Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave

DIPLOMOVÁ PRÁCA

Juraj Kopačka

Bratislava 2004

Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave

Ekonomická a finančná matematika

Metódy Sekvenčného Kvadratického Programovania

Diplomová práca

Diplomant: Juraj Kopačka
Vedúci diplomovej práce: doc. RNDr. Milan Hamala, CSc.

Bratislava 2004

Čestne prehlasujem, že som diplomovú prácu napísal samostatne a použil som len literatúru uvedenú v zozname.

Chcel by som sa poďakovať predovšetkým svojmu diplomovému vedúcemu Milanovi Hamalovi za cenné rady a čas, ktorý mi venoval; potom Rogerovi Fletcherovi za poskytnutie literatúry;
Klausovi Schittkowskému za poskytnutie sady testovacích úloh a napokon spoločnosti MathWorks, ktorá mi udelila licenciu na softvér TOMLAB.

Obsah

Kapitola 1

Úvod

1.1 Čo je SQP

Metódy Sekvenčného Kvadratického Programovania (anglicky Sequential Quadratic Programming - ďalej len SQP) sú efektívne algoritmy pre riešenie Úloh Konvexného Programovania nasledovného tvaru:

$$(\mathbf{UKP}) \begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & g(x) \leq 0_m \end{cases} \tag{1.1}$$

kde $f : R^n \to R, g : R^n \to R^m$; pričom $f, g \in C^2$ sú konvexné. Pod takouto formuláciou budeme rozumieť: *"minimalizuj $f(x)$ (cez $x \in R^n$) za platnosti ohraničení $g(x) \leq 0_m$"*.

SQP metódy sú typickým príkladom iteračných optimalizačných metód. Úlohu (UKP) riešia generovaním postupnosti bodov $\{x^k\}$, v tvare $x^{k+1} = x^k + \alpha^k d^k$, kde smer d^k sa získa riešením pomocnej úlohy kvadratického programovania, ktorá v určitom zmysle aproximuje pôvodnú úlohu (UKP) v okolí bodu x^k a dĺžka kroku $\alpha^k \in R$ sa určí tak, aby sme na jednej strane zaručili pokles účelovej funkcie $f(x)$ a na strane druhej čo najviac rešpektovali ohraničenia $g(x) \leq 0_m$.

Ťažisko výpočtu SQP metód spočíva v riešení pomocnej úlohy kvadratického programovania s lineárnymi ohraničeniami, ktorá pre k-tu iteráciu vyzerá nasledovne:

$$\begin{cases} \min\limits_{d \in R^n} & r_k^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.:} & A_k d + b_k \leq 0_m \end{cases} \tag{1.2}$$

kde B_k je matica typu $n \times n$, A_k je matica $m \times n$, $b_k \in R^m$ a $r_k \in R^n$.

Na konštrukciu kvadratickej formy v účelovej funkcii sa používa buď Hessova matica účelovej funkcie alebo Hessova matica Lagrangeovej funkcie úlohy (UKP) alebo ich aproximácie.

Po vyriešení úlohy kvadratického programovania sa v získanom smere d^k hľadá optimálna dĺžka kroku α^k pomocou špeciálne skonštruovanej penalizačnej funkcie.

Výhodou SQP metód je, že ani inicializačný bod, ani body v ostatných iteráciach nemusia byť prípustné. Hlavne v prípade nelineárnych ohraničení je hľadanie prípustného bodu, od ktorého by sme sa mohli odpichnúť, dosť zložité.

Zhrnutie:
Každá iterácia v bode x^k SQP algoritmu sa skladá z nasledujúcich častí:

- Konštrukcia úlohy kvadratického programovania (1.2), t.z. matíc B_k, A_k a vektorov r^k a b_k.
- Riešenie úlohy kvadratického programovania, čím dostaneme smer $d^k \in R^n$
- Určenie dĺžky kroku α^k pomocou minimalizácie penalizačnej funkcie
- Vypočítanie nasledujúceho bodu $x^{k+1} = x^k + \alpha^k d^k$

1.2 Stručná história SQP metód

Prvá zmienka o algoritme typu sekvenčného kvadratického programovania sa objavila v roku 1963 v PhD práci Wilsona z Harvardskej Univerzity. Wilson prezentoval metódu, ktorá sa zvykne označovať ako Newtonov SQP algoritmus. Za voľbu matice z kvadratickej formy úlohy kvadratického programovania požíva Hessovu maticu Lagrangeovej funkcie a za krok α volí jednotku; to znamená, že potom odpadáva problém konštrukcie penalizačnej fukcie. Prvú serióznejšiu prácu na SQP metódach započal Mangasarian a jeho študenti na Univerzite vo Wisconsine. Garcia-Palomares a Mangasarian skúmali v roku 1976 SQP algoritmus, v ktorom sa Hessova matica Lagrangeovej funkcie aproximovala v každom kroku. Krátko po tom v rokoch 1976-77 sa hlbšiemu výskumu SQP metód venoval Han. Pre úlohy nelineárneho programovania prezentoval tvrdenia o lokálnej konvergencii algoritmov používajúcich na aproximáciu Hessovej matice Lagrangeovej funkcie PSB a BFGS formuly. Pre úlohu konvexného programovania použil l_1 penlizačnú funkciu na dokázanie globálnej konvergencie. V rokoch 1977-78 Powell prezentoval na viacerych konferenciách Hanovu prácu, ktorá sa tým dostala do pozornosti odbornej verejnosti. Odvtedy boli SQP metódy skúmané vo väčšej miere. Dodnes vznikajú rôzne nové SQP algoritmy. Za zmienku stojí spomenúť, že v roku 1992 vznikol prvý SQP algoritmus generujúci prípustné riešenia v každej iterácii. Takýto algoritmus má význam hlavne v praxi, keď treba z určitých dôvodov zastaviť iteračný proces ešte pred dosiahnutím optima. V roku 1995 prezentoval Zoppke-Donaldson vo svojej PhD práci, pod vedením prof. Fletchera, myšlienku filtra ako náhrady za penalizačnú funkciu. Fletcher a Leyffer túto ideu rozpracovali a v roku 1997 publikovali článok prezentujúci filtrový SQP algoritmus.

1.3 Ciele Diplomovej Práce

Predkladaná diplomová práca je venovaná metódam Sekvenčného Kvadratického Programovania (SQP) a stavia si nasledovné ciele:

- Uviesť ucelený pohľad na SQP metódy a ukázať na niektoré aplikácie v praxi.
- Bližšie sa zamerať na filtrové SQP metódy a porovnať ich s klasickými SQP metódami.
- Realizovať numerické experimenty za účelom porovnania filtrovej SQP s klasickými SQP metódami

Prvému cieľu sú venované v prvé tri kapitoly. Snažili sme sa získať najnovšie poznatky z danej oblasti dostupné na internete a poskytnúť aktuálny prehľad SQP metód a ich detailnú charakteristiku pre riešenie úlohy konvexného programovania (UKP). V kapitole tri spomíname špecifické typy SQP metód a ich aplikácie v praxi.

Druhému cieľu je venovaná štvrtá kapitola. Opierali sme sa hlavne o práce Rogera Fletchera a Svena Leyffera. Zaoberali sme sa štruktúrou algoritmu filtrovej SQP metódy, jeho problémami a použitím. Porovnávali sme filter s penalizačnou funkciou.

Náplň tretieho cieľa je obsiahnutá v piatej kapitole. V tejto časti sme použili softvér TOMLAB (- nadstavba MATLABu) a v ňom naprogramované funkcie, používajúce SQP metódy, na numerické experimenty. Riešili sme sadu testovacích úloh nelineárneho programovania. Na základe získaných numerických výsledkov, sme porovnali algoritmy používajúce filtrovú SQP metódu s klasickými SQP algoritmami, založenými na použití penalizačnej funkcie. Všetky testovacie úlohy a experimentami získané veličiny z MATLABu sme zálohovali; k dispozícii sú na priloženom CD, ktoré obsahuje aj Excelovský súbor s tabuľkami výstupov a ich grafmi.

1.4 Použitá symbolika a všeobecné predpoklady

Táto podkapitola obsahuje označenia a predpokady, ktoré budeme používať v texte.

Nasledujúci zápis optimalizačnej úlohy

$$\begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & g(x) \leq 0_m \end{cases} \tag{1.3}$$

budeme chápať takto: *"minimalizuj $f(x)$ (cez $x \in R^n$) za platnosti ohraničení $g(x) \leq 0_m$"*. Rozhodli sme sa pre označenie ako v anglickej literatúre (s.t.: je skratka od subject to:).

Predpokladáme, že všetky funkcie sú dvakrát diferencovateľné. Znamienkom ∇ označujeme gradient skalárnej funkcie, napr.

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_n} \right)^T$$

Pre vektorovú funkciu značíme Jacobiho maticu nasledovne:

$$\nabla g(x) = (\nabla g(x)_1, ..., \nabla g(x)_m)$$

Hessovu maticu pre skalárnu funkciu f definujeme ako symetrickú maticu, ktorej prvok v i-tom riadku a j-tom stĺpci vyzerá takto:

$$\nabla^2 f(x)_{i,j} = \frac{\partial f(x)}{\partial x_i \partial x_j}$$

Označme si indexovú množinu aktívnych ohraničení

$$\mathcal{I} := \{i : g_i(x) = 0, i = 1, ..., m\}.$$

Označme si prípustnú množinu pre úlohu (UKP)

$$X := \{x \in R : g_j(x) \leq 0, j = 1, ..., m\}.$$

Lagrangeova funkcia pre úlohu (UKP) vyzerá nasledovne

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x).$$

Označme si maticu $G(x)$ ako maticu tvorenú gradientmi aktívnych ohraničení, t.z. $\nabla g_i(x)$, $i \in \mathcal{I}(x)$. Sformulujme si *nutné podmienky optimality prvého rádu*. Tieto podmienky sú tiež známe pod názvom Karush-Kuhn-Tuckerove podmienky (skrátene KKT podmienky).

Veta: *Nech x^* je optimálne riečenie (UKP), nech stĺpce matice $G(x^*)$ sú lineárne nezávislé. Potom existuje vektor Lagrangeových multiplikátorov $\lambda^* \in R^m$ taký, že platí:*

$$\nabla f(x^*) + \sum_i^m \lambda^* g_i(x^*) = 0 \tag{1.4}$$

$$(i = 1, ..., m) \begin{cases} g_i(x^*) & \leq 0 \\ \lambda^* g_i(x^*) & = 0 \\ \lambda^* & \geq 0 \end{cases} \tag{1.5}$$

Za predpokladu, že matica $G(x)$ má lineárne nezávislé stĺpce, nulový podpriestor matice $G(x)^T$ je vlastne dotykový priestor k aktívnym ohraničeniam v bode x. Projekcia na tento dotykový podpriestor bude vyzerať nasledovne:

$$\mathcal{P}(x) = I - G(x)(G(x)^T G(x))^{-1} G(x)^T$$

Symbolom $\|.\|$ označujeme Euklidovskú normu.

Definujme si úrovne rýchlosti konvergencie:

Definícia 1. *Nech $\{x^k\}$ je postupnosť bodov konvergujúca k x^* (pre $k \to \infty$). Táto postupnosť* ***konverguje****:*

- **lineárne**, ak existuje kladná konštanta $0 < \xi < 1$ taká, že

  $$\left\|(x^{k+1} - x^*)\right\| \leq \xi \left\|(x^k - x^*)\right\|$$

  pre všetky k dostatočne veľké.

- **superlineárne**, ak existuje postupnosť kladných konštánt $\xi_k \longrightarrow 0$ taká, že

  $$\left\|(x^{k+1} - x^*)\right\| \leq \xi_k \left\|(x^k - x^*)\right\|$$

  pre všetky k dostatočne veľké.

- **kvadraticky**, ak existuje kladná konštanta ξ taká, že

  $$\left\|(x^{k+1} - x^*)\right\| \leq \xi \left\|(x^k - x^*)\right\|^2$$

  pre všetky k dostatočne veľké.

Tieto rýchlosti konvergencie zaznamenávajú pokrok v každom kroku. Takže sa zvyknú nazývať jednokrokové. Taktiež existujú aj m-krokové rýchlosti ($m > 1$); pre ne tiež platia spomenuté definície s tým rozdielom, že $k + 1$ sa nahradí $k + m$. My budeme spomínať ešte jeden druh tempa konvergencie, a síce priemernú rýchlosť konvergencie nazývanú ako R-rýchlostná konvergencia.

Def: *Hovoríme, že postupnosť* $\{x^k\}$ *konverguje* ***R-lineárne****, ak je postupnosť* $\{\|x^k - x^*\|\}$ *ohraničená postupnosťou, ktorá konverguje lineárne k nule.*

Ekvivalentne budú vyzerať aj definície pre R-superlineárnu a R-kvadratickú rýchlosť konvergencie.

Označme si predpoklady, na ktoré sa budeme postupne v texte odvolávať:
A1: Existuje optimálny vektor Lagrangeových multiplikátorov $\lambda^* \geq 0$ taký, že

$$\nabla \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) + \nabla g(x^*)\lambda^* = 0$$

A2: Stĺpce matice $G(x^*)$ sú lineárne nezávislé
A3: Hessova matica Lagrangeovej funkcie je vzhľadom na x kladne definitná na nulovom podpriestore matice $G(x^*)^T$. T.z. pre všetky $d \neq 0$ také, že $G(x^*)^T d = 0$ platí

$$d^T \nabla^2 \mathcal{L}(x^*, \lambda^*) d > 0$$

Pre zjednodušenie zápisu budeme $\mathcal{L}(x^*, \lambda^*)$ označovať $\mathcal{L}^*$.

Kapitola 2

Základná SQP metóda

V tejto kapitole si rozoberieme všeobecnú kostru SQP metódy a budeme sa venovať jednotlivým častiam potrebným pre správne fungovanie algoritmu. Najprv sa pozrieme ako správne zostrojiť pomocnú úlohu kvadratického programovania. Dostatočný priestor venujeme procesu voľby matice B_k z účelovej funkcie úlohy kvadratického programovania; spomenieme niekoľko metód riešiacich danú problematiku. Potom sa bedeme zaoberať penalizačnými funkciami. Vysvetlíme na čo sa používajú a ukážeme si niekoľko druhov známych penalizačných funkcií.

2.1 Konštrukcia pomocnej úlohy kvadratického programovania v bode x^k

Pripomeňme si, že našim cieľom je riešiť úlohu konvexného programovania (UKP) prostredníctvom postupnosti úloh kvadratického programovania (1.2), pričom výsledok d^k riešenia úlohy (1.2) vstupuje do iteračného vzorca $x^{k+1} = x^k + \alpha^k d^k$, kde α^k sa určí postupom opísaným v podkapitole 2.3.

SQP metóda je iteračná metóda v ktorej na to, aby sme dosiahli bod x^{k+1} z bodu x^k, potrebujeme poznať istú informáciu, ktorú získame riešením úlohy kvadratického programovania (angl. quadratic programing - ďalej QP). Táto úloha sa zvyčajne skladá z kvadratickej účelovej funkcie a lineárnych ohraničení. Jednou z najdôležitejších vecí v SQP metóde je správna formulácia QP. Na to aby sme odstránili nelineárnosti z ohraničení, je rozumné použiť linearizáciu pôvodných ohraničení (UKP) v bode x^k. Takže úloha QP by mohla mať nasledujúcu formu:

$$\begin{cases} \min\limits_{d \in R^n} & r_k^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.:} & \nabla g(x^k)^T d + g(x^k) \leq 0_m \end{cases} \tag{2.1}$$

kde $d = x - x^k$. Podľa tohto spôsobu by sme mohli skonštruovať aj účelovú funkciu úlohy QP, t.z. $r^k = \nabla f(x^k)$ a $B_k = \nabla^2 f(x^k)$ alebo $B_k \approx \nabla^2 f(x^k)$. Chceme aby úloha QP mala lineárne ohraničenia, no zároveň je potrebné, aby obsahovala aj "nelineárnu informáciu"

ohraničení pôvodnej úlohy (UKP). Preto tieto nelineárnosti ohraničení zakomponujeme do účelovej funkcie QP. Z toho dôvodu je vhodné zvoliť za účelovú funkciu úlohy QP Lagrangeovu funkciu a linearizovať nasledovnú úlohu:

$$\begin{cases} \min\limits_{x \in R^n} & \mathcal{L}(x, \lambda^*) \\ \text{s.t.:} & g(x) \leq 0_m \end{cases} \tag{2.2}$$

kde Lagrangeova funkcia má tvar

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x). \tag{2.3}$$

Vzhľadom na to že nepoznáme optimálne hodnoty lagrangeových multiplikátorov λ^*, použijeme ich aproximácie λ^k. Postup ako ich získať uvádzame v podkapitole 2.2 . Potom pre daný bod (x^k, λ^k) bude rozvinutie do Taylorovko radu podľa premennej x vyzerať nasledovne:

$$\mathcal{L}(x^k, \lambda^k) + \nabla \mathcal{L}(x^k, \lambda^k)^T d_x + \frac{1}{2} {d_x}^T \nabla^2 \mathcal{L}(x^k, \lambda^k) d_x \tag{2.4}$$

Počítanie celej Hessovej matice Lagrangeovej funkcie je numericky náročné, preto sa radšej používa vhodná aproximácia, ktorá má vlastnosti zaručujúce, aby QP skonvergovalo k riešeniu z ľubovoľného bodu x^k. Nech B_k je aproximácia $\nabla^2 \mathcal{L}(x^k, \lambda^k)$, potom môžme QP napísať nasledovne:

$$\begin{cases} \min\limits_{d \in R^n} & \nabla \mathcal{L}(x^k, \lambda^k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.:} & \nabla g(x^k)^T d + g(x^k) \leq 0_m \end{cases} \tag{2.5}$$

No najčastejšie používaná forma QP je táto:

$$(\mathbf{QP}) \begin{cases} \min\limits_{d \in R^n} & \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.:} & \nabla g(x^k)^T d + g(x^k) \leq 0_m \end{cases} \tag{2.6}$$

Formulácie úlohy QP (2.5) a (2.6) sú ekvivalentné iba v prípade ohraničení typu rovností. Vtedy by na základe linearizácie ohraničení bol v (2.5) výraz $\nabla h(x^k)^T d_x$ konštantný, takže potom by mala účelová funkcia tvar $\nabla f(x^k)^T d + \frac{1}{2} d^T B^k d$. V našom prípade ohraničení typu nerovností tieto dve úlohy nie sú ekvivalentné (ak $\lambda^k \neq 0$ alebo všetky ohraničenia nie sú aktívne). Aby sa dosiahla ekvivalencia (2.5) a (2.6), treba pôvodnú úlohu (UKP) preforlmulovať pridaním doplnkovej premennej na nasledovný tvar:

$$(\mathbf{UKP_2}) \begin{cases} \min\limits_{x,z} & f(x) & \\ \text{s.t.:} & g(x) + & z = 0_m \\ & & z \geq 0_m \end{cases} \tag{2.7}$$

kde $z \in R^m$ je vektor doplnkových premenných.

