Vysoká škola báňská – Technická universita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Metody vnitřního bodu pro úlohy lineárního programování

2010 Radek Riedel

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: …..... Podpis: ….....

Poděkování:

Mé poděkování patří doc. Vítovi Vondrákovi za četné podněty, kterými se snažil mě navést do cíle a rozšířit mé znalosti dané oblasti. Dále bych rád poděkoval prof. Zdeňkovi Dostálovi také za pomoc a rady při řešení problémů, na které jsem narazil. A především své ženě Viktórii za trpělivost a péči, se kterou se stará o naší malou dcerku, která byla taky moc hodná a mě při dokončování studia příliš nazatěžovala i přes svůj velmi malý věk.

Abstrakt

Tato práce pojednává o teorii a praktické implementaci algoritmu pro řešení úloh lineárního programování pomocí metody vnitřního bodu. Prácei se zaměřuje na problémy, které nastávají při praktické implementaci algoritmů metod vnitřního bodu. Ambicí práce je nalézt způsob, kterým by se algoritmus metody vnitřního bodu dal vylepšit z pohledu efektivity řešení úloh velkých rozměrů. Tato práce částečně navazuje na moji předešlou bakalářskou práci, kterou jsem vypracoval na téma efektivní implementace simplexového algoritmu pro úlohy lineárního programování.

Klíčová slova

Lineární programovaní, metody vnitřního bodu, primárně-duální metody, předpodmínění, metoda sdružených gradientů.

Obsah

1. Úvod

Hlavní ambicí této práce je implementovat efektivní algoritmus metody vnitřního bodu. K tomu nejdříve probereme potřebnou teorii. Probereme různé varianty základních algoritmů a postupně se dostaneme ke známému algoritmu primárně duální metody nazývanému prediktor-korektor. Ukážeme si poznatky známých praktických řešení. Výsledkem práce bude jmenovaného algoritmu a dalším cílem je najít způsob jakým by se dal tento algoritmus vylepšit a tím zvýšit jeho efektivitu. Protože pro malé úlohy je jmenovaná implementace efektivní snahou bude zaměření na úlohy velkých rozměrů. Závěrem práce bude srovnání různých způsobů implementace a vyhodnocení testů.

1.1 Úloha lineárního programovaní

Úlohu lineárního programovaní lze obecně definovat jako úlohu nalezení extrému lineární funkce $f(x)$, definované na vektorovém prostoru R^n omezeném zadanými podmínkami. Hledaným extrémem, podle podstaty zadaného problému, může být maximum nebo minimum. Cílem úlohy je nalézt optimální hodnotu účelové funkce $f(x)$ při dodržení zadaných podmínek. Proto je úloha lineárního programovaní chápána jako optimalizační úloha. Na úlohu lineárního programování se budeme v následujícím textu odkazovat také jako na úlohu LP.

Existuje celá řada formulací úloh lineárního programování, nejčastěji se pak setkáme s následujícím zadáním, hledejme

$$min\, c^T x \text{, přičemž platí } Ax = b,\ x \geq 0. \qquad 1.1$$

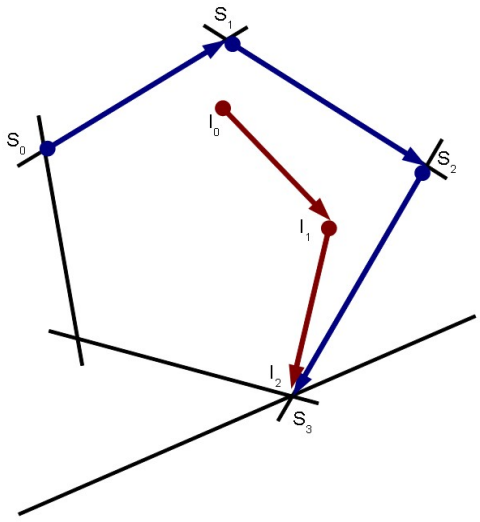
Úlohu LP lze řešit různými metodami. Nejstarší metodou dodnes stále často používanou je Simplexová metoda, kterou jsem se zabýval ve své Bakalářské práci [1]. Dalšími metodami jsou například metody vnitřního bodu, kterými se budeme zabývat v této práci.

1.2 Metody vnitřního bodu

Metody vnitřního bodu jsou jednou z možností jak řešit úlohy LP. Tyto metody se pak dají uplatnit i pro hledání řešení jiných problému jako např. úloh kvadratického programování. Na rozdíl Simplexové metody, která se k optimálnímu řešení přibližuje po obalu přípustné množiny, se metody vnitřního bodu snaží nalézt optimální řešení jako posloupnost bodů, které leží uvnitř oblasti přípustných řešení a které konvergují k řešení optimálnímu. Právě proto, že se k optimálnímu řešení přibližujeme z vnitřku množiny přípustných řešení a v každém kroku hlídáme, abychom

nepřekročili hranici oblasti přípustných řešení, nazýváme tyto metody metodami vnitřního bodu. Pro upřesnění je vhodné zmínit, že praktické algoritmy metody vnitřního bodu nemusí striktně dodržovat přípustnost řešení v každém kroku. Některé algoritmy mohou jednotlivých krocích pracovat i s body mimo oblast přípustnosti, nicméně tyto algoritmy striktně dodržují nezápornost aktuálního řešení. V následujících kapitolách si popíšeme principy, na kterých je tato metoda postavena. Ukážeme si také podmínky, za kterých bude tato metoda úspěšně konvergovat k hledanému řešení.

Jestliže je známo, že optimální řešení úlohy LP leží na hranici přípustné oblasti viz. [1], potom metody vnitřního bodu, které se při hledání řešení snaží od této hranice držet odstup, se k optimálnímu řešení blíží pouze limitně. Rozdíl kterým se liší hledání optimálního řešení pomocí simplexové metody a metod vnitřního bodu je znázorněn na obrázku 1.



Obrázek 1: Znázornění cesty hledání optimálního řešení úlohy LP. S_0-S_3 – simplexová metoda, I_0-I_2 – metody vnitřního bodu.

2. Optimalizační úlohy

V této kapitole představíme optimalizační úlohu s omezením, jejíž speciálním případem je i úloha lineárního programování, kterou se budeme zabývat. Rozvedeme teoretický základ a principy, které jsou dále využívány metodami vnitřního bodu. Probereme podmínky optimality, které musí být splněny pro nalezení optimálního řešení a zavedeme definici duální úlohy. Představení primárně-duální úlohy a podmínek optimality je základ, na kterém staví většina metod vnitřního bodu, zejména potom primárně-duální algoritmy.

2.1 Optimalizační úloha s omezením

Optimalizační úlohou s omezením budeme myslet následující zadání. Hledejte minimum funkce

$$\min_{x \in R^n} f(x) \text{, kde platí } c_i(x) = 0, i \in E$$

$$\text{a} \qquad c_i(x) \geq 0, i \in I. \qquad 2.1$$

Funkce $f(x)$ a $c_i(x)$ jsou spojité reálné funkce proměnné $x \in R^n$, rovnice $c_i(x) = 0$, pro každé $i \in E$ a nerovnice $c_i(x) \geq 0$, pro každé $i \in I$ jsou podmínkami definující oblast přípustnosti hledaného řešení. Tuto oblast můžeme definovat jako množinu

$$\Omega = \{x | c_i(x) = 0, i \in E; c_i(x) \geq 0, i \in I\}. \qquad 2.2$$

Optimalizační úlohu můžeme jednodušeji zapsat jako

$$\min_{x \in \Omega} f(x). \qquad 2.3$$

Řešením optimalizační úlohy bude vektor $x \in R^n$, který vyhovuje podmínkám omezení. Optimální řešení, tedy hledané řešení úlohy, označíme $\mathring{x} \in \Omega$, je vektor $\mathring{x} \in \Omega$ pro který platí $f(\mathring{x}) \leq f(x)$ pro každé $x \in \Omega$. Často se také setkáme s pojmem přípustné řešení, což jsou všechna $x \in \Omega$. V různých textech se také často setkáme s označením bod, přípustný bod nebo optimální bod, v takovém případě v kontextu s řešením optimalizační úlohy považujme výrazy řešení a bod za ekvivalentní.

Úloha lineárního programování zmíněna již v úvodu a také cílová skupina úloh této práce, je speciálním případem výše popsané optimalizační úlohy s omezením, kde funkce $f(x)$ a $c_i(x)$ jsou lineární funkce proměnné $x \in R^n$.

Úlohu lineárního programovaní ve standardní formě definujeme jako

$$min\, c^T x \text{, kde platí } Ax = b,\ x \geq 0. \qquad 2.4$$

Úlohu lineárního programování lze definovat různými způsoby, lze však ukázat, že každé zadání úlohy LP lze převést na výše uvedený standardní tvar viz [1].

2.2 Podmínky optimality

K tomu abychom mohli definovat podmínky optimality pro řešení úlohy s omezením, budeme potřebovat několik vztahů.

Nejdříve definujme Lagrangeovu funkci

$$L(x, \lambda) = f(x) - \sum_{i \in E \cup I} \lambda_i c_i(x), \qquad 2.5$$

která kombinuje hodnotu účelové funkce s hodnotami omezujících funkcí pomocí koeficientů, které nazýváme Lagrangeovy multiplikátory.

Dále si definujme pojem aktivní množiny, kterou definujeme jako množinu přípustných řešení, které leží na hranici oblasti Ω. Tedy body, ve kterých jsou splněny všechny rovnostní omezení, a ve kterých nejsou nerovnostní omezení splněny striktně tedy

$$A(x) = E \cup \{i \in I | c_i(x) = 0\}. \qquad 2.6$$

Dále si definujme pojem lineární nezávislosti omezení v anglické literatuře označované jako LICQ (takto se na ní budeme odkazovat i my).

Definice 2.1 (LICQ)

Mějme bod $\mathring{x}$ a aktivní množinu $A(\mathring{x})$ definovanou 2.6. Říkáme, že podmínka lineární nezávislosti omezení (LICQ) je splněna, pokud množina gradientů aktivních omezení $\{\nabla c_i(\mathring{x}), i \in A(\mathring{x})\}$ je lineárně nezávislá.

Nyní můžeme definovat nutnou podmínku optimality pro obecnou optimalizační úlohu. Tuto podmínku nazveme podmínkou prvního druhu, protože využívá gradientu, tedy derivace prvního řádu, účelové funkce.

Věta 2.1: ***Nutná podmínka optimality prvního druhu***

Předpokládejme, že $\mathring{x}$ je lokálním řešením 2.1 a předpokládejme, že úloha splňuje LICQ v $\mathring{x}$. Potom existuje vektor Lagrangeových multiplikátorů $\mathring{\lambda}$ s prvky $\mathring{\lambda}_i, i \in E \cup I$ takový, že následující podmínky jsou splněny.

$$\nabla_x L(\mathring{x}, \mathring{\lambda}) = 0, \qquad 2.7$$

$$c_i(\mathring{x}) = 0 \qquad \forall i \in E, \qquad 2.8$$

$$c_i(\mathring{x}) \geq 0 \qquad \forall i \in I, \qquad 2.9$$

$$\mathring{\lambda}_i \geq 0 \qquad \forall i \in I, \qquad 2.10$$

$$\mathring{\lambda}_i c_i(\mathring{x}) = 0 \qquad \forall i \in E \cup I. \qquad 2.11$$

Podmínky z předcházející věty jsou známy jako Karush-Kuhn-Tuckerovy (KKT) podmínky. Důkaz věty 2.1 je velmi důležitým pro porozumění podstaty řešení úloh optimalizace s omezením. Celý tento důkaz je poměrně rozsáhlý a je dobře popsán například ve [2, kapitola 12], proto jej zde nebudeme znovu uvádět. Uveďme si jen, že Karush-Kuhn-Tuckerovy podmínky jsou základem, na kterém jsou postaveny metody vnitřního bodu.

2.3 Optimalita LP

Existenci optimálního řešení úlohy lineárního programovaní 2.4 můžeme odvodit z KKT 2.7-2.11. Rozdělme Lagrangeovy multiplikátory na dva vektory λ a s tak, že $\lambda \in R^m$ je vektor multiplikátoru pro podmínky $Ax = b$ a s je vektor multiplikátorů pro omezující podmínku $x \geq 0$. Podle 2.5 můžeme přepsat Lagrangeovu funkci následujícím způsobem

$$L(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x. \qquad 2.12$$

Buď $\mathring{x}$ optimální řešení řešením 2.4, aplikací KKT podmínek zjistíme, že musí existovat vektory λ a s, které vyhovují

$$A^T \lambda + s = c \qquad 2.13$$

$$Ax = b \qquad 2.14$$

$$x \geq 0 \qquad 2.15$$

$$s \geq 0 \qquad 2.16$$

$$x_i s_i = 0, i = 1, 2, \ldots, n. \qquad 2.17$$

Buď $(\mathring{x}, \mathring{\lambda}, \mathring{s})$ trojicí vektorů odpovídajících podmínkám 2.13-2.17. Kombinací těchto podmínek dostaneme

$$c^T \mathring{x} = (A^T \mathring{\lambda} + \mathring{s})^T \mathring{x} = (A \mathring{x})^T \mathring{\lambda} = b^T \mathring{\lambda} . \qquad 2.18$$

V další kapitole si ukážeme, že $b^T \lambda$ bude účelovou funkcí duální úlohy k úloze 2.4 a podle 2.18 vidíme, že řešení primární a duální úlohy jsou si rovny pro trojici vektorů $(\mathring{x}, \mathring{\lambda}, \mathring{s})$, které odpovídají podmínkám 2.13-2.17.

Nyní si můžeme ukázat, že podmínky 2.13-2.17 jsou postačující proto, aby $\mathring{x}$ bylo globálním řešením 2.4. Mějme vektor y, který je jiným přípustným řešením téže úlohy, platí tedy $A y = b$ a $y \geq 0$, potom platí

$$c^T y = (A^T \lambda + s)^T y = b^T \lambda + y^T s \geq b^T \lambda = c^T x . \qquad 2.19$$

Tato nerovnost vyplývá přímo z 2.13 a 2.16. Z této nerovnosti rovněž vidíme, že hodnota účelové funkce $c^T y$ není pro žádné přípustné řešení menší než $c^T x$. Můžeme říct, že přípustné řešení y je optimální právě tehdy, když $y^T s = 0$, jinak je nerovnost 2.19 ostrá. Což znamená že musí platit 2.17. Dospěli jsme k závěru, že podmínky 2.13-2.17 jsou v tomto případě postačující.

