Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave


# DIPLOMOVÁ PRÁCA

Bratislava 2003 Mária Kubíčková

Fakulta Matematiky, Fyziky a Informatiky
Univerzity Komenského v Bratislave

Ekonomická a finančná matematika


# Úlohy O Komplementarite A Ich Ekonomické Aplikácie

Diplomová práca

Diplomant: Mária Kubíčková
Vedúci diplomovej práce: doc. RNDr. Milan Hamala, CSc.

Bratislava 2003

Čestne prehlasujem, že som diplomovú prácu spracovala samostatne a použila som len literatúru uvedenú v zozname.

Touto cestou by som sa chcela poďakovať svojmu vedúcemu diplomovej práce doc. RNDr. Milanovi Hamalovi, CSc. za jeho odborné vedenie, cenné rady a za množstvo času, ktoré mi venoval pri vypracovávaní diplomovej práce.

# Obsah

# Úvod

Predkladaná diplomová práca je venovaná úlohe o komplementarite a k nej príbuzným úlohám. Ciele práce sú nasledovné:

1. Uviesť ucelený pohľad na úlohy o komplementarite, resp. k nim príbuzné úlohy, ako je úloha o variačných nerovnostiach, úloha o sedlovom bode, prípadne niektoré ich zovšeobecnenia.

2. Poukázať na niektoré ekonomické aplikácie modelované pomocou úlohy o komplementarite.

3. Ilustrovať možnosti riešenia nelineárnych úloh o komplementarite.

Pri plnení prvého cieľa sme vychádzali z práce [1] z roku 1997. Pri štúdiu uvedenej práce sme narazili na niektoré nepresnosti a nezrovnalosti, ktoré sme so značným úsilím odstránili a príslušný text sme metodicky vylepšili (kapitola 1).

Pri plnení druhého cieľa sme taktiež vychádzali z [1]. Táto práca však popri ekonomických aplikáciách obsahuje aj niektoré technické aplikácie, ktorými sme sa nezaoberali. Zvolili sme si dva modely (kapitoly 2, 3), ktoré sme podrobne dopracovali. Aj tu sme našli niektoré drobné chyby, ktoré sme odstránili a príslušný text sme uviedli na správnu mieru.

Pri plnení tretieho cieľa sme vychádzali z práce [2] z roku 1999, ktorá bola značne technicky náročná. Z tejto práce sme prevzali algoritmus regularizačnej newtonovej metódy na riešenie nelineárnych úloh o komplementarite (kapitola 4). Uvedený algoritmus sme naprogramovali v jazyku C++ (kapitola 5) a realizovali sme s ním rozsiahle numerické experimenty, ktoré ukázali, že tento algoritmus je primerane efektívny.

Teraz podrobnejšie charakterizujeme jednotlivé kapitoly.

Prvá kapitola obsahuje prehľad rôznych tvarov úlohy o komplementarite. Uvádzame definíciu a popis základného tvaru úlohy o komplementarite a príbuzných úloh. Popisujeme tu aj podmienky, pri ktorých je úloha o komplementarite ekvivalentná s inými úlohami.

Druhá kapitola obsahuje model všeobecnej rovnováhy na trhu s komoditami. Tento model obsahuje sektor výroby a sektor spotreby. Pre oba sektory tu popisujeme, ako sa správajú a ako maximalizujú svoj zisk, resp. úžitok. V závere kapitoly je tento model sformulovaný ako zmiešaná úloha o komplementarite.

Tretia kapitola obsahuje stručný popis základnej úlohy hľadania Nashovho ekvilibria, ktorá sa dá sformulovať ako úloha o komplementarite a taktiež ako úloha o variačných nerovnostiach. V druhej časti je popísaný Nash-Cournotov produkčný model, ktorý sa dá sformulovať v tvare úlohy o variačných nerovnostiach.

V štvrtej kapitole uvádzame popis algoritmu regularizačnej newtonovej metódy, ktorý je jeden z mnohých algoritmov na riešenie úloh o komplementarite.

Piata kapitola obsahuje popis a výsledky numerických experimentov, ktoré sme uskutočnili s nami naprogramovanou verziou algoritmu uvedeného v štvrtej kapitole.

V prílohe je uvedný stručný popis jednotlivých modulov programu a funkcií v nich obsiahnutých. Taktiež je tu aj výpis zdrojového kódu popisovaného programu. Na záver pripájame CD médium s naprogramovanou regularizačnou newtonovou metódou a elektronickou verziou textu diplomovej práce.

# Kapitola 1

# Typy úloh o komplementarite.

Vo všeobecnosti existuje množstvo typov úloh o komplementarite. V tejto kapitole uvedieme len niektoré typy týchto úloh. Neskôr popíšeme aj niektoré ekonomické aplikácie úloh o komplementarite.

## 1.1 Základný typ úlohy o komplementarite.

Úloha o komplementarite v najjednoduchšom tvare (základný typ) sa pre zobrazenie $\mathbf{F} : R^n \to R^n$ formuluje nasledovne :

**Úloha NCP(F) :**

Nájsť také $\mathbf{x}^* \in R^n$ aby platilo

$$\text{NCP}(F): \qquad \mathbf{x}^* \geq \mathbf{0_n}, \quad \mathbf{F}(\mathbf{x}^*) \geq \mathbf{0_n}, \quad (\mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = 0 \tag{1.1}$$

Poznamenávame, že označenie NCP je utvorené zo začiatočných písmen anglického názvu Nonlinear Complementarity problem. Špeciálne, ak je zobrazenie F lineárne, zvykne sa hovoriť o úlohe LCP z anglického Linear Complementarity Problem.

Kvôli úplnosti zapíšeme úlohu NCP v zložkovom tvare

$$x_j^* \geq 0, \quad F_j(\mathbf{x}^*) \geq 0, \quad (x_j^*) F_j(\mathbf{x}^*) \ = 0 \qquad (j = \ 1, \ 2, \ \ldots \ n) \tag{1.1a}$$

Je zrejmé, že z (1.1a) vyplýva zápis (1.1). Platí aj opačná implikácia. Stačí, ak si uvedomíme, že $(\mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = \sum_{j=1}^{n} x_j^* F_j(\mathbf{x}^*)$ a z prvých dvoch podmienok vieme, že sčítance sú nezáporné.

Dá sa ľahko ukázať, že pre $a, b \in R$ platí

$$\{a \geq 0, \quad b \geq 0, \quad ab = 0\} \quad \Leftrightarrow \quad \min(a, b) = 0 \quad ,$$

a teda vzťahy (1.1a) možno ekvivalentne prepísať v tvare

$$\min(x_j, F_j(\mathbf{x})) = 0 \qquad\qquad (j = 1,\ 2,\ \ldots\ n) \tag{1.1b}$$

resp. použitím vektorového zápisu v tvare:

$$\min(\mathbf{x}, \mathbf{F}(\mathbf{x})) = \mathbf{0} \tag{1.1c}$$

Okrem toho sa dá ešte ľahko ukázať, že pre $a, b \in R$ platí aj (tzv. Fisherova transformácia z r. 1992)

$$\{a \geq 0, \quad b \geq 0, \quad ab = 0\} \quad \Leftrightarrow \quad \sqrt{a^2 + b^2} = a + b,$$

a teda (1.1a) možno zapísať tiež v nasledovnom ekvivalentnom tvare

$$\sqrt{x_j^2 + F_j(\mathbf{x})^2} \ = \ x_j + F_j(\mathbf{x}) \qquad\qquad (j = 1,\ 2,\ \ldots\ n) \tag{1.1d}$$

Úlohy o komplementarite tohto typu prirodzene vznikajú z nutných podmienok optimality pre úlohu nelineárneho programovania

$$\min\{ \quad f(\mathbf{x}) \ \mid \ \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \ \ \mathbf{x} \geq \mathbf{0} \quad \}, \tag{1.2}$$

s Lagrangeovou funkciou

$$L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \mathbf{u}^T \mathbf{g}(\mathbf{x}), \qquad \mathbf{x} \in R^n_+, \quad \mathbf{u} \in R^m_+ \tag{1.2a}$$

kde $f : R^n \to R$, $\mathbf{g} : R^n \to R^m$ sú spojite diferencovateľné. V tom prípade k (regulárnemu) optimálnemu riešeniu $\mathbf{x^0}$ úlohy (1.2) existuje bod $\mathbf{u^0}$ tak, že platí

$$\begin{array}{ll} \dfrac{\partial L^0}{\partial \mathbf{x}} \geq \mathbf{0^T_n} & \dfrac{\partial L^0}{\partial \mathbf{u}} \leq \mathbf{0^T_m} \\ \dfrac{\partial L^0}{\partial \mathbf{x}}\ \mathbf{x^0} = 0 & \dfrac{\partial L^0}{\partial \mathbf{u}}\ \mathbf{u^0} = 0 \\ \mathbf{x^0} \geq \mathbf{0_n} & \mathbf{u^0} \geq \mathbf{0_m} \end{array} \tag{1.2b}$$

Ľahko vidieť, že ak označíme $\mathbf{y^0} = (\mathbf{x^0}, \mathbf{u^0})$ a $\mathbf{F} = (\frac{\partial L}{\partial \mathbf{x}}, -\frac{\partial L}{\partial \mathbf{u}})$, podmienky (1.2b) sa dajú zapísať v tvare (1.1).

Ak v úlohe (1.2) nie je ohraničenie $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, dostaneme jednoduchú optimalizačnú úlohu :

$$\min\{\ f(\mathbf{x}) \ \mid \ \mathbf{x} \geq \mathbf{0}\ \} \tag{1.3}$$

Lagrangeovou funkciou $L(\mathbf{x}, \mathbf{u})$ je v tomto prípade samotná funkcia $f(\mathbf{x})$ ($L(\mathbf{x}, \mathbf{u}) \equiv f(\mathbf{x})$), a tak príslušné nutné podmienky optimality majú nasledovný tvar:

$$\mathbf{x} \geq \mathbf{0}, \quad \nabla f(\mathbf{x}) \geq \mathbf{0_n}, \quad \mathbf{x}^T \nabla f(\mathbf{x}) \ = 0 \tag{1.3a}$$

Toto je úloha o komplementarite (1.1), kde $\mathbf{F}(\mathbf{x}) = \nabla f(\mathbf{x})$

## 1.2 Úlohy o komplementarite implikované niektorými variačnými nerovnosťami.

Úloha riešenia variačných nerovníc je svojim obsahom veľmi blízka úlohe o komplementarite. Pre danú množinu $\Omega \in R^n$ a dané zobrazenie $\mathbf{F} : \Omega \to R^n$ ju možno formulovať nasledovne :

**Úloha VI($\mathbf{F}, \Omega$):**

$$\text{VI}(F, \Omega): \qquad \text{nájsť } \mathbf{x} \in \Omega \text{ také, že } \forall\ \mathbf{y} \in \Omega \text{ platí} \quad (\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x}) \geq 0, \tag{1.4}$$

Poznamenávame, že označenie VI je utvorené zo začiatočných písmen anglického názvu Variational Inequality (variačná nerovnosť).

**Veta 1.** *V prípade, že $\Omega = R^n_+$, úloha (1.4) je ekvivalentná s úlohou (1.1).*

*Dôkaz.*

1. **(1.4) $\Rightarrow$ (1.1)**
   (Z úlohy o variačných nerovnostiach vyplýva úloha o komplementarite)

   Úloha VI($F, R^n_+$) má tvar:

   $$\text{nájsť} \quad \mathbf{x} \geq \mathbf{0}: \quad (\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x}) \geq 0 \quad \forall \mathbf{y} \geq \mathbf{0},$$

   čo je ekvivalentné so zápisom:

   $$\text{nájsť} \quad \mathbf{x} \geq \mathbf{0}: \quad \mathbf{y}^T \mathbf{F}(\mathbf{x}) - \mathbf{x}^T \mathbf{F}(\mathbf{x}) \geq 0 \quad \forall \mathbf{y} \geq \mathbf{0}.$$

   Pre $\mathbf{y} = \mathbf{0_n}$ máme $-\mathbf{x}^T \mathbf{F}(\mathbf{x}) \geq 0$, z čoho vyplýva $\mathbf{x}^T \mathbf{F}(\mathbf{x}) \leq 0$.

