.

[25] A. M. Bruaset and A. Tveito. RILU preconditioning; a computational study. *J. Comput. Appl. Math.*, 39(3):259–275, 1992.

[26] A. M. Bruaset, A. Tveito, and R. Winther. On the stability of relaxed incomplete *LU* factorizations. *Math. Comp.*, 54(190):701–719, 1990.

[27] N. I. Buleev. A numerical method for solving two-dimensional diffusion equations (in Russian). *Atomnaja Energija*, 6:338–340, 1959.

[28] N. I. Buleev. A numerical method for solving two-dimensional and three-dimensional diffusion equations (in Russian). *Matematičeskij Sbornik*, 51:227–238, 1960.

[29] L. Cesari. Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive. *Atti Accad. Nazionale Lincei R. Classe Sci. Fis. Mat. Nat.*, 25:422–428, 1937.

[30] A.T. Chang. Application of sparse matrix methods in electric power system analysis. In *Sparse Matrix Proceedings, R.A. Willoughby, ed.*, pages 113–121, IBM, Thomas J. Watson Research Center, Yorktown Heights, New York, 1969.

[31] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Software*, 35(3):Art. 22, 14, 2008.

[32] E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. on Scientific Computing*, 21(5):1804–1822, 2000.

[33] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *J. of Computational and Applied Mathematics*, 86(2):387–414, 1997.

[34] P. Concus, G. H. Golub, and G. A. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. on Scientific and Statistical Computing*, 6:220–252, 1985.

[35] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[36] J. D. F. Cosgrove, Díaz, and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *Int. J. Comput. Math.*, 44:91–110, 1992.

[37] A. R. Curtis and J. K. Reid. The solution of large sparse unsymmetric systems of linear equations. In *Information processing 71 (Proc. IFIP Congress, Ljubljana, 1971), Vol. 2: Applications*, pages 1240–1245, 1972.

[38] Timothy A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):196–199, 2004.

[39] Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):167–195, 2004.

[40] Timothy A. Davis, Sivasankaran Rajamanickam, and Wissam M. Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numer.*, 25:383–566, 2016.

[41] E. F. D'Azevedo, P. A. Forsyth, and Wei-Pai Tang. Two variants of minimum discarded fill ordering. In *Iterative methods in linear algebra (Brussels, 1991)*, pages 603–612. North-Holland, Amsterdam, 1992.

[42] C. de Boor and J. R. Rice. Extremal polynomials with application to Richardson iteration for indefinite linear systems. *SIAM J. on Scientific and Statistical Computing*, 3(1):47–57, 1982.

[43] J. Demmel, S. Eisenstat, J. Gilbert, X. Li, and J. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Math. Anal. Appl.*, page to appear, 2000.

[44] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Anal. Appl.*, 20(3):720–755, 1999.

[45] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16:1–17, 1990.

[46] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of fortran basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 14:1–17, 1988.

[47] J. J. Dongarra, F. G. Gustavson, and A. Karp. Implementing linear algebra algorithms for dense matrices on a vector pipeline machine. *SIAM Review*, 26:91–112, 1984.

[48] I. S. Duff and J. R. Gilbert. Maximum-weighted matching and block pivoting for symmetric indefinite systems. In *Abstract book of Householder Symposium XV*, pages 73–75, Peebles, Scotland, 2002.

[49] I. S. Duff, K. Kaya, and B. Uçar. Design, implementation, and analysis of maximum transversal algorithms. *ACM Transactions on Mathematical Software*, 38(2):13:1–13:31, 2011.

[50] I. S. Duff and J. Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. *SIAM J. on Matrix Analysis and Applications*, 20:889–901, 1999.

[51] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM J. on Matrix Analysis and Applications*, 22:973–996, 2001.

[52] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics*, 29:635–657, 1989.

[53] I. S. Duff and S. Pralet. Strategies for scaling and pivoting for sparse symmetric indefinite problems. *SIAM J. on Matrix Analysis and Applications*, 27:313–340, 2005.

[54] I. S. Duff and S. Pralet. Towards stable mixed pivoting strategies for the sequential and parallel solution of sparse symmetric indefinite systems. *SIAM J. on Matrix Analysis and Applications*, 29(3):1007–1024, 2007.