2.4 Dualita LP

Máme-li zadánu úlohu LP 2.4 pomocí A, b a c, pak můžeme definovat příbuzný problém

$$max\, b^T \lambda \text{, přičemž platí } A^T \lambda \leq c . \qquad 2.20$$

Tuto úlohu budeme nazývat úlohou duální k úloze 2.4, kterou v tomto případě budeme nazývat primární.

Ukažme si nyní, jak spolu souvisí řešení primární a duální úlohy. Úlohu maximalizace můžeme přepsat jako úlohu minimalizace

$$min\, -b^T \lambda \text{, kde platí } c - A^T \lambda \geq 0 . \qquad 2.21$$

Nyní použijeme vektor x jako vektor Langrangeových multiplikátorů a definujeme Lagrangeovu funkci

$$L(\lambda, x) = -b^T \lambda - x^T (c - A^T \lambda) . \qquad 2.22$$

Pro výše uvedenou úlohu napíšeme KKT podmínky

$$\nabla_{\lambda} L(\lambda, x) = 0 \quad \Rightarrow \quad -b + Ax = 0 \quad \Rightarrow \quad Ax = b\,, \tag{2.23}$$

$$c - A^T \lambda \geq 0 \quad \Rightarrow \quad A^T \lambda \leq c\,, \tag{2.24}$$

$$x \geq 0\,, \tag{2.25}$$

$$x_i (c - A^T \lambda)_i = 0\,. \tag{2.26}$$

Zavedeme substituci

$$s = c - A^T \lambda \tag{2.27}$$

a doplníme do 2.24 a 2.26. Zjistíme, že podmínky 2.23-2.27 odpovídají podmínkám 2.13-2.17. Zjistili jsme také, že nalezení řešení duální úlohy je stejným problémem jako nalezení řešení úlohy primární.

Věta 2.2: O dualitě lineárního programování

i. Jestliže má jedna z úloh 2.4 a 2.20 optimální řešení s konečnou hodnotou účelové funkce, potom má optimální řešení i druhá úloha a hodnoty jejích účelových funkcí jsou si rovny.

ii. Jestliže hodnota účelové funkce jedné z úloh 2.4 a 2.20 může neomezeně růst, potom ta druhá nemá žádné přípustné řešení.

Důkaz

i. Předpokládejme, že primární úloha 2.4 má konečné optimální řešení, potom podle věty 2.1 existuje trojice vektorů (x, λ, s), která splňuje podmínky 2.13-2.17. Protože podmínky 2.13-2.17 odpovídají podmínkám 2.23-2.27 potom tato trojice vektorů (x, λ, s) vyhovuje i řešení duální úlohy 2.20.

Podle 2.17 a 2.26 platí

$$0 = x^T s = x^T (c - A^T \lambda) = x^T c - b^T \lambda \Rightarrow x^T c = b^T \lambda\,.$$

Proto obě úlohy mají stejné řešení a jejich účelové funkce mají v tomto řešení stejnou hodnotu.

ii. Předpokládejme, že primární úloha nemá omezené optimální řešení. To znamená, že musí existovat směr $d \in R^n$, podél kterého účelová funkce $c^T x$, klesá bez porušení podmínek přípustnosti. Musí platit

$$\begin{aligned} c^T(x+d) < c^T x \quad &\Rightarrow c^T d < 0, \\ A(x+d) = 0 \quad &\Rightarrow A d = 0, \\ x + d \geq 0 \quad &\Rightarrow d \geq 0. \end{aligned}$$

Předpokládejme, že existuje přípustný bod λ duální úlohy, takový že $A^T \lambda \leq c$. Násobením nezáporným vektorem d^T zleva dostaneme

$$0 = d^T A^T \lambda \leq d^T c < 0$$

což je spor.

c.b.d.

3. Metody vnitřního bodu

Existuje více variací metod vnitřního bodu, z nichž mezi nejznámější patří bariérové metody a primárně duální metody. Je známo, že primárně duální metody bývají efektivnější než metody bariéry. Pro některé základní třídy problému jako jsou např. lineární, kvadratické, geometrické, semidefinitní programování existují algoritmy primárně duální metody, jejichž efektivita převyšuje bariérové metody. V této práci se zaměříme se na primárně duální metody. Mezi významnou vlastnost přispívající k efektivitě primárně duálních metod, patří fakt, že tyto metody umí při hledání optimálního řešení v průběhu algoritmu pracovat i s nepřípustnými body, čímž odpadá problém s nalezením výchozího řešení, které by bylo řešením přípustným. Hledání takového přípustného výchozího bodu může vést k hledání řešení podobného problému jako je zadaná úloha sama.

3.1 Primárně duální metoda

Mějme primární úlohu

$$min\, c^T x \text{ , kde platí } Ax = b \text{ , } \quad x \geq 0 \text{ ,} \qquad 3.1$$

kde $c \in R^n$, $x \in R^n$, $b \in R^m$ a A je matice $m\,x\,n$. K této úloze definujme duální úlohu

$$max\, b^T \lambda \text{ , kde } A^T \lambda + s = c \text{ , } \quad s \geq 0 \text{ .} \qquad 3.2$$

Ukázali jsme si, že optimální řešení primárně duální úlohy musí splňovat podmínky KKT, které jsme definovali jako

$$A^T \lambda + s = c \text{ ,} \qquad 3.3$$

$$A\,x = b \text{ ,} \qquad 3.4$$

$$x_i s_i = 0, i = 1,2, \ldots, n \text{ ,} \qquad 3.5$$

$$(x, s) \geq 0 \text{ .} \qquad 3.6$$

Hledat řešení úlohy LP primárně duální metodou znamená, hledat trojici vektorů (x, λ, s), pro které platí podmínky 3.3-3.6. Toto trojici vektorů hledáme tak, že ze zmíněných podmínek sestavíme soustavu rovnic a použijeme vhodnou metodou k nalezení řešení této soustavy. Použitá metoda v každém kroku generuje trojici vektorů (x^k, λ^k, s^k), které splňují 3.3-3.6. Důležitým prvkem metody vnitřního bodu je podmínka $(x^k, s^k) \geq 0$, která zaručuje, že aktuální řešení neporuší požadovanou podmínku nezápornosti v každém kroku.

3.2 Základní algoritmus

Přepíšeme podmínky 3.3-3.5 jako funkci $F: R^{2n+m} \rightarrow R^{2n+m}$. Pro nalezení řešení primárně duální úlohy budeme hledat bod, ve kterém tato funkce nabývá 0

$$F(x, \lambda, s) = \begin{bmatrix} A^T \lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0, \qquad 3.7$$

$$(x, s) \geq 0, \qquad 3.8$$

kde $X = diag(x_1, x_2, \ldots, x_n)$, $S = diag(s_1, s_2, \ldots, s_n)$ a $e = (1,1,\ldots,1)^T$.

Primárně duální metoda v každém kroku generuje řešení (x^k, λ^k, s^k), které splňuje 3.8 striktně, tedy $x^k > 0$ a $s^k > 0$. K řešení výše definované rovnice 3.7, můžeme použít Newtonovu metodu pro nelineární rovnice. Touto metodou nalezneme pomocí lineárního modelu v okolí aktuálního bodu směr k hledanému řešení. Nalezení tohoto směru popíšeme následující soustavou lineárních rovnic

$$J(x, \lambda, s) \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = -F(x, \lambda, s), \qquad 3.9$$

kde J je Jacobián funkce F. Více o Newtonově metodě lze nalézt v literatuře např. [2 kapitola 11].

Pokud je aktuální řešení striktně přípustné, potom lze tuto soustavu zapsat takto

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe \end{bmatrix}. \qquad 3.10$$

Výše uvedeným způsobem nalezneme směr pro provedení kroku k hledanému řešení. Tento směr budeme nazývat affiní. Provedení úplného kroku v nalezeném směru, ale nemusí být dovoleno, protože by takto mohla být porušena podmínka nezápornosti $(x, s) \geq 0$. Tento problém odstraníme tak, že místo úplného kroku budeme hledat další bod řešení v nalezeném směru tak, aby podmínka nezápornosti nebyla porušena. Tedy nalezneme $\alpha \in (0,1]$ takové, že

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta x^k, \Delta \lambda^k, \Delta s^k) \qquad 3.11$$

bude splňovat $(x^{k+1}, s^{k+1}) \geq 0$.

Nyní může ukázat základní algoritmus metody vnitřního bodu

Mějme počáteční bod (x^0, λ^0, s^0) , kde $(x^0, s^0) > 0$

while $max(x_i s_i) > \epsilon$

Vyřešit $\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe \end{bmatrix}$

Vypočítat $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha^k (\Delta x^k, \Delta \lambda^k, \Delta s^k)$,

kde α_k bylo vybráno tak aby $(x^{k+1}, s^{k+1}) > 0$.

end while

Algoritmus 3.1: Základní algoritmus primárně duální metody vnitřního bodu

3.3 Míra duality

Řešení úlohy pomocí algoritmu 3.1 má tu nevýhodu, že většinou můžeme udělat pouze malý krok tak, aby nebyla porušena podmínka nezápornosti. Z toho důvodu není tento způsob řešení příliš vhodný, protože konverguje velmi pomalu. Proto se používají různé mechanismy, pomocí kterých tento základní algoritmus modifikujeme tak, abychom byli schopni dosáhnout rychlejší konvergence. Prvním takovým mechanismem je zatížení hledání směru k dalšímu řešení, směrem dovnitř oblasti přípustných řešení, tak abychom mohli udělat co nejdelší krok bez porušení omezení $(x, s) \geq 0$. Dalším mechanismem je snaha udělat co nejdelší krok směrem k hledanému optimálnímu řešení. Tyto dva zmíněné mechanismy jsou navzájem protichůdné a proto se snažíme nalézt nějaký kompromis, který by dokázal tyto mechanismy vzájemně kombinovat. Vhodným kompromisem je udělat v daném kroku algoritmu co nejdelší krok tak, aby nalezené řešení bylo zatíženo co nejvíce do středu oblasti přípustnosti a tím byla větší šance pro nalezení co nejdelšího kroku v následující iteraci algoritmu. Za tímto účelem je vhodné zvolit nějaké kritérium úspěšnosti aktuálního kroku. Důležitým faktorem úspěšnosti aktuálně dosaženého řešení je vzdálenost tohoto řešení od hledaného optimálního řešení. Pro tento účel je vhodné použít takzvanou míru duality, kterou definujeme jako

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i = \frac{x^T s}{n} . \qquad 3.12$$

Tato míra udává průměr součinů $x_i s_i$ po složkách. Lze ji využít nejen jako kritérium konvergence, ale její změnu mezi jednotlivými kroky, lze využít jako míru úspěšnosti aktuálního kroku. Míru duality využívají primárně duální algoritmy také k řízení aktuálního kroku, který se nesnaží mířit přímo na optimální řešení, ale spíše do bodu, ve kterém bude aktuální míra duality snížena na nějakou požadovanou hodnotu. Tímto se právě snažíme zatížit směr hledání dovnitř oblasti přípustných řešení. Budeme-li uvažovat o nějaké konkrétní požadované míře duality, které bychom

rádi dosáhli můžeme přepsat 3.5 jako $x_i s_i = \sigma\mu$, kde μ je aktuální míra duality a $\sigma \in (0,1]$ je poměr snížení, kterého chceme dosáhnout. Pro $\sigma = 1$ bude hledaný směr zatížen do středu oblasti přípustnosti, v takovém případě se nám podaří udělat pouze malý krok směrem k optimálnímu řešení. Na druhou stranu pro $\sigma = 0$ dostaneme standardní afinní krok. Zohlednění parametrů σ a μ do soustavy rovnic hledající Newtonův krok naší úlohy zapíšeme jako

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{bmatrix}. \qquad 3.13$$

Nyní můžeme rozšířit základní algoritmus primárně duální metody vnitřního bodu a využít míry duality k řízení jednotlivých iterací.

Mějme počáteční bod (x^0, λ^0, s^0), kde $(x^0, s^0) > 0$

while $\mu_k > \epsilon$

Vybrat $\sigma_k \in \langle 0,1 \rangle$

$$\mu_k = \frac{x_k^T s_k}{n}$$

Vyřešit $\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma_k \mu_k \end{bmatrix}$

Vypočítat $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha^k (\Delta x^k, \Delta \lambda^k, \Delta s^k)$,

kde α_k bylo vybráno tak, aby $(x^{k+1}, s^{k+1}) > 0$.

end while

Algoritmus 3.2: Algoritmus primárně duální metody rozšířený o míru duality

3.4 Centrální cesta

Centrální cesta a princip sledování centrální cesty je velmi důležitým prvkem primárně-duálních algoritmů vnitřního bodu, za jehož pomoci můžeme dosáhnout lepší konvergence. Proto se v této kapitole pokusíme popsat princip centrální cesty a jejího sledování.

Definujme si množinu F přípustných a množinu F_0 striktně přípustných řešení. Tyto jsou množiny bodů (x, λ, s), které vyhovují omezujícím podmínkám jak primární tak duální úlohy.

$$F = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) \geq 0\} \qquad 3.14$$

$$F^0 = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) > 0\} \qquad 3.15$$

Budeme-li mluvit o striktně přípustném řešení, budeme mít na mysli $(x, \lambda, s) \in F^0$.

Centrální cestou C nazveme množinu striktně přípustných bodů (x, λ, s), které můžeme parametrizovat pomocí τ. Potom každý takový bod $(x_\tau, \lambda_\tau, s_\tau) \in C$ splňuje následující podmínky

$$A^T \lambda + s = c, \quad 3.16$$

$$A x = b, \quad 3.17$$

$$x_i s_i = \tau, \quad i = 1,2,\ldots,n., \quad 3.18$$

$$(x, s) > 0. \quad 3.19$$

Tyto podmínky jsou odvozeny z KKT podmínek a liší pouze hodnotou τ ve vztahu 3.18. Podmínky 3.16 a 3.17 jsou podmínkami přípustnosti řešení primární a duální úlohy a podmínka 3.18 nahrazuje podmínku komplementarity z KKT, která pro optimální řešení požaduje, aby byl součin prvků vektorů $x_i s_i = 0$. Místo toho vyžadujeme, aby součin prvků vektorů byl $x_i s_i = \tau$. Odtud definujeme centrální cestu jako množinu bodů vyhovujících 3.16-3.19 parametrizovanou τ

$$C = \{(x_\tau, \lambda_\tau, s_\tau) | \tau > 0\}. \quad 3.20$$

Dá se ukázat [2], že trojice $(x_\tau, \lambda_\tau, s_\tau)$ je dána jednoznačně pro každé τ, pokud je F^0 neprázdná množina.