   Teraz ukážeme, že $\mathbf{F}(\mathbf{x}) \geq \mathbf{0_n}$. Máme $(\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x}) \geq 0 \qquad \forall \mathbf{y} \geq \mathbf{0}$.
   Zvolíme také $\mathbf{y^i}$, že $\mathbf{y^i} - \mathbf{x} = \mathbf{e^i}$, kde $\mathbf{e^i}$ je vektor, ktorý má $i$-ty prvok 1 a ostatné sú nulové. Potom dostávame

   $$\begin{aligned} (\mathbf{e^i})^T \mathbf{F}(\mathbf{x}) \geq 0, \qquad & \text{čo implikuje} \\ F_i(\mathbf{x}) \geq 0 \qquad & \forall i = 1,\ 2,\ \ldots\ n \quad , \end{aligned}$$

   a teda $\mathbf{F}(\mathbf{x}) \geq \mathbf{0_n}$.
   Zo vzťahov $\mathbf{F}(\mathbf{x}) \geq \mathbf{0}$ a $\mathbf{x} \geq \mathbf{0}$ vyplýva, že nerovnosť $\mathbf{x}^T \mathbf{F}(\mathbf{x}) \leq 0$ sa musí realizovať ako rovnosť $\mathbf{x}^T \mathbf{F}(\mathbf{x}) = 0$, čo je presne úloha NCP(F).

2. **(1.1) $\Rightarrow$ (1.4)**
   (Z úlohy o komplementarite vyplýva úloha o variačných nerovnostiach)

   Predpokladáme, že $\mathbf{x}^*$ je riešením úlohy (1.1) a teda spĺňa :

$$\mathbf{x}^* \geq \mathbf{0_n}, \quad \mathbf{F}(\mathbf{x}^*) \geq \mathbf{0_n}, \quad (\mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = 0$$

   Vieme, že

$$(\mathbf{y} - \mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = \mathbf{y}^T \mathbf{F}(\mathbf{x}^*) - \mathbf{x}^{*T} \mathbf{F}(\mathbf{x}^*)$$

   Keďže $\mathbf{x}^{*T}\mathbf{F}(\mathbf{x}^*) = 0$ a $\mathbf{F}(\mathbf{x}^*) \geq \mathbf{0}$, tak $\mathbf{x}^*$ je riešením úlohy (1.4)

$$\text{nájsť} \quad \mathbf{x} \geq \mathbf{0} : \quad (\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x}) \geq 0 \quad \forall \mathbf{y} \geq \mathbf{0},$$

□

Poznamenávame však, že množina $\Omega$ musí byť zhora neohraničená. Ak by $\Omega$ bola zhora ohraničená, tak už nemôžeme jednoznačne zaručiť nezápornosť $\mathbf{F}(\mathbf{x})$ a teda úlohu (1.4) nemožno previesť na úlohu (1.1). Ľahko si to uvedomíme, keď vezmeme do úvahy $\mathbf{x} \in \partial\Omega$.

V mnohých aplikáciách sa medzi ohraničeniami vyskytujú aj rovnice a nerovnice a príslušné premenné môžu byť buď voľné, alebo nezáporné. Takáto situácia vedie k zmiešanej úlohe o komplementarite, ktorú možno formulovať nasledovne:

**Zmiešaná úloha o komplementarite:**

$$\begin{array}{lll} \text{Nech} \quad \mathbf{x} \in R^n_+ & \text{( nezáporné premenné )} & \\ \quad \mathbf{y} \in R^m & \text{( voľné premenné )} & \\ \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) \geq \mathbf{0_n} & \text{( ohraničenia v tvare nerovností )} & (1.5) \\ \quad \mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{0_m} & \text{( ohraničenia v tvare rovností )} & \\ \quad \mathbf{x}^T \mathbf{F}(\mathbf{x}, \mathbf{y}) = 0 & \text{( podmienka komplementarity )} & \end{array}$$

Špeciálny prípad zmiešanej úlohy o komplementarite je, keď sú všetky premenné voľné a všetky ohraničenia sú v tvare rovníc. V zmysle zápisu (1.5) dostávame úlohu:

**Úloha NE(G):**

$$\text{NE(G)} : \qquad \mathbf{G}(\mathbf{y}) = \mathbf{0_m}, \qquad \mathbf{y} \in R^m \qquad (1.6)$$

Poznamenávame, že označenie NE je z anglického názvu Nonlinear Equality.

Úloha VI($\mathbf{F}, \Omega \cap X$), kde $X = \{\mathbf{x} : A\mathbf{x} = \mathbf{b}\}$ a $A$ je matica $m \times n$, sa ľahko prevedie na tvar (1.4) zavedením multiplikátorov pre ohraničenia dané lineánymi rovnicami. Úloha VI($\mathbf{F}, \Omega \cap X$)je potom ekvivalentná s úlohou VI($\mathbf{F}, \Omega \times R^m$):

$$\begin{aligned} \text{nájsť} \qquad & \mathbf{x} \in \Omega,\ \lambda \in R^m \quad : \qquad \forall \mathbf{y} \in \Omega,\ \forall \lambda_{\mathbf{y}} \in R^m \\ & \left((\mathbf{y}, \lambda_{\mathbf{y}}) - (\mathbf{x}, \lambda)\right)^T \mathbf{H}(\mathbf{x}, \lambda) \geq 0 \end{aligned} \tag{1.7}$$

$$\mathbf{H}(\mathbf{x}, \lambda) = \begin{bmatrix} \mathbf{F}(\mathbf{x}) + A^T \lambda \\ -A\mathbf{x} + \mathbf{b} \end{bmatrix}. \tag{1.8}$$

Všeobecnejšie, ak $X = \{\mathbf{x} : \mathbf{g}(\mathbf{x}) \leq \mathbf{0_m}, \mathbf{h}(\mathbf{x}) = \mathbf{0_s}\}$, kde $\mathbf{g} : R^n \to R^m$ a $\mathbf{h} : R^n \to R^s$ sú vektorové, spojite diferencovateľné funkcie, tak pri regularite ohraničení pre ľubovoľné riešenie $\mathbf{x}$ úlohy VI($\mathbf{F}, \Omega \cap X$) musia existovať multiplikátory $\lambda \in R^m$ a $\mu \in R^s$ také, že vektory $\mathbf{x}, \lambda, \mu$ sú riešením úlohy VI($\mathbf{H}, \Omega \times R^m \times R^s$), kde

$$\mathbf{H}(\mathbf{x}, \lambda, \mu) = \begin{bmatrix} \mathbf{L}(\mathbf{x}, \lambda, \mu) \\ -\mathbf{g}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix}$$

a

$$\mathbf{L}(\mathbf{x}, \lambda, \mu) \equiv \mathbf{F}(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}) + \sum_{j=1}^{s} \mu_j \nabla h_j(\mathbf{x})$$

je vektorová Lagrangeova funkcia pre VI($\mathbf{F}, \Omega \cap X$).

## 1.3 Vertikálne úlohy o komplementarite.

Ako vidieť z formulácie (1.1b) základnej úlohy o komplementarite, existuje určitá asymetria medzi prvým a druhým argumentom v relácii $\min(\mathbf{x}, \mathbf{F}(\mathbf{x})) = \mathbf{0}$. (Pokiaľ druhý argument je ľubovoľným zobrazením, prvý argument reprezentuje identické zobrazenie.) Za účelom dosiahnutia symetrie prichádzame k nasledovnej, všeobecnejšej formulácii:

$$\min(\mathbf{F^1}(\mathbf{x}), \mathbf{F^2}(\mathbf{x})) = \mathbf{0} \tag{1.9}$$

kde $\mathbf{F^1}, \mathbf{F^2} : R^n \to R^n$ sú dve zobrazenia, čo v tvare (1.1) znamená

$$\mathbf{F^1}(\mathbf{x}) \geq \mathbf{0}, \quad \mathbf{F^2}(\mathbf{x}) \geq \mathbf{0}, \quad \left(\mathbf{F^1}(\mathbf{x})\right)^T \mathbf{F^2}(\mathbf{x}) = 0 \tag{1.9a}$$

Poznamenávame, že niektoré reálne úlohy sa vyskytujú práve v tomto tvare.

Uvedené zovšeobecnenie (1.9) možno indukciou rozšíriť na tvar

**Úloha VCP(F):**

$$\text{VCP}(F): \qquad \min(\mathbf{F^1(x)}, \mathbf{F^2(x)}, \ldots, \mathbf{F^m(x)}) = \mathbf{0} \tag{1.10}$$

Poznamenávame, že označenie VCP je utvorené zo začiatočných písmen anglického názvu Vertical Complementarity Problem.
Úlohu (1.10) môžeme preformulovať na tvar (1.1a):

$$\left.\begin{array}{r} F_i^1(\mathbf{x}) \geq 0,\ F_i^2(\mathbf{x}) \geq 0,\ \ldots, F_i^m(\mathbf{x}) \geq 0 \\ F_i^1(\mathbf{x})\ .\ F_i^2(\mathbf{x})\ .\ \ldots F_i^m(\mathbf{x}) = 0 \end{array}\right\} \quad i = 1,\ 2,\ \ldots\ n \tag{1.11}$$

To znamená, že v sústave m × n nerovníc $F_i^k(\mathbf{x}) \geq 0$ $(k = 1, 2, \ldots m,\ i = 1, 2, \ldots n)$ sa v každej m-tici realizuje aspoň jedna nerovnosť ako rovnica.

Úloha VCP sa dá pomocou dodatočných premenných zapísať ako zmiešaná úloha o komplementarite (1.5). Zavedú sa pomocné vektory $\mathbf{z^j} \in R^n$, $j = 2, 3, \ldots m$ a formulácia tejto úlohy je potom nasledovná:

$$\begin{array}{lcll} \mathbf{F^1(x)} & - \sum_{k=2}^{m} \mathbf{z^k} & = \mathbf{0_n} & \\ \mathbf{F^j(x)} & - \sum_{k=j+1}^{m} \mathbf{z^k} & \geq \mathbf{0_n} & (j = 2,\ 3,\ \ldots\ m-1) \\ \mathbf{F^m(x)} & & \geq \mathbf{0_n} & \\ & & \mathbf{z^k} \geq \mathbf{0_n} & (k = 2,\ 3,\ \ldots\ m) \\ (\mathbf{z^j})^T(\mathbf{F^j(x)} - & \sum_{k=j+1}^{m} \mathbf{z^k}) & = \mathbf{0} & (j = 2,\ 3,\ \ldots\ m-1) \\ (\mathbf{z^m})^T\mathbf{F^m(x)} = 0 & & & \end{array} \tag{1.12}$$

Vidíme, že ak $\mathbf{x}$ rieši VCP$(F)$, potom vyššie uvedené rovnice sú splnené pre

$$\mathbf{z^j} \equiv \min(\mathbf{F^1(x)},\ \mathbf{F^2(x)},\ \ldots\ \mathbf{F^{j-1}(x)}) - \textstyle\sum_{k=j+1}^{m} \mathbf{z^k}, \qquad j = 2,\ 3,\ \ldots\ m-1,$$

$$\mathbf{z^m} \equiv \min(\mathbf{F^1(x)},\ \mathbf{F^2(x)},\ \ldots\ \mathbf{F^{m-1}(x)});$$

a naopak, ak (1.12) platí, potom $\mathbf{x}$ rieši VCP$(F)$.

Poznamenajme, že množstvo premenných v úlohe (1.12) je $m \times n$, zatiaľ čo pôvodná formulácia VCP(F) pomocou minima mala len $n$ premenných.

Tak, ako sme zovšeobecnením úlohy o komplementarite (1.1) dostali úlohu (1.9) [ a neskôr úlohu (1.10) ], možno analogicky zovšeobecniť aj úlohu o variačných nerovnostiach (1.4) na dve a viac zobrazení. Tým dostávame akúsi analógiu "vertikálnej" úlohy o variačných nerovnostiach. (V ďalšom sa obmedzíme len na prípad dvoch zobrazení.)