[55] I. S. Duff and T. Wiberg. Remarks on implementations of $O(n^{1/2}\tau)$ assignment algorithms. *ACM Transactions on Mathematical Software*, 14(3):267–287, 1988.

[56] T. Dupont. A factorization procedure for the solution of elliptic difference equations. *SIAM J. on Numerical Analysis*, 5:735–782, 1968.

[57] T. Dupont, R. P. Kendall, and H. H. Jr. Rachford. An approximate factorization procedure for the solving self-adjoint elliptic difference equations. *SIAM J. on Numerical Analysis*, 5:559–573, 1968.

[58] V. Eijkhout. Beware of unperturbed modified incomplete factorizations. In *Iterative methods in linear algebra (Brussels, 1991)*, pages 583–591. North-Holland, Amsterdam, 1992.

[59] V. Eijkhout. On the existence problem of incomplete factorisation methods. Technical Report UT-CS-99-435, (also Lapack Working Note No. 144), Department of Computer Science, University of Tennessee, 1999.

[60] S. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. on Scientific and Statistical Computing*, 2:1–4, 1981.

[61] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman. The Yale Sparse Matrix Package (YSMP) – II : The non-symmetric codes. Technical Report No. 114, Department of Computer Science, Yale University, 1977.

[62] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman. The Yale Sparse Matrix Package (YSMP) – I : The symmetric codes. *International J. of Numerical Methods in Engineering*, 18:1145–1151, 1982.

[63] Stanley Eisenstat and Joseph W. H. Liu. The theory of elimination trees for sparse unsymmetric matrices. *SIAM J. Matrix Anal. Appl.*, 26(3):686–705, 2005.

[64] Stanley C. Eisenstat and Joseph W. H. Liu. A tree-based dataflow model for the unsymmetric multifrontal method. *Electron. Trans. Numer. Anal.*, 21:1–19, 2005.

[65] Stanley C. Eisenstat and Joseph W. H. Liu. Algorithmic aspects of elimination trees for sparse unsymmetric matrices. *SIAM J. Matrix Anal. Appl.*, 29(4):1363–1381, 2007.

[66] H. C. Elman. A stability analysis of incomplete lu factorizations. *Mathematics of Computation*, 47:191–217, 1986.

[67] H. C. Elman. Relaxed and stabilized incomplete factorizations for non-self-adjoint linear systems. *BIT Numerical Mathematics*, 29:890–915, 1989.

[68] G. Estrin. Organization of computer systems—the fixed plus variable structure computer. In *Proc. Western Joint Comput. Conf., San Francisco, May 1960*, pages 33–40, 1992.

[69] Ky Fan. Note on M-matrices. *Quart. J. Math. Oxford Ser. (2)*, 11:43–49, 1960.

[70] Bernd Fischer and Roland W. Freund. On adaptive weighted polynomial preconditioning for Hermitian positive definite matrices. *SIAM J. Sci. Comput.*, 15(2):408–426, 1994.

[71] G. E. Forsythe and E. G. Straus. On best conditioned matrices. *Proc. Amer. Math. Soc.*, 6:340–345, 1955.

[72] L. Fox, H. D. Huskey, and J. H. Wilkinson. Notes on the solution of algebraic linear simultaneous equations. *Quart. J. Mech. and Appl. Math.*, 1:149–173, 1948.

[73] Stanley P. Frankel. Convergence rates of iterative treatments of partial differential equations. *Math. Tables Aids Comput.*, 4:65–75, 1950.

[74] Roland Freund. On polynomial preconditioning and asymptotic convergence factors for indefinite Hermitian matrices. *Linear Algebra Appl.*, 154/156:259–288, 1991.

[75] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ., 1981.

[76] A. George and E. Ng. An implementation of Gaussian elimination with partial pivoting for sparse systems. *SIAM J. Sci. and Stat. Comput.*, 6:390–409, 1985.

[77] A. George and E. Ng. Symbolic factorization for sparse gaussian elimination with partial pivoting. *SIAM J. Sci. Stat. Comput.*, 8:877–898, 1987.

[78] J. R. Gilbert. Predicting structure in sparse matrix computations. *SIAM J. on Matrix Analysis and Applications*, 15:62–79, 1994.