Dalším způsobem jak lze definovat centrální cestu je za použití rovnice 3.7, potom

$$F(x_\tau, \lambda_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}. \quad 3.21$$

Podmínky 3.16-3.18 aproximují 3.3-3.5, tato aproximace se stává tím přesnější čím více se τ blíží k 0. Proto jestliže $(x_\tau, \lambda_\tau, s_\tau) \in C$ konverguje, tím jak se τ blíží k 0, potom konverguje k optimálnímu řešení primárně duální úlohy. Centrální cesta nám tedy zajišťuje posloupnost řešení, které konvergují k optimálnímu řešení a vyhýbají se řešením, která nejsou přípustná. Dodržovaní centrální cesty nám zajišťuje udržení nezápornosti složek vektorů x a s a navíc tento postup dodržuje pokles součinu $x_i s_i$ pro každé i tak, že udržuje stejnou míru poklesu pro všechny složky řešení.

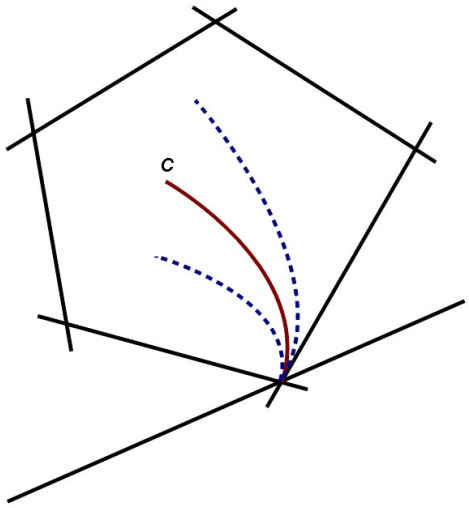
Hledání řešení ležícího na centrální cestě nám dává Newtonům krok který je zatížený do středu množiny přípustných řešení. Což má za následek možnost udělat delší krok dříve než porušíme hranici přípustných řešení. Položením $\tau = \sigma\mu$ a použitím Newtonovy metody pro rovnici 3.21 dostaneme soustavu rovnic 3.13, jejímž řešením dostaneme směr kroku k řešení $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu})$. Zvolíme-li $\sigma = 1$ potom řešení 3.13 určuje tzv. centrující krok který směřuje přímo do bodu centrální cesty $(x_\mu, \lambda_\mu, s_\mu)$. V takovém bodě je součin všech složek $x_i s_i = \mu$. Centrující směr je silně zatížen dovnitř nezáporného ortantu a zpravidla umožňuje malý nebo dokonce žádný pokrok směrem k optimálnímu řešení. Na druhou stranu pokud zvolíme $\sigma = 0$, dostaneme standardní Newtonův afinní směr. Algoritmy vnitřního bodu často volí různé hodnoty $\sigma \in (0,1]$ tak, aby co nejlépe využily protichůdných požadavků na jedné straně na snížení μ a na druhé straně přiblížení

se k centrální cestě.

Centrální cestu sledující algoritmy se v jednotlivých iteracích omezují na řešení, které leží v okolí centrální cesty. Udržováním řešení v jednotlivých iteracích dostatečně daleko od hranice nezáporného orthantu bude zajištěno, že algoritmus bude schopný zlepšit aktuální řešení alespoň o nějaký malý krok směrem k optimálnímu řešení v každém kroku. Již bylo řešeno, že aktuální přípustné řešení ležící na centrální cestě konverguje k optimálnímu řešení tak, jak τ konverguje k 0. To ovšem platí i pro jiné přípustné body, které nemusí ležet přímo na centrální cestě, ale budou ležet v blízkém okolí této cesty. Pro tento účel je vhodné definovat tzv. okolí centrální cesty. Toto okolí lze definovat různými způsoby. Ukážeme si jeden často používaný způsob, kterým toto okolí definuje jako množinu

$$N_{-\infty}(\gamma)=\{(x,\lambda,s)\in F^0 \mid x_i s_i \geq \gamma \mu \; \forall i=1,2,\ldots,n\}. \qquad 3.22$$

pro $\gamma\in(0,1]$.



Okolí 2: Centrální cesta a její okolí

Okolí centrální cesty můžeme použít jako jistý prvek pro úpravu algoritmu metody vnitřního bodu. Takový algoritmus bude potom charakterizován právě vlastnostmi použitého okolí centrální cesty. Ukažme si nyní algoritmus sledující okolí centrální cesty $N_{-\infty}(\gamma)$ s parametrem γ blízkým 0, tento algoritmus je známý jako algoritmus sledující cestu dlouhým krokem (long-step path-following). Závisí na dvou parametrech σ_{min} a σ_{max}, které nastavují spodní a horní hranici centrujícího parametru σ. Směr kroku je jako obvykle určen pomocí 3.21 a α je vybráno tak aby řešení v každé iteraci patřilo do $N_{-\infty}(\gamma)$.

Mějme γ , σ_{min} , σ_{max} , $\gamma \in (0,1)$, $0 < \sigma_{min} < \sigma_{max} < 1$ a $(x_{0,} \lambda_{0,} s_0) \in N_{-\infty}(\gamma)$

for k = 1,2, …

Vyberme $\sigma_k \in \langle \sigma_{min}, \sigma_{max} \rangle$

Řešme 3.21 a získáme $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$

Nejděme α^k tak aby

$(x^k, \lambda^k, s^k) + \alpha^k (\Delta x^k, \Delta \lambda^k, \Delta s^k) \in N_{-\infty}(\theta)$

Vypočteme $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha^k (\Delta x^k, \Delta \lambda^k, \Delta s^k)$

end for

Algoritmus 3.3: Long step path-following

3.5 Nepřípustné body

Doposud jsme předpokládali, že bod $(x_{0,} \lambda_{0,} s_0)$ byl striktně přípustným řešením. Tedy že $A x_0 = b$ a $A^T \lambda_0 + s_0 = c$ a díky tomu jak je postavena úloha hledání dalšího řešení je zaručeno, že i každé další řešení bude striktně přípustné. Bohužel nalezení striktně přípustného řešení úlohy může být časově velmi náročné. Z toho důvodu je výhodné použít algoritmus, který umí pracovat i s nepřípustnými body. Takové algoritmy zachovávají podmínku nezápornosti $(x, s) \geq 0$, nicméně vektory trojice (x_k, λ_k, s_k) nemusí vyhovovat podmínkám přípustnosti primární a duální úlohy. Takový algoritmus se potom snaží v každém kroku přiblížit aktuální řešení směrem k řešení optimálnímu a zároveň zlepšit přípustnost aktuálního řešení. K tomu nám postačí jemně modifikovat soustavu rovnic 3.13

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma \mu e \end{bmatrix}, \qquad 3.23$$

kde $r_c = A^T \lambda + s - c$ a $r_b = A x - b$. Řešením soustavy 3.23 je Newtonův krok, který směřuje směrem k bodu centrální cesty $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu})$ a navíc se snaží opravit nepřípustnost řešení v daném kroku. Pokud by se nám podařilo udělat plný krok v nalezeném směru, znamenalo by to nalézt přípustné řešení.

3.6 Konvergence metody

V této podkapitole si ukážeme, jakým způsobem můžeme dokázat konvergenci algoritmu 3.3. A ukážeme si, že algoritmus bude konvergovat v nejhorším případě v $O(n|\log\epsilon|)$ krocích, kde ϵ je koeficient redukce míry duality výchozího bodu. V praxi počet kroků zpravidla roste mnohem pomaleji než dimenze problému n.

Hledáme řešení (x^k, λ^k, s^k), pro které platí $\mu_k \leq \epsilon \mu_0$ kde μ_0 je míra duality výchozího bodu (x^0, λ^0, s^0). Vhodnou volbou ϵ můžeme dosáhnout μ_k velmi blízkého 0. V takovém případě (x^k, λ^k, s^k) téměř splňují podmínky optimality a můžeme říct, že (x^k, λ^k, s^k) je s malou chybou hledaným optimálním řešením zadané úlohy.

Následující tvrzení a důkazy byly převzaty z [2].

Důkaz začneme následujícím lemmatem.

Lemma 3.1:

Mějme dva libovolné vektory $u, v \in R^n$ takové, že $u^T v \geq 0$ potom platí

$$\|UVe\|_2 \leq \frac{1}{2^{\frac{3}{2}}} \|u+v\|_2^2, \tag{3.24}$$

kde $U = diag(u_1, u_2, \ldots, u_n)$, $V = diag(v_1, v_2, \ldots, v_n)$.

Důkaz:

Mějme dvě libovolné hodnoty α a β, pro které platí $\alpha\beta \geq 0$. Potom platí

$$\sqrt{|\alpha\beta|} \leq \frac{1}{2}|\alpha+\beta| \ . \tag{3.25}$$

Protože $u^T v \geq 0$, platí

$$0 \leq u^T v = \sum_{u_i v_i \geq 0} u_i v_i + \sum_{u_i v_i < 0} u_i v_i = \sum_{i \in P} |u_i v_i| + \sum_{i \in M} |u_i v_i| \ , \tag{3.26}$$

kde P a M jsou množiny indexů, pro které platí

$$P\{i | u_i v_i \geq 0\} \ , \ M\{i | u_i v_i < 0\} \ . \tag{3.27}$$

Nyní položme

$$\begin{aligned}
\|UVe\| &= \left(\left\| [u_i v_i]_{i\in P} \right\|^2 + \left\| [u_i v_i]_{i\in M} \right\|^2 \right)^{1/2} \\
&\le \left(\left\| [u_i v_i]_{i\in P} \right\|_1^2 + \left\| [u_i v_i]_{i\in M} \right\|_1^2 \right)^{1/2} && \Leftarrow \|x\|_2 \le \|x\|_1 \\
&\le \left(2 \left\| [u_i v_i]_{i\in P} \right\|_1^2 \right)^{1/2} && \Leftarrow \text{ 3.26} \\
&\le \sqrt{2} \left\| [\frac{1}{4}(u_i + v_i)^2]_{i\in P} \right\|_1 && \Leftarrow \text{ 3.25} \\
&= 2^{1/2} 2^{-2} \sum_{i\in P} (u_i + v_i)^2 \\
&\le 2^{-3/2} \sum_{i=1}^{n} (u_i + v_i)^2 \\
&\le 2^{-3/2} \left\| u_i + v_i \right\|^2
\end{aligned}$$

c.b.d.

V dalším kroku vezmeme řešení Δx a Δs soustavy z 3.23 a vyřkneme následující tvrzení.

Lemma 3.2:

Buď $(x, \lambda, s) \in N_{-\infty}(\gamma)$ *, potom*

$$\|\Delta X \Delta S e\| \le 2^{-\frac{3}{2}} (1 + \frac{1}{\gamma}) n \mu , \qquad 3.28$$

kde $\Delta X = diag(\Delta x_1, \Delta x_2, \ldots, \Delta x_n)$, $\Delta S = diag(\Delta s_1, \Delta s_2, \ldots, \Delta s_n)$.

Důkaz:

Z 3.23 lze ukázat, že

$$\Delta x^T \Delta s = 0 . \qquad 3.29$$

Násobením posledního blokového řádku 3.23 výrazem $(XS)^{-1/2}$ a použitím rovnosti $D = X^{1/2} S^{-1/2}$, dostaneme

$$D^{-1} \Delta x + D \Delta s = (XS)^{-1/2} (-XSe + \sigma \mu e) . \qquad 3.30$$

Protože $(D^{-1}\Delta x)^T(D\Delta s)=\Delta x^T\Delta s=0$ můžeme použít Lemma 3.1 kde $u=D^{-1}\Delta x$ a $v=D\Delta s$ a dostaneme

$$\begin{aligned}\|\Delta X\Delta S e\| &= \|(D^{-1}\Delta X)(D\Delta S)e\| \\ &\le 2^{-3/2}\|D^{-1}\Delta x+D\Delta s\|^2 \\ &= 2^{-3/2}\|(XS)^{-1/2}(-XSe+\sigma\mu e)\|^2 .\end{aligned}$$

Mějme $x^T s=n\mu$ a $e^T e=n$, potom výše uvedenou normu můžeme rozepsat

$$\begin{aligned}\|\Delta X\Delta S e\| &\le 2^{-\frac{3}{2}}\left[x^T s-2\sigma\mu e^T e+\sigma^2\mu^2\sum_{i=1}^{n}\frac{1}{x_i s_i}\right] \\ &\le 2^{-\frac{3}{2}}\left[x^T s-2\sigma\mu e^T e+\sigma^2\mu^2\frac{n}{\gamma\mu}\right] \\ &\le 2^{-\frac{3}{2}}\left[1-2\sigma+\frac{\sigma^2}{\gamma}\right]n\mu \\ &\le 2^{-\frac{3}{2}}\left(1+\frac{1}{\gamma}\right)n\mu .\end{aligned}$$

c..b.d.

Dále si dokážeme následující větu.

Věta 3.1

V algoritmu 3.3 mějme parametry γ, σ_{min} a σ_{max}, potom existuje konstanta δ, taková že pro každé $k\ge 0$ nezávisle na n bude platit

$$\mu_{k+1}\le\left(1-\frac{\delta}{n}\right)\mu_k . \qquad 3.31$$

Důkaz:

Nejdříve dokážeme, že platí

$$(x^k(\alpha),\lambda^k(\alpha),s^k(\alpha))\in N_{-\infty}(\gamma) \qquad 3.32$$

pro každé $\alpha\in\left[0,2^{\frac{3}{2}}\gamma\frac{1-\gamma}{1+\gamma}\frac{\sigma_k}{n}\right]$.