Majme množiny $\Omega_1 \subset R^n$, $\Omega_2 \subset R^m$, $\Omega = \Omega_1 \times \Omega_2$ a zobrazenia $\mathbf{F^1} : \Omega \to R^n$, $\mathbf{F^2} : \Omega \to R^m$. Zovšeobecnenú úlohu o variačných nerovnostiach pre množinu $\Omega$ a zobrazenia $\mathbf{F^1}, \mathbf{F^2}$ definujeme nasledovne:

**Úloha GVI(F, $\Omega$):**

nájsť $\mathbf{x}^* = (\mathbf{x_1^*}, \mathbf{x_2^*}) \in \Omega$ také, že $\forall \mathbf{y} = (\mathbf{y_1}, \mathbf{y_2}) \in \Omega$ platí

$$\text{GVI}(F, \Omega) \qquad \begin{aligned} (\mathbf{y_1} - \mathbf{x_1^*})^T \mathbf{F^1}(\mathbf{x}^*) &\geq 0 \\ (\mathbf{y_2} - \mathbf{x_2^*})^T \mathbf{F^2}(\mathbf{x}^*) &\geq 0 \end{aligned} \tag{1.13}$$

Túto úlohu sme nazvali GVI (z anglického Generalized Variational Inequality - zovšeobecnené variačné nerovnosti).

## 1.4 Úlohy o komplementarite implikované úlohou o sedlovom bode.

Úloha o sedlovom bode je jednou z kľúčových úloh teórie matematického programovania a ako taká sa tiež dá previesť na úlohu o komplementarite, ako aj na úlohu o variačných nerovnostiach.

Úlohu o sedlovom bode pre množiny $\emptyset \neq X \subset R^n$, $\emptyset \neq Y \subset R^n$ a pre funkciu $f : X \times Y \to R$ možno formulovať nasledovne.

**Úloha SB(f):**

Nájsť bod $(\mathbf{x}^*, \mathbf{y}^*) \in X \times Y$ taký, že platí:

$$\text{SB}(f): \qquad \forall (\mathbf{x}, \mathbf{y}) \in X \times Y \qquad f(\mathbf{x}^*, \mathbf{y}) \leq f(\mathbf{x}^*, \mathbf{y}^*) \leq f(\mathbf{x}, \mathbf{y}^*) \tag{1.14}$$

Bod $(\mathbf{x}^*, \mathbf{y}^*)$ nazývame sedlovým bodom typu minmax funkcie $f$.

Aby sme ukázali, že úlohu SB($f$) možno previesť na úlohu GVI, musíme dodatočne predpokladať, že v úlohe SB($f$) sú množiny $X$, $Y$ konvexné, funkcia $f$ je konvexno - konkávna a spojite diferencovateľná.

Poznamenávame, že funkcia $f : X \times Y \to R$ sa nazýva konvexno - konkávna, ak:

$$\begin{aligned} &\forall\, \overline{\mathbf{y}} \in Y \;\; \varphi(\mathbf{x}) = f(\mathbf{x}, \overline{\mathbf{y}}) \text{ je konvexná na } X, \\ &\forall\, \overline{\mathbf{x}} \in X \;\; \psi(\mathbf{y}) = f(\overline{\mathbf{x}}, \mathbf{y}) \text{ je konkávna na } Y. \end{aligned}$$

**Veta 2.** *Majme úlohu o sedlovom bode (1.14), kde $X$, $Y$ sú konvexné množiny a $f : X \times Y \to R$ je konvexno-konkávna a spojito diferencovateľná funkcia. Potom úloha (1.14) je ekvivalentná s nasledovnou zovšeobecnenou úlohou o variačných nerovnostiach (1.15).*

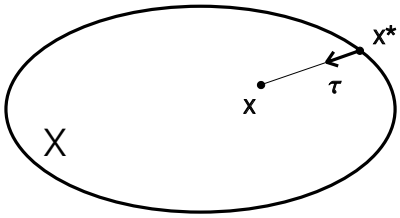
$$\begin{aligned} &\textit{nájsť} \quad (\mathbf{x}^*, \mathbf{y}^*) \in X \times Y \textit{ také, že} \quad \forall\, (\mathbf{x}, \mathbf{y}) \in X \times Y \textit{ platí} \\ &\qquad\qquad (\mathbf{x} - \mathbf{x}^*)^T \nabla_x f(\mathbf{x}^*, \mathbf{y}^*) \geq 0 \\ &\qquad\qquad -(\mathbf{y} - \mathbf{y}^*)^T \nabla_y f(\mathbf{x}^*, \mathbf{y}^*) \geq 0 \end{aligned} \tag{1.15}$$

*Dôkaz.*

1. **(1.14) $\Rightarrow$ (1.15)**
   (t.j. ak $(\mathbf{x}^*, \mathbf{y}^*)$ je sedlovým bodom, potom je riešením úlohy (1.15))
   Definujeme si funkciu $\varphi : X \to R$, $\varphi(\mathbf{x}) = f(\mathbf{x}, \mathbf{y}^*)$, t.j. $\nabla\varphi(\mathbf{x}) = \nabla_x f(\mathbf{x}, \mathbf{y}^*)$. Môžu nastať dva prípady : buď je $\mathbf{x}^*$ vnútorným bodom množiny $X$, alebo je $\mathbf{x}^*$ hraničným bodom množiny $X$.

   (a) Nech $\mathbf{x}^*$ je vnútorným bodom množiny $X$. Keďže $(\mathbf{x}^*, \mathbf{y}^*)$ je sedlovým bodom funkcie $f$, zrejme $\mathbf{x}^*$ je bodom minima funkcie $\varphi$. A keďže je tento bod minima $\mathbf{x}^*$ vnútorným bodom, musí platiť $\nabla\varphi(\mathbf{x}^*) = \mathbf{0_n}$. Z toho vyplýva, že bod $\mathbf{x}^*$ spĺňa prvú variačnú nerovnicu v (1.15).

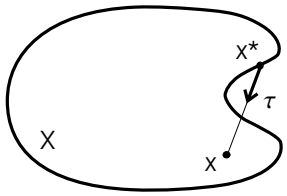
   (b) Nech $\mathbf{x}^*$ je hraničným bodom konvexnej množiny $X$. Funkcia $\varphi(\mathbf{x})$ nadobúda v tomto bode minimum na množine $X$.




Obr.1.1

   Vezmeme si ľubovoľný vnútorný bod množiny $X$ (Obr.1.1). Definujeme si funkciu $\Psi(\tau) = \varphi(\mathbf{x}^* + \tau(\mathbf{x} - \mathbf{x}^*))$. Keďže $\mathbf{x}^*$ je bodom minima, funkcia $\Psi(\tau)$ nesmie v žiadnom smere do vnútra množiny $X$ klesať. Z toho vyplýva, že $0 \leq \Psi'(0) = \varphi(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*)$, t.j. platí prvá variačná nerovnosť v (1.15)

Pre $\mathbf{y}$ analogicky, no namiesto minima tu bude maximum, čo v konečnom dôsledku obráti znamienko na opačné. (Pozn.: Bez predpokladu konvexnosti množín $X$, $Y$ príslušná implikácia neplatí, ako ukazuje obrázok Obr.1.2)




Obr.1.2

2. **(1.15) ⇒ (1.14)**
   (t.j. ak bod $(\mathbf{x}^*, \mathbf{y}^*)$ je riešením úlohy (1.15), potom je sedlovým bodom.)

   (a) Z konvexnosti $f$ v premennej $\mathbf{x}$ máme :

   $$f(\mathbf{x}, \mathbf{y}^*) - f(\mathbf{x}^*, \mathbf{y}^*) \geq (\mathbf{x} - \mathbf{x}^*)^T \nabla_x f(\mathbf{x}^*, \mathbf{y}^*)$$

   Z (1.15) vieme, že

   $$(\mathbf{x} - \mathbf{x}^*)^T \nabla_x f(\mathbf{x}^*, \mathbf{y}^*) \geq 0,$$

   a teda $f(\mathbf{x}, \mathbf{y}^*) - f(\mathbf{x}^*, \mathbf{y}^*) \geq 0$, čo dokazuje druhú nerovnosť v (1.14).

   (b) Z konkávnosti $f$ v premennej $\mathbf{y}$ máme :

   $$f(\mathbf{x}^*, \mathbf{y}) - f(\mathbf{x}^*, \mathbf{y}^*) \leq (\mathbf{y} - \mathbf{y}^*)^T \nabla_y f(\mathbf{x}^*, \mathbf{y}^*)$$

   a z (1.15) vieme, že

   $$(\mathbf{y} - \mathbf{y}^*)^T \nabla_y f(\mathbf{x}^*, \mathbf{y}^*) \leq 0,$$

   a teda $f(\mathbf{x}^*, \mathbf{y}) - f(\mathbf{x}^*, \mathbf{y}^*) \leq 0$ , čo dokazuje prvú nerovnosť v (1.14).

   Keď výsledky z (a) a (b) spojíme, dostaneme:

   $$f(\mathbf{x}^*, \mathbf{y}) \leq f(\mathbf{x}^*, \mathbf{y}^*) \leq f(\mathbf{x}, \mathbf{y}^*),$$

   t.j. $(\mathbf{x}^*, \mathbf{y}^*)$ je sedlovým bodom funkcie $f$.

□

V prípade, keď $X = R_+^n$ a $Y = R_+^m$ (pričom funkcia $f$ nemusí byť konvexno - konkávna), úloha (1.15) je podľa Vety 1 ekvivalentná s nasledujúcou úlohou o komplementarite:

$$\mathbf{x} \geq \mathbf{0_n} \qquad \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \geq \mathbf{0_n} \qquad \mathbf{x}^T \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) = 0 \qquad (1.16a)$$

$$\mathbf{y} \geq \mathbf{0_m} \qquad -\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \geq \mathbf{0_m} \qquad \mathbf{y}^T \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = 0 \qquad (1.16b)$$

Podľa zápisu (1.3a) možno vidieť, že (1.16a) sú nutné podmienky optimality pre úlohu $\min\{\ f(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \geq 0\ \}$ a (1.16b) sú nutné podmienky optimality pre úlohu $\max\{\ f(\mathbf{x},\mathbf{y}) \mid \mathbf{y} \geq 0\ \}$. Tieto dve úlohy spolu dávajú presne úlohu o sedlovom bode. (Ak by príslušná funkcia $f$ bola konvexno - konkávna, uvedené nutné podmienky sú aj postačujúcimi podmienkami optimality.)

Uvedené vzájomné vzťahy ilustrujeme na najjednoduchšom prípade kvadratickej funkcie:

$$f(\mathbf{x},\mathbf{y}) = \mathbf{p}^T\mathbf{x} + \frac{1}{2}\,\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{B}\mathbf{y} + \mathbf{q}^T\mathbf{y} - \frac{1}{2}\,\mathbf{y}^T\mathbf{C}\mathbf{y}, \tag{1.17}$$

kde $\mathbf{A}$, $\mathbf{C}$ sú kladne semidefinitné matice, $\mathbf{B}$ je obdĺžniková $m \times n$ matica a $\mathbf{p}$, $\mathbf{q}$ sú vektory príslušných dimenzií.