Riešenie d_x úlohy (QP) použijeme na získanie nového bodu x^{k+1} tak, že z bodu x^k spravíme krok v smere d_x podľa iteračnej schémy $x^{k+1} = x^k + \alpha^k d_x^k$. V ďalšej iterácii budeme

taktiež potrebovať nové odhady lagrangeových multiplikátorov λ^{k+1}. Jedným zo spôsobov ich voľby je použiť optimálne multiplikátory z quadratického podproblému; označme ich λ_{qp}. Pomocou nich si vyjadríme krok $d_\lambda = \lambda_{qp} - \lambda^k$. Potom môžme aktualizovať bod (x^k, λ^k) nasledovne:

$$x^{k+1} = x^k + \alpha^k d_x^k,$$
$$\lambda^{k+1} = \lambda^k + \alpha^k d_\lambda^k,$$

pre istú dĺžku kroku α^k. Po získaní nových bodov (x^k, λ^k) sa vyčísli hodnota funkcií f, g, ich derivácií Δf, Δg a napokon sa podľa daného spôsobu určí B_{k+1}.

Úlohu (QP) možno riešiť ľubovoľnou metódou kvadratického programovania. Väčšina algoritmov z tejto oblasti sa skladá z dvoch hlavných častí. V prvej, ktorá sa zvykne nazývať Fáza 1, sa zisťuje, či je inicializačný bod prípustný. Ak nie rieši sa jednoduchá úloha lineárneho programovania s cieľom nájsť prípustný bod. Vo Fáze 2 sa generuje postupnosť prípustných bodov, ktoré konvergujú k riešeniu d^*.

2.2 Voľba Hessovej matice Lagrangeovej funkcie

Jedným z podstatných prvkov, na ktorých závisí rýchlosť konvergencie SQP metód, je vhodná voľba matice B_k v pomocnej úlohe kvadratického programovania. Samozrejme existuje niekoľko spôsobov ako to spraviť. Prvým z nich je priamo zvoliť $B_k = \nabla^2\mathcal{L}(x^k, \lambda^k)$, tzv. newtonovská voľba B_k. Tento prístup má svoje výhody, no v praxi je výpočtovo dosť náročný na čas. Preto sa ukazujú efektívnejšie metódy, ktoré aproximujú maticu $\nabla^2\mathcal{L}(x^k, \lambda^k)$, či už celú alebo len jej určitú časť. V tejto podkapitole si ukážeme jeden spôsob priamej voľby a 2 prístupy používajúce aproximáciu Hessovej matice Lagrangeovej funkcie.

Na začiatok si uveďme pár predpokladov. Po prvé predpokladáme, že poznáme aktívne ohraničenia (UKP) v bode x^*. Potom, za podmienky, že bod x^k je blízko bodu x^*, bude mať (QP) v bode x^k rovnakú indexovú množinu aktívnych ohraničení ako (UKP). Takže môžme neaktívne ohraničenia pre (UKP) v bode x^* ignorovať a aktívne ohraničenia prehlásiť za ohraničenia typu rovnosti bez toho, aby sa zmenilo riešenie (QP). Za takýchto podmienok môžme ďalšiu analýzu lokálneho správania algoritmu obmedziť na úlohu s ohraničeniami typu rovností:

$$(\mathbf{U_R}) \begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & h(x) = 0_m \end{cases}$$

Zodpovedajúca pomocná úloha kvadratického programovania pre (U_R) bude vyzerať nasledovne:

$$(\mathbf{QP_R}) \begin{cases} \min\limits_{d \in R^n} \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.: } \nabla h(x^k)^T d + h(x^k) = 0 \end{cases} \tag{2.8}$$

Druhým predpokladom je, že pri voľbe dĺžky kroku $\alpha = 1$, klesá v novom bode hodnota penalizačnej funkcie.

Pre úlohu s ohraničeniami typu rovností si označme lagrangeove multiplikátory písmenom u. Z podmienky **A1** možno pri platnosti predpokladu **A2** nasledovne získať formulku na odhad vektoru lagrangeových multiplikátorov:

$$\nabla^2\mathcal{L}(x^*, u^*) = 0$$
$$\nabla f(x^*) + \nabla h(x^*)u^* = 0$$

túto rovnosť vynásobíme zľava kladne definitnou, regulárnou maticou $A_{n\times n}$

$$A\nabla f(x^*) + A\nabla h(x^*)u^* = 0 \ / \nabla h(x^*)^T \text{zľava}$$
$$\nabla h(x^*)^T A \nabla f(x^*) + \nabla h(x^*)^T A \nabla h(x^*) u^* = 0$$
$$\nabla h(x^*)^T A \nabla h(x^*) u^* = -\nabla h(x^*)^T A \nabla f(x^*),$$

keďže je A regulárna, aj $\nabla h(x^*)^T A \nabla h(x^*)$ je regulárna, takže pre ňu existuje inverzná matica; potom

$$u^* = -\left[\nabla h(x^*)^T A \nabla h(x^*)\right]^{-1} \nabla h(x^*)^T A \nabla f(x^*), \tag{2.9}$$

Ak za A zvolíme jednotkovú maticu (s jenotkami na diagonále), potom uvedená formulka definuje MNŠ (Metóda Nejmenších štvorcov) riešenie nutných podmienok optimality prvého rádu. Za platnotnosti predpokladov hladkosti f a h platí, že

$$u^0 = -\left[\nabla h(x^0)^T \nabla h(x^0)\right]^{-1} \nabla h(x^0)^T \nabla f(x^0) \tag{2.10}$$

sa nachádza blízko u^*, ak je x^0 blízko x^*.

Označme si u_{qp} ako optimálny lagrangeov multiplikátor pre (QP_R). Podmienky optimality prvého rádu pre úlohu (QP_R) sú

$$\begin{array}{rl} B_k d_x + \nabla h(x^k) u_{qp} & = -\nabla f(x^k) \\ \nabla h(x^k)^T d_x & = -\nabla h(x^k). \end{array} \tag{2.11}$$

Pomocou $u^{k+1} = u_{qp} = u^k + d_u$ môžme tieto rovnosti napísať takto:

$$\begin{array}{rl} B_k d_x + \nabla h(x^k) d_u & = -\nabla \mathcal{L}(x^k, u^k) \\ \nabla h(x^k)^T d_x & = -\nabla h(x^k) \end{array} \tag{2.12}$$

2.2.1 Newtonova SQP metóda

Prvým prístupom k voľbe matice B_k je, že za ňu volíme priamo Hessovu maticu lagrangeovej funkcie, čiže:

$$B_k = \nabla^2 \mathcal{L}(x^k, u^k). \tag{2.13}$$

Predpokladáme, že počiatočný bod x^0 je blízko bodu x^*. Z toho podľa (2.10) vyplýva, že u^0 je blízko u^*, a teda $\nabla^2 \mathcal{L}(x^k, u^k)$ je blízko $\nabla^2 \mathcal{L}(x^*, u^*)$. Z nutných podmienok prvého rádu a ohraničení dostaneme systém nelineárnych rovníc:

$$\Psi(x, u) = \begin{bmatrix} \nabla \mathcal{L}(x, u) \\ h(x) \end{bmatrix} = 0 \tag{2.14}$$

Podľa predpokladov **A1** a **A3** je Jakobián tohto systému v bode (x^*, u^*) regulárny a vyzerá nasledovne:

$$J(x^*, u^*) = \begin{bmatrix} \nabla^2 \mathcal{L}(x^*, u^*) & \nabla h(x^*) \\ \nabla h(x^*)^T & 0 \end{bmatrix} = 0 \tag{2.15}$$

Potom postupnosť bodov $\left\{(x^k, u^k)\right\}$ získaná iteračnou schémou

$$x^{k+1} = x^k + s_x,$$
$$u^{k+1} = u^k + s_u,$$

kde vektor $s = (s_x, s_u)$ je riešením

$$J(x^k, u^k) s = -\Psi(x^k, u^k), \tag{2.16}$$

konvergujú k (x^*, u^*) kvadraticky za podmienky, že (x^0, u^0) je dostatočne blízko (x^*, u^*). Rovnice (2.16) sú identické s rovnicami (2.12), ak položíme $B_k = \nabla^2 \mathcal{L}(x^k, u^k)$, $d_x = s_x$ a $d_u = s_u$. Túto verziu algoritmu voláme (lokálna) Newtonova SQP metóda.

Základné výsledky lokálnej konvergencie sú zhrnuté v nasledujúcej vete:

Veta 1. *Nech x^0 je štartovací bod pre riešenie (UKP) a u^0 je dané vzťahom (2.10). Predpokladajme, že postupnosť bodov $\{(x^k, u^k)\}$ je generovaná nasledovne*

$$\begin{aligned} x^{k+1} &= x^k + d_x, \\ u^{k+1} &= u^k + d_u, \end{aligned} \tag{2.17}$$

kde d_x a $u_{qp} = u^k + d_u$ sú riešenie a lagrangeov multiplikátor úlohy kvadratického programovania (QP_R) s $B_k = \nabla^2 \mathcal{L}(x^k, u^k)$. Potom ak $\|x^0 - x^*\|$ je dostatočne malé, je táto postupnosť dobre definovaná na (x, u)- priestore a kvadraticky konverguje k dvojici (x^*, u^*). ([1], str.14)

Treba zdôrazniť, že v tejto verzii algoritmu je dĺžka kroku definovaná $\alpha := 1$. To je obrovská výhoda v tom, že nepotrebujeme v smere d_x hľadať lepšiu dĺžku kroku, takže sa nepotrebujeme zaoberať problematikou penalizačnej funkcie (podkapitola 2.4). Newtonove SQP metódy, ktoré namiesto jednotkového kroku hľadajú optimálnu dĺžku kroku sa po anglicky označujú - damped Newton Methods. V takýchto prípadoch dochádza k zníženiu rýchlosti konvergencie algoritmu, niekedy až na lineárnu úroveň.

Aj napriek dobrým vlastnostiam, je Newtonov SQP algoritmus z praktického hľadiska nepoužiteľný. Najväčším problémom je zvoliť vhodný štartovací bod, ktorý by bol dostatočne blízko riešenia (U_R), aby zaručil konvergenciu algoritmu k minimu účelovej funkcie. V prílišnej vzdialenosti od riešenia (U_R) takisto nie je zaručené, že $\nabla^2 \mathcal{L}(x^k, u^k)$ bude kladne definitná na vhodnom podpriestore, a teda ani riešenie (QP_R) nemusí vôbec existovať. Táto metóda sa považuje za určitý teoretický štandard s ktorým sa porovnávajú ostatné metódy v oblasti konvergencie.

2.2.2 Aproximácie matíc B_k

V ďalšom sa budeme zaoberať takým prístupom k voľbe matice B_k, ktorý nepoužíva priamo Hessovu maticu Lagrangeovej funkcie, ale túto maticu aproximuje. Na začiatok si uveďme predpoklady pre maticu B_k:

B1: Matice B_k sú pre všetky k *rovnomerne kladne definitné* na nulovom podpriestore matíc $\nabla h(x^k)^T$. To znamená, že existuje nejaké $\beta_1 > 0$ také, že pre všetky k platí

$$d^T B_k d \geq \beta_1 \|d\|^2,$$

pre všetky $d \neq 0$ spĺňajúce

$$\nabla h(x^k)^T d = 0.$$

B2: Postupnosť $\{B_k\}$ je *rovnomerne ohraničená*; to znamená, že existuje $\beta_2 > 0$ také, že pre všetky k platí

$$\|B_k\|^2 \leq \beta_2.$$

B3: Inverzné matice k B_k sú tiež *rovnomerne ohraničené*; to znamená, že existuje $\beta_3 > 0$ také, že

$$\left\|B_k^{-1}\right\| \leq \beta_3$$

pre všetky k.

Postačujúce podmienky optimality druhého rádu pre úlohu (QP_R) vyžadujú, aby bola matica B_k kladne definitná na nulovom podpriestore $\nabla h(x^k)^T$. Podmienka **B1** tento požiadavok ešte zosilňuje.

Za platnosti podmienok **B1-B3** a za predpokladu, že (x^k, u^k) je dostatočne blízko (x^*, u^*), majú podmienky (2.12) riešenie d_x a d_u.

Najprv si vyjadrime d_x z prvej rovnice (2.12):

$$\begin{aligned} B_k d_x + \nabla h(x^k) d_u &= -\nabla \mathcal{L}(x^k, u^k) \\ B_k d_x &= -\left[\nabla h(x^k) d_u + \nabla f(x^k) + \nabla h(x^k) u^k\right] \qquad / u^k + d_u = u^{k+1} \\ B_k d_x &= -\left[\nabla f(x^k) + \nabla h(x^k) u^{k+1}\right] \end{aligned}$$

$$d_x = -B_k^{-1} \nabla \mathcal{L}(x^k, u^{k+1}); \tag{2.18}$$

potom d_u z druhej rovnice (2.12):

$$\begin{aligned} \nabla h(x^k)^T B_k^{-1} \nabla \mathcal{L}(x^k, u^{k+1}) &= \nabla h(x^k) \\ \nabla h(x^k)^T B_k^{-1} \nabla f(x^k) + \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^{k+1} &= \nabla h(x^k) \\ \nabla h(x^k)^T B_k^{-1} \nabla f(x^k) + & \\ + \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^k + \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) d_u &= \nabla h(x^k) \\ \nabla h(x^k)^T B_k^{-1} \nabla \mathcal{L}(x^k, u^k) + \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) d_u &= \nabla h(x^k) \\ \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) d_u &= \nabla h(x^k) \\ &\quad - \nabla h(x^k)^T B_k^{-1} \nabla \mathcal{L}(x^k, u^k) \end{aligned}$$

$$d_u = [\nabla h(x^k)^T B_k^{-1} \nabla h(x^k)]^{-1} [h(x^k) - \nabla h(x^k)^T B_k^{-1} \nabla \mathcal{L}(x^k, u^k)]. \tag{2.19}$$

Potom možno $u^{k+1} = u_{qp} = d_u + u^k$ prepísať nasledovne:

$$\begin{aligned} u^{k+1} &= \left[\nabla h(x^k)^T B_k^{-1} \nabla h(x^k)\right]^{-1} \left[h(x^k) - \nabla h(x^k)^T B_k^{-1} \nabla \mathcal{L}(x^k, u^k)\right] + u^k \\ \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^{k+1} &= h(x^k) - \nabla h(x^k)^T B_k^{-1} \nabla f(x^k) \\ &\quad - \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^k + \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^k \\ \nabla h(x^k)^T B_k^{-1} \nabla h(x^k) u^{k+1} &= h(x^k) - \nabla h(x^k)^T B_k^{-1} \nabla f(x^k) \end{aligned}$$

$$u^{k+1} = \left[\nabla h(x^k)^T B_k^{-1} \nabla h(x^k)\right]^{-1} \left[h(x^k) - \nabla h(x^k)^T B_k^{-1} \nabla f(x^k)\right]$$