Zde $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha))$ je definováno jako $(x^{k+1}, \lambda^{k+1}, s^{k+1})$ z 3.11. Z uvedeného vyplývá, že délka kroku α_k je maximálně tak velká, jako je horní hranice uvedeného intervalu, to je

$$\alpha_k \le 2^{\frac{3}{2}} \gamma \frac{1-\gamma}{1+\gamma} \frac{\sigma_k}{n} . \qquad 3.33$$

Podle lemmatu 3.2 platí pro každé $i = 1, 2, \ldots, n$

$$|\Delta x_i^k \Delta s_i^k| \le \|\Delta X^k \Delta S^k e\|_2 \le 2^{-\frac{3}{2}} \left(1 + \frac{1}{\gamma}\right) n \mu_k \qquad 3.34$$

Z 3.23 vyplývá $x_i^k s_i^k \ge \gamma \mu_k$ a podle 3.34

$$\begin{aligned} x_i^k(\alpha) s_i^k(\alpha) &= (x_i^k + \alpha \Delta x_i^k)(s_i^k + \alpha \Delta s_i^k) \\ &= x_i^k s_i^k + \alpha (x_i^k \Delta s_i^k + s_i^k \Delta x_i^k) + \alpha^2 \Delta x_i^k \Delta s_i^k \\ &\ge x_i^k s_i^k (1-\alpha) + \alpha \sigma_k \mu_k - \alpha^2 |\Delta x_i^k \Delta s_i^k| \\ &\ge \gamma (1-\alpha) \mu_k + \alpha \sigma_k \mu_k - \alpha^2 2^{-\frac{3}{2}} \left(1 + \frac{1}{\gamma}\right) n \mu_k . \end{aligned}$$

Sečtením všech n komponent třetího řádku 3.23 a použitím 3.29 dostaneme

$$\mu(\alpha) = (1 - \alpha(1 - \sigma_k)) \mu_k .$$

Z posledních dvou uvedených vztahů, můžeme vidět, že následující podmínka

$$x_i(\alpha) s_i(\alpha) \ge \gamma \mu_k(\alpha)$$

je splněna, když platí

$$\gamma(1-\alpha)\mu_k + \alpha \sigma_k \mu_k - \alpha^2 2^{-\frac{3}{2}} \left(1 + \frac{1}{\gamma}\right) n \mu_k \ge \gamma(1 - \alpha + \alpha \sigma_k) \mu_k .$$

Přeuspořádáním výrazu dostaneme

$$\alpha \sigma_k \mu_k (1-\gamma) \ge \alpha^2 2^{-\frac{3}{2}} n \mu_k \left(1 + \frac{1}{\gamma}\right),$$

což platí, jestliže platí $\alpha_k \le 2^{\frac{3}{2}} \gamma \frac{1-\gamma}{1+\gamma} \frac{\sigma_k}{n}$.

Dokázali jsme, že $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha))$ leží v $N_{-\infty}(\gamma)$ pokud α leží v intervalu definovaném 3.32.

Důkaz dokončíme odhadem zmenšení μ v k-tém kroku. Protože platí 3.29 a 3.33 dostaneme posledního řádku 3.23

$$\begin{aligned}
\mu_{k+1} &= x_k(\alpha_k)^T s^k(\alpha_k)/n \\
&= \left[(x^k)^T s^k + \alpha((x^k)^T \Delta s^k (s^k)^T \Delta x^k) + \alpha_k^2 (\Delta x^k)^T \Delta s^k\right]/n \\
&= \mu_k + \alpha_k(-(x^k)^T s^k/n + \sigma_k \mu_k) \\
&= (1-\alpha_k(1-\sigma_k))\mu_k \\
&\le \left(1 - 2^{-\frac{3}{2}} \gamma \frac{1-\gamma}{n} \frac{1-\gamma}{1-\gamma} \sigma_k(1-\sigma_k)\right)\mu_k . \qquad 3.35
\end{aligned}$$

Protože $\sigma(1-\sigma)$ je konkávní kvadratická funkce víme, že na jakémkoliv intervalu nabývá své minimum v jednom z krajních bodů. Proto

$$\sigma_k(1-\sigma_k) \ge min\{\sigma_{min}(1-\sigma_{min}), \sigma_{max}(1-\sigma_{max})\} ,$$

pro každé $\sigma_k \in [\sigma_{min}, \sigma_{max}]$.

Důkaz dokončíme dosazením tohoto odhadu do 3.35 jako

$$\delta = 2^{-\frac{3}{2}} \gamma \frac{1-\gamma}{1-\gamma} min\{\sigma_{min}(1-\sigma_{min}), \sigma_{max}(1-\sigma_{max})\} .$$

c.b.d.

A zakončíme poslední větou, ze které vyplyne původní tvrzení o horní hranici složitosti uvedeného algoritmu.

Věta 3.2

Mějme $\epsilon \in (0,1)$ a $\gamma \in (0,1)$, předpokládejme výchozí bod v algoritmu 3.3 leží v okolí centrální cesty tedy $(x_0, \lambda_0, s_0) \in N_{-\infty}(\gamma)$. Potom existuje index $K = O(n \log 1/\epsilon)$ takový, že $\mu_k \le \epsilon \mu_0$, pro všechna $k > K$.

Důkaz:

Vztah 3.31 z minulé věty logaritmujeme a dostaneme

$$\log \mu_{k+1} \leq \log\left(1-\frac{\delta}{n}\right)+\log \mu_k \, .$$

Dosazením μ_k opakovaně dostaneme

$$\log \mu_{k+1} \leq k \log\left(1-\frac{\delta}{n}\right)+\log \mu_0 \, .$$

Použijeme následující odhad logaritmické funkce $\log(1+\beta) \leq \beta$, pro každé $\beta > -1$, a vyplyne

$$\log\left(\frac{\mu_k}{\mu_0}\right) \leq k\left(-\frac{\delta}{n}\right) \, .$$

Proto podmínka $\frac{\mu_k}{\mu_0} \leq \epsilon$ je splněno, pokud platí $k\left(-\frac{\delta}{n}\right) \leq \log \epsilon$.

Tato nerovnost platí pro všechna k, pro která platí

$$k \geq K = \frac{n}{\delta} \log \frac{1}{\epsilon} = \frac{n}{\delta} |\log \epsilon| \, .$$

c.b.d.

3.7 Mehrotra prediktor-korektor

Nejslabším článkem algoritmů metod vnitřního bodu je výpočet směru k dalšímu řešení v každém kroku. K tomu je potřeba nalézt řešení soustavy rovnic pro určení Newtonova kroku. Pokud toto řešení hledáme například pomocí faktorizace úlohy, je tato faktorizace zpravidla nejdražší operací v každém kroku. Proto je výhodné, aby se další řešení v nalezeném směru co nejvýhodněji přiblížilo k hledanému řešení. Snaha nalézt nejvýhodnější následující bod vede k celkovému snížení počtu kroků a tedy k menšímu počtu provedení nejdražší operace. Metoda Mehrotra prediktor-korektor spočívá ve dvoukrokovém zpracování každé iterace. V prvním kroku nalezneme afinní směr (predikce) a v druhém kroku tento směr opravíme tak, aby se zlepšila výhodnost řešení v nalezeném směru (korekce).

V prvním kroku určíme afinní směr řešením soustavy

$$\begin{bmatrix} 0 & A_T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta \lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix} . \qquad 3.36$$

Pokud uděláme plný krok tímto směrem dostaneme

$$\begin{aligned} &(x_i + \Delta x_i^{aff})(s_i + \Delta s_i^{aff}) = \\ &= x_i s_i + x_i \Delta s_i^{aff} + s_i \Delta x_i^{aff} + \Delta x_i^{aff} \Delta s_i^{aff} = \\ &= \Delta x_i^{aff} \Delta s_i^{aff} . \end{aligned} \qquad 3.37$$

Dostaneme tedy nějakou odchylku od ideálního řešení ve kterém by $x_i s_i = 0$. Tuto odchylku se můžeme pokusit opravit. Hledáme tedy směr korekce řešením soustavy

$$\begin{bmatrix} 0 & A_T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{cor} \\ \Delta \lambda^{cor} \\ \Delta s^{cor} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Delta X^{aff} \Delta S^{aff} e \end{bmatrix} . \qquad 3.38$$

Již taková úprava směru z pravidla vylepšuje míru duality dosaženou v jednom kroku algoritmu. Pro ještě lepší opravu afinního směru můžeme použít princip sledování cesty, popsaného v předchozí podkapitole. V praxi se využívá algoritmů s proměnlivým centrujícím parametrem σ_k v každém kroku. První krok prediktor-korektor algoritmu se využívá právě pro predikci zlepšení míry duality aktuálně hledaného řešení a na základě tohoto zlepšení, můžeme upravit parametr σ tak, aby se podle potřeby rozšiřovalo okolí centrální cesty. Podstatné zlepšení míry duality znamená, že aktuální řešení není potřeba příliš centrovat a σ_k může být malé. Naopak malý pokrok vyžaduje zvětšení centrování a tím hledání řešení, ze kterého bude větší pravděpodobnost na větší pokrok v dalším kroku.

Pro určení σ_k využijeme míru duality vypočítanou po určení afinního směru. Nalezneme α_{pri} a α_{dual} tak, aby nebyla narušena podmínka nezápornosti a určíme μ_{aff}

$$\mu_{aff} = (x + \alpha_{aff}^{pri} \Delta x^{aff})(s + \alpha_{aff}^{dual} \Delta s^{aff}) / n . \qquad 3.39$$

Centrující parametr σ_k je vybírán pomocí heuristiky. Často se používá hodnota

$$\sigma = \left(\frac{\mu_{aff}}{\mu} \right)^3 , \qquad 3.40$$

která se v praxi osvědčila, ale nemá žádný analytický základ. Po určení parametru σ můžeme přistoupit k určení opraveného směru a to korekcí, kterou jsme si ukázali v 3.38. Zkombinujeme-li 3.38 a 3.23, můžeme korekci nahradit trochu sofistikovanějším směrem doplněným o korekci s určením centrujícího parametru. Takové určení směru shrnuje následující soustava rovnic

$$\begin{bmatrix} 0 & A_T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe - \Delta X^{aff} \Delta S^{aff} e + \sigma \mu e \end{bmatrix}. \qquad 3.41$$

Tato soustava kombinuje všechny předchozí kroky, tedy predikci, korekci i zohlednění centrujícího parametru. Všimněme si, že matice pro výpočet obou směrů, afinního pro určení centrování i výsledného opraveného směru, jsou totožné. To má za následek, že faktorizaci, která je zapotřebí pro výpočet této soustavy, je potřeba dělat pouze jednou v každém kroku.

Nyní si můžeme ukázat algoritmus který kombinuje všechny dosavadní poznatky a který bude základem pro naší praktickou implementaci.

Mějme počáteční bod (x^0, λ^0, s^0) , kde $(x^0, s^0) > 0$

for k = 1,2, …

Řešením 3.36 a získáme $(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff})$

Vypočteme α_{pri}^{aff} , α_{dual}^{aff} a μ_{aff}

Nastavme $\sigma = (\mu_{aff} / \mu)^3$

Řešením 3.41 a získáme $(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff})$

Určíme α_{pri}^{k} , α_{dual}^{k}

Vypočteme

$$x^{k+1} = x^k + \alpha_{pri}^k \Delta x^k$$

$$(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + \alpha_{dual}^k (\Delta \lambda^k, \Delta s^k)$$

end for

Algoritmus 3.4: Mehrotra prediktor-korektor

4. Praktická implementace algoritmu

Praktické implementace většiny algoritmů se mnohdy mohou lišit od teorie ze které vychází, zpravidla se využívá různých optimalizací a heuristik, které se v praxi osvědčily. Nezřídka se ignorují některé teoretické předpoklady. V předchozí teoretické části jsme se zaměřili na primárně duální metodu vnitřního bodu a nyní si popíšeme praktickou implementaci jejího algoritmu. Postupně si popíšeme jednotlivé kroky její implementace. Pokud daný krok lze implementovat různými způsoby, pokusíme se popsat výhody a nevýhody různých přístupů.

4.1 Nalezení výchozího řešení

Nalezení výchozího bodu je velmi důležitým krokem pro úspěšné řešení úlohy a často má vliv úspěšnost i stabilitu algoritmu. Špatná volba výchozího bodu může mít podstatný vliv na konvergenci algoritmu.

Pokusíme se popsat způsob jakým nalézt vhodný výchozí bod, který nebude porušovat podmínky nezápornosti $(x, s) \geq 0$ a zároveň nebude nabývat příliš velkých hodnot, což by mělo za následek velkou vzdálenost od optimálního řešení díky $\frac{x^T s}{n} = \mu$.

Už jsme si vysvětlili, že při použití vhodného algoritmu nemusí aktuální řešení být striktně přípustné. Nicméně hledání výchozího bodu začneme tak, že nalezneme vektory (x, λ, s) tak, aby platilo $Ax = b$ a $A^T \lambda + s = c$. Zároveň budeme vyžadovat, aby vektory x a s měly co nejmenší normu.

To je hledáme řešení

$$\min_x \frac{1}{2} x^T x \text{ , kde platí } Ax = b \qquad 4.1$$

a

$$\min_{\lambda s} \frac{1}{2} s^T s \text{ , kde platí } A\lambda + s = c . \qquad 4.2$$

Můžeme ukázat, že řešení těchto problému lze zapsat následujícím způsobem

$$x = A^T (AA^T)^{-1} b , \qquad 4.3$$

$$\lambda = (AA^T)^{-1} Ac , \qquad 4.4$$

$$s = c - A^T \lambda . \qquad 4.5$$

Tímto získáme bod, který sice vyhovuje omezením úlohy, ale není zaručeno, že nebude porušena podmínka nezápornosti. Proto posuneme vektory x a s tak, aby byly nezáporné. Použijeme

$$\tilde{x} = x + \delta_x e \text{ , kde } \delta_x = max(\alpha \min_i x_i, 0) \qquad 4.6$$

a

$$\tilde{s} = s + \delta_s e \text{ , kde } \delta_s = max(\alpha \min_i s_i, 0) \text{ , } \qquad 4.7$$

kde $\alpha < -1$ zvolíme $-3/2$ a $e = (1,1, ...,1)^T$ je jednotkový vektor dimenze n .