Potom príslušná úloha o sedlovom bode vedie k nasledovnej "afínnej" úlohe GVI($\mathbf{F}, K$) v ktorej $K = R^n_+ \times R^m_+$ a

$$\mathbf{F}(\mathbf{x},\mathbf{y}) = \begin{cases} \nabla_{\mathbf{x}} f(\mathbf{x},\mathbf{y}) &= \mathbf{p} + \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{y} \\ -\nabla_{\mathbf{y}} f(\mathbf{x},\mathbf{y}) &= -\mathbf{q} + \mathbf{B}\mathbf{x} - \mathbf{C}\mathbf{y} \end{cases}$$

Ináč povedané, ide o nasledovnú úlohu:

nájsť $(\mathbf{x}^*,\mathbf{y}^*) \in K$ také, že $\forall(\mathbf{x},\mathbf{y}) \in K$ platí

$$\begin{aligned} (\mathbf{x}-\mathbf{x}^*)^T(\mathbf{p}^T + \mathbf{A}\mathbf{x}^* - \mathbf{B}\mathbf{y}^*) \geq 0 \\ (\mathbf{y}-\mathbf{y}^*)^T(-\mathbf{q} + \mathbf{B}\mathbf{x}^* - \mathbf{C}\mathbf{y}^*) \geq 0 \end{aligned} \tag{1.18}$$

Príslušná (ekvivalentná) úloha o komplementarite vyzerá takto:

$$\mathbf{x} \geq \mathbf{0_n} \qquad \mathbf{p}^T + \mathbf{x}^T\mathbf{A} - \mathbf{B}\mathbf{y} \geq \mathbf{0_n} \qquad \mathbf{x}^T(\mathbf{p} + \mathbf{x}^T\mathbf{A} - \mathbf{B}\mathbf{y}) = 0 \tag{1.19a}$$

$$\mathbf{y} \geq \mathbf{0_m} \qquad -\mathbf{q}^T + \mathbf{x}^T\mathbf{B} - \mathbf{y}^T\mathbf{C} \geq \mathbf{0_m} \qquad \mathbf{y}^T(-\mathbf{q} + \mathbf{x}^T\mathbf{B} - \mathbf{y}^T\mathbf{C}) = 0 \tag{1.19b}$$

## 1.5 Úloha o komplementarite v úlohe nelineárneho programovania s ohraničením v tvare variačnej nerovnosti.

V úlohe nelineárneho programovania sa niekedy môže vyskytnúť ohraničenie v tvare "parametrizovanej" variačnej nerovnosti. Preto najskôr zovšeobecníme úlohu VI($\mathbf{F}, \Omega$) na parametrizovanú úlohu PVI($\mathbf{F_t}, \Omega_\mathbf{t}$), kde $\mathbf{t} \in T \subseteq R^m$ je parametrom. Teda namiesto zafixovanej množiny $\Omega \subset R^n$ uvažujeme celú triedu množín $\Omega_\mathbf{t} \subset R^n$ a namiesto zafixovaného

zobrazenia $\mathbf{F} : \Omega \to R^n$ budeme uvažovať parametrizované zobrazenie $\mathbf{F_t} : \Omega_\mathbf{t} \to R^n$, ktoré v ďalšom označíme ako $\mathbf{F_t}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, \mathbf{t})$. Potom "parametrizovanú" úlohu možno formulovať nasledovne:

**Úloha PVI($\mathbf{F_t}, \mathbf{\Omega_t}$):**

Pre každé $\mathbf{t} \in T$

$$\text{(II)} \qquad \text{nájsť } \mathbf{x}^*(\mathbf{t}) \in \Omega_\mathbf{t} \ : \qquad \forall\, \mathbf{y} \in \Omega_\mathbf{t} \quad (\mathbf{y} - \mathbf{x}^*(\mathbf{t}))^T \mathbf{F}(\mathbf{x}^*(\mathbf{t}), \mathbf{t}) \geq 0, \tag{1.20}$$

Optimalizačné úlohy, v ktorých vystupujú parametrizované variačné nerovnosti ako ohraničenia, sa nazývajú úlohy matematického programovania s rovnovážnymi ohraničeniami (ďalej len MPEC, z anglického názvu Mathematic Programming with Equilibrium Constraints.)

Úlohy MPEC pozostávajú z dvoch "úrovní" premenných: "I úroveň", premenná $\mathbf{t} \in T \subseteq R^n$ a "II úroveň", premenná $\mathbf{x} \in R^m$. Úloha MPEC má nasledujúci tvar:

**Úloha MPEC:**

(I) $\min\limits_{\mathbf{x},\mathbf{t}} \{\ \Phi(\mathbf{x}, \mathbf{t}) \mid \mathbf{t} \in T,\ \mathbf{x} \in \Omega_\mathbf{t},\ \mathbf{x} \text{ rieši úlohu (II) } \}$

(II) nájsť $\mathbf{x}(\mathbf{t}) \in \Omega_\mathbf{t}$ : $\forall\, \mathbf{y} \in \Omega_\mathbf{t}\ (\mathbf{y} - \mathbf{x}^*(\mathbf{t}))^T \mathbf{F}(\mathbf{x}^*(\mathbf{t}), \mathbf{t}) \geq 0$ (pre každé $\mathbf{t} \in T$)

Kde $\Phi(\mathbf{x}, \mathbf{t}) : R^{n+m} \to R$ je prvoúrovňová účelová funkcia, $\mathbf{F}(\mathbf{x}, \mathbf{t}) : R^{n+m} \to R^n$ je druhoúrovňová "rovnovážna" funkcia a pre každé $\mathbf{t} \in T \subseteq R^m$ je $\Omega_\mathbf{t}$ množina ohraničení druhej úrovne ( môže byť aj prázdna. )

V špeciálnom prípade, ak $\Omega_\mathbf{t} = R^n_+,\ \forall \mathbf{t} \in T$, t.j. ak sa parametrizácia vzťahuje len na zobrazenie $\mathbf{F}$, úloha (II) prejde na tvar (IIa):

$$\text{(IIa)} \qquad \text{nájsť } \mathbf{x} \in R^n_+ \ : \forall\, \mathbf{y} \in R^n_+ \quad (\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x}, \mathbf{t}) \geq 0,$$

Úloha (IIa) je potom ekvivalentná parametrizovanej úlohe o komplementarite:

$$\text{(IIb)} \qquad \mathbf{x} \geq \mathbf{0} \qquad \mathbf{F}(\mathbf{x}, \mathbf{t}) \geq \mathbf{0} \qquad \mathbf{x}^T \mathbf{F}(\mathbf{x}, \mathbf{t}) = 0$$

Úloha MPEC v tomto prípade bude mať tvar:

(I) $\min\{\ \Phi(\mathbf{x}, \mathbf{t}) \mid \mathbf{x} \geq 0,\ \mathbf{x} \text{ rieši úlohu (IIb) } \}$,

čo je vlastne úloha

$$\min_{\mathbf{x},\mathbf{t}} \{\ \Phi(\mathbf{x}, \mathbf{t}) \mid \mathbf{x} \geq 0,\ \mathbf{F}(\mathbf{x}, \mathbf{t}) \geq \mathbf{0}, \quad \mathbf{x}^T \mathbf{F}(\mathbf{x}, \mathbf{t}) = 0 \ \}$$

Úlohu (IIb) možno pridaním vektora $\mathbf{u}$ previesť na zmiešanú úlohu o komplementarite:

$$\mathbf{x} \geq \mathbf{0_n}$$
$$\mathbf{u} \geq \mathbf{0_n}$$
$$\mathbf{F}(\mathbf{x}, \mathbf{t}) - \mathbf{u} = \mathbf{0_n}$$
$$\mathbf{x}^T \mathbf{u} = 0$$

kde parameter $\mathbf{t}$ vystupuje už len v ohraničení v tvare rovnosti.

# Kapitola 2

# Model rovnováhy na trhu s komoditami.

Ekonomiku budeme modelovať nasledovne. Uvažujeme, že obsahuje $l$ komodít s cenovým vektorom $\mathbf{p} \in R^l_{++}$, ktorý bude premennou v našom modeli. Ekonomika sa delí na dva sektory: sektor výroby a sektor spotreby.

Sektor výroby obsahuje $n$ výrobcov, pričom $j$-ty výrobca je charakterizovaný produkčnou množinou $Y_j \subset R^l$. Každý výrobca maximalizuje svoj zisk $\pi_j(\mathbf{p}) = \mathbf{p}^T\mathbf{y}$, vhodnou voľbou svojho výrobného vektora $\mathbf{y^j} \in Y_j$.

Sektor spotreby obsahuje $m$ spotrebiteľov. Na začiatku $i$-ty spotrebiteľ disponuje komoditami $\mathbf{e^i} \in R^l_+$, ktoré majú celkovú hodnotu $w_i = \mathbf{p}^T\mathbf{e^i}$. Spotrebiteľ voľbou spotreby $\mathbf{d^i} \in D_i$ maximalizuje svoj úžitok podľa funkcie užitočnosti $u_i(\mathbf{d^i})$ vzhľadom na svoj rozpočet $w_i$.

Úlohou trhu je teda stanoviť optimálny vektor cien $\mathbf{p} \in R^l_{++}$ a množstvá $\mathbf{y^i} \in Y_i$, ktoré budú jednotliví výrobcovia vyrábať.

## 2.1 Základné pojmy a predpoklady.

Na začiatku uvedieme všetky pojmy a označenia, ktoré budeme ďalej používať. Dolné indexy budeme používať na indexovanie prvkov vektorov, horné na indexovanie vektorov spojených s výrobcami, alebo spotrebiteľmi.

| | |
|---|---|
| $l$ | množstvo komodít na trhu (pričom každá komodita môže byť vstupom aj výstupom) |
| $m$ | množstvo spotrebiteľov |

| | |
|---|---|
| $n$ | množstvo výrobcov |
| $\mathbf{p} \in R^l_{++}$ | vektor cien (označenie pochádza z anglického price) |

Sektor výroby

| | |
|---|---|
| $Y_j \subset R^l$ | produkčná možina výrobcu $j$ ($j = 1,\ 2,\ \ldots\ n$), udáva funkcionálnu závislosť medzi vstupmi a výstupmi pri jednotkovej úrovni výroby. V našom modeli bude produkčná množina reprezentovať nejakú polpriamku v $R^l$, t.j $Y_j = \{\ \mathbf{y} \in R^l \mid \mathbf{y} = \zeta \mathbf{a^j}, \quad \zeta \geq 0\ \}$ |
| $A \in R^{l \times n}$ | matica technológií, $A = [\ \mathbf{a^1},\ \mathbf{a^2},\ \ldots\ \mathbf{a^n}\ ]$ popisuje výrobný proces |
| $z_j \in R_+$ | úroveň produkcie (intenzita) $j$-teho výrobcu ($\mathbf{y^j} = z_j \mathbf{a^j}$) |
| $\mathbf{y^j} \in Y_j$ | výrobný vektor $j$-teho výrobcu, popisuje koľko ktorej komodity výrobca $j$ vyrába, resp. spotrebúva (ak je jeho niektorá zložka záporná, výrobca túto komoditu len spotrebúva). Vektor $\mathbf{y^i}$ možno vyjadriť ako rozdiel výstupu $\mathbf{q^i} \geq 0$ a vstupu $\mathbf{x^i} \geq 0$. |
| $\pi_j(\mathbf{p})$ | zisk $j$-teho výrobcu ($\pi$ z anglického profit) |
| $f_j(\mathbf{x})$ | produkčná funkcia určujúca intenzitu $z_j = f_j(\mathbf{x})$ produkcie pri daných vstupoch $\mathbf{x}$ |
| $g_j(\mathbf{q})$ | produkčná funkcia určujúca intenzitu $z_j = g_j(\mathbf{q})$ produkcie pri daných výstupoch $\mathbf{q}$ |
| $t_k$ | daňová sadzba uvalená na vstup $x_k$ ($k = 1, 2, \ldots l$) (z anglického tax) |
| $\tau_k$ | daňová sadzba uvalená na výstup $q_k$ |
| $\mathbf{T} \in R^n$ | kumulované dane pre každého výrobcu $j$ pri jednotkovej úrovni výroby, $\left( T_j = \sum_{k=1}^{l} (q_k^j \tau_k + x_k^j t_k) p_k \right)$ |

Sektor spotreby

| | |
|---|---|
| $C_i \subset R^l_+$ | spotrebná množina $i$-teho spotrebiteľa ($i = 1, 2, \ldots m$), modeluje ohraničenia pre minimálnu spotrebu spotrebiteľa |
| $\mathbf{c^i} \in C_i$ | spotrebný vektor $i$-teho spotrebiteľa (z anglického consumption) |
| $u_i(\mathbf{c^i})$ | úžitková funkcia, popisuje preferencie $i$-teho spotrebiteľa (z anglického utility function) |
| $\mathbf{e^i} \in R^l_+$ | počiatočné rozdelenie komodít pre $i$-teho spotrebiteľa (z anglického endowment) |
| $w_i \in R_+$ | rozpočet $i$-teho spotrebiteľa, $w_i = \mathbf{p}^T \mathbf{e^i}$ (z anglického weatlh) |

CES funkcie funkcie s konštantnou elasticitou substitúcie
(skratka z anglického názvu Constant Elasticity of Substitution)

Sformulujeme ešte základné predpoklady tohoto modelu. Prvým je predpoklad dokonalej súťaže. Spotrebitelia nemajú uprednostňovaného výrobcu, nemôžu ovplyvňovať ceny (zjednávať sa) a akceptujú ponúkanú cenu. Každý spotrebiteľ aj výrobca má dokonalú informáciu o cenách na trhu.