[79] J. R. Gilbert and J. W. H. Liu. Elimination structures for unsymmetric sparse lu factors. *SIAM J. on Matrix Analysis and Applications*, 14:334–352, 1993.

[80] J. R. Gilbert, E. Ng, and B. W. Peyton. An efficient algorithm to compute row and column counts for sparse Cholesky factorization. *SIAM J. on Matrix Analysis and Applications*, 15(4):1075–1091, 1994.

[81] J. R. Gilbert and T. Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. on Scientific and Statistical Computing*, 9:862–874, 1988.

[82] N. I. M. Gould and J. A. Scott. Sparse approximate-inverse preconditioners using norm-minimization techniques. *SIAM J. on Scientific Computing*, 19(2):605–625, 1998.

[83] J. Grcar. Analyses of the lanczos algorithm and of the approximation problem in richardson's method. Technical Report No. 1074, Department of Computer Science, University of Illinois, Urbana, IL, 1981. Ph.D Thesis.

[84] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. on Scientific Computing*, 18(3):838–853, 1997.

[85] I. Gustafsson. On modified incomplete Cholesky factorization methods for the solution of problems with mixed boundary conditions and problems with discontinuous material coefficients. *International J. of Numerical Methods in Engineering*, 14(8):1127–1140, 1979.

[86] I. Gustafsson. Modified incomplete Cholesky (MIC) methods. In *Preconditioning methods: analysis and applications*, volume 1 of *Topics in Computational Mathematics*, pages 265–293. Gordon & Breach, New York, 1983.

[87] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95 of *Applied Mathematical Sciences*. Springer, [Cham], second edition, 2016.

[88] J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *Conference Record 1971 Twelfth Annual Symposium on Switching and Automata Theory*, East Lansing, Michigan, USA, 1971. IEEE.

[89] T. Huckle. Approximate sparsity patterns for the inverse of a matrix and preconditioning. *Applied Numerical Mathematics*, 30(2-3):291–303, 1999.

[90] T. K. Huckle. Efficient computation of sparse approximate inverses. *Numerical Linear Algebra with Applications*, 5(1):57–71, 1998.

[91] D. Hysom and A. Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM J. on Scientific Computing*, 22:2194–2215, 2001.

[92] A. Jennings and G. M. Malik. Partial elimination. *J. of the Institute of Mathematics and its Applications*, 20(3):307–316, 1977.

[93] A. Jennings and G. M. Malik. The solution of sparse linear equations by the conjugate gradient method. *International J. of Numerical Methods in Engineering*, 12(1):141–158, 1978.

[94] O. G. Johnson, C. A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. on Numerical Analysis*, 20(2):362–376, 1983.

[95] G. Karypis and V. Kumar. Parallel threshold-based ilu factorization. In *Proceedings of 9th Supercomputing Conference, Barcelona, Spain*, pages 1–24. ACM, 1997.

[96] D. S. Kershaw. The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *J. of Computational Physics*, 26:43–65, 1978.

[97] D. E. Knuth. *The art of computer programming. Vol. 1.* Addison-Wesley, Reading, MA, 1997. Fundamental algorithms, Third edition.

[98] D. E. Knuth. *The art of computer programming. Vol. 2.* Addison-Wesley, Reading, MA, 1998. Seminumerical algorithms, Third edition.

[99] D. E. Knuth. *The art of computer programming. Vol. 3.* Addison-Wesley, Reading, MA, 1998. Sorting and searching, Second edition.

[100] A. Koubková and V. Koubek. *Datové struktury I.* MatfyzPress, 2011.

[101] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *Application for Computing Machinery Transactions on Mathematical Software*, 5:308–323, 1979.

[102] Xiaoye S. Li. An overview of SuperLU: algorithms, implementation, and user interface. *ACM Trans. Math. Software*, 31(3):302–325, 2005.

[103] Xiaoye S. Li and James W. Demmel. SuperLU_DIST: a scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Math. Software*, 29(2):110–140, 2003.

[104] J. Liesen and Z. Strakoš. *Krylov subspace methods.* Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2013. Principles and analysis.

[105] J. W. H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11:141–153, 1985.

[106] J. W. H. Liu. A compact row storage scheme for Cholesky factors using elimination trees. *ACM Transactions on Mathematical Software*, 12(2):127–148, 1986.