Ak v (2.9) položíme $A = B_k^{-1}$, potom platí

$$u^{k+1} - u^* = [\nabla h(x^*)^T B_k^{-1} \nabla h(x^*)]^{-1} \nabla h(x^*)^T B_k^{-1} (B_k - \nabla^2 \mathcal{L}^*)(x^k - x^*) + \omega_k, \qquad (2.20)$$

kde podľa predpokladov **B2** a **B3**,

$$\omega_k \leq \kappa \left\| x^k - x^* \right\|^2,$$

pre nejakú konštantu κ, nezávislú od k. Rovnice (2.18) a (2.20) a predpoklad **A2** dávajú dohromady

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* - B_k^{-1} \left[\nabla \mathcal{L}(x^k, u^{k+1}) - \nabla \mathcal{L}(x^*, u^*) \right] \\ &= B_k^{-1} \left[(B_k - \nabla^2 \mathcal{L}^*)(x^k - x^*) - \nabla h(x^*)(x^{k+1} - x^*) \right] \\ &\quad + \mathcal{O}\left(\left\| x^k - x^* \right\|^2 \right) \\ &= B_k^{-1} V_k (B_k - \nabla^2 \mathcal{L}^*)(x^k - x^*) + \mathcal{O}\left(\left\| x^k - x^* \right\|^2 \right), \end{aligned} \qquad (2.21)$$

kde

$$V_k = I - \nabla h(x^*) \left[\nabla h(x^*)^T B_k \nabla h(x^*) \right]^{-1} \nabla h(x^*)^T B_k^{-1}$$

Operátor projekcie (definovaný v podkapitole 1.4), ktorý má v prípade ohraničení typu rovností tvar

$$\mathcal{P}_k = I - \nabla h(x) \left[\nabla h(x)^T \nabla h(x) \right]^{-1} \nabla h(x)^T$$

spĺňa

$$V_k \mathcal{P}^k = V_k.$$

Takže z (2.21) vyplýva nasledujúca nerovnosť

$$\left\| x^k - x^* \right\| \leq \left\| B_k^{-1} \right\| \left\| V_k \right\| \left\| \mathcal{P}^k (B_k - \nabla^2 \mathcal{L}^*)(x^k - x^*) \right\| + \mathcal{O}\left(\left\| x^k - x^* \right\|^2 \right).$$

Horeuvedená analýza nás privádza k nasledujúcej vete:

Veta 2. *Nech platia predpoklady **B1-B3**. Potom existujú kladné konštanty ε a γ také, že ak platí*

$$\begin{aligned} \left\| x^0 - x^* \right\| &< \varepsilon, \\ \left\| u^0 - u^* \right\| &< \varepsilon \end{aligned}$$

a pre všetky k

$$\left\| \mathcal{P}^k (B_k - \nabla^2 \mathcal{L}^*)(x^k - x^*) \right\| < \gamma \left\| x^k - x^* \right\|, \qquad (V2c)$$

potom postupnosti $\left\{ x^k \right\}$ a $\left\{ (x^k, u^k) \right\}$ generované SQP algoritmom sú dobre definované a konvergujú lineárne k x^* a (x^*, u^*). Postupnosť $\left\{ u^k \right\}$ konverguje R-lineárne k u^*.
([1], str.17)

Platí aj: Ak sú podmienky vety okrem nerovnosti (V2c) splnené a postupnosť x^k konverguje lineárne, potom existuje ξ také, že

$$\left\|\mathcal{P}^k(B_k - \nabla^2\mathcal{L}^*)(x^k - x^*)\right\| < \xi \left\|x^k - x^*\right\|.$$

Z týchto výsledkov možno vidieť význam projekcie $(B_k - \nabla^2\mathcal{L}^*)$, to znamená vzdialenosti matice B_k od Hessovej matice v optimálnom bode (x^*, u^*). Ako možno vidieť, aby platilo (V2c) musí platiť

$$\left\|\mathcal{P}^k(B_k - \nabla^2\mathcal{L}^*)\right\| < \gamma$$

alebo

$$\left\|(B_k - \nabla^2\mathcal{L}^*)\right\| < \gamma \tag{2.22}$$

Na to aby bola splnená podmienka (2.22) nie je nutné, aby postupnosť aproximácií $\|B_k\|$ konvergovala k Hessovej matici, ale stačí udržať ich vzdialenosť $\|(B_k - \nabla^2\mathcal{L}^*)\|$ v určitom rozumnom ohraničení.

Definícia 2. *Hovoríme, že postupnosť aproximácii $\{B_k\}$ pre SQP metódu má vlastnosť* ***ohraničenej odchýlky****, ak existujú konštanty α_1 a α_2 nezávislé od k také, že*

$$\left\|B_{k+1} - \nabla^2\mathcal{L}^*\right\| \leq (1 + \alpha_1\sigma_k)\left\|B_k - \nabla^2\mathcal{L}^*\right\| + \alpha_2\sigma_k, \quad (D1a)$$

kde

$$\sigma_k = max\left\{\left\|x^{k+1} - x^*\right\|, \left\|x^k - x^*\right\|, \left\|u^{k+1} - u^*\right\|, \left\|u^k - u^*\right\|\right\}.$$

Ak do SQP algoritmu zahrnieme podmienku *ohraničenej odchýlky* (pre matice B_k), postupnosť matíc $\{B_k\}$ bude spĺňať podmienky **B1**, **B2** a (V2c). Potom bude matica B_k dostatočne blízko Hessovej matice Lagrangeovej funkcie a (x^k, u^k) bude dostatočne blízko k (x^*, u^*). O tomto svedčí nasledovná veta

Veta 3. *Predpokladajme, že postupnosť bodov $\left\{(x^k, u^k)\right\}$ je generovaná SQP algoritmom a postupnosť aproximačných matíc $\left\{B^k\right\}$ spĺňa podmienky* ***B1*** *a (D1a). Ak $\|x^0 - x^*\|$ a $\|B_0 - \nabla^2\mathcal{L}^*\|$ sú dostatočne malé a u^0 je dané vzťahom (2.10), potom platia všetky predpoklady vety 2.*
([1], str.17)

Doterajšie tvrdenia zaručujú iba lineárnu rýchlosť konvergencie. Vzhľadom na kvadratickú konvergenciu Newtonovej metódy to nie je veľmi uspokojivé. Preto treba zosilniť isté predpoklady týkajúce sa vzťahu aproximamačných matíc B_k a Hessovej matice Lagrangeovej funkcie. Tento "vzťah" závisí na projekcii rozdielu B_k a Hessovej matice na nulový priestor matice $\nabla h(x^k)^T$. Platí nasledujúca veta:

Veta 4. *Nech platia predpoklady **B1-B3** a postupnosť bodov* $\{(x^k, u^k)\}$ *je generovaná SQP algoritmom. Ďalej predpokladajme, že* x^k *konverguje k* x^**. Potom postupnosť* $\{x^k\}$ *konverguje k* x^* *superlineárne vtedy a len vtedy, ak aproximačné matice spĺňajú*

$$\lim_{k\to\infty} \frac{\left\|\mathcal{P}^k(B_k - \nabla^2\mathcal{L}^*)(x^{k+1} - x^k)\right\|}{\|(x^{k+1} - x^k)\|} = 0. \quad (V4a)$$

Ak toto platí, potom postupnosť $\{u^k\}$ *konverguje R-superlineárne k* u^**. Takže postupnosť* $\{(x^k, u^k)\}$ *konverguje superlineárne k* (x^*, u^*)*.*
([1], str.19)

V tejto vete je dobré si všimnúť, že rovnica (V4a) sama o sebe nezaručuje konvergenciu, a preto je v predpokladoch uvedené, že x^k musí konvergovať k x^*.

Nie je jednoduché nájsť vhodné a hlavne použiteľné aktualizačné formulky pre B_k, ktoré by spĺňali či už superlineárnu alebo aj lineárnu konvergenciu. My si ukážeme dva základné prístupy k danej problematike. Plnú aproximáciu Hessovej matice Lagrangeovej funkcie a aproximáciu redukovaného Hessiánu.

2.2.3 Plná aproximácia Hessiánu

V tejto podkapitole sa budeme zaoberať plnou aproximáciu Hessovej matice Lagrangeovej funkcie. Pre Lagrangeovu funkciu platí

$$\nabla\mathcal{L}(x^{k+1}, u^{k+1}) - \nabla\mathcal{L}(x^k, u^{k+1}) \approx \nabla^2\mathcal{L}(x^{k+1}, u^{k+1})(x^{k+1} - x^k)$$

Takže je rozumné aproximovať Hessovu maticu Lagrangeovej funkcie v bode (x^{k+1}, u^{k+1}) tak, aby B_{k+1} spĺňala

$$B_{k+1}(x^{k+1} - x^k) = \nabla\mathcal{L}(x^{k+1}, u^{k+1}) - \nabla\mathcal{L}(x^k, u^{k+1}) \quad (2.23)$$

Rovnica (2.23) sa zvykne nazývať *kvázinewtonovská podmienka.*

Ako sa teda bude generovať postupnosť aproximačných matíc $\{B_k\}$? Prirodzené je vytvoriť maticu B_{k+1} nasledovnou schémou.

$$B_{k+1} = B_k + \Delta B_k, \quad (2.24)$$

kde B_k je matica z predošlej iterácie a ΔB_k je príslušná korečná matica.

Sformulujme si požadované vlastnosti matíc B_k:
1) Symetria matícB_k
2) Kladná definitnosť matíc B_k
3) Jednoduchosť korekčnej matice ΔB_k, ktorá sa dosahuje jej malou hodnosťou. Zvyčajne to býva hodnosť jedna alebo dva. Podľa toho klasifikujeme aj jednotlivé formulky na tvorbu aproximačných matíc.

Uveďme si 2 známe kvázinewtonovské aproximačné formuly:

PSB formula

Jednou z aproximačných formúl hodosti dva je PSB formula (Powell-symetric-Broyden). Vyzerá nasledovne:

$$B_{k+1} = B_k + \frac{1}{s_k^T s_k}\left[(y_k - B_k s_k)s_k^T + s_k(y_k - B_k s_k)^T\right] - \frac{(y_k - B_k s_k)^T s_k}{(s_k^T s_k)^2} s_k s_k^T \tag{2.25}$$

kde

$$s_k = x^{k+1} - x^k$$

a

$$y_k = \nabla\mathcal{L}(x^{k+1}, u^{k+1}) - \nabla\mathcal{L}(x^k, u^{k+1})$$

SQP algoritmus používajúci PSB aproximačnú formulu dosahuje síce superlineárnu konvergenciu, ale nezahŕňa predpoklad kladnej definitnosti aproximačných matíc B_k. To je závažný problém, lebo potom úloha (QP_R) nemusí mať riešenie, ak počiatočný bod nie je blízko riešenia.

BFGS formula

Nasledujúci aproximačný vzorec hodnosti dva spĺňa aj podmienku kladnej definitnosti. Túto kvázinewtonovskú formulu odvodili v rokoch 1970-71 rôznym spôsobom štyria matematici: Broyden, Fletcher, Goldfarb a Shano. Podľa nich nesie názov BFGS formula. Je považovaná za najefektívnejšiu a najpopulárnejšiu. Jej tvar vyzerá nasledovne:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} + \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \tag{2.26}$$

kde s_k a y_k sú definované rovnako ako pre PSB formulu (2.25).

Dôležitou vlastnosťou postupnosti aproximačných matíc $\{B_k\}$ generovaných BFGS formulou je, že spĺňa vlastosť *ohraničenej odchýlky* (Definícia 2). Ďalším plusom je, že matice B_k dedia symetrickosť a kladnú definitnosť. Ak je B_k kladne dofinitná a platí

$$y^T s > 0, \tag{2.27}$$

potom B_{k+1} je kladne definitná. To znamená, že ak je počiatočná matica B_0 kladne definitná, potom BFGS algoritmus generuje symetrické kladne definitné matice. Vďaka tejto vlastnosti je (QP_R) ľahko riešiteľný a SQP kroky sú dobre definované. V tomto prípade má BFGS-SQP algoritmus rovnaké lokálne konvergenčné vlastnosti ako PSB-SQP algoritmus.

Veta 5. *Nech je $\nabla^2\mathcal{L}^*$ kladne definitná a nech počiatočná matica B_0 je tiež kladne definitná. Ďalej predpokladajme, že postupnosť bodov $\{(x^k, u^k)\}$ je generovaná SQP algoritmom, ktorý na tvorbu aproximačných matíc B_k používa BFGS formulu (2.26). Potom ak $\|B_k - \nabla^2\mathcal{L}^*\|$ a $\|(x^{k+1} - x^k)\|$ sú dostatočne malé a u^0 je dané pomocou (2.10), má postupnosť $\{B_k\}$ vlastnosť "ohraničenej odchýlky" a je splnené (V4a). Preto postupnosť bodov (x^k, u^k) konverguje superlineárne k (x^*, u^*). x konverguje superlineárne k x^* a lagrangeove multiplikátory konvergujú R-superlineárne.*
([1], str. 19)

2.2.4 Aproximácia redukovaného Hessiánu

Tento prístup je úplne odlišný. Vychádza z predpokladu, že (podm **A3**) je potrebné, aby $\nabla^2\mathcal{L}(x^*, u^*)$ bola kladne definitná iba na istom podpriestore. Metódy aproximácie redukovaného Hessiánu aproximujú iba časť Hessovej matice Lagrangeovej funkcie prislúchajúcu k tomuto podpriestoru. Výhodou takéhoto prístupu je, že možno použiť "štandardné aktualizácie kladnej definitnosti" a pritom sa zníži rozmer úlohy na $n - m$, čo môže byť dosť významná redukcia.

Nech x^k je bod, pre ktorý $\nabla h(x^k)$ má plnú hodnosť. Ďalej nech Z_k a Y_k sú také matice, že stĺpce Z_k tvoria bázu pre nulový podpriestor $\nabla h(x^k)^T$ a stĺpce Y_k tvoria bázu pre stĺpcový podpriestor $\nabla h(x^k)$. Navyše predpokladajme, že stĺpce Z_k sú navzájom ortogonálne. Z_k a Y_k sa dajú získať napr. QR rozkladom matice $\nabla h(x^k)$.

Veta 6. *Ľubovoľnú maticu $A \in M_{m\times n}$ s LN stĺpcami možno vyjadriť v tvare $A = Q.R$, kde Q je matica s ortogonálnymi stĺpcami, R je horná trojuholníková matica ($reg. \Rightarrow \exists R^{-1}$). Ak A je $n \times n$, potom Q je ortonormálna matica $n \times n$ a R je horná trojuholníková matica $n \times n$.*

Definícia 3. *Nech (x^k, u^k) je daná dvojica (riešenie a multiplikátor) a nech $\nabla h(x^k)$ má plnú hodnosť. Nech matica Z_k má ortogonálne stĺpce, ktoré tvoria bázu nulového podpriestoru $\nabla h(x^k)^T$ Potom sa matica*

$$Z_k^T \nabla^2 \mathcal{L}(x^k, u^k) Z_k$$

*nazýva **redukovaná Hessova matica** Lagrangeovej funkcie v bode (x^k, u^k).*

Redukovaná Hessova matica nie je jediná. Jej forma závisí na voľbe bázy pre nulový priestor matice $\nabla h(x^k)^T$. Podľa predpokladu **A3** je redukovaná Hessova matica v riešení kladne definitná. Takisto platí : ak (x^k, u^k) je blízko (x^*, u^*) potom redukovaná Hessova matica je kladne definitná v bode (x^k, u^k). Rozložme vektor d_x nasledovne:

$$d_x = Z_k p_Z + Y_k p_Y. \tag{2.28}$$

Keďže $\nabla h(x^k)^T Z_k p_Z = 0_n$, budú ohraničenia (QP_R) vyzerať takto

$$\nabla h(x^k)^T Y_k p_Y = -h(x^k).$$

Z toho si za platnosti **A2** možno vyjadriť p_Y:

$$p_Y = -\left[\nabla h(x^k)^T Y_k\right]^{-1} h(x^k). \tag{2.29}$$

Úloha (QP_R) sa potom zmení na neohraničenú úlohu s $n - m$ premennými takéhoto tvaru

$$\min_{p_Z} \quad \frac{1}{2} p_Z^T Z_k^T B_k Z_k p_Z + \left(\nabla f(x^k)^T + p_Y^T B_k\right) Z_k p_Z \tag{2.30}$$

Matica $Z_k^T B_k Z_k$ v úlohe (2.30) je aproximácia redukovanej Hessovej matice Lagrangeovej funkcie v bode x^k. Pri jej aktualizácii by sa dalo postupovať spôsobom, že by sa aktualizovala najpr matica B_k a potom by sa vypočítalo $Z_k^T B_k Z_k$. No výhodnejšie je aktualizovať hneď celú redukovanú Hessovu maticu.