O produkčnej množine $Y_i$ predpokladáme, že je uzavretá polpriamka a obsahuje nulový vektor, takže je možná aj nulová produkcia. O produkčných funkciách budeme predpokladať, že majú konštantné výnosy z rozsahu. To znamená, že sú pozitívne homogénne 1. stupňa: $f_j(\mu\mathbf{x}) = \mu f_j(\mathbf{x})$, $g_j(\mu\mathbf{q}) = \mu g_j(\mathbf{q})$ pre všetky $\mu > 0$. Ak pre $\mathbf{x}$ platí $f_j(\mathbf{x}) = 1$ hovoríme, že $\mathbf{x}$ sú jednotkové vstupy a ak pre $\mathbf{q}$ platí $g_j(\mathbf{q}) = 1$ hovoríme, že $\mathbf{q}$ sú jednotkové výstupy z produkčného procesu $j$ teho výrobcu.

Funkcia užitočnosti $u_i$ popisuje preferencie $i$-teho spotrebiteľa a má vlastnosť trichotómie, t.j. že pre dva spotrebné vektory komodít $\mathbf{a}$, $\mathbf{b}$ platí jedna z troch vlastností: $u^i(\mathbf{a}) > u^i(\mathbf{b})$, $u^i(\mathbf{a}) < u^i(\mathbf{b})$, alebo $u^i(\mathbf{a}) = u^i(\mathbf{b})$. (Spotrebiteľ uprednostní $\mathbf{a}$ pred $\mathbf{b}$, $\mathbf{b}$ pred $\mathbf{a}$, alebo neuprednostní ani jeden z nich.) Funkcia užitočnosti je spojitá a vo všeobecnosti sa predpokladá, že je to CES funkcia. Ďalej predpokladáme nepresýtenie komoditami. To znamená, že keď spotrebiteľ $i$ má komodity $\mathbf{a^i} \in D_i$, tak existuje iný balík komodít $\mathbf{b^i} \in D_i$, ktorý bude spotrebiteľ $i$ preferovať pred $\mathbf{a^i}$ ($u^i(\mathbf{b^i}) > u^i(\mathbf{a^i})$).

Napokon predpokladáme, že oba sektory budú vyvíjať svoje aktivity tak, aby boli splnené aj nasledovné vzťahy:

$$\text{žiadny výrobca neprodukuje kladný zisk} \tag{2.1}$$

$$\text{ponuka prevyšuje dopyt} \tag{2.2}$$

$$\text{výdaje spotrebiteľov sa rovnajú príjmu výrobcov} \tag{2.3}$$

Všetky tieto predpoklady dávajú intuitívny zmysel. Vzťah (2.1) hovorí o tom, že keby mohol nejaký výrobca dosiahnuť kladný zisk, tak by zdvojnásobením svojej aktivity mohol dosiahnuť dvakrát taký zisk, čím by sa mohol jeho zisk stať neohraničeným. Toto je spôsobené predpokladom o konštantných výnosoch z rozsahu produkčnej funkcie ($f(2\mathbf{x}) = 2f(\mathbf{x})$). Preto budeme predpokladať, že zisk nie je kladný. Vzťah (2.2) vychádza zo snahy výrobcu maximalizovať zisk. Ak je po nejakej komodite väčší dopyt než ponuka, potom bude pre nejakú firmu výhodné vyrábať viac tejto komodity, čím stúpne ponuka.

Toto sa bude diať dovtedy, kým sa dopyt nenasýti. Posledná podmienka hovorí, že celkové výdaje spotrebiteľov na komodity sú rovné príjmu z predaja týchto komodít.

Neskôr ukážeme, že všetky uvedené predpoklady umožňujú formuláciu ekonomického modelu rovnováhy v tvare zmiešanej úlohy o komplementarite. Príslušný model uvedieme najprv v jednoduchšom tvare a potom ho mierne zovšeobecníme. Na záver tento model uvedieme aj s prihliadnutím na dane a dotácie. Najprv však podrobnejšie popíšeme sektor výroby a spotreby.

## 2.2 Sektor výroby.

### 2.2.1 Matica technológií a produkčné funkcie.

V tejto časti popíšeme, čo sa deje v sektore výroby. Ako sme už spomenuli, sektor výroby obsahuje $n$ výrobcov a títo sú popísaní svojimi výrobnými vektormi $\mathbf{y^j} \in R^l$. $K$-ta zložka, $y_k^j$, výrobného vektora je kladná, ak $j$-ty výrobca produkuje $k$-tu komoditu vo väčšom množstve než ju spotrebúva. Naopak, $y_k^j \leq 0$, ak sa $k$-ta komodita spotrebúva vo väčšom množstve, než ju výrobca vyrába. Vektor $\mathbf{y^j}$ sa tiež nazýva ako vektor čistého výstupu (netput vektor), pretože popisuje čistý výstup $j$-teho výrobcu. Dá sa určiť ako rozdiel vektorov výstupu ($\mathbf{q^j}$) a vstupu ($\mathbf{x^j}$) komodít vo výrobnom procese daného výrobcu, t.j. $\mathbf{y^j} = \mathbf{q^j} - \mathbf{x^j}$. Každý výrobca je ohraničený svojou výrobnou technológiou. V našom modeli budeme na popis výrobného procesu používať maticu technológií

$$A = [\ \mathbf{a^1} \quad \mathbf{a^2} \quad \ldots \quad \mathbf{a^n}\ ] \ \in \ R^{l \times n},$$

$J$-ty stĺpec $\mathbf{a^j}$ matice technológií popisuje vzťah medzi jednotkovými vstupmi $\mathbf{x^j}$ ($f_j(\mathbf{x^j}) = 1$) a jednotkovými výstupmi $\mathbf{q^j}$ ($g_j(\mathbf{q^j}) = 1$) vo výrobnom procese $j$-teho výrobcu

$$\mathbf{a^j} = \mathbf{q^j} - \mathbf{x^j}$$

Poznamenávame, že $\mathbf{a^j}$ je vlastne vektor čistého výstupu $\mathbf{y^j}$ pri jednotkovej úrovni výroby. Produkčnú množinu $j$-teho výrobcu budeme teda popisovať ako závislosť medzi jednotkovými vstupmi a výstupmi $\mathbf{a^j}$ a úrovňou produkcie $z_j > 0$ :

$$\mathbf{y^j} = \mathbf{a^j} z_j \qquad j \ = \ 1,\ 2,\ \ldots n. \tag{2.4}$$

Teraz popíšeme, ako sa dá výrobná technológia určiť pomocou produkčných funkcií. Funkcia $f_j(\mathbf{x^j})$ určuje intenzitu výroby pri danom vstupe $\mathbf{x^j}$ a funkcia $g_j(\mathbf{q^j})$ určuje intenzitu výroby pri danom výstupe $\mathbf{q^j}$.

Najrozšírenejšou triedou funkcií v ekonomickej literatúre sú takzvané CES funkcie. Všeobecný tvar týchto funkcií je

$$\left(\sum_{k=1}^{l} \lambda_k x_k^{\rho}\right)^{1/\rho}, \qquad \lambda_k \geq 0, \quad k = 1,\ 2,\ \ldots, l \tag{2.5}$$

Definičný obor CES funkcií je $R_+^l$ a hodnoty v hraničných bodoch sa dodefinúvajú zo spojitosti. Dá sa ukázať, že CES funkcie sú konvexné pre $\rho \geq 1$ a konkávne pre $\rho \leq 1$.

Uvedieme niektoré špeciálne prípady CES funkcií, ktoré sa bežne používajú.

1. Pre $\rho = 1$ je to lineárna produkčná funkcia :

$$\sum_{k=1}^{l} \lambda_k x_k.$$

2. Ak pre $\lambda_k$ platí $\sum_{k=1}^{l} \lambda_k > 0$ ,potom pre $\rho \to 0$ môžeme vypočítať limitu pomocou l´Hospitalovho pravidla :

$$\prod_{k=1}^{l} x_k^{\mu_k},$$

kde $\mu_k \equiv \lambda_k / \sum_{j=1}^{l} \lambda_j$. Tento tvar sa nazýva Cobb-Douglasova produkčná funkcia.

3. Pre $\rho \to -\infty$ potom máme funkciu

$$\min_{1 \leq k \leq l} \lambda_k x_k.$$

Ide o Leontiefovu produkčnú funkciu, ktorá je konkávna.

### 2.2.2 Optimalizácia zisku výrobcu.

Na konkurenčnom trhu má $j$-ty výrobca pri cenách $\mathbf{p}$ zisk $\pi_j(\mathbf{p})$. Tento zisk je určený riešením nesledovnej optimalizačnej úlohy

$$\pi_j(\mathbf{p}) = \max\{ \quad \mathbf{p}^T \mathbf{y^j} \mid \mathbf{y^j} \in Y_j \quad \} \ \leq \ 0 \qquad j \ = \ 1,\ 2,\ \ldots n \tag{2.6}$$

Keď namiesto všeobecnej produkčnej množiny $Y_j$ dosadíme produkčnú množinu popísanú pomocou matice technológií (2.4), dostávame úlohu

$$\pi_j(\mathbf{p}) = \max\{ \quad \mathbf{p}^T \mathbf{a^j} z_j \mid z_j \geq 0 \quad \} \ \leq \ 0 \qquad j \ = \ 1,\ 2,\ \ldots n.$$

Nutné a postačujúce podmienky optimality pre celý sektor výroby sú potom nasledujúce

$$\mathbf{z} \geq \mathbf{0_n}, \quad -A^T\mathbf{p} \geq \mathbf{0_n}, \quad \mathbf{z}^T A^T \mathbf{p} = 0. \tag{2.7}$$

Teraz budeme predpokladať, že výrobca operuje na úrovni produkcie $z_j$. Vektor čistého výstupu $\mathbf{y^j}$ pri úrovni výroby $z_j$ rozdelíme na vektor vstupov $\overline{\mathbf{x}}^\mathbf{j}$ a vektor výstupov $\overline{\mathbf{q}}^\mathbf{j}$ ($\mathbf{y^j} = \overline{\mathbf{q}}^\mathbf{j} - \overline{\mathbf{x}}^\mathbf{j}$). Z úlohy (2.6) a vzťahu (2.4) pre produkčnú množinu $Y_j$ dostávame úlohu

$$\max_{\overline{\mathbf{x}}^\mathbf{j}, \overline{\mathbf{q}}^\mathbf{j}} \{ \quad \mathbf{p}^T(\overline{\mathbf{q}}^\mathbf{j} - \overline{\mathbf{x}}^\mathbf{j}) \mid f_j(\overline{\mathbf{x}}^\mathbf{j}) = z_j, \quad g_j(\overline{\mathbf{q}}^\mathbf{j}) = z_j, \quad \overline{\mathbf{x}}^\mathbf{j} \geq \mathbf{0_l}, \ \overline{\mathbf{q}}^\mathbf{j} \geq \mathbf{0_l} \quad \}. \tag{2.8}$$

Kde $f_j : R^l \to R$ je konkávna CES funkcia a $g_j : R^l \to R$ je konvexná CES funkcia (vo väčšine praktických modelov sa používa niektorý z vyššie uvedených tvarov CES funkcií.) Keďže $f_j$ a $g_j$ sú pozitívne homogénne prvého stupňa, platí: $\exists \mathbf{x^j} : f_j(\mathbf{x^j}) = 1$, kde $\mathbf{x^j}$ sú jednotkové vstupy do výrobného procesu ($\mathbf{x^j} z_j = \overline{\mathbf{x}}^\mathbf{j}$). Podobne $\exists \mathbf{q^j} : g_j(\mathbf{q^j}) = 1$, kde $\mathbf{q^j}$ sú jednotkové výstupy z výrobného procesu ($\mathbf{q^j} z_j = \overline{\mathbf{q}}^\mathbf{j}$).