[107] J. W. H. Liu. The role of elimination trees in sparse factorizations. *SIAM J. on Matrix Analysis and Applications*, 11(1):134–172, 1990.

[108] J. W. H. Liu. A generalized envelope method for sparse factorization by rows. *ACM Transactions on Mathematical Software*, 17(1):112–129, 1991.

[109] J. W. H. Liu and A. H. Sherman. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM J. on Numerical Analysis*, 13:198–213, 1976.

[110] Joseph W. H. Liu. On threshold pivoting in the multifrontal method for sparse indefinite systems. *ACM Trans. Math. Software*, 13(3):250–261, 1987.

[111] Joseph W. H. Liu. A partial pivoting strategy for sparse symmetric matrix decomposition. *ACM Trans. Math. Software*, 13(2):173–182, 1987.

[112] T. A. Manteuffel. The Tchebychev iteration for nonsymmetric matrices. *Numerische Mathematik*, 28:307–327, 1977.

[113] T. A. Manteuffel. Adaptive procedure for estimation of parameter for the nonsymmetric Tchebychev iteration. *Numerische Mathematik*, 31:183–208, 1978.

[114] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation*, 34:473–497, 1980.

[115] M. Mareš and T. Valla. *Průvodce labyrintem algoritmů*. CZ.NIC, 2017.

[116] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3:255–269, 1957.

[117] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31(137):148–162, 1977.

[118] N. S. Mendelsohn. Some properties of approximate inverses of matrices. *Trans. Roy. Soc. Canada Sect. III. (3)*, 50:53–59, 1956.

[119] G. Meurant. *Computer Solution of Large Linear Systems*. Elsevier, Amsterdam – Lausanne – New York – Oxford – Shannon – Singapore – Tokyo, 1999.

[120] J. Morris. An escalator process for the solution of linear simultaneous equations. *Philos. Mag.*, 37:106–120, 1946.

[121] A. Neumaier and M. Olschowka. A new pivoting strategy for Gaussian elimination. *Linear Algebra and its Applications*, 240:131–151, 1996.

[122] Michael Neumann and Richard S. Varga. On the sharpness of some upper bounds for the spectral radii of S.O.R. iteration matrices. *Numer. Math.*, 35(1):69–79, 1980.

[123] E. G. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM J. on Scientific Computing*, 14:1034–1056, 1993.

[124] E. G. Ng and B. W. Peyton. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors. *SIAM J. on Scientific Computing*, 14:761–769, 1993.

[125] Yvan Notay. DRIC: a dynamic version of the RIC method. *Numer. Linear Algebra Appl.*, 1(6):511–532, 1994.

[126] Dianne P. O'Leary. Yet another polynomial preconditioner for the conjugate gradient algorithm. *Linear Algebra Appl.*, 154/156:377–388, 1991.

[127] J. M. Ortega. *Introduction to Parallel and Vector Computing*. Plenum Press, New York, 1988.

[128] O. Østerby and Z. Zlatev. *Direct methods for sparse matrices*, volume 157 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1983.

[129] A. M. Ostrowski. On the linear iteration procedures for symmetric matrices. *Rend. Mat. e Appl. (5)*, 14:140–163, 1954.

[130] S. Pissanetzky. *Sparse matrix technology*. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], London, 1984.

[131] G. Radicati di Brozolo and Y. Robert. Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor. *Parallel Comput.*, 11(2):223–239, 1989.

[132] Edgar Reich. On the convergence of the classical iterative method of solving linear simultaneous equations. *Ann. Math. Statistics*, 20:448–451, 1949.

[133] J. K. Reid and I. S. Duff. Ma48–a fortran code for direct solution of sparse unsymmetric linear systems of equations. Technical Report RAL-93-072, Rutherford Appleton Laboratories, 1993.

[134] L.F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations with an application to stresses in a masonry dam. *Phil. Trans. Roy. Soc., Sec. A*, 210:307–357, 1910.

[135] James D. Riley. Iteration procedures for the Dirichlet difference problem. *Math. Tables Aids Comput.*, 8:125–131, 1954.

[136] Romeo Rizzi. A short proof of König's matching theorem. *J. Graph Theory*, 33(3):138–139, 2000.