Takže predpokladajme, že v k-tej iterácii máme danú pozitívne definitnú maticu R_k, ktorá má rozmery $(n-m) \times (n-m)$ a je aproximáciou redukovanej Hessovej matice Lagrangeovej funkcie, bod x^k a odhad Lagrangeových multiplikátorov u^k. Ako získame "informáciu" k+prvej iterácie? Najprv si podľa vzťahu (2.29) vypočítame p_Y. Potom si vyjadríme p_Z podľa

$$p_Z = -R_k^{-1} Z_k^T (\nabla f(x^k) + B_k p_Y).$$

d_x sa vypočíta zo vzťahu (2.28) a dosadí do $x^{k+1} = x^k + d_x$. Vygeneruje sa nový multiplikátor u^{k+1}. Napokon aktualizujeme aproximáciu redukovanej Hessovej matice; na to použijeme jemne upravenú BFGS formulu

$$R_{k+1} = R_k + \frac{y_k y_k^T}{y_k^T s_k} + \frac{R_k s_k s_k^T R_k}{s_k^T R_k s_k} \tag{2.31}$$

kde

$$s_k = Z_k^T (x^{k+1} - x^k) = Z_k^T Z_k p_Z$$

a

$$y_k = Z_k^T \left[\nabla \mathcal{L}(x^k + Z_k p_Z, u^k) - \nabla \mathcal{L}(x^k, u^{k+1}) \right].$$

Vzhľadom na to, že úloha (2.30) nie je ohraničená, nepočítajú sa Lagrangeove multiplikátory priamo. Môže sa použiť napr. MNŠ odhad (2.10). Vo všeobecnosti jediné čo treba zaručiť je, aby multiplikátory spĺňali

$$\left\| u^k - u^* \right\| = \mathcal{O}\left(\left\| x^k - x^* \right\| \right). \tag{2.32}$$

Platí $\mathcal{P}^k = Z_k Z_k^T$; takže aproximáciu

$$Z_k R_k Z_k^T$$

chápeme ako aproximáciu

$$\mathcal{P}^k \nabla^2 \mathcal{L}(x^k, u^k) \mathcal{P}^k.$$

Takže vzhľadom na to, že táto metóda neaproximuje $\mathcal{P}^k \nabla^2 \mathcal{L}^*$, tu neplatia tvrdenia o lokálnej konvergencii ako pri plnej aproximácii Hessovej matice.

Predpokladá sa, že sa matice Z_k volia tak, že platí

$$\| Z_k - Z(x^*) \| = \mathcal{O}\left(\left\| x^k - x^* \right\| \right). \tag{2.33}$$

Veta 7. *Predpokladajme, že horeuvedený algoritmus používajúci redukovanú Hessovu maticu Lagrangeovej funkcie volí u^k a Z_k tak, aby spĺňali (2.32) a (2.33). Ak postupnosť $\{x^k\}$ konverguje k x^* R-lineárne, potom sú postupnosti $\{R_k\}$ a $\{R_k^{-1}\}$ rovnomerne ohraničené a $\{x^k\}$ konverguje dvojkrokovo superlineárne.*
([1]*, str. 26)*

Pre Lagrangeove multiplikátory by nemal byť problém splniť podmienku (2.32). Zložitejšia je voľba matíc Z_k, aby spĺňali (2.33).

2.3 Penalizačné funkcie

Penalizačná funkcia ϕ sa používa za účelom dosiahnutia globálnej konvergencie (t.z. konvergencie k lokálnemu minimu). Určujeme pomocou nej optimálnu dĺžku kroku α^k v smere d^k z bodu x^k do bodu x^{k+1}, pričom sa snažíme aby hodnota ϕ bola čo najmenšia a súčasne aby platilo

$$\phi(x^k + \alpha^k d^k) < \phi(x^k). \tag{2.34}$$

Proces hľadania optimálnej dĺžky kroku sa zvykne v anglickej literatúre nazývať "linesearch". Dôležitou vlastnosťou penalizačnej funkcie je, aby sa jej inflexné body zhodovali s inflexnými bodmi účelovej funkcie. Ďalej musí byť možné znížiť funkčnú hodnotu ϕ použitím smeru d_x, ktorý sme získali ako riešenie pomocnej úlohy kvadratického programovania (ďalej len QP). To znamená, že d_x musí byť spádový smer pre ϕ.

Pri neohraničenej optimalizácii nie je voľba penalizačnej funkcie žiadny problém, pretože ňou bude samotná účelová funkcia, takže budeme merať priamo pokles účelovej funkcie. V prípade ohraničenej optimalizácie nastáva viacero problémov. Okrem merania poklesu účelovej funkcie potrebujeme poznať aj informáciu o prípustnosti riešenia. Takže musíme do penalizačnej funkcie vhodným spôsobom zakomponovať aj ohraničenia. Dôležitou otázkou, ktorá sa vynára je čomu máme prikladať väčší význam - meraniu neprípustnosti alebo meraniu poklesu. Pomer týchto dvoch cieľov sa v penalizačnej funkcii vyjadruje pomocou penalizačného parametra, ktorého určenie je najväčšou slabinou penalizačnej funkcie. Väčšinou existuje istá prahová hodnota, pod ktorou penalizačná funkcia nemá lokálne minimum práve v riešení úlohy (UKP). Táto hodnota nie je explicitne známa. Príliš nízka hodnota môže spôsobiť získanie neprípustného bodu (UKP). Na druhej strane príliš vysoká hodnota zatieňuje vplyv účelovej funkcie, čo môže spôsobovať napr. pomalú konvergenciu . Existujú aj metódy, ktoré uvažujú o premenlivom penalizačnom parametri, no tu sa taktiež vyskytujú problémy. Ďalšou ťažkosťou je efekt nediferencovateľnosti niektorých populárnych penalizačných funkcií. Toto viedlo k vývoju metód, v ktorých je dovolený aj rast penalizačnej funkcie počas istého ohraničeného počtu iterácii.

V ďalšom si ukážeme tri populárne penalizačné funkcie: diferencovateľnú Augmented Lagrangian funkciu, jednoduchšiu, ale nediferencovateľnú l_1 penalizačnú funkciu a penalizačnú funkciu navrhnutú Hanom a Powellom . Takisto si ukážeme vhodné voľby parametrov, aby bola dosiahnutá konvergencia k optimálnemu riešeniu. Pre jednoduchosť sa na začiatku obmedzíme na úlohu s ohraničeniami v tvare rovností. Takže budeme uvažovať úlohu:

$$(\mathbf{U_R}) \begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & h(x) = 0 \end{cases} \tag{2.35}$$

A potom rozšírime analýzu aj o úlohu s ohraničeniami typu nerovností :

$$(\mathbf{UKP}) \begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & g(x) \leq 0_m \end{cases} \tag{2.36}$$

Označme si písmenom λ lagrangeove multiplikátory pri ohraničeniach typu nerovností a písmenom u pri rovnostných ohraničeniach.

Uveďme si predpoklady, ktoré budeme potrebovať pre ďalšiu analýzu:
C1: Štartujúci bod iteračného procesu a všetky nasledujúce body ležia v nejakej kompaktnej množine C.
C2: Stĺpce matice $\nabla h(x)$ sú lineárne nezávislé pre všetky $x \in C$.

2.3.1 Augmented Lagrangian penalizačná funkcia

Prvým príkladom je augmented Lagrangian penalizačná funkcia. Existuje niekoľko verzií tejto funkcie, no my sa budeme venovať nasledujúcemu tvaru:

$$\phi_F(x, \eta) = f(x) + h(x)^T \bar{u}(x) + \frac{\eta}{2} \|h(x)\|_2^2, \tag{2.37}$$

kde η je penalizačný parameter a $\bar{u}(x)$ sú odhady lagrangeových multiplikátorov získané metódou najmenších štvorcov , pričom platí, že $\bar{u}(x^*) = u^*$. Tieto odhady sú definované podľa (2.10):

$$\bar{u}(x) = -\left[\nabla h(x)^T \nabla h(x)\right]^{-1} \nabla h(x)^T \nabla f(x). \tag{2.38}$$

Nasledujúca veta hovorí o tom, že takto zadefinovaná funkcia má nevyhnutné vlastnosti, aby mohla byť penalizačnou funkciou pre úlohu s ohraničeniami typu rovností.

Veta 8. *Nech sú splnené predpoklady* ***B1, B2, C1*** *a* ***C2****. Potom pre dostatočne veľké η platí:*
(1) *$x^* \in C$ je ostré lokélne minimum ϕ_F vtedy a len vtedy, keď x^* je ostré lokálne minimum (U_R).*
(2) *ak x nie je inflexný bod (U_R), potom d_x je spádový smer pre ϕ_F.*
([1], str. 29)

Dôkaz vety je uvedený v [1] na str. 30-31. Dôsledkom vety je nasledujúca nerovnosť

$$cos(\theta_k) = -\frac{\nabla \phi_F(x^k, \eta)^T d_x}{\|\nabla \phi_F(x^k, \eta)\| \, \|d_x\|} \geq \gamma_1 > 0, \tag{2.39}$$

kde

$$\nabla \phi_F(x, \eta) = \nabla f(x) + \nabla h(x)^T \bar{u}(x) + \nabla \bar{u}(x)^T h(x) + \eta \nabla h(x)^T h(x) \tag{2.40}$$

a γ_1 je vhodná konštanta.

Táto nerovnosť hovorí o tom, že kosínus uhla medzi smerom d_x a záporným gradientom ϕ_F je nenulový pre všetky k. Ak k tejto podmienke pridáme vhodné *line-search* kritéria, určujúce voľbu α - dĺžku iteračného kroku, dostaneme postačujúce podmienky pre konvergenciu postupnosti $\{x^k\}$. Vhodnými line-search kritériami sú napr. Wolfove podmienky, ktoré vyžadujú aby dĺžka kroku α spĺňala nasledujúcie nerovnosti:

$$\phi_F(x^k + \alpha d_x, \eta) \leq \phi_F(x^k, \eta) + \sigma_1 \alpha \nabla \phi_F(x^k, \eta)^T d_x, \tag{2.41}$$

$$\nabla \phi_F(x^k + \alpha d_x, \eta)^T d_x \geq \sigma_2 \nabla \phi_F(x^k, \eta)_x^d, \tag{2.42}$$

kde $0 < \sigma_1 < \sigma_2 < 1$. Nerovnosť (2.41) zaručuje, že nastane dostatočný pokles vo funkčnej hodnote ϕ_F. Nerovnosť (2.42) garantuje, že dĺžka kroku α nebude príliš malá. Ak je d_x spádový smer, čo nám zaručuje Veta 8, tak je možné vždy nájsť dĺžku kroku α, spĺňajúcu podmienky (2.41) a (2.42). Okrem toho zmenšovanie hodnoty ϕ_F pre takýto krok spĺňa nasledujúcu rovnosť

$$\phi_F(x^{k+1}, \eta) \leq \phi_F(x^k, \eta) - \gamma_2 cos(\theta_k)^2 \left\|\nabla \phi_F(x^k, \eta)\right\|^2, \tag{2.43}$$

pre nejakú kladnú konštantu γ_2. Z toho

$$\sum_{k=1}^{\infty} cos(\theta_k)^2 \left\|\nabla \phi_F(x^k, \eta)\right\|^2 < \infty \tag{2.44}$$

a pretože $cos(\theta_k)$ je odrazený od nuly, platí

$$\lim_{x \to \infty} \phi_F(x^k, \eta) = 0 \tag{2.45}$$

Toto implikuje, že postupnosť x^k konverguje k inflexnému bodu ϕ_F

Veta 9. *Zvoľme parameter η tak, aby platilo (2.39). Potom je SQP algoritmus s dĺžkou kroku α, ktorá spĺňa Wolfove podmienky (2.41), globálne konvergentný k inflexnému bodu ϕ_F.* ([1], str. 32)

Ak sa všetky inflexné body ϕ_F zhodujú s lokálnymi minimami (U_R), potom algoritmus konverguje k lokálnemu riešeniu ϕ_F. Toto je zriedka zaručené. Zaručená je iba konvergencia k inflexnému bodu a nie k lokálnemu minimu ϕ_F. Môžme si to ukázať na nasledujúcom príklade:

$$\begin{cases} \min\limits_{x} & x_1 \\ \text{s.t.:} & x_1^2 + x_2^2 - 1 = 0 \end{cases} \tag{2.46}$$

so štartovacím bodom (2,0) a $B_k = I$ pre všetky k. Postupnosť iteračných bodov skonverguje k bodu (1,0), ktorý je lokálne maximum úlohy. No ϕ_F postupne klesá v každej iterácii. Bod (1,0) je sedlovým bodom ϕ_F.

V predošlých úvahách sa spomína, že voľba penalizačného parametra η má byť dostatočne veľká. Čo to znamená dostatočne veľká? Vzhľadom na to, že sa dopredu explicitne nedá určiť potrebná veľkosť η, je nutné v praxi pri programovaní algoritmu zahrnúť vhodný spôsob na zistenie dostatočného η.

Teraz sa pozrieme na to, ako bude vyzerať penalizačná funkcia, ak budeme riešiť úlohu s ohraničeniami typu nerovností. Opäť existuje niekoľko spôsobov ako na to. My si ukážeme dva najbejžnejšie.

V prvom spôsobe sa využíva pojem aktívnej množiny. V každej iterácii sa spomedzi nerovnostných ohraničení vyberú aktívne ohraničenia, s ktorými sa potom manipuluje

ako keby to boli ohraničenia typu rovností. K zostávajúcim neaktívnym nerovnostným ohraničeniam sa pristupuje osobitne. Definujme si indexovú množinu

$$\mathcal{D}_k = \left\{ i : g_i(x^k) \geq -\frac{(\lambda_{qp})_i}{\eta} \right\} \tag{2.47}$$

Táto množina obsahuje indexy všetkých nesplnených ohraničení + ohraničení, ktoré nie sú "bezpečne" splnené (t.z. bod x^k leží veľmi blízko hranice). Penalizačná funkcia bude definovaná nasledovne:

$$\begin{aligned}\phi_{FI}(x, \lambda_{qp}, \eta) =& f(x) + \sum_{i \in \mathcal{D}_k} (g_i(x)(\lambda_{qp}(x))_i + \frac{\eta}{2} g_i(x)^2) \\ &+ \frac{1}{2\eta} \sum_{i \notin \mathcal{D}_k} (\lambda_{qp}(x))_i^2\end{aligned}$$

Takáto penalizačná funkcia je diferencovateľná. Tentokrát sa nepoužívajú MNŠ odhady lagrangeových multiplikátorov, ale multiplikátory získané po riešení QP.

Druhý prístup používa myšlienku doplnkových premenných. Pôvodný problém (UKP) si preformulujeme do nasledovného tvaru

$$(\mathbf{UKP_{dopl}}) \begin{cases} \min\limits_{x,t} f(x) \\ \text{s.t.:} \quad g_i(x) + (t_i)^2 = 0, \quad i = 1, ..., m \end{cases} \tag{2.48}$$

Táto úloha je ekvivalentná s úlohou (UKP) v zmysle, že obidve majú ostré lokálne minimum v bode x^*, pričom v (UKP_{dopl}) platí $(t_i^*)^2 = -g_i(x^*), \quad i = 1, ..., m$.
Označme si vektor

$$t^2 = \begin{pmatrix} t_1^2 \\ \cdot \\ \cdot \\ t_m^2 \end{pmatrix},$$

potom definujeme

$$\bar{h}(x, t^2) = \begin{pmatrix} g(x) + t^2 \end{pmatrix}. \tag{2.49}$$

Potom môžeme skonštruovať nasledujúcu penalizačnú funkciu

$$\phi_F(x, \eta) = f(x) + \bar{h}(x, t^2)^T \bar{u}(x, t^2) + \frac{\eta}{2} \left\| \bar{h}(x, t^2) \right\|^2, \tag{2.50}$$

V tomto prípade je $\bar{u}(x, t^2)$ MNŠ odhad lagrangeových multiplikátorov. Na skonštruovanie úlohy QP sa nepoužíva úloha (UKP_{dopl}), ale pôvodná (UKP). Vypočíta sa smer d_x a potom sa aktualizuje doplnková premenná t^2. Smer z $(t^2)^k$ do $(t^2)^{k+1}$ určíme nasledovne:

$$d_{t^2} = -(\nabla g(x^k)^T d_x + g(x^k) + (t^2)^k). \tag{2.51}$$

Potom

$$(t^2)^{k+1} = (t^2)^k + \alpha d_{t^2}, \quad \text{kde} \quad \alpha \in (0, 1]. \tag{2.52}$$

2.3.2 l_1 penalizačná funkcia

K jednej z prvých penalizačných funkcií patrí l_1 penalizačná funkcia. Jej tvar pre ohraničenia typu rovností je nasledovný:

$$\phi_1(x, \rho) = f(x) + \rho \left\| h(x) \right\|_1, \tag{2.53}$$

kde ρ je kladný penalizačný parameter. Tak ako aj augmented Lagrangian funkcia, tak aj l_1 penalizačná funkcia je "exaktná" penalizačná funkcia, t.z., že existuje kladný penalizačný parameter ρ^* taký, že pre všetky $\rho \geq \rho^*$ zodpovedá neohraničené minimum penalizačnej funkcie minimu (U_R). Tvar ϕ_1 je síce jednoduchý, no táto funkcia nie je diferencovateľná na množine prípustných riešení. Na to, aby sa dosiahla diferencovateľnosť by sa mohla upraviť ϕ_1 tak, že by sa porameter ρ umocnil na druhú. V tom prípade by sa ale stratila vlastnosť exaktnosti, t.z., že body minima ϕ_1 by len konvergovali k riešeniu (U_R) pre $\rho \longrightarrow \infty$.