Kvôli tejto vlastnosti je úloha (2.8) ekvivalentná s úlohou (2.9)

$$\pi_j(\mathbf{p}) = z_j \ \max_{\mathbf{x^j}, \mathbf{q^j}} \{ \ \mathbf{p}^T(\mathbf{q^j} - \mathbf{x^j}) \mid f_j(\mathbf{x^j}) = 1, \ g_j(\mathbf{q^j}) = 1, \ \mathbf{x^j} \geq \mathbf{0_l}, \ \mathbf{q^j} \geq \mathbf{0_l} \ \}. \tag{2.9}$$

Zisk $j$-teho výrobcu je daný ako

$$\pi_j(\mathbf{p}) = \mathbf{p}^T \mathbf{a^j} z_j,$$

kde

$$\mathbf{a^j} = \mathbf{q^j} - \mathbf{x^j}, \tag{2.10}$$

Vektory $\mathbf{x^j}, \mathbf{q^j}$ riešia optimalizačnú úlohu

$$\max_{\mathbf{x^j}, \mathbf{q^j}} \{ \quad \mathbf{p}^T(\mathbf{q^j} - \mathbf{x^j}) \mid f_j(\mathbf{x^j}) = 1, \ g_j(\mathbf{q^j}) = 1, \quad \mathbf{x^j} \geq \mathbf{0}, \ \mathbf{q^j} \geq \mathbf{0} \quad \}. \tag{2.11}$$

Túto úlohu možno rozdeliť na dve samostatné úlohy

$$\min_{\mathbf{x^j}} \{ \ \mathbf{p}^T \mathbf{x^j} \mid f_j(\mathbf{x^j}) = 1, \ \mathbf{x^j} \geq \mathbf{0} \ \} \tag{2.11a}$$

$$\max_{\mathbf{q^j}} \{ \ \mathbf{p}^T \mathbf{q^j} \mid g_j(\mathbf{q^j}) = 1, \ \mathbf{q^j} \geq \mathbf{0} \ \} \tag{2.11b}$$

Poznamenávame, že úloha (2.11a) popisuje minimalizáciu výrobných nákladov $j$-teho výrobcu a úloha (2.11b) zas maximalizáciu príjmu z predaja vyrobených komodít. Vo všeobecnosti sú riešenia oboch úloh dané analyticky výrazmi $\mathbf{x^j}(\mathbf{p})$ a $\mathbf{q^j}(\mathbf{p})$. Preto môžeme stĺpce matice technológií $\mathbf{a^j}(\mathbf{p})$ uvažovať ako funkcie vektora cien $\mathbf{p}$ ($\mathbf{a^j}(\mathbf{p}) = \mathbf{q^j}(\mathbf{p}) - \mathbf{x^j}(\mathbf{p})$).

Pre ilustráciu uvádzame konkrétny tvar fukncie $\mathbf{x^j}(\mathbf{p})$, keď $f_j$ je Cobb-Douglasova produkčná funkcia ($f_j(\mathbf{x}) = \prod_{k=1}^{l} x_k^j \lambda_k^j$). Jednotkový vstup $k$-tej komodity do výrobného procesu $j$-teho výrobcu má v tomto prípade nasledovný tvar:

$$x_k^j(\mathbf{p}) = \frac{\lambda_k \prod_{i=1}^{l} (p_i/\lambda_i)^{\lambda_i}}{p_k}$$

Na koniec zhrnieme všetky výsledky týkajúce sa sektora výroby. Zisk $j$-teho výrobcu je daný ako $\pi_j(\mathbf{p}) = \mathbf{p}^T \mathbf{a^j}(\mathbf{p}) z_j$ a úroveň produkcie $z_j$ je daná z riešenia úlohy o komplementarite (2.7). Celková ponuka komodít na trhu potom bude $A(\mathbf{p})\mathbf{z}$. Ostáva určiť dopyt spotrebiteľov, ktorý je predmetom ďalšej časti.

## 2.3 Sektor spotreby.

Sektor spotreby má $m$ spotrebiteľov a spotrebný vektor $\mathbf{c^i} \in D_i$ $i$-teho spotrebiteľa udáva množstvo komodít, ktoré spotrebuje (napr. $c_k^i$ udáva množstvo komodity $k$, ktoré spotrebuje spotrebiteľ $i$). Minimálna spotreba $i$-teho spotrebiteľa je modelovaná spotrebnou množinou $C_i \subset R_+^l$.

Kľúčovou pre spotrebiteľov je myšlienka preferencií. Preferencie sú kompletné usporiadanie na množine komodít. Znamená to, že spotrebiteľ môže uprednostniť spotrebný balík komodít $\mathbf{a}$ pred spotrebným balíkom $\mathbf{b}$, môže uprednostniť $\mathbf{b}$ pred $\mathbf{a}$, alebo preň môžu byť indiferentné, t.j. neuprednostní ani jeden spotrebný balík, má z nich rovnaký úžitok. Preferencie spotrebiteľa $i$ sú popísané funkciou užitočnosti $u^i$.

Na začiatku je $i$-ty spotrebiteľ je vybavený bohatstvom $w_i$ ($w_i = \mathbf{p}^T \mathbf{e^i}$, kde $\mathbf{e^i}$ je počiatočné rozdelenie komodít). Spotrebiteľ maximalizuje svoj úžitok zo spotreby vzhľadom na svoj rozpočet $w_i$ a ohraničenia spotreby. Rieši teda úlohu:

$$\max\{ \quad u^i(\mathbf{c^i}) \mid \mathbf{p}^T \mathbf{c^i} \leq w_i, \ \mathbf{c^i} \in C_i \quad \}. \tag{2.12}$$

Za predpokladu lokálnej nenasýtenosti komoditami, t.j. že pre funkciu užitočnosti $u_i$ je žiadúce väčšie množstvo ľubovoľnej komodity, sa optimum nadobudne na hranici rozpočtovej množiny (nastane rovnosť $\mathbf{p}^T \mathbf{c^i} = w_i$).

Pri naších predpokladoch spotreba $c_k^i$ spotrebiteľa $i$ komodity $k$ závisí od funkcie užitočnosti $u_i$, jeho bohatstva $w_i$ a vektora cien komodít $\mathbf{p}$. Pre ilustráciu uvedieme príklady pre niektoré tvary funkcií užitočnosti.

Pre Leontiefovu funkciu užitočnosti:

$$c_k^i(\mathbf{p}, w_i) = \frac{w_i}{\lambda_k^i \sum_{j=1}^{l} \frac{p_j}{\lambda_j^i}} \tag{2.13}$$

Pre Cobb-Douglasovu funkciu užitočnosti:

$$c_k^i(\mathbf{p}, w_i) = \frac{\lambda_k^i w_i}{p_k} \tag{2.14}$$

Pre všeobecnú CES funkciu:

$$c_k^i(\mathbf{p}, w_i) = \frac{(p_k \lambda_k^i)^{r-1} w_i}{\sum_{j=1}^{l} \frac{p_j^r}{(\lambda_j^i)^{r-1}}} \tag{2.15}$$

kde $r = \rho/(\rho - 1)$.

Poznamenajme, že funkcia $\mathbf{c^i}$ je pozitívne homogénna nultého stupňa vo všetkých vyššie uvedených prípadoch. Ďalej, keď $w_i = \mathbf{p}^T\mathbf{e^i}$ , potom $\mathbf{c^i}$ uvažovaná len ako funkcia $\mathbf{p}$ je stále pozitívne homogénna nultého stupňa.

Na záver uvedieme, že dopyt $j$-teho spotrebiteľa po komoditách na trhu $\mathbf{d^i}$ sa určí ako rozdiel medzi jeho optimálnou spotrebou $\mathbf{c^i}$ a komoditami $\mathbf{e^i}$, ktoré mal na začiatku $\mathbf{d^i}(\mathbf{p}) = \mathbf{c^i}(\mathbf{p}) - \mathbf{e^i}$.

## 2.4 Formulácia modelu.

Všetky výsledky týkajúce sa sektora výroby a spotreby sa dajú zhrnúť do nasledovnej zmiešanej úlohy o komplementarite.

$$\begin{array}{rcll} \mathbf{z} \in R_+^n, \qquad \mathbf{p} \in R_{++}^l & & & \\ -A(\mathbf{p})^T\mathbf{p} & \geq & \mathbf{0_n} & \\ A(\mathbf{p})\mathbf{z} \; - \; \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i}) & \geq & \mathbf{0_l} & \\ w_i \quad - \quad \mathbf{p}^T\mathbf{e^i} & = & 0 \quad \forall\; i = 1,\; 2\; \ldots\; m & \\ \mathbf{z}^T A(\mathbf{p})^T\mathbf{p} & = & 0 & \\ \mathbf{p}^T[A(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i})] & = & 0 & \end{array} \tag{2.16}$$

Matica technológií $A(\mathbf{p})$ je daná vzťahmi (2.10) a (2.11), ktoré popisujú maximalizáciu zisku výrobcov. Spotreba $\mathbf{c^i}(\mathbf{p}, w_i)$ je určená maximalizovaním užitočnosti spotrebiteľmi s použitím jedného zo vzťahov (2.13), (2.14), alebo (2.15) pre jednotlivé funkcie užitočnosti.

Prvá nerovnosť spolu s druhou rovnosťou a podmienkou nezápornosti vektora $\mathbf{z}$ popisujú nutné podmienky optimality pre maximalizáciu zisku výrobcov. Druhá nerovnosť popisuje podmienku (2.2), ktorá hovorí, že ponuka prevyšuje dopyt. Prvá rovnosť popisuje rozpočtové ohraničenie spotrebiteľov, presnejšie fakt, že optimum sa nadobudne na hranici tejto množiny. A posledná, tretia rovnosť hovorí o tom, že celkové výdaje spotrebiteľov na komodity sa rovnajú celkovému príjmu výrobcov z predaja týchto komodít.

Poznamenajme, že ak vektory $\mathbf{p}, \mathbf{z}, \mathbf{w}$ sú riešením vyššie uvedenej zmiešanej úlohy o komplementarite, tak aj $\lambda\mathbf{p}, \mathbf{z}, \lambda\mathbf{w}$ pre ľubovoľné $\lambda > 0$ (t.j. ak $\lambda$-násobne stúpne cenová hladina, v riešení sa to odzrkadlí len na premenných, ktoré súvisia s cenami. Množstvo vyrobeného tovaru sa nezmení. Inými slovami, model je škálovo invariantný vzhľadom na cenovú hladinu.)