Nyní již máme $\tilde{x} \geq 0$ a $\tilde{s} \geq 0$. Další úpravou se pokusíme vyvážit oba vektory tak, aby jejich hodnoty byly pokud možno rozprostřeny okolo nějaké stejné hodnoty je to určitý způsob vycentrování. Vypočteme vyvažovací koeficienty tak, že nalezneme průměrnou velikost prvků vektorů $\tilde{x}$ vyváženou vzhledem k $\tilde{s}$, a naopak průměr prvků $\tilde{s}$ vzhledem k $\tilde{x}$. Vyvažovací koeficienty určíme jako

$$\tilde{\delta}_x = \frac{1}{2} \frac{\tilde{x}^T \tilde{s}}{e^T \tilde{s}} \text{ , } \qquad 4.8$$

$$\tilde{\delta}_s = \frac{1}{2} \frac{\tilde{s}^T \tilde{x}}{e^T \tilde{x}} \text{ . } \qquad 4.9$$

A nyní vypočteme počáteční vektory x^0 , λ^0 a s^0

$$x^0 = \tilde{x} + \tilde{\delta}_x e \text{ , } \lambda^0 = \lambda \text{ a } s^0 = \tilde{s} + \tilde{\delta}_s e \text{ . } \qquad 4.10$$

Složitost nalezení výchozího bodu přibližně odpovídá jedné iteraci algoritmu. Což můžeme vidět, když si uvědomíme, že musíme vyřešit soustavu $(AA^T)^{-1} b$ a to je vlastně řešení $A I A^T x = b$. Tedy velmi podobný problém, jaký řešíme v každé iteraci algoritmu metody vnitřního bodu. Tento problém můžeme řešit například pomocí LU rozkladu.

4.2 Hledání délky kroku

Nejjednodušším, ale z algoritmického hlediska velmi důležitým krokem je určení délky kroku v daném nalezeném směru. Jde nám o to v daném kroku nalézt řešení, které bude použito jako výchozí bod pro další iteraci. Toto řešení nemusí splňovat kritéria přípustnosti, tedy nemusí byt striktně přípustné, ale musí striktně splňovat podmínky nezápornosti $(x, s) > 0$.

Mějme aktuální řešení (x^k, λ^k, s^k) kde $(x^k, s^k) > 0$ a nalezený Newtonův směr $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$, potom je jednoduché určit α_p a α_d takové aby $x^k + \alpha_p \Delta x^k > 0$, $s^k + \alpha_d \Delta s^k > 0$.

Mějme α_{pmax} a α_{dmax}

$$\alpha_{pmax} = max\left(\min_{i:\Delta x_i^k<0} -\frac{x_i^k}{\Delta x_i^k} \right), \qquad 4.11$$

$$\alpha_{dmax} = max\left(\min_{i:\Delta s_i^k<0} -\frac{s_i^k}{\Delta s_i^k} \right). \qquad 4.12$$

Koeficienty α_p a α_d budeme vybírat z intervalu $\alpha_p \in (0, \alpha_{pmax})$ a $\alpha_d \in (0, \alpha_{dmax})$.
Řešení $(x^{k+1}, \lambda^{k+1}, s^{k+1})$ získáme takto

$$x^{k+1} = x^k + \alpha_p^k \Delta x^k$$

a

$$(\lambda^{k+1}, s^{k+1}) = (\lambda^k + \alpha_d^k \Delta \lambda^k, s^k + \alpha_d^k \Delta s^k). \qquad 4.13$$

Směr $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$ odstraňuje chybu nepřípustnosti aktuálního řešení primární i duální úlohy

$$A \Delta x^k = -r_b^k = -A x^k + b$$

a

$$A^t \Delta \lambda^k + \Delta s^k = -r_c^k = -A^T \lambda^k - s^k + c.$$

Snadnou úpravou vidíme, že

$$r_b^{k+1} = (1-\alpha_p^k) r_b^k \text{ a } r_c^{k+1} = (1-\alpha_d^k) r_c^k. \qquad 4.14$$

Odtud lze také vidět, že hledané α_p a α_d nesmějí být větší jedné.
Pro praktickou implementaci můžeme použít následující formule

$$\hat{\alpha}_p^k = min(1, \eta \alpha_p^k), \ \hat{\alpha}_d^k = min(1, \eta \alpha_d^p), \qquad 4.15$$

kde parametr η volíme z intervalu $\eta \in [0.9, 1)$. Je vhodné volit η tak, že se bude zvyšovat, jak se bude řešení přibližovat optimálnímu řešení.

4.3 Nalezení Newtonova směru

Nejsložitější operací v každém kroku primárně-duálního algoritmu je řešení soustavy lineárních rovnic definované v 3.41. Matice soustavy je v tomto případě obvykle velká a řídká. Soustava definovaná v 3.41 je dimenze $2n+m$. Díky struktuře úlohy 3.41, ale můžeme tuto úlohu přeformulovat tak, že výsledná úloha bude menší, přesněji dimenze m a navíc matice této soustavy bude symetrická, normální a pozitivně definitní. Symetrie a pozitivní definitnosti lze potom využít při volbě způsobu řešení soustavy.

Úlohu 3.41 v její základní podobě přepíšeme jako

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -r_{xs} \end{bmatrix}. \qquad 4.16$$

Matice X a S jsou regulární matice, to víme z jejich konstrukce, jde o diagonální matice vytvořené ze striktně kladných vektorů. Proto můžeme eliminovat Δs tím, že vynásobíme třetí řádek maticí $-X^{-1}$ a přičteme jej k prvnímu. Dostaneme

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_c + X^{-1} r_{xs} \\ -r_b \end{bmatrix}, \qquad 4.17$$

$$\Delta s = -X^{-1} r_{xs} - X^{-1} S \Delta x, \qquad 4.18$$

$$D = S^{-\frac{1}{2}} X^{\frac{1}{2}}. \qquad 4.19$$

V dalším kroku vynásobíme první řádek maticí AD^2 a přičteme jej k druhému. Dostaneme

$$A D^2 A^T \Delta \lambda = -r_b - AXS^{-1} r_c + AS^{-1} r_{xs} \qquad 4.20$$

$$\Delta s = -r_c - A^T \Delta \lambda \qquad 4.21$$

$$\Delta x = -S^{-1} r_{xs} - X S^{-1} \Delta s, \qquad 4.22$$

kde 4.21 je odvozeno přímo z 4.16.

Touto úpravou jsme původní problém dimenze $2n+m$ redukovali na soustavu rovnic dimenze m. Navíc matice $A D^2 A^T$ bude pozitivně definitní normální maticí.

Řešení soustavy rovnic 4.20 můžeme hledat různými způsoby, které si rozdělíme do dvou kategorií:

- Přímé metody
  - Choleského rozklad
  - LU rozklad
- Iterační metody
  - Metoda sdružených gradientů

Přímé metody spočívají v přesném nalezení řešení soustavy rovnic. Nejčastěji se hledá řešení pomocí faktorizace problému na trojúhelníkové, popřípadě diagonální matice jejichž řešení je snadné. Takové řešení dává přesné výsledky a možnost pokračovat nalezeným směrem k přímo hledanému řešení.

Druhý způsob je založený na použití některé z iteračních metod, která hledá jenom přibližné řešení soustavy 4.20. To může být efektivnější než úplná faktorizace soustavy. Protože řešením soustavy rovnic hledáme směr, ve kterém se potřebujeme přiblížit k optimálnímu řešení úlohy, může za určitých podmínek takové přibližné řešení postačovat.

V následujících podkapitolách jednotlivě rozebereme tyto metody a blíže popíšeme jejich výhody a nevýhody. Jedním společným jmenovatelem, který vychází z definice soustavy 4.20, je špatná podmíněnost matice $A D^2 A^T$ způsobena tím jak se blížíme k optimálnímu řešení a diagonála definovaná 4.19 obsahuje buď čísla extrémně malá odpovídající prvkům $x_i \to 0$ nebo naopak čísla extrémně velká odpovídající prvkům $s_i \to 0$. Toto společně se zaokrouhlovacími chybami počítačové aritmetiky způsobuje problémy se stabilitou a může vést k selhání algoritmu při řešení soustavy 4.20, což může vést ke zkresleným výsledkům úlohy a dokonce úplnému selhání konvergence algoritmu.

4.3.1 Choleského rozklad

Nejznámějším a nejčastěji popisovaným způsobem řešení soustavy rovnic $A D^2 A^T x = b$ na jejímž řešení jsou postaveny metody vnitřního bodu je Choleského rozklad. Prostředí Matlab nám přímo poskytuje balík funkcí s efektivní implementací Choleského rozkladu a to i pro řídké matice. Proto použití Choleského rozkladu, pokud uvažujeme o implementaci v prostředí Matlab, je z hlediska implementace výhodné.

Problém tohoto řešení může být v již zmíněném velmi špatném podmínění matice $A D^2 A^T$ ve chvíli kdy se nacházíme velmi blízko k optimálnímu řešení a na diagonále se objevují čísla buď velmi malá nebo velmi velká. Díky nepřesnostem počítačové aritmetiky dochází k chybám, které mohou vést k selhání rozkladu z důvodu porušení positivní definitnosti matice soustavy. K tomu dochází v případech, kdy klademe velké nároky na přesnost řešení.

Hledání řešení soustavy rovnic pomocí Choleského rozkladu probíhá následujícím způsobem. Mějme soustavu $A D^2 A^T x = b$, matici $A D^2 A^T$ pomocí Choleského rozkladu rozložíme

$$A D^2 A^T = L L^T, \tag{4.23}$$

máme tedy

$$A D^2 A^T x = L L^T x = b, \tag{4.24}$$

odkud použitím zpětné substituce na horní trojúhelníkovou matici L^T a dopředné substituce na dolní trojúhelníkovou matici L dostaneme

$$x = L^T \backslash y, \quad y = L \backslash z \text{ a } z = P b.$$

4.3.2 LU rozklad

LU rozklad nemusí tak efektivní jako zmíněný Choleského rozklad. Teoretická složitost LU rozkladu je udávaná jako O($2n^3/3$), na rozdíl od složitosti Choleského rozkladu, která je udávaná jako O($n^3/3$). Přesto i tento způsob řešení jistě najde své uplatnění zejména při zpracování řídkých matic, kdy se složitost rozkladu výrazně liší od horní teoretické hranice. Další jeho předností může být větší odolnost proti zaokrouhlovacím chybám vzniklým díky nepřesnosti počítačové aritmetiky. Tím je myšleno hlavně to, že ve chvílích kdy Choleského rozklad selže z důvodu porušení positivní definitnosti, LU rozklad proběhne a budeme schopni nalézt řešení soustavy. Ale i tento způsob má své limity a při dosažení určité blízkosti optimálního řešení, se stávají trojúhelníkové matice získané z rozkladu singulární a tento způsob implemntace také přestává fungovat. Stejně tak jako u Choleského rozkladu i pro tuto metodu existuje velmi slušná podpora v prostředí Matlab, která také zohledňuje efektivitu pro řídké matice.

Hledání řešení soustavy rovnic pomocí LU rozkladu probíhá následujícím způsobem. Mějme soustavu $A D^2 A^T x = b$, matici $A D^2 A^T$ pomocí LU rozkladu rozložíme

$$P A D^2 A^T = L U$$

máme tedy

$$P A D^2 A^T x = LU x = P b,$$

odkud použitím zpětné substituce pro horní trojúhelníkovou matici U a dopředné substituce pro dolní trojúhelníkovou matici L dostaneme

$$x = U \backslash y, \quad y = L \backslash z \text{ a } z = P b.$$

4.3.3 Metoda sdružených gradientů

Dalším způsobem, kterým lze řešit soustavu rovnic $A D^2 A^T x = b$ je metoda sdružených gradientů. Tento způsob řešení soustavy rovnic patří mezi iterační metody. Výhoda tohoto přístupu spočívá v tom, že se snaží nahradit nákladnou operaci řešení soustavy rovnic, hledáním přibližného řešení efektivní iterační metodou. Dá se očekávat, že v některých případech, obvzláště pro velká n, nebude nalezení řešení tak nákladné jako faktorizace. Bohužel tento způsob má své problémy, které se nám zatím nepodařilo úplně vyřešit. První problém souvisí s již zmíněnou vzrůstající podmíněností soustavy, která roste s tím jak se mění diagonála soustavy 4.20. Tento problém mívá zpravidla za následek, že metoda sdružených gradientů není schopna konvergovat ani k přibližnému řešení, které by vyhovovalo dovolené toleranci. Toto se dá řešit pomocí vhodného předpodmínění. Bohužel se ukazuje, že není snadné nalézt vhodný předpodmiňovač a už vůbec se nám nepodařilo najít jednoduchý obecný předpodmiňovač, který by fungoval na různé problémy. Velkým problémem s předpodmíněním úlohy je to, že levné a jednoduché předpodmiňovače s rostoucí podmíněností selhávají a složitější předpodminovače mohou být natolik výpočetně náročné, že efektivitu této metody degradují a staví jí svojí efektivitou za zmíněné přímé metody.

4.3.4 Ortogonalizace úlohy

Jednou z myšlenek, kterou jsme chtěli uplatnit pro řešení pomocí metody sdružených gradientů, byla ortogonalizace matice omezení. Předpokladem bylo, že takovou úpravou dostaneme soustavu rovnic 4.20 ve vhodnějším tvaru jako součin matic $Q^T D^2 Q$ (ortogonální-diagonální-ortogonální). Tím jsme doufali, že bude snadnější nalézt řešení takto strukturované soustavy, tedy že díky ortogonalitě matice Q bude metoda sdružených gradientů lépe konvergovat.

Ortogonalizaci úlohy provedeme následujícím postupem. Mějme úlohu

$$min\, c^T x \text{ , kde } Ax = b \text{ a } x \geq 0 . \qquad 4.25$$

Matici $A \in R^{m\,x\,n}$, kde $m < n$, můžeme rozložit jako

$$A^T = Q\,R \qquad \Rightarrow A = R^T Q^T ,$$

kde $Q \in R^{n\,x\,m}$ je ortogonální a $R \in R^{m\,x\,m}$ je trojúhelníková. Omezující podmínky úlohy reprezentované soustavou rovnic $Ax = b$, můžeme nyní transformovat následujícím způsobem

$$\begin{aligned} Ax = R^T Q^T x &= b \\ Q^T x &= R^{-T} b \\ Q^T x &= \hat{b} \end{aligned}$$

A úlohu 4.25 můžeme nyní přepsat takto

$$min\, c^T x \text{ , kde } Q^T x = \hat{b} \text{ a } x \geq 0 . \qquad 4.26$$

Následkem toho budeme v každé iteraci řešit soustavu rovnic 4.20 ve tvaru

$$Q^T D^2 Q \lambda = r . \qquad 4.27$$

Tato soustava rovnic nápadně připomíná soustavu rovnic předpodmíněnou ortogonálním projektorem viz. [6]. V uvedené literatuře se také dočteme o přepodmiňovacím efektu takovéto úpravy. Stručně řečeno, řádková ortogonalita matice Q^T nám zaručí, že

$$\kappa(Q^T D^2 Q) \leq \kappa(D^2) . \qquad 4.28$$

Toto by nám mohlo zaručit, že podmíněnost soustavy 4.20 bude růst pomaleji než samotná podmíněnost diagonály D^2 .