Výhodou l_1 penalizačnej funkcie je, že sa ľahko rozšíri pre úlohu s ohraničeniami typu nerovností. Vzhľadom na to, že diferencovateľnosť tu nehrá rolu, môžme ϕ_1 definovať takto:

$$\phi_1(x, \rho) = f(x) + \rho \left\| g^+(x) \right\|_1, \tag{2.54}$$

kde

$$g_i^+(x) = \begin{cases} 0 \text{ ,ak} & g_i(x) \leq 0 \\ g_i(x) \text{ ,ak} & g_i(x) > 0 \end{cases} \tag{2.55}$$

Všetky teoretické výsledky platiace pre prípad s ohraničeniami typu rovností platia aj tu.

l_1 penalalizačná funkcia je populárna hlavne kvôli svojej jednoduchosti. Na overenie možného bodu, ktorý by prichádzal do úvahy pri *line-search* procese nám stačí vypočítanie f a g.

Na druhej strane pri augmented Lagrangian funkcii potrebujeme okrem f, g počítať aj ich gradienty. Ďalšou nepríjemnosťou je počítanie $\bar{u}(x)$ pri úlohach väčších rozmerov. Preto sa zvyknú v praxi používať aproximácie augmented Lagrangian penalizačnej funkcie.

2.3.3 Han-Powell penalizačná funkcia

Uveďme si ešte Hanovu-Powellovu penalizačnú funkciu, ktorá sa zvykne často používať v algoritmoch používaných softvérom MATLAB.

Han a Powell používajú nasledovnú penalizačnú funkciu:

$$\phi(x) = f(x) + \sum_{i=1}^{m_a} r_i g_i(x) \; + \sum_{i=m_a}^{m} r_i \max\{0, g_i(x)\}, \tag{2.56}$$

kde m_a je počet aktívnych ohraničení a m počet všetkých ohraničení. Powell odporúča voliť penalizačný parameter nasledovne:

$$r_i = (r_{k+1})_i = \max_i \{\lambda_i, \frac{1}{2}((r_k)_i + \lambda_i)\}, \quad i = 1, ..., m \tag{2.57}$$

Pri inicializácii sa parameter nastaví takto:

$$r_i = \frac{\|\nabla f(x)\|_2}{\|\nabla g_i(x)\|_2}, \quad i = 1, ..., m. \tag{2.58}$$

Kapitola 3

Špecifické typy SQP metód

V tejto kapitole si predstavíme rôzne druhy SQP metód, ktoré sú založené na všeobecnej SQP schéme, no špecializujú sa na isté druhy úloh, resp. sú konštruované so špecifickými vlastnosťami. Opisujeme SQP metódu generujúcu v každej iterácii body ležiace v množine prípustných riešení - *feasible* SQP; hovoríme o metóde na riešenie úloh veľkého rozmeru; spomíname viacero aplikácií v praxi, kde sa využívajú SQP algoritmy.

3.1 Feasible SQP metóda

SQP metódy nemajú vo všeobecnosti vlastnosť, že by generovali v každej iterácii prípustné body. Ani štartovací bod ani body v jednotlivých iteráciách nemusia ležať v množine prípustných riešení. To je veľkou výhodou, pretože hlavne v prípade nelineárnych ohraničení môže byť hľadanie prípustného bodu, od ktorého by sme sa mohli odpichnúť, dosť zložité (dalo by sa to náročnosťou prirovnať k vyriešeniu samotného (UKP)). Existujú prípady, kedy postupnosť bodov skonverguje k riešeniu rýchlejšie, ak môže využívať aj body mimo množiny prípustných riešení.

Napriek spomenutým výhodám algoritmov, pracujúcich aj s neprípustnými bodmi, jestvuje veľa prípadov, kedy potrebujeme, aby metóda generovala body striktne prípustné. SQP metóda generujúca postupnosť bodov, ktorá leží v prípustnej množine (anglicky *feasible set*) sa v angličtine zvykne nazývať *feasible SQP* metóda. Povedzme si prečo je užitočný prístup generujúci iteračnú postupnosť len v množine prípustných riešení. Najprv sa pozrime na výhody z hľadiska vnútornej stavby metódy. Vo *feasible* SQP algoritme:

- sú pomocné úlohy kvadratického programovania vždy konzistentné (vždy existuje prípustné riešenie),
- môžme v line-search procese použiť priamo učelovú funkciu ako penalizačnú funkciu.

V praxi (napr. v strojárstve) sú *feasible* SQP metódy výhodné, lebo

- účelová funkcia často nebýva definovaná mimo množiny prípustných riešení,

- algoritmus možno zastaviť pred dosiahnutím optimálneho riešenia, získajúc prípustný bod (tento bod je dôležitý hlavne pri aplikáciach, prebiehajúcich v reálnom čase, kedy nejaký ďalší proces vyžaduje prípustný bod ešte pred tým ako stihne algoritmus skonvergovať k optimu).

Hlavným argumentom proti feasible metódam, je náročnosť získania prípustného štartovacieho bodu pre (UKP); okrem časového aspektu tu môže hrať úlohu aj to, že pri procese hľadania tohto bodu (ktorý sa zvykne vo viacerych algoritmoch označovať ako "Fáza 1"(angl. Phase 1)) môže príliš stúpnuť hodnota účelovej funkcie. No pri numerických experimentoch sa ukázalo, že celková časová strata spôsobená Fázou 1 býva malá.

Kľúčovým prvkom v SQP algoritmoch generujúcich prípustné body je tzv. *prípustný smer*. Definujme si tento pojem:

Definícia 4. *Nech* $X := \{x \in R : g_j(x) \leq 0, j = 1, ..., m\}$ *je množina prípustných riešení úlohy (UKP).* ***Prípustný smer*** *vychádzajúci z bodu* $x \in X$ *sa definuje ako ľubovoľný vektor* $d \in R^n$ *taký, že* $x + td$ *patrí do* X*, pre všetky* $t \in [0, T]$*, kde* $T > 0$.

Tak ako aj vo všeobecnej SQP metóde aj v tomto prípade sa v každej iterácii rieši istá pomocná úloha, ktorej riešením je smer ktorým sa vydáme z bodu x^k do bodu x^{k+1}. No teraz treba postupovať tak, aby sme získali *prípustný smer*; t.z. aby bod x^{k+1} ležal v množine prípustných riešení. Treba si uvedomiť, že smer d získaný ako riešenie (QP), nemusí byť prípustným. To znamená, že budeme musieť tento smer "nakloniť" tak, aby smeroval do prípustnej množiny. Počiatočné algoritmy s prípustnými smermi využívali na získanie smeru iba informáciu prvého rádu (použili sa iba prvé derivácie). Navyše smer bol počítaný pomocou úloh lineárneho a nie kvadratického programovania, takže bolo možné dosiahnuť maximálne lineárnu konvergenciu. Zahrnutie informácie druhého rádu použil Polak. No ani v tomto prípade nebola zaručená superlineárna konvergencia, pretože týmto algoritmom chýbal machanizmus, ktorý by kontroloval koľko času treba venovať samotnému "nakláňaniu" smeru d.

Ukázalo sa, že najlepšia cesta (zatiaľ) vedie cez riešenie úlohy kvadratického programovania. Algoritmus vytvorený Panierom a Titsom sa zakladá na riešení úlohy (QP) a nasledovnom nahradení d^* (optimálneho smeru (QP)) konvexnou kombináciou $d = (1 - \rho)d^* + \rho d^L$, kde d^L je ľubovoľný prípustný klesajúci smer. Kvôli zachovaniu lokálnej konvergencie SQP iterácie, sa ρ volí ako funkcia od d^* v takom zmysle, aby sa d približoval k d^*, keď sa blížime k optimálnemu bodu (UKP). Zväčša $\rho(d^*) = \mathcal{O}(\|d^*\|^2)$. Aby sme sa vyhli Maratosovmu efektu (čo je stručne povedané kmitanie okolo bodu optima, pričom sa tento bod nikdy nedosiahne) a zaručili superlineárnu konvergenciu, sa používa korekčný krok druhého rádu d^K. d^K spôsobí "ohnutie" smeru z x^k do x^{k+1}. Po zahrnutí všetkých spomenutých úprav napokon hľadáme bod x^{k+1} po oblúku, takže nový bod bude $x + td + t^2 d^K$. Ako nájdeme d a d^K? Panier a Tits v oboch prípadoch používajú úlohy kvadratického programovania, no d^K možno počítať aj lineárnou MNŠ bez ovplyvnenia

asymptotických vlastností konvergencie. Takže z pohľadu výpočtovej náročnosti rieši tento Panierov a Titsov algoritmus 3 úlohy kvadratického programovania (resp. 2 QP + 1 MNŠ). Toto nie je z časového hľadiska príliš vhodné hlavne pre úlohy veľkých rozmerov.

Aby sa vyhli prílišnému počtu úloh kvadratického programovania, skonštruovali Lawrence a Tits algoritmus zakladajúci sa na vhodnej transformácii (QP). Takže v každej iterácii sa počíta jedna úloha kvadratického programovania nasledovného tvaru:

$$(\mathbf{QP_{TRANS}}) \begin{cases} \min\limits_{d \in R^n} & \frac{1}{2} d^T B_k d + \eta \\ \text{s.t.:} & \nabla f(x^k)^T d \quad \leq \gamma \\ & \nabla g_j(x^k)^T d + g_j(x^k) \quad \leq \gamma\eta, \quad j = 1, ..., m, \end{cases} \tag{3.1}$$

kde η je kladný sklár nazývaný aj "*nakláňací parameter*" , B_k je kladne definitná a $x \in X$; γ je doplnková skalárna premenná. Pointa spočíva v tom, že ďalej od bodov spĺňajúcich KKT podmienky úlohy (UKP), bude $\gamma < 0$ a preto bude d, vzhľadom na prvé ohraničenie, klesajúci smer pre f; navyše ak $\eta > 0$, bude d prípustným smerom, vzhľadom na ďalších m phraničení. Nie je ťažké ukázať, že ak položíme $\eta = 0$, riešením (QP_{TRANS}) bude bežný SQP smer. Veľké hodnoty η zdôrazňujú prípustnosť, malé hodnoty η zvýrazňujú pokles.

3.2 Trust-region SQP metódy

Doteraz sme chápali nasledovnú iteračnú schému

$$x^{k+1} = x^k + \alpha^k d^k$$

tak, že nachádzajúc sa v bode x^k najprv nájdeme optimálny smer d^k a potom v tomto smere spravíme optimálny krok α^k.

V metódach používajúcich *trust-region* prístup sa postupuje inak. Stojac v bode x^k sa najprv určí *trust-region* polomer ρ. V oblasti vymedzenej polomerom ρ^k sa potom hľadá optimálny "smer+krok" d^k. Nový bod sa získa nasledovne

$$x^{k+1} = x^k + d^k.$$

V d^k je teraz zahrnutá informácia aj o smere aj o dĺžke kroku. Potom sa zisťuje či má penalizačná funkcia v bode x^{k+1} nižšiu hodnotu ako v bode x^k. Ak nie, *trust-region* polomer ρ sa zmenší (väčšinou o polovicu) a hľadá sa nový bod x^{k+1}.

3.3 SQP metóda na riešenie úloh veľkého rozmeru

V praxi sa často stretávame s úlohami, ktoré majú veľmi veľa (rádovo tisícky) premenných a takisto aj ohraničení. Pre takéto úlohy treba zostaviť špeciálne algoritmy, ktoré sú schopné pracovať práve s veľkými rozmermi. Kvôli ich dobrým vlastnostiam boli SQP metódy rozšírené aj pre veľkorozmerné prípady. Ujali sa napr. v oblasti optimálneho riadenia, konkrétne pri výpočte optimálnej trajektórie. V leteckom priemysle sa dodnes úspešne používa tzv. " systém optimálnej trajektórie" (angl. optimal trajectory system OTIS), ktorý na riešenie istých optimalizačných úloh nelineárneho programovania využíval algoritmus NPSOL. Táto metóda používa transformáciu Hessovej matice Lagrangeovej funkcie a potom ju kvázi-newtonovsky aproximuje. Úlohu kvadratického programovania rieši požitím metódy najmenších Štvorcov. NPSOL je schopná riešiť úlohy s 2000 ohraničeniami a viac ako 1000 premennými. No na problémy s ešte väčším rozmerom už nestačí.

Preto vznikol nový "riedky" SQP algoritmus. Nazýva sa SNOPT (angl. Sparse Nonlinear Optimizer - Riedky Nelineárny Optimalizér). Táto metóda využíva riedkosť matice Jakobiánu ohraničení. Nepoužíva plnú aproximáciu Hessovej matice Lagrangeovej funkcie, ale práve aproximáciu redukovaného Hessiánu (spomínanú v podkapitole 2.3.4), čo je výhodné najmä z hľadiska využita pamäte. Pomocnú úlohu kvadratického programovania rieši istým typom metódy *aktívnej množiny*, ktorá umožňuje premenným vystupovať lineárne aj v účelovej funkcii, aj v ohraničeniach QP. Pre hľadanie optimálnej dĺžky kroku sa používa l_1 penalizačná funkcia (spomínaná v podkapitole 2.4.2).

3.4 Ďalšie praktické využitie SQP algoritmov

Metódy sekvenčného kvadratického programovania majú bohaté praktické využitie. Prístup k najrôznejším testovaniam typu pokus-omyl býva nahradzovaný lacnejšou a efektívnejšou alternatívou počítačových simulácii. V nich sa vyskytuje veľa optimalizačných úloch, pre ktoré je ideálne použitie SQP algoritmov.

Optimálny dizajn

Jednou z najčastejších oblastí využitia SQP metód je oblasť optimálneho dizajnu.

V odvetviach ako napr. automobilový a letecký priemysel je potrebná čo najlacnejšia, najkvalitnejšia a časovo nenáročná výroba. Spoločnosti takéhoto druhu vyvýjajú množstvo nových modelov svojich produktov a preto, v rámci ušetrenia materiálu a prostriedkov, používajú na vytváranie a testovanie nových prototypov počítačové simulácie. Pomocou nich sa okrem ideálneho tvaru, povrchu, hrúbky, typu materiálu dájú určiť potenciálne chyby, poškodenia, atď. V týchto priemyselných odvetviach sa najčastejšie jedná o spracovanie plechu. V simuláciach vystupuje viacero druhou optimalizačných úloh nelineárneho programovania. Na ich riešenie sa využívajú prevažne SQP algoritmy.

V oblasti optimalizácie aerodynamického tvaru sa riešia úlohy pre ideálny tvar trupu lietadiel a rakiet, tvar a zakrivenie krídiel; simuluje sa stav tlaku pod krídlami, stav tlaku pri prechode do nadzvukovej rýchlosti, atď.

Ďalšou oblasťou optimálneho dizajnu využívajúcou SQP metódy je výroba elektronických súčiastok. Napríklad pri navrhovaní elektro-magnetických filtrov vystupuje niekoľko parametrov pre optimalizáciu, napr. dĺžka, hrúbka, počet istých vrstiev, jednoducho celková vnútorná geometrická štruktúra. Špecifické časti vytvoreného matematického modelu sú optimalizované SQP algoritmom.

Optimalizácia chemických reaktorov

Algoritmy sekvenčného kvadratického programovania sa vyžívaju aj na optimalizáciu pri rôznych procesoch v chemických reaktoroch. V simuláciach pre danú problematiku sa vyskytujú napr. úlohy pre výpočet optimálneho množstva jednotlivých prvkov, časového rozloženia rozpustnosti látok, ideálneho intervalu čistenia reaktora, atď.

Kalenie oceli

Optimálne riadenie sa uplatňuje aj pri leserovom kalení oceli . Fáza stavu oceli pri vysokých teplotách sa popisuje diferenciálnymi rovnicami. Dôležité sú ohraničenia teploty, aby sa zabránilo roztopeniu materiálu. Daný problém sa dá naformulovať ako úloha nelineárneho programovania; zvykne sa riešiť SQP algoritmom s aproximáciou redukovaného Hesiánu.

Kapitola 4

Filtrová SQP metóda

V tejto kapitole si ukážeme iný prístup v oblasti SQP metód, ako sme videli doteraz. Oboznámime sa s pojmom filtra, ako náhrady penalizačnej funkcie; spomenieme jeho výhody i nevýhody. Filtrový algoritmus používa trust-region prístup. To znamená, že nachádzajúc sa v bode x^k, sa najprv ohraničí dĺžka kroku a až potom sa hľadá smer, ktorým sa bude postupovať do ďalšieho bodu x^{k+1}. Bližšie si ukážeme filtrový algoritmus vytvorený R. Fletcherom a S. Leyfferom.