Poznamenajme, že tento model možno formulovať aj bez matice technológií $A$, s menej explicitnými predpokladmi na funkciu zisku $j$-teho výrobcu $\pi_j(\mathbf{p})$. Podľa Hotellingovej lemy, ktorá uvádza, že ak zisková funkcia definovaná ako

$$\pi_j(\mathbf{p}) = \max_{\mathbf{y^j}} \{ \quad \mathbf{p}^T\mathbf{y^j} \mid \mathbf{y^j} \in Y_j \quad \} \ \leq \ 0$$

má riešenie pre nejaké $\mathbf{y^j}$, potom je diferencovateľná a vektor optimálneho riešenia (čo je vektor čistého výstupu) je $\nabla\pi_j(p) = \mathbf{y^j}$. Takéto riešenie možno použiť na formuláciu modelu rovnováhy a bude mať nasledovný tvar:

$$\begin{aligned}
\mathbf{z} \geq \mathbf{0_n}, \qquad \mathbf{p} \geq \mathbf{0_l} & & \\
-\pi(\mathbf{p}) & \geq \mathbf{0_n} & \\
\nabla\pi(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i}) & \geq \mathbf{0_l} & \\
w_i - \mathbf{p}^T\mathbf{e^i} & = 0 \quad & \forall\ i = 1,\ 2\ \ldots\ m \\
\mathbf{z}^T\pi(\mathbf{p}) & = 0 & \\
\mathbf{p}^T[\nabla\pi(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i})] & = 0 &
\end{aligned} \tag{2.17}$$

Opätovné získanie antisymetrie nám umožní Eulerova veta. Tá hovorí, že ak je funkcia $\pi_j(\mathbf{p})$ pozitívne homogénna prvého stupňa a je diferencovateľná, potom

$$\pi_j(\mathbf{p}) = \nabla\pi_j(\mathbf{p})^T\mathbf{p}.$$

Po dosadení tohto vzťahu do modelu (2.17) dostaneme nasledujúcu zmiešanú úlohu o komplementarite.

$$
\begin{aligned}
\mathbf{z} \geq \mathbf{0_n}, \qquad & \mathbf{p} \geq \mathbf{0_l} \\
-\nabla\pi(\mathbf{p})^T\mathbf{p} \quad & \geq \mathbf{0_n} \\
\nabla\pi(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i}) \quad & \geq \mathbf{0_l} \\
w_i - \mathbf{p}^T\mathbf{e^i} \quad & = 0 \qquad \forall\ i = 1,\ 2\ \ldots\ m \\
\mathbf{z}^T\nabla\pi(\mathbf{p})^T\mathbf{p} \quad & = 0 \\
\mathbf{p}^T[\nabla\pi(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i})] \quad & = 0
\end{aligned}
\tag{2.18}
$$

## 2.5 Rozšírenie modelu so zreteľom na danenie a dotácie.

Keď do nášho modelu započítame aj dane na vstupy a výstupy, ovplyvní sa ziskovosť jednotlivých výrobcov a chod produkčných procesov v sektore výroby. Daňová sadzba na vstupný tovar $x_k$ je $t_k$ a $\tau_k$ je daňová sadzba na výstupný tovar $q_k$. Tieto sadzby sú rovnaké pre každého výrobcu. Podobné následky nastanú aj s dotáciami, ale otázky dotácií sa dajú vyriešiť s použitím daní so zápornými znamienkami.

Podmienené faktorové vstupy $\mathbf{x^j}$ a podmienené výstupy $\mathbf{q^j}$ výrobcu $j$ sa dajú tak, ako v predchádzajúcej časti, určiť samostatne. Keď sú vstupné náklady skreslené daňami, výrobca $j$ rieši nasledujúce dve úlohy:

$$
\min_{\mathbf{x^j}} \left\{ \quad \sum_{k=1}^{l}(1 + t_k)p_k x_k^j \quad | \quad f_j(\mathbf{x^j}) = 1,\ \mathbf{x^j} \geq \mathbf{0} \quad \right\} \tag{2.19a}
$$

$$
\max_{\mathbf{q^j}} \left\{ \quad \sum_{k=1}^{l}(1 - \tau_k)p_k q_k^j \quad | \quad g_j(\mathbf{q^j}) = 1,\ \mathbf{q^j} \geq \mathbf{0} \quad \right\}. \tag{2.19b}
$$

Poznamenajme, že úlohy (2.19a) a (2.19b) sú analógiou úloh (2.11a) a (2.11b), len ceny sú upravené príslušnými daňovými sadzbami. Úloha (2.19a) obsahuje minimalizáciu nákladov na vstupné komodity $\mathbf{x^j}$, ktoré sú navýšené o dane. Úloha (2.19b) obsahuje maximalizáciu príjmu z predaja vyrobených komodít $\mathbf{q^j}$, ktorý je znížený o daň uvalenú na tieto komodity. Poznamenávame, že opäť ide o jednotkovú úroveň výroby.

Celková ponuka je daná z matice technológií ako $A(\mathbf{p})\mathbf{z}$ kde stĺpce matice $A$ sú vytvorené s použitím (2.10) a riešení úloh (2.19a) a (2.19b).

Ak si zadefinujeme maticu $C(\mathbf{p})$

$$C(\mathbf{p}) \equiv [\ \mathbf{c^1} \quad \mathbf{c^2} \quad \dots \quad \mathbf{c^n}\ ],$$

a jej prvky $c_i^j(\mathbf{p}) = q_i^j(1-\tau_i) - x_i^j(1+t_i)$, potom zisk $j$-teho výrobcu po zdanení pri jednotkovej úrovni výroby je daný vzťahom $\pi_j = \mathbf{c^j}(\mathbf{p})^T\mathbf{p}$.

Daň $\mathbf{T} \equiv (A(\mathbf{p}) - C(\mathbf{p}))^T\mathbf{p}$ je pre každého výrobcu určená ako súčet všetkých daní, ktoré odvedie pri jednotkovej úrovni výroby.

$$\mathbf{T} = \left[\sum_{k=1}^{l}(q_k^1\tau_k + x_k^1 t_k)p_k,\ \sum_{k=1}^{l}(q_k^2\tau_k + x_k^2 t_k)p_k,\ \dots \sum_{k=1}^{l}(q_k^n\tau_k + x_k^n t_k)p_k\right]$$

Predpokladáme, že táto daň sa rozdelí medzi spotrebiteľov. Nech $0 \le \theta_{ji} \le 1$ reprezentuje podiel daní $j$-teho výrobcu ktoré pripadli $i$-temu spotrebiteľovi. To znamená, že $i$-ty spotrebiteľ časť svojho rozpočtu $w_i$ vynaloží na zaplatenie daní. Jeho rozpočet sa tak zmenší o celkovú sumu $\sum_{j=1}^{n}\theta_{ji}T_j z_j$.

Príslušná úloha o komplementarite má teraz nasledovný tvar

$$\begin{aligned}
\mathbf{z} \ge \mathbf{0_n}, \qquad \mathbf{p} \ge \mathbf{0_l} & & \\
-C(\mathbf{p})^T\mathbf{p} & \ge \mathbf{0_n} & \\
A(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i}) & \ge \mathbf{0_l} & \\
w_i - \mathbf{p}^T\mathbf{e^i} - \sum_{j=1}^{n}\theta_{ji}T_j z_j & = 0 & \forall\ i = 1,\ 2\ \dots\ m \\
\mathbf{z}^T C(\mathbf{p})^T\mathbf{p} & = 0 & \\
\mathbf{p}^T[A(\mathbf{p})\mathbf{z} - \sum_{i=1}^{m}(\mathbf{c^i}(\mathbf{p}, w_i) - \mathbf{e^i})] & = 0 &
\end{aligned}$$

Vidíme, že jediná zmena oproti základnému modelu (2.16) nastala v rozpočtovom ohraničení spotrebiteľa.

# Kapitola 3

# Model z teórie hier.

## 3.1 Úloha hľadania Nashovho ekvilibria.

Veľké množstvo ekonomických modelov je formulovaných v pomenovaniach nekooperatívnej hry. Budeme uvažovať hru $n$ hráčov v ktorej má $i$-ty hráč ($i = 1,\ 2, \ldots n$) vlastnú množinu stratégií $X_i = R_+^{m_i},\ m_i \in N$ a funkciu užitočnosti $u_i : X_i \to R$. Úžitok hráča $i$ závisí aj od stratégií $\mathbf{x^j},\ i \neq j$ ostatných hráčov. Keď hráme podľa Nashovho princípu, každý hráč maximalizuje svoj úžitok za predpokladu, že stratégie ostatných hráčov sú známe. Pre dané stratégie ostatných hráčov $\mathbf{z^j}, j \neq i$, hráč $i$ volí najlepšiu stratégiu $\mathbf{x^i}$ a rieši nasledovnú optimalizačnú úlohu:

$$\max\{\quad u_i(\ \mathbf{z^1},\ \ldots,\ \mathbf{z^{i-1}},\ \mathbf{x^i},\ \mathbf{z^{i+1}},\ \ldots,\ \mathbf{z^n}\ )\quad |\quad \mathbf{x^i} \in X^i\quad \}. \tag{3.1}$$

V ďalšom texte budeme vektor $(\mathbf{z^1},\ \ldots,\ \mathbf{z^{i-1}},\ \mathbf{z^{i+1}},\ \ldots,\ \mathbf{z^n})$ stratégií ostatných hráčov označovať ako $\mathbf{z^{-i}}$. Úloha (3.1) sa teda prepíše nasledovne

$$\max\{\quad u_i(\ \mathbf{x^i},\ \mathbf{z^{-i}}\ )\quad |\quad \mathbf{x^i} \in X_i\quad \} \tag{3.1a}$$

Hovoríme, že vektor $\mathbf{z} \in X \equiv X_1 \times X_2 \times \ldots \times X_n$ tvorí Nashovo ekvilibrium hry, ak pre každé $i = 1,\ 2,\ \ldots,\ n$,

$$u_i(\mathbf{z^1},\ \ldots,\ \mathbf{z^{i-1}},\ \mathbf{x^i},\ \mathbf{z^{i+1}},\ \ldots,\ \mathbf{z^n}) \leq u_i(\mathbf{z}) \qquad \forall \mathbf{x^i} \in X_i.$$

Ak je funkcia užitočnosti $u_i$ hráča $i$ diferencovateľná, úloha (3.1a) je ekvivalentná nasledujúcej úlohe o komplementarite.

$$\mathbf{x^i} \geq \mathbf{0}_{m_i} \qquad -\nabla_{\mathbf{x^i}} u_i(\mathbf{x^i}, \mathbf{z^{-i}}) \geq \mathbf{0}_{m_i} \qquad \mathbf{x^i}^T \nabla_{\mathbf{x^i}} u_i(\mathbf{x^i}, \mathbf{z^{-i}}) = 0 \tag{3.2}$$

Problém hľadania Nashovho ekvilibria sa teda dá formulovať ako systém navzájom prepojených úloh o komplementarite

$$\mathbf{x^i} \geq \mathbf{0}_{m_i} \qquad -\nabla_{\mathbf{x^i}} u_i(\mathbf{x}) \geq \mathbf{0}_{m_i} \qquad \nabla_{\mathbf{x^i}} u_i(\mathbf{x})^T \mathbf{x^i} = 0 \qquad i = 1,\ 2,\ \ldots,\ n \tag{3.3}$$

kde $\mathbf{x^i}$ je riešením $i$-tej úlohy, v ktorej je jedinou premennou.

Keďže $X_i = R_+^{m_i}$, tak úloha o komplementarite (3.2) je podľa Vety 1 (kapitola 1) ekvivalentná s úlohou o variačných nerovnostiach

$$\text{nájsť } \mathbf{x^i} \in R^{m_i} \quad : \quad \forall\ \mathbf{y^i} \in R^{m_i} \qquad (\mathbf{y^i} - \mathbf{x^i})^T \left(-\nabla_{\mathbf{x^i}} u_i(\mathbf{x^i}, \mathbf{z^{-i}})\right) \geq 0 \tag{3.4}$$

Systém (3.3) sa potom prevedie na zovšeobecnenú úlohu o variačných nerovnostiach GVI($\mathbf{F}, X$), kde $X = R_+^{m_1} \times R_+^{m_2} \times \ldots \times R_+^{m_n}$ a

$$\mathbf{F}(\mathbf{x}) \equiv \begin{pmatrix} -\nabla_{\mathbf{x^1}} u_1(\mathbf{x}) \\ \vdots \\ -\nabla_{\mathbf{x^n}} u_n(\mathbf{x}) \end{pmatrix} \quad \text{pre } \mathbf{x} = (\mathbf{x^1},\ \mathbf{x^2},\ \ldots,\ \mathbf{x^n}) \in X,$$

$$\begin{aligned} \text{nájsť } \ \mathbf{z} = (\mathbf{z^1},\ \mathbf{z^2},\ \ldots \mathbf{z^n}) \in X \ : \quad & \forall\ \mathbf{y} = (\mathbf{y^1},\ \mathbf{y^2},\ \ldots \mathbf{y^n}) \in X \\ & (\mathbf{y^i} - \mathbf{z^i})^T F^i(\mathbf{z}) \geq 0 \qquad i = 1,\ 2,\ \ldots,\ n \end{aligned} \tag{3.5}$$

Riešením tejto zovšeobecnenej úlohy o variačných nerovnostiach je vektor $\mathbf{z}$, ktorý je Nashovo ekvilibrium.