V kapitole zabývající se numerickými výsledky si ukážeme jaké výsledky tento předpodminovací efekt vykazuje. Již nyní však prozradíme, že tento způsob řešení není dostačující, protože podmíněnost soustavy 4.20 i tak narůstá do nepříjemné velikosti a konvergence metody sdružených gradientů postupně selhává tak, jak se blížíme k optimálnímu řešení úlohy. Další nevýhodou tohoto řešení je, že ortogonalizací matice A se zvýší koeficient zaplnění často původně velmi řídké matice, čímž se stávají všechny operace s takovou maticí časově náročnější. A právě řídkost matice omezujících podmínek hraje velký význam při efektivním řešení úloh LP.

4.3.5 Předpodmínění

Použijeme-li k řešení soustavy rovnic $A D^2 A^T x = b$ metodu sdružených gradientů, souvisí rychlost její konvergence s číslem podmíněnosti κ , které se dá vyjádřit jako poměr největšího a nejmenšího singulárního čísla matice. Navíc pokud je matice soustavy normální, což v našem případě je, dá se toto číslo vyjádřit jako poměr maximálního a minimálního vlastního čísla $\kappa(M) = \left| \frac{\lambda_{max}(M)}{\lambda_{min}(M)} \right|$.

Předpodmínění potom spočívá v nalezení takové matice T, pomocí které transformujeme původní soustavu tak, že číslo podmíněnosti nové transformované soustavy bude nižší.

Máme-li

$$A D^2 A^T x = b$$

potom platí

$$T A D^2 A^T x = T A D^2 A^T T^T \hat{x} = T b \text{ , kde } \hat{x} = T^{-T} x \text{ .} \qquad 4.29$$

Hledáme takové T kde

$$\kappa(T A D^2 A^T T^T) < \kappa(A D^2 A^T) \text{ .}$$

4.3.6 Báze s maximální váhou

Bází vektorového prostoru s maximální váhou, budeme rozumět bázi vektorového prostoru $R^{m x m}$ daného lineárním obalem sloupců matice $A \in R^{m x n}$ a vybranou ze sloupců matice A tak, že při výběru sloupců dáváme přednost těm sloupcům, které odpovídají prvkům diagonální matice $D \in R^{n x n}$ s vyšší hodnotou. Takto získaná báze lze vhodně použít k sestrojení předpodmiňovače pro soustavu rovnic zadanou ve formě $A D^2 A^T$. Tento předpodmiňovač byl původně navrhnut Vaydiou [4], pro řešení problémů nejméně nákladné cesty v síti a později dalšími rozšířen a zobecněn pro obecné použití při řešení úloh LP.

V anglické literatuře se taková báze označuje jako „Maximum Weight Basis (MWB)“ a my ji v následujícím textu budeme takto také označovat.

Následující algoritmus 4.1 ukazuje jakým způsobem nalezneme MWB bázi matice A přiřazenou odpovídající diagonále D .

Seřadíme prvky diagonály D jako vektor d tak, že $d_1 \geq ... = d_n$, a stejným způsobem přeskupíme sloupce matice A .

Mějme B (matici báze), prázdnou matici.

for i=1...n

  Jestliže je sloupec A_i lineárně nezávislý s prvky báze danou sloupci B , potom vložíme sloupec do báze $B = B \cup A_i$

  if $|B| = m$ break

end for

Algoritmus 4.1: Sestavení MWB

4.3.7 Předpodmínění pomocí MWB

Použití MWB pro předpodmínění spočívá v tom, že nalezneme MWB pro matici A váženou diagonálou D a pomocí této báze sestavíme transformační matici, kterou použijeme jako předpodmiňovač. Tuto bázi označíme A_B a z prvků D odpovídajících sloupcům A, které se zařadily do báze A_B, sestavíme diagonální matici, kterou označíme D_B. Jako transformační matici T pro předpodmínění soustavy $A D^2 A^T x = b$ použijeme $T = D_B^{-1} A_B^{-1}$. Podle 4.29 dostaneme

$$\begin{aligned} D_B^{-1} A_B^{-1} A D^2 A^T x &= D^{-1} A_B^{-1} b \\ D_B^{-1} A_B^{-1} A D^2 A^T A_B^{-T} D_B^{-T} D_B^T A_B^T x &= D^{-1} A_B^{-1} b \\ D_B^{-1} A_B^{-1} A D^2 A^T A_B^{-T} D_B^{-T} \hat{x} &= \hat{b}, \end{aligned} \qquad 4.30$$

kde $D_B^{-1} A_B^{-1} A D^2 A^T A_B^{-T} D_B^{-T}$ je symetrická a můžeme ji řešit metodou sdružených gradientů.

Rozepišme si matici výše transformované soustavy následujícím způsobem

$$\begin{aligned} & D_B^{-1} A_B^{-1} A D^2 A^T A_B^{-T} D_B^{-T} = \\ & = D_B^{-1} A_B^{-1} [A_B, A_N] \begin{bmatrix} D_B^2 & 0 \\ 0 & D_N^2 \end{bmatrix} \begin{bmatrix} A_B^T \\ A_N^T \end{bmatrix} A_B^{-T} D_B^{-T} \\ & = D_B^{-1} A_B^{-1} A_B D_B^2 A_B^T A_B^{-T} D_B^{-T} + D_B^{-1} A_B^{-1} A_N D_N^2 A_N^T A_B^{-T} D_B^{-T} \\ & = I + D_B^{-1} A_B^{-1} A_N D_N^2 A_N^T A_B^{-T} D_B^{-T} \\ & = I + T A_N D_N^2 A_N^T T^T \end{aligned} \qquad 4.31$$

Dá se ukázat že podmínění $\kappa(T A D^2 A^T T^T) \le K$, kde K je omezené a nezávislé na prvcích diagonály D. Ukažme si jakým způsobem se dá tato podmíněnost odhadnout. Následující tvrzení a důkazy byli převzaty z [3].

Lemma 4.1

Buď $(A, d) \in R^{m x n} x R^n_{++}$ a buď hodnost $A = m$. Předpokládejme, že B je báze s maximalní vahou (MWB) přiřazená dvojici (A, d). Dále buď $D = diag(d)$, $D_B = diag(d_B)$. Potom pro každé $j = 1, \ldots, n$

$$d_j \| D_B^{-1} B^{-1} A_j \| \le \| B^{-1} A_j \|. \qquad 4.32$$

Jako důsledek platí

$$\|D_B^{-1} B^{-1} A D\| \le \|D_B^{-1} B^{-1} A D\|_F \le \|B^{-1} A_j\|_F . \qquad 4.33$$

Důkaz:

Buď I_B množina indexů odpovídajících sloupcům A báze B a I_N množina zbylých indexů sloupců matice A nepatřících do báze B. Potom pro každé $j \in I_B$ jsou si obě strany 4.32 rovny a proto 4.32 platí. Jestliže je $j \in I_N$ potom mohu nastat dva případy: a) sloupec A_j se nedostal do báze, protože byl lineární kombinací sloupců s větší vahou již zařazených do báze. Předpokládejme, že A_j měl vstoupit do báze jako r-tý sloupec B, ale protože byl lineární kombinací předchozích $r-1$ sloupců, platí

$$B^{-1} A_j = \begin{pmatrix} u \\ 0 \end{pmatrix},$$

pro nějaké $u \in R^{r-1}$. Platí také, že $d_i \ge d_j$ pro všechna $i \in 1, \dots, r-1$. Zjistíme, že

$$d_j \|D_B^{-1} B^{-1} A_j\| \le d_j \left(\sum_{i=1}^{r-1} \frac{u_i^2}{d_i^2} \right)^{\frac{1}{2}} \le \left(\sum_{i=1}^{r-1} u_i^2 \right)^{\frac{1}{2}} = \|u\| = \|B^{-1} Aj\| ,$$

takže 4.32 platí; b) sloupec A_j se nedostal do báze, protože odpovídá prvku diagonály s malou vahou. V tomto případě platí $d_j \le min(d_B)$ a proto platí

$$d_j \|D_B^{-1} B^{-1} A_j\| \le \frac{d_j}{min(d_B)} \|B^{-1} A_j\| \le \|B^{-1} A_j\| \text{ , takže 4.32 platí také.}$$

Nyní můžeme dokázat 4.33. První nerovnost $\|A\| \le \|A\|_F$ je známá a druhá vychází z 4.32. Pro každé $R \in R^{m x m}$ platí $\|R\|_F^2 = \sum_{j=1}^{n} \|R_j\|^2$. Uvědomíme-li si každý sloupec $D_B^{-1} B^{-1} A D$ odpovídá $d_j D_B^{-1} B^{-1} A_j$ a aplikujeme 4.32 dostaneme

$$\|D_B^{-1} B^{-1} A D\|_F^2 = \sum_{i=1}^{n} d_j \|D_B^{-1} B^{-1} A_j\|^2 \le \sum_{i=1}^{n} \|B^{-1} A_j\|^2 = \|B^{-1} A\|_F^2$$

c.b.d.

Lemma 4.2

Buď $(A, d) \in R^{m x n} x R^{n}_{++}$ *a buď hodnost* $A = m$. *Předpokládejme, že* B *je báze s maximalní vahou (MWB) přiřazená dvojici* (A, d). *Dále buď* $D = diag(d)$, $D_B = diag(d_B)$ *a* $T = D_B^{-1} B^{-1}$. *Potom platí*

$$\kappa(TAD^2 A^T T^T) \leq \|B^{-1} A\|_F^2 .$$

Důkaz:

Podle lemmatu 4.1 platí $\|TAD\| \leq \|B^{-1} A\|_F^2$. Proto platí

$$\lambda_{max}(T A D^2 A^T T^T) = \|T A D\|^2 \leq \|B^{-1} A\|_F^2 .$$

Víme že

$$TA D^2 A^T = D^{-1_B} B^{-1} (B D_B^2 B^T + N D_N^2 N^T) B^{-T} D^{-T} = I + WW^T ,$$

kde WW^T bude positivně semidefinitní. Proto odvodíme, že $\lambda_{min}(TAD^2 A^T T^T) \geq 1$ a odtud

$$\kappa(TAD^2 A^T T^T) = \frac{\lambda_{max}}{\lambda_{min}} \leq \|B^{-1} A\|_F^2 .$$

c.b.d.

Pro účel následující věty si definujme

$$\varphi_A = max \|B^{-1} A\|_F : B \text{ je bází } A . \qquad 4.34$$

Následující věta říká, že podmíněnost matice soustavy $TAD^2 A^T T^T$ je shora omezená maximem nezávislým na diagonále D^2.

Věta 4.1

Buď $(A, d) \in R^{m x n} x R^{n}_{++}$ *a buď hodnost* $A = m$. *Předpokládejme, že* B *je báze s maximalní vahou (MWB) přiřazená dvojici* (A, d). *Dále buď* $D = diag(d)$, $D_B = diag(d_B)$ *a* $T = D_B^{-1} B^{-1}$. *Potom*

$$\kappa(TAD^2 A^T T^T) \leq \varphi_A^2 .$$

Věta 4.1vyplývá přímo z tvrzení 4.2 a definice φ_A.

Z uvedené věty vyplývá, že podmíněnost soustavy předpodmíněné pomocí MWB je shora omezená hranicí nezávislou na diagonále D^2. Toto nám dává naději, že úloha bude rozumně konvergovat i v krocích kdy samotná matice D^2 je velmi špatně podmíněné. To ovšem pouze za podmínky, že φ_A má rozumnou velikost.

Problém tohoto způsobu předpodmínění spočívá v náročnosti nalezení MWB pro zadanou matici A a vektor diagonály d. Tento problém se nám zatím nepodařilo vyřešit, ale existuje naděje, že pro určité třídy úloh může být nalezen způsob jak sestrojení MWB efektivně dosáhnout. Toto by mohlo být cílem dalšího bádání.

4.3.8 Chyba iterační metody

Jedním z dalších problémů při použití iterační metody k řešení Newtonova kroku je to, že pomocí této metody nalezneme zpravidla pouze přibližné řešení. Což má za následek, že přiblížení se optimálnímu řešení v určeném přibližném směru je oproti správnému směru zkreslené. Následkem tohoto zkreslení může být selhání konvergence algoritmu.

Máme-li přibližné řešení soustavy, potom podle 4.20 odpovídá

$$A D^2 A^T \Delta\lambda = -r_b - AXS^{-1} r_c + AS^{-1} r_{xs} + \xi , \qquad 4.35$$

kde ξ označuje chybu řešení. Potom dosazením do 4.21 dostane Δs, ve kterém je promítnuta odchylka $\Delta\lambda$. Dosazením do 4.21 dostaneme $\Delta\hat{s} = \Delta s + \nu$, kde ν je odchylka promítnuta z nepřesnosti $\Delta\lambda$. Dosazením $\Delta\hat{s}$ do 4.22 nemáme zaručeno, že bude platit $-r_b = A\Delta x$. Abychom toto zaručili upravíme 4.22 takto

$$\Delta x = -S^{-1} r_{xs} - X S^{-1} \Delta s + S^{-1}\nu , \qquad 4.36$$

kde ν je vektor odchylky splňující

$$AS^{-1}\nu = \xi . \qquad 4.37$$

Dosazením 4.21,4.22 a 4.35 dostaneme

$$\begin{aligned} A\Delta\hat{x} &= A\Delta x + S^{-1}\nu \\ &= A(-S^{-1}r_{xs} - XS^{-1}\Delta s + S^{-1}\nu) \\ &= -AS^{-1}r_{xs} - AXS^{-1}\Delta s + AS^{-1}\nu \\ &= -AS^{-1}r_{xs} + AXS^{-1}r_c + AXS^{-1}A^T\Delta\lambda + AS^{-1}\nu \\ &= -r_b + \xi - AS^{-1}\nu \end{aligned}$$

Odtud vidíme, že chyba ξ bude odstraněna právě tehdy, když $\xi - AS^{-1}\nu = 0$ jak jsme uvedli v 4.37.