4.1 Filter v nelineárnom programovaní

Ukázalo sa, že existuje aj lepší spôsob ako používať penalizačnú funkciu. Zoppke-Donaldson, študent profesora Rogera Fletchera, prezentoval vo svojej Ph.D práci SQP algoritmus, ktorý nepoužíva žiadnu penalizačnú funkciu. Táto nová metóda je založená na trust-reagion prístupe a namiesto penalizačnej funkcie používa "filter". Pripomeňme si úlohu konvexného programovania, ktorou sa zaoberáme

$$(\mathbf{UKP}) \begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & g(x) \leq 0_m \end{cases} \tag{4.1}$$

Preformulujme si túto úlohu inak. Doteraz sme sa na (UKP) pozerali tak, že sme minimalizovali účelovú funkciu $f(x)$, pričom museli byť splnené ohraničenia $g(x)$. Teraz zmeníme cieľ splnenia ohraničení na ďalší minimalizačný problém. Takže naša úloha bude:

$$\min_{x \in R^n} \quad f(x) \tag{4.2}$$

a súčasne

$$\min_{x \in R^n} \quad h(g(x)), \tag{4.3}$$

pričom

$$h(g(x)) := \left\| g^+(x) \right\|_1 := \sum_{i=1}^{m} g_i^+(x)$$

je l_1 norma "*odchýlenia od ohraničení*" a $g^+(x) = max(0, g_i)$.

Penalizačná funkcia kombinuje (4.2) a (4.3) do jedného minimalizačného problému. My sa budeme na (4.2) a (4.3) pozerať ako na 2 rôzne ciele, ktoré chceme dosiahnuť.

Označme si $f^k := f(x^k)$ a $h^k := h(g(x^k))$ a zaveďme si pojem dominancie:

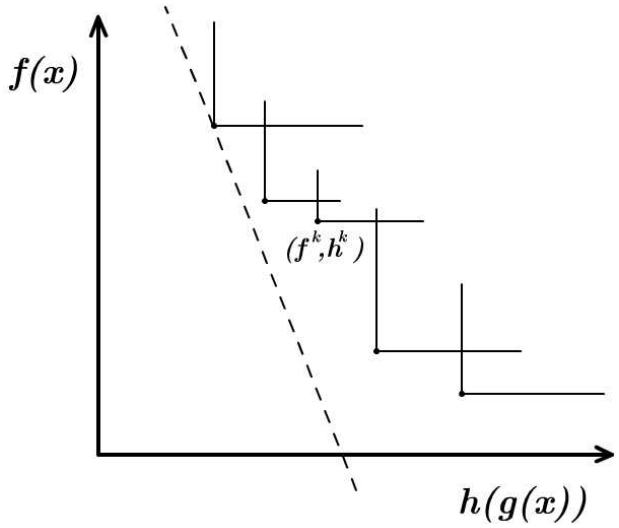
Definícia 5. *Hovoríme, že dvojica* (f^k, h^k) ***dominuje*** *inej dvojici* (f^l, h^l) $\iff$ *platí* $f^k \leq f^l$ *a súčasne* $h^k \leq h^l$.

Inak povedané x^k je prinajmenšom tak dobré ako x^l vzhľadom na (4.2) a (4.3). Teraz si môžeme zadefinovať pojem $filtra$, ktorý budeme používať v *trust-region* typoch algoritmov ako kritérium pre prijatie alebo odmietnutie nového kroku.

Definícia 6. ***Filtrom*** *nazývame takú postupnosť dvojíc* (f^l, h^l), *v ktorej žiadna dvojica nedominuje inej. Označme si* $\mathcal{F}^k$ *množinu indexov* $j < k$ *takú, že* (f^j, h^j) *patrí do filtra. Bod* (f^k, h^k) *sa nazýva* ***akceptovateľný*** *pre zahrnutie do filtra, ak nie je dominovaný žiadnym iným bodom z filtra; t.z. že buď* $f^k \leq f^j$ *alebo* $h^k \leq h^j$, *pre všetky* $j \in \mathcal{F}^k$.

V každom bode filtra sú zahrnuté iba 2 skaláre. Neukladáme žiadne vektory (ako napr. x^l). Toto je z hľadiska jednoduchosti a zaťaženia pamäťe veľkou výhodou. Nevýhodou je, že nám odpadá možnosť spätnej kontroly postupnosti predošlých bodov.

Grafickú reprezentáciu filtra možno vidieť na nasledujúcom obrázku 4.1.



Obrázok 4.1: Grafická reprezentácia filtra.

Ako možno vidieť na obrázku 4.1, každý prvok filtra vytvára blok neakceptovateľných bodov pre filter. Zjednotenie týchto blokov tvorí množinu všetkých bodov, ktoré nie sú akceptovateľné pre aktuálny filter. Čiarkovane je znázornená grafická reprezentácia istej penalizačnej funkcie. Vpravo od tejto čiarkovanej priamky je množina neprijateľných bodov pre penalizačnú funkciu, t.z. tieto body spôsobujú nárast hodnoty danej penalizačnej funkcie. Vidno, že existujú body, ktoré sú akceptovateľné pre filter, no spôsobujú nárast penalizačnej funkcie. To znamená, že filter povoľuje používať väčšiu množinu bodov pri vytváraní iteračnej postupnosti $\{x^k\}$, konvergujúcej k optimálnemu riešeniu x^*.

Hlavnou myšlienkou je použiť filter ako kritérium pre akceptovanie resp. odmietanie kroku v SQP metóde. Takže kritérium klesajúcosti penalizačnej funkcie je nahradené požiadavkou akceptovania filtrom.

Najväčšie zmeny v algoritme sa udejú v pomocnej úlohe kvadratického programovania (QP). Pridá sa do nej ohraničenie dĺžky kroku - *trust-region* polomer:

$$(\mathbf{QP^f}) \begin{cases} \min\limits_{d \in R^n} & q(d) = \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.:} & \nabla g(x^k)^T d + g(x^k) \leq 0_m \\ & \|d\|_\infty \leq \rho \end{cases} \tag{4.4}$$

Riešenie d^* úlohy (QP^f) už nebude len smerový vektor, ale celkový krok (ako pred tým $\alpha^k d^k$).

Takže k-ta iterácia bude vyzerať nasledovne:
Začíname v bode x^k, ktorý vstupuje do (QP^f). (QP^f) vyprodukuje riešenie - krok d^k. Získavame ďalší bod $x^{k+1} = x^k + d^k$.

Nový bod x^{k+1} je akceptovaný filtrom, ak zodpovedajúca dvojica (f^{k+1}, h^{k+1}) nie je dominovaná žiadnym prvkom filtra. V tomto prípade pokračujeme ďalšou iteráciou.

Ak bod x^{k+1} nie je akceptovaný filtrom, zamietame ho, zmenšíme trust-region polomer ρ a znova riešime (QP^f) (s pôvodným x^k).

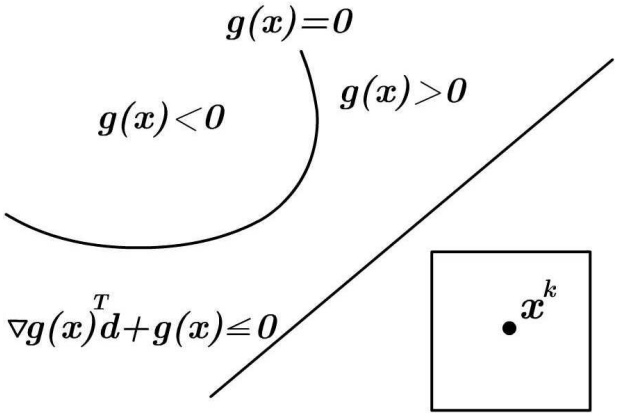
Hlavnou výhodou filtrového prístupu, je jeho jednoduchosť. Do filtra vkladáme iba dvojice skalárnych hodnôt a ukladáme si ich indexy. Odpadáva nám problém konštrukcie penalizačnej funkcie a voľby penalizačného parametra.

Mohla by sa naskytnúť otázka, či môže nastať situácia, kedy by bolo vo filtri príliš veľa prvkov. Pri numerických experimentoch Fletchera a Leyffera sa ukázalo, že vo filtri bolo uložených väčšinou menej ako 10 prvkov; maximálny počet sa vyšplhal na 43 prvkov (čo je relatívne malé číslo).

Pochopiteľne aj vo filtrových algoritmoch sa vyskytuje pár problémov:

1) Ak je pokles účelovej funkcie dostatočný, *odchýlka od ohraničení* nám môže ujsť do nekonečna a opačne. Toto sa rieši zavedením horných hraníc funkcií f a h. Z praktického hľadiska stačí ohraničiť iba *odchýlku od ohraničení*, čiže $h < a$. Takáto podmienka zavedie tak, že sa do filtra vloží bod $(a, -\infty)$.

2) Závažnejším problémom je, že po zamietnutí niekoľkých bodov nasledujúcich po sebe, sa môže *trust-region* polomer zmenšiť až natoľko, že sa úloha (QP^f) stane neprípustnou. Takúto situáciu možno vidieť na nasledovnom obrázku 4.2.



Obrázok 4.2: Príliš malý *trust-region* polomer.

Riešením je zavedenie tzv. *obnovovacej fázy*, ktorá má za úlohu nájsť prípustný bod, ktorý by bol zároveň akceptovateľný filtrom. Pre tento účel sa používajú algoritmy schopné nájsť riešenie nelineárneho algebraického systému nerovníc. Vhodným je napr. algoritmus popísaný v [2] na str.9.

Myšlienka použitia filtra síce vznikla v oblasti SQP algoritmov, no preberajú ju aj iné oblasti optimalizácie. Napr. Benson, Shanno a Vanderbei implementovali použitie filtra pre metódy vnútorného bodu. Preto existuje viacero rôznych druhov filtra. Za spomenutie stojí *Bariérový filter*, *Markovov filter*, *Multirozmerný filter*. My sa v ďalšej podkapitole budeme venovať *Fletcherovmu-Leyfferovmu filtru*.

4.2 SQP algoritmus používajúci Fletcherov-Leyfferov filter

Ako sme už spomínali predtým, myšlienka filtra prišla do styku s odbornou verejnosťou z vedeckých kruhov, ktorých stredom je profesor Roger Fletcher. V tejto podkapitole sa budeme zaoberať SQP algoritmom, ktorý používa filter vytvorený Rogerom Fletcherom a Svenom Leyfferom.

Špecifikujme si teda ich algoritmus. Doterajšia definícia filtra dovoľuje vytvárať nové body príliš blízko už známych bodov filtra, ktoré majú $h^j > 0$. Z hľadiska rýchlosti konvergencie metódy je to dosť nevýhodné. Preto sa v prípustnej oblasti definuje istý malý obal okolo hranice neprípustnosti; body z tohto obalu sa neakceptujú. Takže podmienka akceptovateľnosti filtrom sa zmení nasledovne

Definícia 7. *Bod* (f^k, h^k) *je* ***akceptovateľný*** *pre filter, ak spĺňa buď*

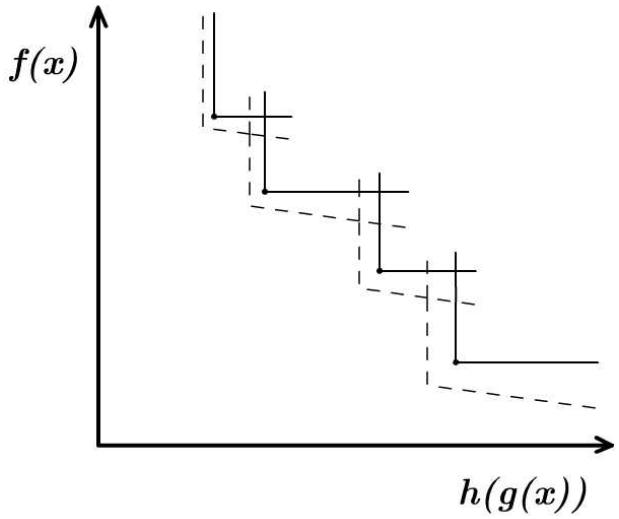
$$h^k \leq \beta h^j \tag{4.5}$$

alebo

$$f^k + \gamma h^k \leq f^j, \tag{4.6}$$

pre všetky $j \in \mathcal{F}^k$*; kde* β *a* γ *sú predvolené parametre:* $0 < \gamma < \beta < 1$*;* β *je blízko 1 a* γ *je blízko 0.*

Nerovnosť (4.5) z definície zabezpečuje dostatočné zmenšovanie h. Nerovnosť (4.6) zaručuje dostatočný pokles v f. Na ľavej strane (4.6) je vhodne zakomponované aj h tak, aby boli jednotlivé body postupnosti (f^k, h^k) tlačené k prípustnosti (t.z. $h \longrightarrow 0$; graficky je to znázornené klesajúcou polpriamkou so zvyšujúcim sa h). Grafická reprezentácia takéhoto obalu je na nasledujúcom obrázku 4.3.



Obrázok 4.3: "Zamietací" obal okolo hranice akceptovateľnosti bodov filtrom.

Ako sme už spomínali v predošlej podkapitole, pri riešení (UKP) sledujeme 2 ciele. Minimalizujeme účelovú funkciu a súčasne minimalizujeme *odchýlenie od ohraničení* (reprezentované funkciou h). V skutočnosti tieto dva ciele nie sú celkom rovnocenné. Vzhľadom na to, že optimálne riešenie (UKP)(ako úlohy ohraničenej optimalizácie) musí ležať v množinbe prípustných riešení, dostáva cieľ minimalizovať *odchýlenie od ohraničení* vyššiu prioritu. V bode optima (UKP) musí byť $h = 0$. Inak by (UKP) nemala prípusné riešenie. Na grafe teda môžme rozoznať prípustné body podľa toho, že ležia priamo na osi f, kde je $h = 0$. O dosiahnuteľnosti množiny prípustných bodov hovorí nasledovná Lemma:

Lema 1. Nech $\{f^k\}$ a $\{h^k\}$ sú také postupnosti, že $h^k \geq 0$ a f^k je monotónne klesajúca a zospodu ohraničená. Nech β a γ sú zvolené konštanky také, že
$0 < \gamma < \beta < 1$. Ak platí buď

$$h^{k+1} \leq \beta h^k \quad \text{alebo} \quad f^k - f^{k+1} \leq \gamma h^{k+1} \quad \text{, pre všetky k,}$$

potom $h^k \longrightarrow 0$.
([3], str.4)

Dôležitým prvkom Fletcherovho-Leyfferovho algoritmu je kritérium *dostatočného poklesu*:

$$\Delta f \geq \sigma \Delta q, \tag{4.7}$$

kde
$\Delta f = f(x) - f(x+d)$ je *skutočný pokles* f,
$\Delta q = q(0) - q(d) = -g^T d - \frac{1}{2} d^T B d$ je *predpovedaný pokles* f a $\sigma \in (0,1)$.

Ďalším prvkom je horná hranica h, t.z.

$$h(g(x)) \leq \beta a, \quad \text{kde } a > 0.$$

Takáto podmienka sa prakticky zavedie tak, že sa pri inicializácii vloží do filtra bod $(a, -\infty)$.

Pripomeňme si, že máme 2 ciele pre minimalizáciu - minimalizovanie účelovej funkcie $f(x)$ a funkcie *odchýlenia od ohraničení* $h(g(x))$. Definujme si preto dva druhy iterácií:

- *f-iterácia* - je to iterácia v ktorej je prvoradým cieľom pokles účelovej funkcie f (je dovolený prípadný nárast h), platí vtedy $\Delta q > 0$;
- *h-iterácia* - iterácia v ktorej je primárnym cieľom pokles *odchýlenia od ohraničení* h (je dovolený prípadný nárast f).