[137] Y. Saad. Practical use of polynomial preconditionings for the conjugate gradient method. Research Report RR–282, Department of Computer Science, Yale Univerisity, 1983.

[138] Y. Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *J. of Computational and Applied Mathematics*, 24(1-2):89–105, 1988. Iterative methods for the solution of linear systems.

[139] Y. Saad. SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report 90–20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.

[140] Y. Saad. Finding exact and approximate block structures for ILU preconditioning. *SIAM J. on Scientific Computing*, 24(4):1107–1123, 2003.

[141] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[142] Paul E. Saylor. Second order strongly implicit symmetric factorization methods for the solution of elliptic difference equations. *SIAM J. Numer. Anal.*, 11:894–908, 1974.

[143] Paul E. Saylor and Dennis C. Smolarski. Computing the roots of complex orthogonal and kernel polynomials. *SIAM J. Sci. Statist. Comput.*, 9(1):1–13, 1988.

[144] G. Schulz. Iterative Berechnung der reziproken Matrix. *ZAMM*, 13:57–59, 1933.

[145] Mark K Seager. Parallelizing conjugate gradient for the cray x-mp. *Parallel Computing*, 3(1):35 – 47, 1986.

[146] John W. Sheldon. On the numerical solution of elliptic difference equations. *Math. Tables Aids Comput.*, 9:101–112, 1955.

[147] A. H. Sherman. On the efficient solution of sparse systems of linear and non-linear equations. Tech. Report 46, Dept. of Computer Science, Yale Univ., 1975.

[148] George Shortley. Use of Tschebyscheff-polynomial operators in the numerical solution of boundary-value problems. *J. Appl. Phys.*, 24:392–396, 1953.

[149] Eduard L. Stiefel. *Kernel polynomials in linear algebra and their numerical applications. Four lectures on solving linear equations and determining eigenvalues.* National Bureau of Standards, Washington, D. C., 1955.

[150] Eduard L. Stiefel. Kernel polynomials in linear algebra and their numerical applications. *Nat. Bur. Standards Appl. Math. Ser.*, 49:1–24, 1958.

[151] Wei-Pai Tang. Generalized Schwarz splittings. *SIAM J. Sci. Statist. Comput.*, 13(2):573–595, 1992.

[152] R.E. Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1983.

[153] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. Assoc. Comput. Mach.*, 22:215–225, 1975.

[154] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. In *Proceeedings of the IEEE*, volume 55, pages 1801–1809, 1967.

[155] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications*, 154–156:331–353, 1991.

[156] A. van der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969/70.

[157] H. A. van der Vorst. High performance preconditioning. *SIAM J. on Scientific and Statistical Computing*, 10(6):1174–1185, 1989.

[158] Henk A. van der Vorst. Analysis of a parallel solution method for tridiagonal linear systems. *Parallel Comput.*, 5(3):303–311, 1987.

[159] R. S. Varga. Factorizations and normalized iterative methods. In *Boundary problems in differential equations*, pages 121–142, Madison, WI, 1960. University of Wisconsin Press.

[160] R. S. Varga, E. B. Saff, and V. Mehrmann. Incomplete factorizations of matrices and connections with h-matrices. *SIAM J. on Numerical Analysis*, 17:787–793, 1980.

[161] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. Assoc. Comput. Mach.*, 8:281–330, 1961.

[162] J. H. Wilkinson. *Rounding errors in algebraic processes.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.

[163] Gangjoon Yoon and Chohong Min. A simple proof of gustafsson's conjecture in case of poisson equation on rectangular domains. *American Journal of Computational Mathematics*, 05:75–79, 01 2015.

[164] David Young. Iterative methods for solving partial difference equations of elliptic type. *Trans. Amer. Math. Soc.*, 76:92–111, 1954.

[165] David Young. On Richardson's method for solving linear systems with positive definite matrices. *J. Math. Physics*, 32:243–255, 1954.

[166] Z. Zlatev. *Computational Methods for General Sparse Matrices.* Kluwer, Dordrecht, 1991.

[167] Z. Zlatev, J. Wasniewski, and K. Schaumburg. *Y12M : Solution of Large and Sparse Systems of Linear Algebraic Equations, Documentation of Subroutines*, volume 121 of *Lecture Notes in Computer Science.* Springer-Verlag, 1981.