Protože existuje nekonečně mnoho řešení ν můžeme vybrat některé, které bude vyhovovat. Logickou volbou může být nalezení řešení metodou nejmenších čtverců. Bohužel tohle je typ výpočtů, kterému se snažíme pro jeho náročnost vyhnout. Dobrou volbou může být $\nu = (\nu_B, \nu_N) = (S_B\tilde{B}^{-1}\xi, 0)$, kde $\tilde{B}$ volíme jako vybranou MWB použitou pro předpodmínění.

5. Numerické experimenty

V této kapitole si ukážeme praktické výsledky pokusů při řešení testovacích úloh. Pro testování naprogramovaného algoritmu jsme používali úlohy z webového repository www.netlib.org [5], které poskytuje data různých matematických problémů. Testování probíhalo v prostředí Matlab verze 7.6.0 (R2008a) na poměrně zastaralém hardwaru s procesorem Intel Pentium 4 2,8GHz a 1GB paměti. Také proto jsme se zaměřily spíše na relativní rychlosti zpracovaní v poměru na velikost a složitost úlohy než na absolutní výkon, proto tabulky ani grafy neuvádějí jednotky naměřených hodnot, které považujeme za nepodstatné.

Uvedeme výsledky všech různých implementací metody prediktor-korektor, které jsme vytvořili. Kód jednotlivých funkcí je velmi podobný, hlavní rozdíl, který tyto funkce odlišuje, je ve způsobu řešení Newtonova kroku, kde je uvedená soustava rovnic řešena buď Choleského rozkladem, LU rozkladem nebo metodou sdružených gradientů.

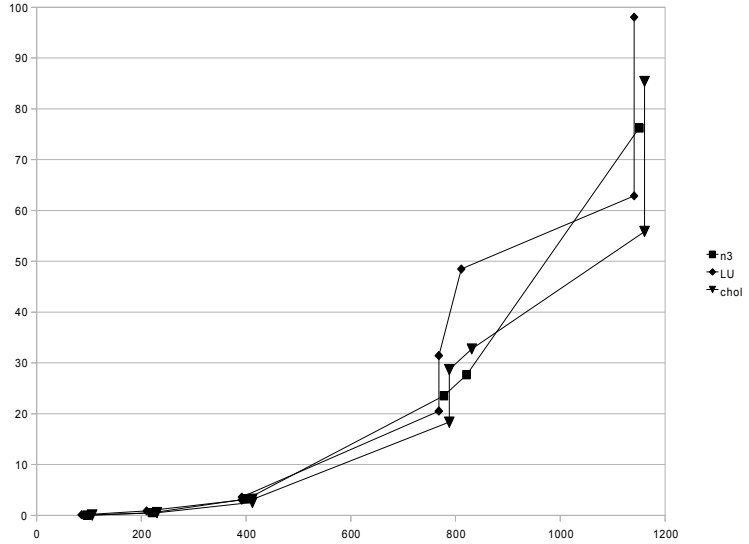
5.1 LU versus Choleského rozklad

Porovnejme nejdříve implementace algoritmů využívajících Choleského a LU rozklad. Následující tabulka 1, ukazuje dimenze omezujících podmínek a doby zpracování různých problémů. V této tabulce jsou uvedené časy zpracování úloh, které byly před zpracováním upraveny QR rozkladem, abychom odstranily závislé řádky omezení. Tato úprava pomocí QR rozkladu má za následek, že matice omezení A je trojúhelníková, ale relativně plná. Proto je i řešená soustava rovnic ve tvaru $A D A^T$ téměř plná. To má za následek, že doby zpracování jsou relativně dlouhé. Tento způsob zpracování je neefektivní a do výsledků jsme jej zařadily proto, abychom ukázali horní hranice, ke kterým se složitost algoritmu blíží.

Název úlohy	share2b	brandy	ship04s	ship04l	ship08s	ship08l	25FV47	ship12s	ship12l
počet omezení	96	220	402	402	778	778	821	1151	1151
dimenze řešení	162	303	1506	2166	2467	4363	1876	2869	5533
Doba řešení CHOL	0,06	0,53	2,53	3,15	18,34	28,67	32,73	55,85	85,37
Doba řešení LU	0,09	0,88	3,01	3,58	20,53	31,45	48,48	62,86	98,03

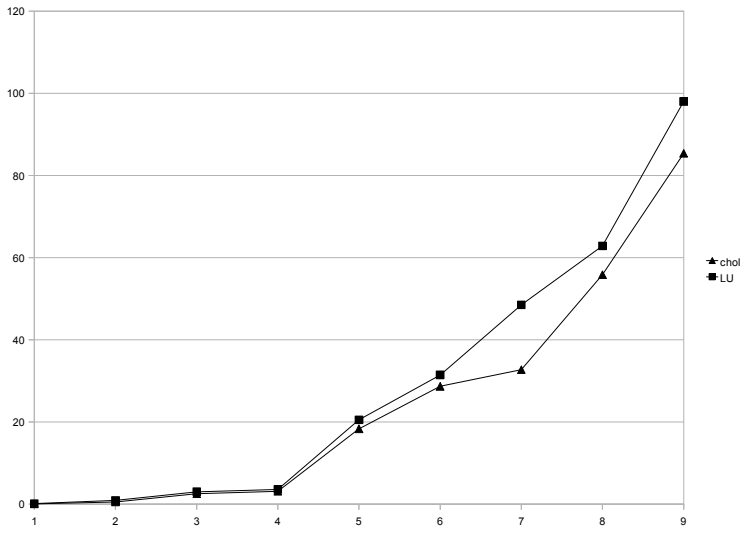
Tabulka 1: Doby zpracování úloh, Choleského a LU rozklad.

Hodnoty z tabulky 1, použije k sestrojení grafu 1, ve kterém ukážeme závislost doby zpracování na dimenzi problému a porovnáme tuto závislost s očekávaným teoretickým předpokladem. Graf 1 sleduje trend růstu doby trvání algoritmu v závislosti na počtu omezujících podmínek, tj. m pokud je dimenze matice $A \quad m \times n$. V tomto grafu jsou zaznamenané tři křivky. Jedna sledující trend růstu složitosti třídy $O(n^3)$, tu jsme do grafu zařadily proto, abychom ukázali, že trend našeho algoritmu odpovídá teoretickému předpokladu. Další dvě křivky ukazují doby zpracování oběmi metodami, tedy při použití Choleského i LU rozkladu. Z grafu lze vidět, že obě tyto metody sledují trend růstu složitosti třídy $O(n^3)$.



Graf 1: Trend doby trvání algoritmu v závislosti na počtu omezujících podmínek

Graf 2 porovnává doby zpracování úloh z tabulky 1 oběma zmíněnými způsoby. Podle teoretických odhadů je LU rozklad, při rozkladu plných matic, až dvakrát pomalejší. Z grafu 2 je však vidět, že řešení praktických úloh s využitím LU rozkladu je sice pomalejší než s využitím Choleského rozkladu, ale zdaleka ne dvakrát. I proto lze algoritmus využívající LU rozklad považovat za dobrou alternativu. Důvodem použití LU rozkladu místo Choleského rozkladu může být požadavek na dosažení větší přesnosti, ve chvílích kdy Choleského rozklad selhává kvůli nepřesnosti počítačové aritmetiky.



Graf 2: Porovnání doby zpracování Choleského a LU rozdkladem.

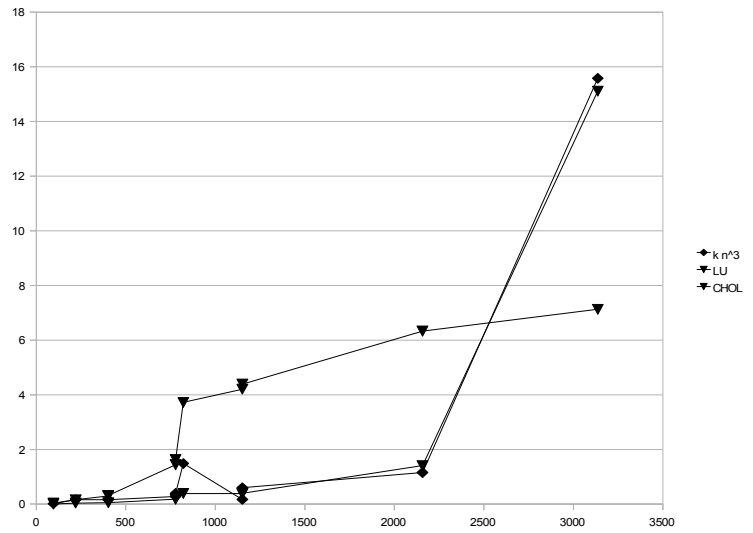
Předchozí odstavce jsou zařazeny pouze proto, abychom si ukázaly teoretickou složitost algoritmu, kterou algoritmus dosahuje při zpracování plných matic. Protože v praxi se častěji setkáme s úlohami kde matice A je řídká, ukažme si nyní výsledky takový úloh. Použijeme stejné úlohy jako

v předchozím případě, jenom pro odstranění závislých řádků použijeme LU rozkladu, který zachová řídkost matice podmínek omezení úlohy. V tabulce 2 vidíme časy zpracování těchto úloh.

Název úlohy	počet omezení	dimenze řešení	CHOL	LU
share2b	96	162	0,03	0,02
brandy	220	303	0,16	0,16
ship04s	402	1506	0,3	0,16
ship04l	402	2166	0,31	0,16
ship08s	778	2467	1,44	0,28
ship08l	778	4363	1,63	0,39
25fv47	821	1876	3,72	1,48
ship12s	1151	2869	4,2	0,17
ship12l	1151	5533	4,39	0,59
stocfor2	2157	3045	6,33	1,16
Maros-r7	3136	9408	7,13	15,58

Tabulka 2: Doby zpracování úloh, Choleského a LU rozklad (řídké matice).

Graf 3 zobrazuje data tabulky 2 a porovnává zpracování pomocí algoritmu využívajícího Choleského rozklad a algorimus využívající LU rozklad. Do grafu je rovněž zanesena křivka, která ukazuje trend růstu složitosti $k\,m^3$, kde m je počet omezení a k je konstanta zohledňující zaplnění matice A. Tato křivka nápadně kopíruje časy zpracování pomocí algoritmu využívajícího LU rozklad. Naopak při použití Choleského rozkladu se tento trend výrazně liší. Z uvedeného bychom mohli odhadovat, že použití LU rozkladu se vyplatí ve chvílích, kdy poměr zaplnění matice je nízký. Naopak při určitém zaplnění se vyplatí Choleského rozklad, který je pro rozklad hustších matic efektivnější a samotné zahuštěním matice vlivem Choleského rozkladu, již nehraje takovou roli.



Graf 3: Porovnání doby zpracování Choleského a LU rozdkladem (řídké matice)

Ještě si ukažme jaký vliv na dosažitelnou přesnost má to, jakou metodu rozkladu zvolíme pro řešení Newtonova kroku, LU nebo Choleského rozklad. V tabulce 3 vidíme 4 různé problémy, u kterých ukazujeme poslední krok, ve kterém nalezení řešení danou metodou faktorizace ještě neselhalo. Tabulka také ukazuje odchylku od požadovaného řešení a odchylky od oblasti přípustnosti pro primární i duální úlohu. Nutno podotknout, že dosažené zlepšení přesnosti není nějak závratné, a proto je na rozhodnutí uživatele a na potřebách aplikace, jestli se to vyplatí.

Název úlohy		ship04s	ship04l	ship08s	ship08l
počet omezení		402	402	778	778
dimenze řešení		1506	2166	2467	4363
μ -míra duality (podmínka ukončení)	LU	3.8209e-012	6.5408e-014	1.1502e-007	4.896e-012
	CHOL	2.8229e-006	5.0488e-008	1.1502e-007	3.3588e-008
počet iterací	LU	14	13	13	15
	CHOL	12	11	13	14
relativni odchylka optimálního řešení	LU	-2.4917e-011	-2.509e-012	-2.1484e-011	6.0443e-012
	CHOL	-2.1553e-009	-3.1456e-011	-1.6396e-010	-6.4079e-011
odchylka primární úlohy	LU	1.3515e-010	4.0453e-011	8.0463e-011	3.0057e-009
	CHOL	3.0962e-008	5.5969e-011	3.8967e-010	4.8299e-010
odchylka primární úlohy	LU	7.224e-011	1.7276e-010	9.4365e-011	5.9942e-011
	CHOL	9.4319e-011	2.9628e-008	9.3014e-011	3.711e-009

Tabulka 3: Dosažená přesnost řešení

5.2 Metoda sdružených gradientů

Pro metodu sdružených gradientů jsme zmiňovali dva způsoby, kterými jsme se snažily odstranit problémy s podmíněností soustavy rovnic řešené ve vnitřním kroku algoritmu. Šlo o předpodmínění pomocí MWB a ortogonalizaci úlohy.

5.2.1 Předpodmínení pomocí MWB

Výsledky algoritmu využívající metody sdružených gradientů, jak jsme jej implementovali pro účely našich testů, nelze přímo srovnávat s metodami z předchozí podkapitoly, protože jsou zatíženy velmi nákladným způsobem nalezení MWB předpomíňovače. A proto si zde uvedeme pouze vliv předpodmíňovače na podmínění řešené soustavy rovnic. A vliv tohoto předpodmínění na rychlost konvergence metody sdržených gradientů.

Pro nalezení řešení soustavy rovnic 4.20 metodu sdružených gradientů, byla použita vestavěná funkce prostředí Matlab PCG, která řeší soustavu rovnic metodou sdružených gradientů s předpodmíněním. Ukažme si na třech úlohách jak tato metoda byla schopna konvergovat k přibližnému řešení s požadovanou přesností. Požadovaná přesnost přibližného řešení, která se pro naše experimenty osvědčila jako dostatečná, byla ve všech příkladech byla volena $\epsilon = 1e\text{-}6$.