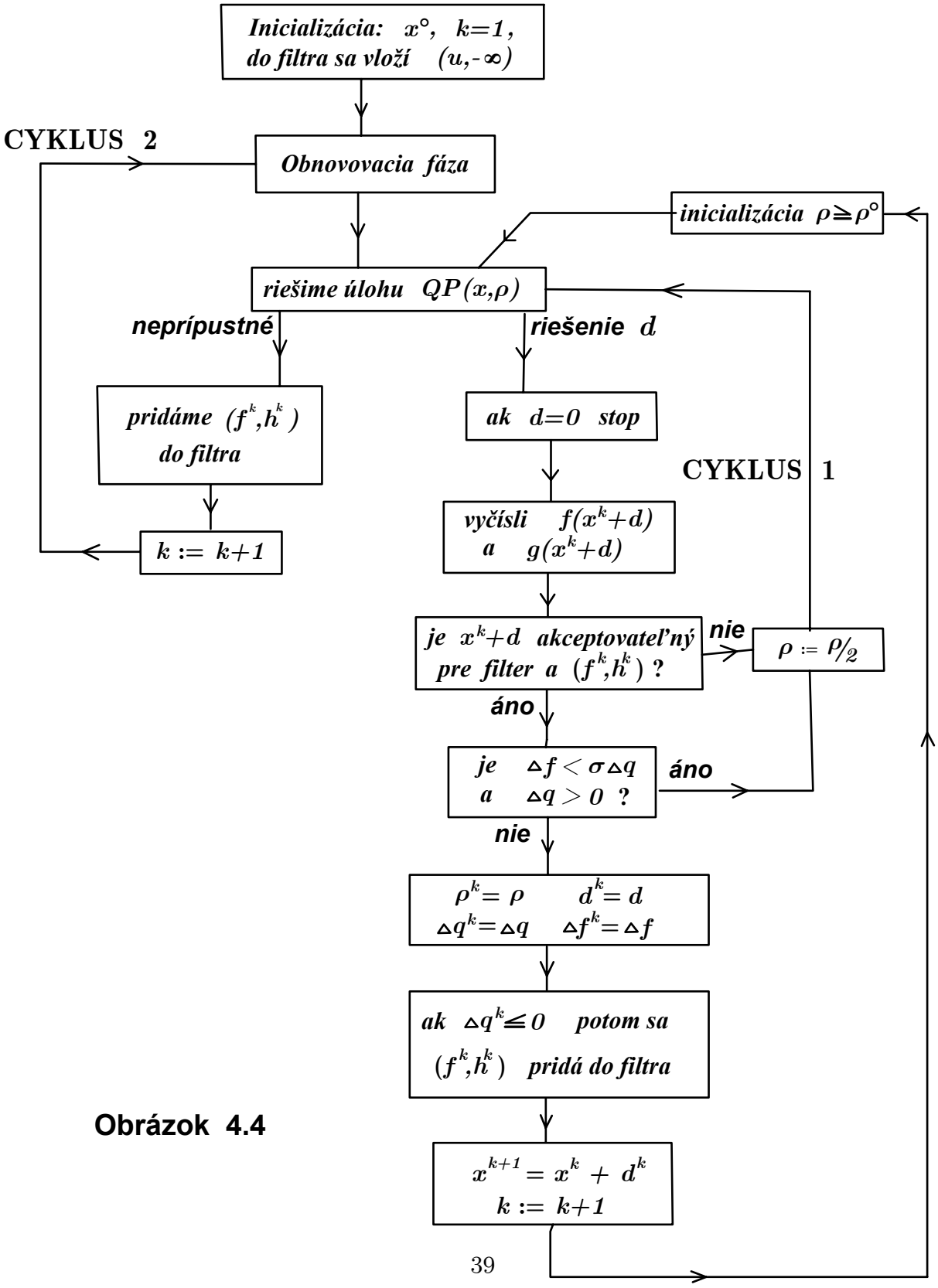
Na obrázku 4.4 je zobrazená schéma Fletcherovho-Leyfferovho filtrového SQP algoritmu. Popíšme si teraz ako daný algoritmus funguje.

Po inicializovaní vstupuje nový bod do pomocnej úlohy kvadratického programovania. Tu nastávajú 2 možnosti. Po prvé (QP^f) je konzistentná a má riešenie d, ktoré vstupuje do tzv. "cyklu 1" algoritmu. Po druhé (QP^f) je nekonzistentná alebo nemá riešenie; vtedy nastupuje "cyklus 2".

Najprv sa pozrime na cyklus 1. Do k-tej iterácie vstupujeme s dvojicou (f^k, h^k), ktorá je akceptovateľná filtrom, ale zatiaľ nie je doňho vložená. Zvolí sa hodnota *trust-region* polomeru $\rho \geq \rho^0$, kde $\rho^0 > 0$ je vopred zvolený parameter (väčšinou blízky nule, napr. 10^{-4}). Bod x^k a polomer ρ vstupujú do úlohy kvadratického programovania (QP^f). Ak je (QP^f) riešiteľná, získavame krok d. V prípade, že $d = 0$, získali sme optimálny bod úlohy (UKP). Ak $d \neq 0$, testujeme či je nový bod $x^k + d$ akceptovateľný pre súčasný filter +

"potenciálny bod filtra" (f^k, h^k). Zmysel tohto testu spočíva v tom, že ak by sme vložili do súčasného filtra (f^k, h^k) vieme, že v ďalšej (k+1) iterácii bude dvojica (f^{k+1}, h^{k+1}) akceptovateľná pre filter. Ďalej si vyčíslime Δf a Δq a testujeme, či nastal dostatočný pokles v f (inak povedané či nastala *f-iterácia*). Ak nie, zmenší sa *trust-region* polomer ρ a znovu sa počíta úloha kvadratického programovania (QP) s novým ρ. Zmenšovanie *trust-region* polomeru sa môže robiť ľubovoľne, ale najbežnejšie je položiť $\rho := \rho/2$. Ak nastala *f-iterácia*, parametre ρ, d, Δf a Δq sa prehlásia za vhodné parametre k-tej iterácie ρ^k, d^k, Δf^k a Δq^k. Testuje sa, či sa jedná aj o *h-iteráciu*; ak áno, bod (f^k, h^k) sa pridá do filtra. Napokon sa vyčísli ďalší bod $x^{k+1} = x^k + d^k$, inicializuje sa $\rho \geq \rho^0$ a nový bod x^{k+1} vstupuje v k+1 iterácii do (QP^f). V tomto cykle je kľúčovým prvkom znižovanie ρ až pokiaľ nie sú splnené určité testy alebo sa QP nestane neprípustný. Vtedy nastupuje cyklus 2.

V cykle 2 sa najprv zahrnie dvojica (f^k, h^k) do filtra, posunie sa počítadlo na $k := k + 1$ a potom sa hľadá nový bod x^k pomocou *obnovovacej fázy*. Účelom tohto procesu je nájsť bod, ktorý by bol prípustný pre pomocnú úlohu kvadratického programovania (pre isté $\rho \geq \rho^0$) a zároveň akceptovateľný pre filter. Samozrejme, že v tomto novom bode môže narásť hodnota účelovej funkcie f, ale vzhľadom na to, že sa hľadá prípustný bod, zredukuje sa h (takže sa jedná o *h-iteráciu*). Môže sa stať, že *obnovovacia fáza* skonverguje k neprípustnému bodu - to znamená, že existuje nenulové minimum $h(g(x))$. Z toho vyplýva, že pôvodná úloha (UKP) je neprípustná.


Inicializácia: x°, $k=1$,
do filtra sa vloží $(u,-\infty)$
CYKLUS 2
Obnovovacia fáza
inicializácia $\rho \geqq \rho^\circ$
riešime úlohu $QP(x,\rho)$
neprípustné
riešenie d
pridáme (f^k,h^k)
do filtra
ak $d=0$ stop
CYKLUS 1
$k := k+1$
vyčísli $f(x^k+d)$
a $g(x^k+d)$
je x^k+d akceptovateľný
pre filter a (f^k,h^k) ?
nie
$\rho := \rho/2$
áno
je $\triangle f < \sigma \triangle q$
a $\triangle q > 0$?
áno
nie
$\rho^k = \rho$ $d^k = d$
$\triangle q^k = \triangle q$ $\triangle f^k = \triangle f$
ak $\triangle q^k \leqq 0$ potom sa
(f^k,h^k) pridá do filtra
$x^{k+1} = x^k + d^k$
$k := k+1$

Obrázok 4.4

Kapitola 5

Numerické experimenty

5.1 Popis testovania

V tejto kapitole budeme prezentovať naše výsledky z numerických experimentov. Testovali sme 112 vybraných úloh z testovacej sady Klausa Schittkowskeho pre úlohy tvaru

$$\begin{cases} \min\limits_{x \in R^n} & f(x) \\ \text{s.t.:} & g(x) \leq 0_{m_n} \\ & h(x) = 0_{m_r} \end{cases} \tag{5.1}$$

Z toho bolo 96 úloh len s ohraničeniami typu nerovností, t.z. tvaru našej úlohy (UKP); 6 úloh len s rovnostnými ohraničeniami a 10 úloh s ohraničeniamy typu rovností aj nerovností. Každá z úloh mala minimálne jedno kvadratické ohraničenie a 12 z nich malo navyše aj lineárne ohraničenia. Hlavné testovanie prebehlo na úlohách rozmeru 2 až 15 premenných a navyše boli vyskúšané jedna úloha s 20 a jedna s 50 premennými (tieto 2 úlohy boli dosť jednoduché; mali len po jednom rovnostnom ohraničení). Maximálny počet ohraničení bol 35 (samotná úloha mala 10 premenných). Počet rovnostných ohraničení v úlohách pochopiteľne nepresiahol počet premenných.

Sada cca 400 testovacích úloh Klausa Schittkowského bola špeciálne vytváraná tak, aby boli jednotlivé úlohy náročné a aby obsahovali rôzne problémy. K dispozícii sme mali aj optimálne riešenia, poskytnuté autorom úloh.

V nasledujúcej tabuľke je uvedené koľko úloh sme testovali pre jednotlivé rozmery úloh:

Rozmer úloh:	2	3	4	5	6	7	8	9	10	13	15	20	50
Počet úloh:	47	18	10	5	5	7	3	3	4	2	6	1	1

Obrázok 5.1: Tabuľka početnosti úloh jednotlivých rozmerov

Na experimenty sme použili softvér TOMLAB - nadstavba pre MATLAB. Testovali sme jeho 4 algoritmy požívajúce SQP metódy:

- **conSolve** je algoritmus používajúci SQP metódu vytvorenú Klausom Schittkowskim. Na hľadanie optimálneho kroku používa *Augmented Lagrangian* penalizačnú funkciu. Hessovu maticu Lagrangeovej funkcie z pomocnej úlohy kvadratického programovania aproximuje BFGS metódou.

- **snopt** popisujeme v podkapitole 3.3.

- **filterSQP** je SQP metóda využívajúca filter popísaná v podkapitole 4.2.

- **fmincon** je rozhranie používajúce podľa potreby najčastejšie buď *snopt* alebo *nlpSolve*, čo je staršia verzia *filterSQP*

Našim hlavným cieľom bolo porovnať výkony filtrového SQP algoritmu oproti klasickým SQP algoritmom používajúcim penalizačnú funkciu. *conSolve* a *snopt* sme zvolili ako predstaviteľov klasického SQP prístupu. *filterSQP* zastupuje filtrové SQP metódy. Rozhranie *fminon* sme zaradili ako reprezentanta zjednotenia obidvoch prístupov.

Testy prebehli na počítači s procesorom Pentium 4 - 2 GHz, 512 MB RAM.
Pri testovaní sme porovnávali 5 "veličín":

- Čas, ktorý procesor potreboval na vyriešenie úlohy.

- Počet iterácii.

- Relatívnu presnosť získaného bodu optima $\longrightarrow \frac{\left\|x^*_{\text{známe}} - x^*_{\text{získané}}\right\|}{\left\|x^*_{\text{známe}}\right\|}$.

- Relatívnu presnosť hodnoty účelovej funkcie v získanom bode optima $\longrightarrow \frac{\left\|f(x^*_{\text{známe}}) - f(x^*_{\text{získané}})\right\|}{\left\|f(x^*_{\text{známe}})\right\|}$.

- Úspešnosť vyriešenia úloh algoritmom.

5.2 Výsledky

Najpr uvedieme priemerné hodnoty jednotlivých testovaných veličín pre všetkých 112 úloh. Potom ukážeme priemerné hodnoty testovaných veličín pre úlohy rozdelené do skupín podľa rozmeru premennej x.

Nasledujúca tabuľka obsahuje priemerné hodnoty testovaných veličín pre všetkých 112 úloh riešených jednotlivými algoritmami:

	conSolve	fmincon	snopt	filterSQP
Čas	10,4583	1,5042	2,1692	0,7273
Počet iterácií	26,8	44,7	51,3	35,5
Rel. Presnosť x*	5,1584	444,8575	0,3591	3,0464
Rel. Presnosť f(x*)	1,0561	0,4034	0,2179	0,2239
Úspešnosť	82,14%	92,86%	100,00%	100,00%

Obrázok 5.2: Tabuľka 5.1

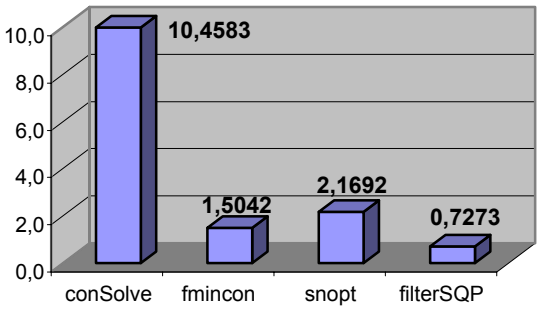
Ukázalo sa, že algoritmus *conSolve* nezvláda riešiť úlohy väčšieho rozmeru ako 7 v rozumnom čase. Na porovnanie uvádzame aj tabuľku len pre rozmery 2 až 7 (spolu 92 úloh).

	conSolve	fmincon	snopt	filterSQP
Čas	9,1373	1,7441	2,1261	0,4977
Počet iterácií	19,7	26,8	56,6	53,4
Rel. Presnosť x*	6,5599	5,5743	0,6404	5,7423
Rel. Presnosť f(x*)	1,4082	0,4792	0,4720	0,4850
Úspešnosť	97,83%	94,57%	100,00%	100,00%

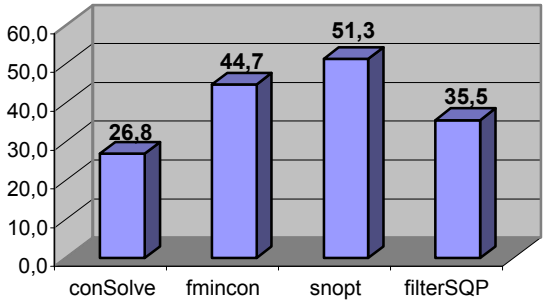
Obrázok 5.3: Tabuľka 5.2

Vzhľadom na to, že z hľadiska porovnávania ostatných algoritmov s algoritmom *conSolve* sa pomerné výsledky v tabuľke 5.2 príliš nezmenili oproti tabuľke 5.1 (aj kvôli tomu, že v tabuľke 5.2 je iba o 20 úloh menej), celkovú analýzu robíme iba podľa tabuľky 5.1. Uveďme si preto grafy porovnávajúce jednotlivé testované veličiny spravené na základe tabuľky 5.1.

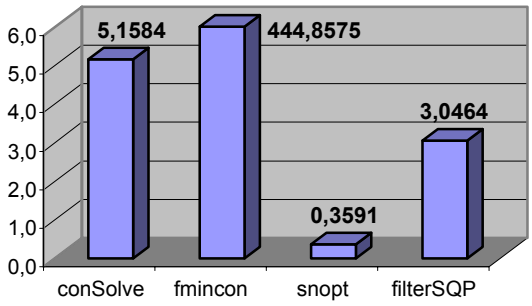
Graf 5.1: Čas



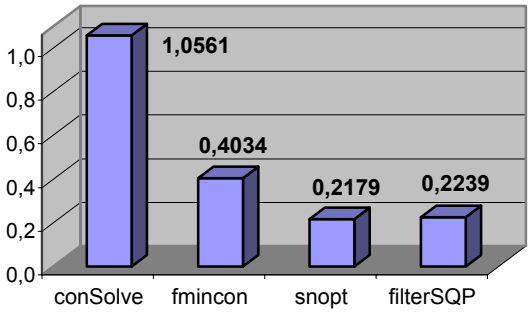
Graf 5.2: Počet iterácii



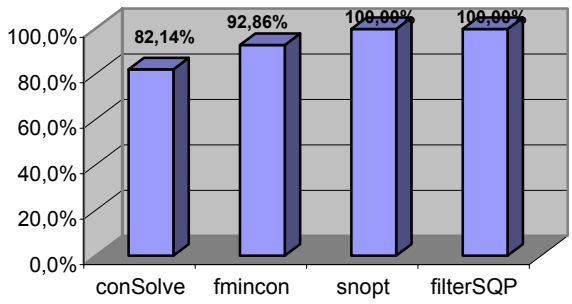
Graf 5.3: Relatívna presnosť x*



Graf 5.4: Relatívna presnosť f(x*)



Graf 5.5: Úspešnosť



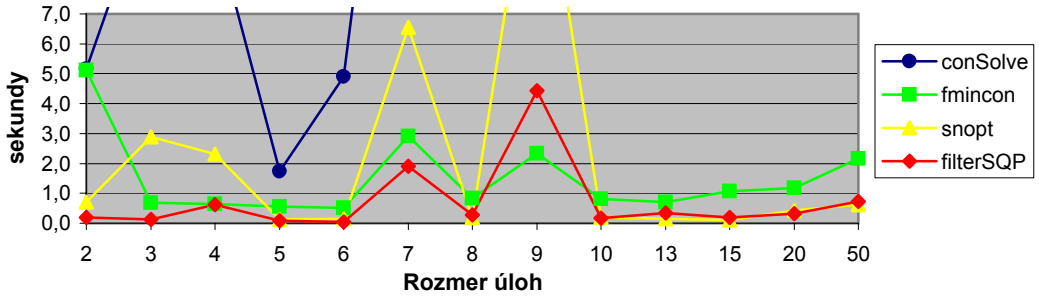
Experimentálne úlohy sme si rozdelili na skupiny podľa rozmeru premennej x. Jednotlivé testované veličiny sme porovnávali pre jednotlivé rozmery úloh. Kvôli prehľadnosti uvádzame ku každej tabuľke aj graf zobrazujúci jej hodnoty.

Porovnanie času pre jednotlivé rozmery úloh

Tab 5.3: Čas

Počet úloh	Rozmer	conSolve	fmincon	snopt	filterSQP
47	2	5,1563	5,1252	0,6989	0,1918
18	3	11,3096	0,6883	2,8916	0,1241
10	4	9,8924	0,6524	2,3197	0,6296
5	5	1,7406	0,5592	0,1344	0,0812
5	6	4,9000	0,5180	0,1564	0,0532
7	7	21,8248	2,9217	6,5559	1,9063
3	8		0,8333	0,1823	0,2707
3	9		2,3227	13,7550	4,4373
4	10		0,8177	0,1990	0,1755
2	13		0,7035	0,1480	0,3515
6	15		1,0810	0,1120	0,2032
1	20	14,2500	1,1720	0,4220	0,3120
1	50	14,5930	2,1600	0,6250	0,7180
	Spolu:	10,4583	1,5042	2,1692	0,7273

Graf 5.6: Čas

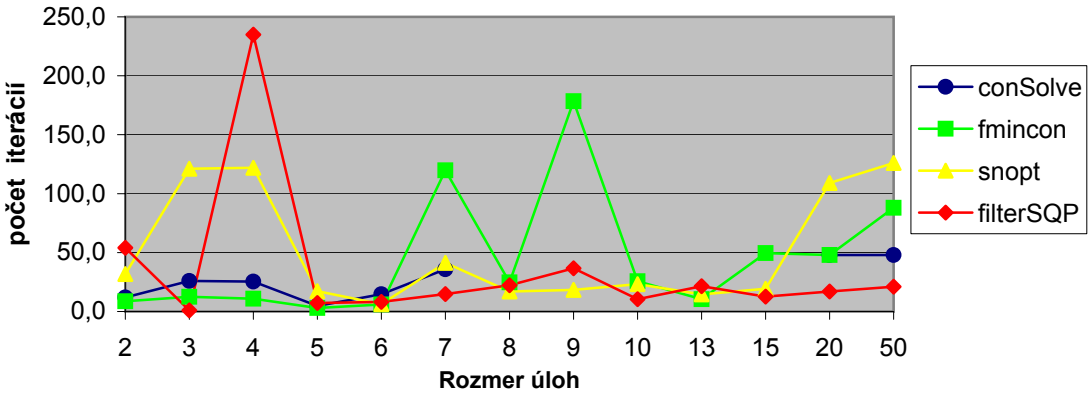


Porovnanie počtu iterácií pre jednotlivé rozmery úloh

Tab 5.4: Počet iterácií

Počet úloh	Rozmer	conSolve	fmincon	snopt	filterSQP
47	2	11,9	8,7	31,7	54,1
18	3	25,9	12,4	121,1	1,0
10	4	25,4	10,9	121,9	234,9
5	5	4,8	3,0	17,2	7,2
5	6	14,6	6,0	6,2	8,2
7	7	35,8	119,8	41,3	14,9
3	8		24,7	17,0	22,3
3	9		178,5	18,3	36,7
4	10		25,7	23,3	10,5
2	13		10,5	14,5	21,5
6	15		49,5	19,3	12,7
1	20	48,0	48,0	109,0	17,0
1	50	48,0	88,0	126,0	21,0
	Spolu:	26,8	45,0	51,3	35,5

Graf 5.7: Počet Iterácií

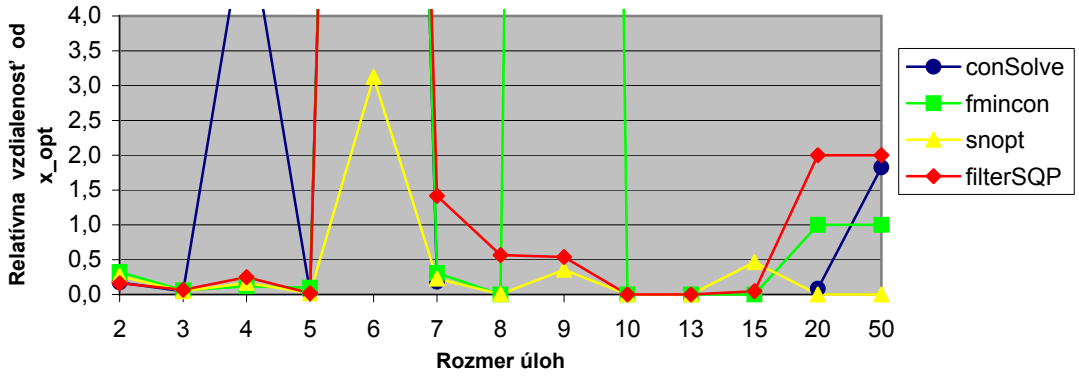


Porovnanie relatívnej presnosti x^* pre jednotlivé rozmery úloh

Tab 5.5: Relatívna presnosť x*

Počet úloh	Rozmer	conSolve	fmincon	snopt	filterSQP
47	2	0,1717	0,3221	0,2634	0,1600
18	3	0,0453	0,0588	0,0464	0,0673
10	4	5,5406	0,1163	0,1664	0,2494
5	5	0,0277	0,0968	0,0187	0,0187
5	6	33,3910	32,5430	3,1207	32,5430
7	7	0,1833	0,3087	0,2270	1,4153
3	8		0,0010	0,0056	0,5656
3	9		5 747,7000	0,3551	0,5371
4	10		0,0000	0,0000	0,0000
2	13		0,0009	0,0007	0,0007
6	15		0,0003	0,4642	0,0464
1	20	0,0817	1,0000	0,0001	2,0000
1	50	1,8257	1,0000	0,0002	2,0000
	Spolu:	5,1584	444,8575	0,3591	3,0464

Graf 5.8: Relatívna presnosť x*

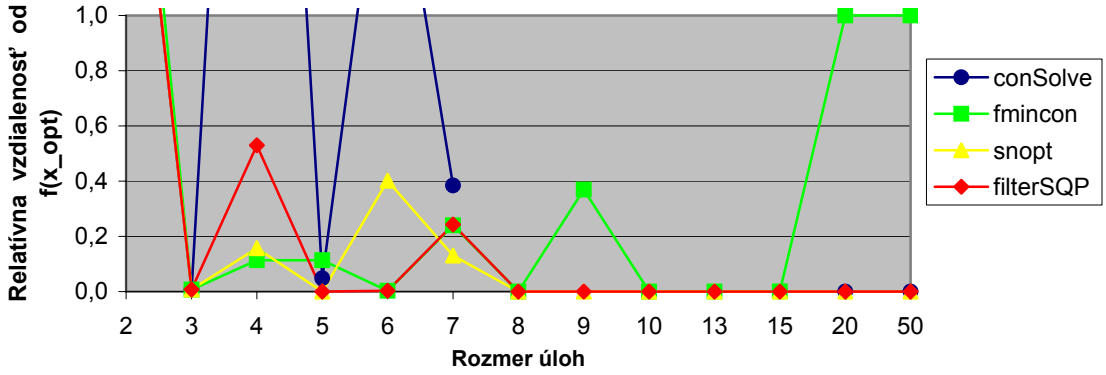


Porovnanie relatívnej presnosti $f(x^*)$ pre jednotlivé rozmery úloh

Tab 5.6: Relatívna presnosť *f*(x*)

Počet úloh	Rozmer	conSolve	fmincon	snopt	filterSQP
47	2	2,1218	2,3974	2,1351	2,1274
18	3	0,0073	0,0072	0,0073	0,0074
10	4	4,0531	0,1139	0,1573	0,5297
5	5	0,0482	0,1139	0,0000	0,0000
5	6	1,8346	0,0029	0,4016	0,0029
7	7	0,3842	0,2398	0,1307	0,2426
3	8		0,0001	0,0000	0,0000
3	9		0,3691	0,0000	0,0000
4	10		0,0000	0,0000	0,0000
2	13		0,0000	0,0000	0,0000
6	15		0,0003	0,0002	0,0002
1	20	0,0000	1,0000	0,0000	0,0000
1	50	0,0000	1,0000	0,0000	0,0000
	Spolu:	1,0561	0,4034	0,2179	0,2239

Graf 5.9: Relatívna presnosť f(x*)



5.3 Zhrnutie

Za najdôležitejšiu testovaciu veličinu považujeme čas, za ktorý algoritmus vyrieši zadanú úlohu. Ako možno vidieť v tejto kategórii má najlepšie výsledky *filterSQP*. Ďalšími dôležitými hodnotami pre porovnávanie sú relatívna presnosť funkčnej hodnoty v získanom riešení a relatívna presnosť získaneho riešenia. Tu sú dominantné *snopt* a *filterSQP*, pričom *snopt* má jemnú prevahu. V rebríčku úspešnosti takisto vedie skupina *snopt* a *filterSQP*. Čo sa týka veličiny - počet iterácii, tú uvádzame len kvôli prehľadu; pre porovnávanie to nie je príliš smerodajné, pretože každý algoritmus v jednej iterácii vykonáva iný počet operácii.

Za celkových výťazov považujeme dvojicu *snopt* a *filterSQP*. Ukázalo sa, že *conSolve* je podstatne horší ako tieto 2 algoritmy. *fmincon* tiež nesplnil naše očakávania. Vzhľadom na to, že kombinuje použitie viacerych algoritmov predpokladali sme lepšie výsledky. V kategórii času zastáva síce 2. miesto, no v ostatných oblastiach sú jeho výsledky horšie.

Porovnajme teda *snopt* (ako najúspešnejšieho zástupcu SQP metód používajúcich penalizačnou funkciou) a filtrový algoritmus *filterSQP*. Z hľadiska úspešnosti vyriešili obidva algoritmy všetky testovacie úlohy. *filterSQP* je lepší v rýchlosti riešenia úloh, ale *snopt* je v riešení presnejší.

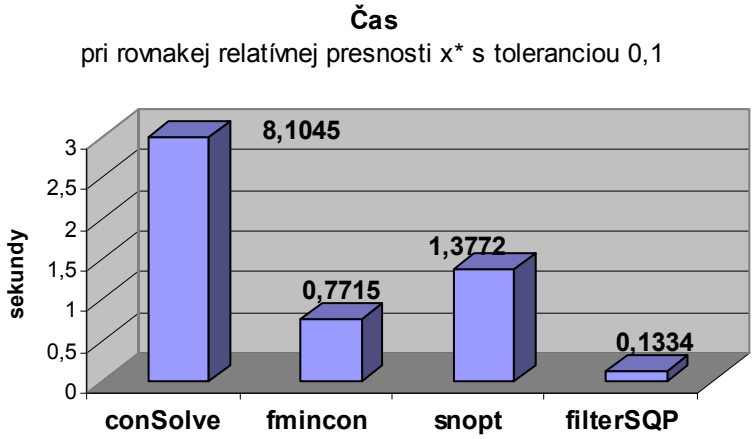
Ako rozšírenie k našim numerickým experimentom, by bolo v budúcnosti vhodné vykonať testy daných algoritmov aj pre úlohy väčšieho rozmeru - rádovo stoviek až tisícov premenných. Tieto pokusy by sa mali prevádzať na výkonnejších počítačoch.

Doplnok pre numerické experimenty

Za najdôležitejšie porovnávacie veličiny pokladáme *Čas* a *Relatívnu presnosť x**. V ďalšom sme zafixovali jednu z týchto veličín a porovnávali sme druhú a naopak. Inak povedané vybrali sme také príklady, ktoré dosahovali takmer rovnaké hodnoty fixovanej veličiny a potom sme porovnali hodnoty ich druhej veličiny.

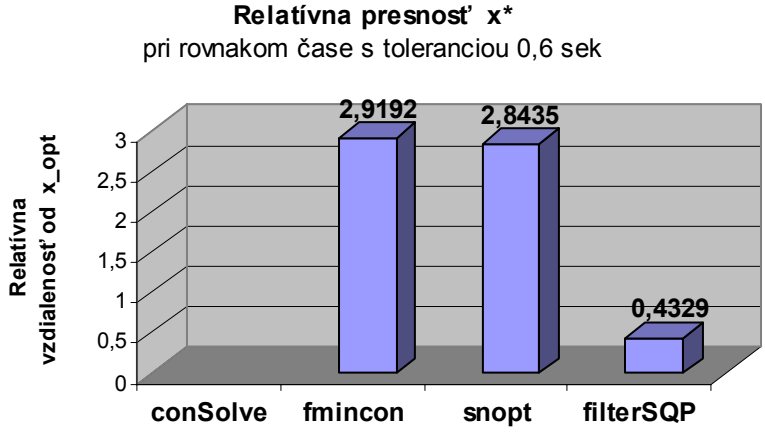
Porovnanie času pri rovnakej "relatívnej presnosti x*" s toleranciou 0,1

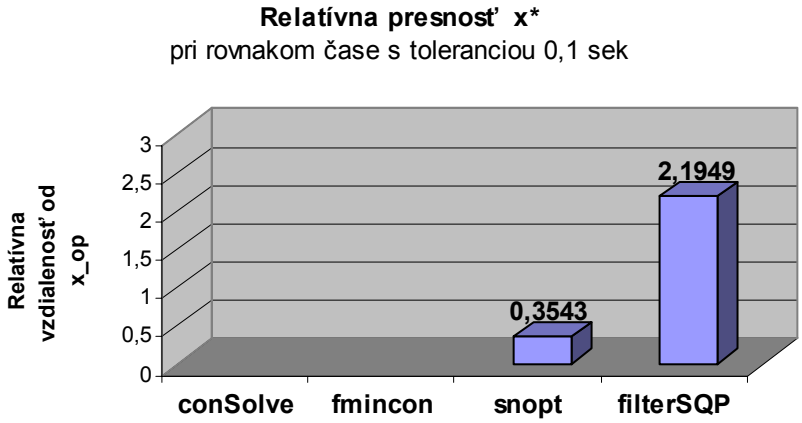
(každý z algoritmov spĺňal danú toleranciu v 63 prípadoch)



Porovnanie relatívnej presnosti x* pri rovnakom "čase" s toleranciou 0,1 a 0,6 sekúnd

(Pre toleranciu 0,6: v 59 prípadoch 3 algoritmy splnili toleranciu)
(Pre toleranciu 0,1: v 79 prípadoch 2 algoritmy splnili toleranciu)





Zhrnutie

Opäť sa ukázalo, že vzhľadom na *Čas* je najrýchlejši algoritmus *filterSQP*. Vzhľadom k *Relatívnej Presnosti x** je dominantný *snopt*. Ktorý z týchto algoritmov je lepší? Na túto otázku nie je jasná odpoveď. Veľmi závisí od použitia algoritmov v praxi. Ak je prioritné čo najrýchlejšie vyriešenie problému, je najvhodnejším algoritmom *filterSQP*. Ak rýchlosť nehrá hlavnú úlohu a je zapotreby čo najväčšia presnosť, najlepším je *snopt*.

Kapitola 6

Záver

Predkladaná diplomová práca je venovaná metódam Sekvenčného Kvadratického Programovania (SQP). Snažili sme sa uviesť ucelený pohľad na SQP metódy a ukázať na ich aplikácie v praxi.

Porovnávali sme filtrové SQP metódy s klasickými SQP algoritmami, používajúcimi penalizačnú funkciu. Zaoberali sme sa štruktúrou filtrového SQP algoritmu Rogera Fletchera a Svena Leyffera.

Realizovali sme numerické experimenty za účelom porovnania filtrovej SQP s klasickými SQP metódami. Použili sme softvér TOMLAB a v ňom naprogramované funkcie, používajúce SQP metódy. Na základe získaných numerických výsledkov, sme porovnali jednotlivé SQP algoritmy.

Všetky testovacie úlohy a experimentami získané veličiny z MATLABu sme zálohovali; k dispozícii sú na priloženom CD, ktoré obsahuje aj Excelovský súbor s tabuľkami výstupov a ich grafmi.

Literatúra

[1] Paul T. Boggs, Jon W. Tolle, *Sequential Quadratic Programming*, Acta Numerica, pp. 1-51, 1995.

[2] Roger Fletcher, Sven Leyffer, *Nonlinear Programming without a penalty function*, University of Dundee Numerical Analysis Report, NA/171, 22 September 1997, revised 22 December 1998, revised 24 August 2000.

[3] Roger Fletcher, Sven Leyffer, Philippe L. Toint *On the Global Convergence of a Filter-SQP Algorithm*, Numerical Analysis Report NA/197, October 2000.

[4] Craig T. Lawrence, Andre L. Tits *A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm* , SIAM J. OPTIM. Vol.11, No.4, pp. 1092-1118, 2001.

[5] Philip E. Gill, Walter Murray, Michael A. Saunders *Snopt: An SQP Algorithm for Large-Scale Constraoned Optimization*, http://www.siam.org/journals/siopt/00-0/00000.html, published electronicaly 2002.

Príloha 1

Listing programu *filterSQP*

Listing programu *snopt*