Tabulky 4 a 5 ukazují dva příklady, ve kterých tento způsob implementace dobře fungoval a předpodmínění matice soustavy 4.20 bylo dostačující. V těchto případech se nalezení přibližného řešení metodou sdružených gradientů vyplatilo a v některých krocích algoritmu bylo toto řešení nalezeno pomocí velmi malého počtu vnitřních iterací metody sdružených gradientů.

ship04s 404x1506	podmínění		PCG			
			predikor		korektor	
iterace	ADA'	TADA'T'	počet iteraci	přesnost	počet iteraci	přesnost
1	7146,2	1.4633e+005	92	8,97E-007	94	9,68E-007
2	17336	3518	45	7,06E-007	45	9,37E-007
3	1,08E+005	557.39	30	4,77E-007	30	8,55E-007
4	1,39E+006	767.21	28	9,95E-007	29	5,37E-007
5	1,94E+007	173.91	21	6,73E-007	21	7,63E-007
6	2,46E+008	44.26	18	6,42E-007	19	4,19E-007
7	2,78E+009	7,63	16	5,42E-007	17	5,56E-007
8	1,19E+011	7,79	15	4,55E-007	16	3,52E-007
9	2,26E+012	7,78	12	3,09E-007	14	7,73E-007
10	5,18E+013	7,77	9	8,70E-007	12	5,36E-007
11	1,77E+016	7,94	6	1,85E-007	8	4,00E-007
12	9,44E+018	8,2	2	2,57E-007	7	9,63E-007
13	5,93E+022	937.85	2	1,06E-007	2	7,15E-005

Tabulka 4: Podmínění a konvergence úlohy ship04s

Tabulka 5 ukazuje druhý příklad úlohy s větší dimezí řešení. A ukazuje, že i s rostoucí dimenzí problému podmíněnost předpodmíněné soustavy nějak závratně neroste a tato metoda potřebuje jen nepatrně vyšší počet vnitřních iterací k nalezení požadovaného přibližného řešení.

ship04l 404x2166	podmínění		PCG			
			predikor		korektor	
iterace	ADA'	TADA'T'	počet iteraci	přesnost	počet iteraci	přesnost
1	7400,2	1,49E+005	102	7,33E-007	101	5,72E-007
2	14853	3743,6	51	9,76E-007	45	7,18E-007
3	1,58E+005	429,09	28	9,27E-007	30	6,05E-007
4	2,13E+006	605,44	29	9,78E-007	30	6,17E-007
5	3,78E+007	427,4	22	6,42E-007	22	4,85E-007
6	8,68E+008	95,87	21	6,66E-007	21	9,30E-007
7	1,25E+010	22	19	6,22E-007	19	9,33E-007
8	1,29E+011	15,18	15	6,11E-007	16	7,06E-007
9	1,87E+012	19,96	13	9,81E-007	16	8,30E-007
10	3,37E+013	26,52	12	3,40E-007	15	9,51E-007
11	3,66E+015	912,22	8	8,76E-007	11	5,38E-007
12	7,33E+018	7,86	2	3,61E-008	2	3,66E-008

Tabulka 5: Podmínění a konvergence úlohy ship04l

Poslední příklad zachycený v tabulce 6, naopak demonstruje úlohu u níž uvedená metoda selhává kvůli neschopnosti metody sdružených gradientů konvergovat k řešení s požadovanou přesností. V tabulce vidíme, že přesto že metoda sdružených gradientů vyčerpala maximum kroků, ve kterém by teoreticky měla být schopna nalézt přesné řešení, nebyla schopna konvergovat ani k přibližnému řešení s požadovanou přesností $\epsilon = 1e\text{-}6$. Toto selhání konvergence je způsobeno nevhodným podmíněním v kombinaci se zaokrouhlovacími chybami počítačové aritmetiky. Přesto, že je zde aplikován MWB předpodmiňovač, podmíněnost soustavy je stále vysoká. Což je patrně zapříčiněno vysokou horní mezí φ_A definovanou 4.34. Z tohoto příkladu je také vidět, že špatná konvergence této metody není způsobena tím jak se blížíme k optimálnímu řešení, ale je špatná již od začátku algoritmu. To je patrně dáno strukturou matice A podmínek omezení úlohy.

brandy 193x303	podmínění		PCG			
			predikor		korektor	
iterace	ADA'	TADA'T'	počet iteraci	přesnost	počet iteraci	přesnost
1	2,97E+007	9,77E+012	177	2,08E-001	179	2,56E-001
2	2,12E+007	1,76E+014	193	1,21E-001	185	1,04E-001
3	9,46E+006	2,00E+015	189	1,30E-002	193	4,73E-002
4	5,80E+007	2,27E+016	193	5,59E-004	193	5,08E-003
5	2,61E+007	1,32E+016	193	1,24E-005	191	5,65E-005
6	6,96E+006	6,56E+015	192	6,26E-003	191	6,96E-003
7	2,29E+008	5,05E+018	193	1,16E-001	193	1,22E-001
8	1,39E+009	2,60E+021	193	1,10E-001	193	1,32E-001
9	7,26E+008	9,74E+021	188	3,36E-003	192	2,32E-002
10	6,53E+008	9,74E+021	193	1,47E-001	159	1,93E-001
11	.....					

Tabulka 6: Podmínění a konvergence úlohy brandy

Protože úlohu LP zadanou ve tvaru

$$min\, c^T x \text{, kde platí } Ax = b \text{ a } x \geq 0, \qquad 5.1$$

lze přeformulovat jako

$$min\, c^T x \text{, kde platí } TAx = Tb \text{ a } x \geq 0, \qquad 5.2$$

nabízí se možnost přetransformovat omezující podmínky tak, abychom dosáhli nižší horní meze φ_A, která závisí právě pouze na struktuře matice A. K tomu abychom nalezli vhodný způsob transformační matice T z 5.2, jsme se v této práci již nedopátrali a ponecháváme toto jako předmět k dalšímu bádání.

5.2.2 Ortogonalizace úlohy

Ukažme si ještě vliv ortogonalizace úlohy na podmíněnost soustavy rovnic pro určení Newtonova kroku. V tabulce 7 je vidět jakým způsobem roste podmíněnost diagonály D^2 a vedle toho jakým způsobem roste podmíněnost soustavy rovnic 4.27 ($Q^T DQ$). Na první pohled je vidět, že ortogonalizovaná úloha udržuje nižší podmíněnost řešené soustavy, nicméně v uvedeném příkladě již ve čtvrtém kroku algoritmu dosahuje podmíněnosti řádově 1e5 a metoda sdružených gradientů již selhává a není schopna konvergovat ani k přibližnému řešení s alespoň nějakou rozumnou přesností. Čímž celý algoritmus selhává a není schopen konvergovat.

krok	cond D	cond Q'DQ
1	5,49	3,59
2	49336	1194,1
3	1,78E+006	3452,4
4	2,35E+006	1,16E+005
5	1,01E+007	1,42E+005
6	5,48E+007	1,91E+005
7	9,67E+007	1,34E+006
8	9,35E+008	4,48E+006
9	2,42E+011	1,06E+007
10	5,52E+013	2,32E+007
11	1,90E+016	7,90E+007
12	5,29E+018	1,86E+008

Tabulka 7: Podmínění ortogonalizované úlohy

6. Závěr

V této práci byly probrány metody vnitřního bodu a teorii potřebná k vytvoření algoritmu této metody. Rovněž byly probrané praktické poznatky, které lze využít při implementaci algoritmů metod vnitřního bodu. Snahou bylo ukázat různé možné způsoby řešení a použít metodu sdružených gradientů s očekáváním zrychlení algoritmu, zejména pro úlohy velkých rozměrů.

Byl implementován algoritmus prediktor-korektor metody vnitřního bodu s využitím třech různých způsobů řešení Newtonova směru hledajícího krok směrem k řešení v každé iteraci. Dva z těchto způsobů řeší soustavu rovnic přímou metodou, to je faktorizací úlohy a určením přesného řešení. Faktorizace je řešena jako Choleského nebo LU rozklad. Třetí způsob hledá řešení soustavy rovnic určující Newtonův krok pomocí metody sdružených gradientů, která hledá pouze přibližné řešení.

Přístup využívající Choleského rozklad je standardní často popisovaný způsob používaný při implementaci metod vnitřního bodu. Díky prostředí Matlab, které nám poskytuje dobrou a efektivní implementaci Choleského rozkladu, je tento způsob snadný k implementaci. Tento způsob implementace relativně efektivní a poskytuje stabilní a přesné výsledky.

Druhým způsobem implementace algoritmu metody vnitřního bodu je využití LU rozkladu. I pro tuto variantu existuje velmi dobrá podpora v prostředí Matlab. Tento způsob řešení je teoreticky trochu pomalejší, zato je odolnější proti selhání a nalezne řešení i v případě kdy už Choleského rozklad selže. Navíc v praxi, při hledání řešení úloh s řídkými maticemi omezujících podmínek, vykazoval výsledky srovnatelné s algoritmem využívajícím Choleského rozkladu a nezřídka jej svojí rychlostí i předčil.

Třetí způsob implementace s využitím metody sdružených gradientů se nepodařilo implementovat zcela podle našich představ. Tento způsob zatím není použitelný pro praktická řešení, zejména protože se nám nepodařilo nalézt efektivní způsob nalezení MWB předpodmíňovače. Není však vyloučeno, že by se tato metoda dala použít pro určité třídy problémů. Předmětem dalšího bádání by mohlo být hledaní způsobu jak využít poznatků získaných s MWB předpodmíněním. Další námět k bádání může být hledaní vhodné transformace matice A omezujících podmínek tak, abychom dosáhli co nejnižší horní hranice φ_A omezující podmíněnost předpodmíněné soustavy rovnic, pomocí které hledáme krok směrem k řešení úlohy.

Dalším mechanismus, který jsme se snažily uplatnit spolu s použitím metody sdružených gradientů, spočíval v ortogonalizaci úlohy. Tímto jsme se snažili dosáhnout lepší konvergence metody sdružených gradientů. Bohužel ani tento způsob nám nepomohl odstranit problém se vzrůstající podmíněností soustavy řešení ve vnitřním kroku algoritmu.

Jedním z cílů této úlohy bylo implementovat efektivní algoritmus pro řešení úloh LP pomocí metod vnitřního bodu. Proto jsme v prostředí Matlab implementovali tři algoritmy, ve kterých jsme využily poznatků uvedených v předchozích kapitolách. Výsledkem jsou tři variace metody prediktor-korektor, které se liší ve způsobu řešení Newtonova kroku, jakožto nejdražšího kroku algoritmu s největším potenciálem pro zlepšení efektivity. Výsledkem jsou funkce ipm_pc_chol, využívající Choleského rozkladu, ipm_pc_lu využívající LU rozkladu a ipm_pc_cg_mwb využívající metodu sdružených gradientů s předpodmínením pomocí MWB. Tyto funkce jsou součástí této práce jako příloha. Implementaci algoritmů metod vnitřního bodu v prostředí Matlab bychom mohli zhodnotit jako v celku jednoduchou záležitost. Také díky tomu, že prostředí Matlab nám samo o sobě poskytuje dobré prostředky v podobě vestavěných funkcí, které lze využít. Nicméně i celý algoritmus implementovaný v podobě prediktor-korektor tak, jak byl popsán ve 4. kapitole není velmi náročný na implementaci. Například v porovnání s implementací simplexového

algoritmu, implementovaného v rámci bakalářské práce [1], který obsahuje daleko větší množství kódu a je rozdělen do dvou fází algoritmu, lze algoritmus metody vnitřního bodu implementovat v podstatě jako jeden cyklus, skoro tak snadno jak popisuje algoritmus 3.4. Největší problémy sebou přináší přesnost počítačové aritmetiky a s tím již zmíněné problémy s počítáním vnitřního kroku algoritmu. V přílohách jsou také přibaleny úlohy, které byly použity k testování implementovaných algoritmů. Testovací úlohy obsahují každá matici A, vektory b, c a očekávanou hodnotu učelové funkce v bodě optimálního řešení. Tyto úlohy byly získány z repository www.netlib.org [5], kde můžeme najít řadu problémů různých velikostí.

Úplným závěrem bych rád zhodnotil tuto práci tak, že se podařilo implementovat algoritmus metody vnitřního bodu standardním způsobem s využitím Choleského a LU rozkladu. Tato implementace nepřináší žádné nové poznatky z oblasti metod vnitřního bodu, ale poskytuje stabilní a relativně efektivní způsob řešení úloh lineárního programování. Sice se nepodařilo naplnit ambice řešení úloh LP metodou vnitřního bodu, která by využívala metody sdružených gradientů, ale jako autor této práce jsem získal cenné poznatky, které by mohli posloužit jako základ k dalšímu bádání nad řešením tohoto problému.

7. Reference

[1] Radek Riedel. *Efektivní implementace simplexového algoritmu do knihovny OOSol.* Bakalářská práce 2008, <*http://www.am.vsb.cz/theses/bakalari/2008/pdfs/rie057.pdf*>

[2] Jorge Nocedal, Stephen J. Wright. *Numerical Optimization . 2. vydání, Springer, 2006. ISBN: 978-0-387-30303-1*

[3] Jerome W. O'Neal. *The Use of Preconditioned Iterative Linear Solvers in Interior-Point Methods and Related Topics.* Georgia Institute of Technology. 2005, <http://hdl.handle.net/1853/11647>

[4] Doron Chen, Sivan Toledo. *Vaidya's preconditioners: implementation and experimental study.* 2003, <http://www.highbeam.com/doc/1G1-187770758.html>

[5] Netlib Repository at UTK and ORNL, <http://www.netlib.org/lp/data/readme>

[6] Zdeněk Dostál. *Optimal Quadratic Programing Algorithms.* Springer, 2009, ISBN: 978-0-387-84805-1

[7] Stephen J. Wright. *Primal-Dual Interior-Point Methods. SIAM, 1997, ISBN: 089871382X.*

[8] Stephen Boyd, LievenVandenberghe. *Convex Optimization*. Cambridge University Press, 2004, ISBN: 978-0-521-83378-3

8. Přílohy

Součástí originálu této práce je CDROM obsahující elektronickou verzi tohoto dokumentu a zdrojové kódy. Kromě funkcí implementující samotné algoritmy jsou přiloženy u pomocné funkce a funkce potřebné k načtení úloh a předzpracování úloh. Zároveň jsou na přiloženém CD nahrané testovací úlohy ve formátu .mat.

Obsah CDROM

```
matlab\
  ulohy\
    25fv47.mat
    brandy.mat
    maros-r7.mat
    share2b.mat
    ship04l.mat
    ship04s.mat
    ship08l.mat
    ship08s.mat
    ship12l.mat
    ship12s.mat
    stocfor2.mat

  ipm_pc_cg_mwb.m
  ipm_pc_cg_ort.m
  ipm_pc_chol.m
  ipm_pc_lu.m
  ipm_start_point.m
  ipm_step.m
  load_problem.m
  mwb.m
  prepare_problem.m
  test_ipm.m
  uptrifullrank.m

IPM.pdf
```