## 3.2 Nash-Cournotov produkčný model.

Nash-Cournotov produkčný model je aplikáciou konceptu Nashovho ekvilibria. Tento model obsahuje $n$ firiem, každá z nich produkuje jeden tovar, ktorý je spoločný na trhu. Každá firma si určí úroveň produkcie tak, aby maximalizovala svoj zisk za predpokladu, že produkcia ostatných firiem ostane konštantná. Intuitívne bude Nashovo ekvilibrium taký spôsob výroby, pri ktorom si už žiadna firma nemôže zvýšiť svoj zisk pri jednostrannom zvyšovaní produkcie. Keďže žiadna firma už nechce meniť úroveň svojej produkcie, nastane rovnováha. Ekvilibrium tu teda znamená, že každá firma robí to najlepšie, čo môže pri daných voľbách ostatných firiem.

Ak označíme počet firiem ako $n$, $x_i$ množstvo tovaru ktoré vyrába $i$-ta firma, $\xi \equiv \sum_{i=1}^{n} x_i$ celkové množstvo tovaru vyrobeného všetkými firmami, potom problém

hľadania Nashovho ekvilibria sa dá popísať dvoma funkciami o dvoch premenných. Prvá, inverzná funkcia dopytu $p(\xi)$ (price), určuje jednotkovú cenu pri ktorej bude celkový dopyt na trhu $\xi$. Druhá, $C_i(x_i)$ (cost), určuje výrobné náklady firmy $i$, plynúce z výroby množstva $x_i$. V zhode so všeobecne uznávaným ekonomickým správaním, uvažujeme inverznú dopytovú funciu $p$ ako klesajúcu funkciu množstva $\xi$, nákladovú funkciu $C_i$ ako konvexnú a "výnosovú krivku odvetvia" $\xi \cdot p(\xi)$ ako konkávnu v skalárnej premennej $\xi \geq 0$. Pri týchto predpokladoch je funkcia užitočnosti každej firmy daná nasledovne

$$u_i(x_i) := x_i \; p\left(x_i + \sum_{j \neq i} x_j^*\right) - C_i(x_i) = x_i \; p(\xi) - C_i(x_i)$$

Funkcia užitočnosti $u_i$ je konkávna v $x_i$ a množina stratégií hráča $i$ je $X_i = R_+$. (Stratégiou tu rozumieme množstvo vyrobeného tovaru a hráčom firmu.) Každá firma sa snaží voľbou vhodnej stratégie maximalizovať svoj úžitok a rieši nasledovnú optimalizačnú úlohu

$$\max\{ \quad u_i(x_i) \mid x_i \geq 0 \quad \} \tag{3.6}$$

Vieme, že úloha (3.6) je ekvivalentná s nasledovnou úlohou o komplementarite

$$x_i \geq 0, \qquad -u_i'(x_i) \geq 0, \qquad x_i u_i'(x_i) = 0 \tag{3.7}$$

Takže keď spojíme všetkých hráčov, úloha hľadania Nashovho ekvilibria je ekvivalentná nelineárnej úlohe o komplementarite

$$\mathbf{x} \geq 0, \qquad \mathbf{F}(\mathbf{x}) \geq 0, \qquad \mathbf{x}^T \mathbf{F}(\mathbf{x}) = 0$$

kde $\mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}),\ F_2(\mathbf{x}),\ \ldots,\ F_n(\mathbf{x}))$ so zložkami

$$F_i(\mathbf{x}) \equiv -u_i'(x_i) = \ C_i{}'(x_i) - p(\xi) - x_i p'(\xi) \qquad \mathbf{x} \in R_+^n, \quad i = 1,\ 2,\ \ldots,\ n$$

a $\quad \xi = \sum_{i=1}^{n} x_i$

## 3.3 Bimaticové hry s $n$ hráčmi.

Opäť predpokladáme $n$ hráčov, ktorých indexujeme ako $j \in N \equiv \{1,\ 2,\ \ldots\ n\}$ . Pre jednoduchosť predpokladajme, že každý hráč má voľbu čistej stratégie $i$ z konečnej množiny čistých stratégií $S = \{1,\ 2,\ \ldots\ m\}$. Strategický profil hry $\mathbf{s}$ je $n$-rozmerný vektor, ktorého $j$-ty element je prvok množiny $S$ korešpondujúci s čistou stratégiou priradenou $j$ - temu hráčovi. Strategický profil hry popisuje konečnú voľbu čistých stratégií, ktorú hráči uskutočnili. Výplata hráča $j$ plynúca zo strategického profilu $\mathbf{s}$ je $A_{j\mathbf{s}}$. V hre volí hráč $j$ vektor

pravdepodobností $\mathbf{p^j}$, kde $p_i^j$ reprezentuje jeho pravdepodobnosť zvolenia čistej stratégie $i$ v ekvilibriu. Tento vektor nazývame zmiešanou stratégiou hráča $j$.

$$p_j^i \geq 0 \quad \forall i \in S, j \in N, \qquad \sum_{i \in S} p_i^j = 1 \quad \forall j \in N.$$

Teraz uvedieme ďalšie označenia. Nech $\mathbf{p^{-j}}$ reprezentuje $(n-1) \times m$ rozmerný vektor $(\mathbf{p^1},\ \ldots\ \mathbf{p^{j-1}},\ \mathbf{p^{j+1}},\ \ldots\ \mathbf{p^n})$ zmiešaných stratégií ostatných hráčov a $i(j, \mathbf{s})$ označuje čistú stratégiu priradenú hráčovi $j$ v strategickom profile hry $\mathbf{s}$. Očakávaný výnos hráča $j$ je daný

$$u_j \equiv \sum_{\mathbf{s}} A_{j\mathbf{s}} p(\mathbf{s}),$$

kde $p(\mathbf{s})$, pravdepodobnosť zvolenia strategického profilu $\mathbf{s}$, sa dá vypočítať nasledovne

$$p(\mathbf{s}) = \prod_{k \in N} p^k_{i(k,\mathbf{s})}.$$

Keďže hra je konečná, sumácia v očakávanom výnose sa dá prepísať, ako

$$u_j = \sum_{i \in S} p_i^j \ x_i^j(\mathbf{p^{-j}}),$$

kde

$$x_i^j(\mathbf{p^{-j}}) \equiv \sum_{\mathbf{s}:\ i(j,\mathbf{s})=i} A_{j\mathbf{s}} \prod_{k \in N \setminus \{j\}} p^k_{i(k,\mathbf{s})}.$$

Nelineárna funkcia $x_i^j$ reprezentuje očakávanú výplatu hráča $j$ používajúceho čistú stratégiu $i$ za predpokladu, že ostatní hráči zvolia stratégie dané $\mathbf{p^{-j}}$. Množiny stratégií sú jednotkové simplexy $P_j = \{\mathbf{p} \in R^m \ :\ \mathbf{p} \geq \mathbf{0},\ \sum_{i=1}^m p_i = 1\}$. Koncept hľadania Nashovho ekvilibria prešiel teda na úlohu hľadania takého pravdepodobnosného profilu hráčov $\bar{\mathbf{p}} = (\bar{\mathbf{p}}^{\mathbf{1}}, \ldots \bar{\mathbf{p}}^{\mathbf{n}})$, že pre každé $j \in N$

$$\bar{\mathbf{p}}^{\mathbf{j}} \in P_j, \qquad \sum_{i \in S} p_i^j \ x_i^j(\bar{\mathbf{p}}^{-\mathbf{j}}) \leq \sum_{i \in S} \bar{p}_i^j \ x_i^j(\bar{\mathbf{p}}^{-\mathbf{j}}) \quad \forall p^j \in P_j.$$

Toto je ekvivalentné s úlohou $\mathrm{VI}(\mathbf{F}, P)$ :

$$\begin{aligned} \text{nájsť } \bar{\mathbf{p}} = (\bar{\mathbf{p}}^{\mathbf{1}},\ \bar{\mathbf{p}}^{\mathbf{2}},\ \ldots\ \bar{\mathbf{p}}^{\mathbf{n}}) \in P \ :\quad & \forall \mathbf{p} = (\mathbf{p^1},\ \mathbf{p^2},\ \ldots\ \mathbf{p^n}) \in P \\ & (\mathbf{p^j} - \bar{\mathbf{p}}^{\mathbf{j}})^T \mathbf{F^j}(\bar{\mathbf{p}}) \geq 0 \qquad j = 1,\ 2,\ \ldots\ n \end{aligned} \tag{3.8}$$

kde jednotlivé zložky zobrazenia $\mathbf{F}$ sú $F_i^j(p) = -x_i^j(p^{-j})$ a $P = P_1 \times P_2 \times \ldots \times P_n$.

# Kapitola 4

# Regularizačná Newtonova metóda.

V tejto kapitole popíšeme jednu z metód, určených na riešenie nelineárnych úloh o komplementarite ( NCP(F) ). Pre pripomenutie uvedieme základný tvar úlohy o komplementarite:

**Úloha NCP(F) :**

Nájsť také $\mathbf{x}^* \in R^n$ aby platilo

$$\text{NCP}(F): \qquad \mathbf{x}^* \geq \mathbf{0_n}, \quad \mathbf{F}(\mathbf{x}^*) \geq \mathbf{0_n}, \quad (\mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = 0$$

kde $F : R^n \to R^n$ je ľubovoľné dané zobrazenie, o ktorom predpokladáme, že je spojite diferencovateľné.

Na riešenie úloh tohto typu sú známe dva prístupy. Metódy "proximal point methods" a regularizačné metódy (regularization methods). Tu popíšeme metódu, ktorá je z triedy regularizačných metód. Metódy v tejto triede sa snažia obísť problém singularity príslušnej Jacobiho matice uvažovaním postupnosti perturbovaných úloh, ktoré môžu mať lepšie vlastnosti. Pre nelineárne úlohy o komplementarite je najjednoduchšou regularizačnou technikou takzvaná Tichonovova regularizácia. Táto pozostáva v nahradení funkcie $\mathbf{F}(\mathbf{x})$ funkciou

$$\mathbf{F}_\varepsilon(\mathbf{x}) := \mathbf{F}(\mathbf{x}) + \varepsilon \mathbf{x}, \tag{4.1}$$

kde $\varepsilon$ je kladný parameter idúci k nule a následnom riešení postupnosti úloh o komplementarite NCP($\mathbf{F}_\varepsilon$).

**Úloha NCP($\mathbf{F}_\varepsilon$) :**

Nájsť také $\mathbf{x}^* \in R^n$ aby platilo

$$\text{NCP}(F_\varepsilon): \qquad \mathbf{x}^* \geq \mathbf{0_n}, \quad \mathbf{F}_\varepsilon(\mathbf{x}^*) \geq \mathbf{0_n}, \quad (\mathbf{x}^*)^T \mathbf{F}_\varepsilon(\mathbf{x}^*) = 0$$

## 4.1 Základné definície a označenia.

**Definícia 1.** Matica $W \in R^{n\times n}$ sa nazýva

- $P_0$-matica, ak každý jej hlavný minor je nezáporný.
- $P$-matica, ak každý jej hlavný minor je kladný.

Poznamenávame, že kladne semidefinitná matica je $P_0$-matica a kladne definitná matica je $P$-matica.

**Definícia 2.** Zobrazenie $F : R^n \to R^n$ sa nazýva

- $P_0$ zobrazenie, ak pre $\forall \mathbf{x}, \mathbf{y} \in R^n, \mathbf{x} \neq \mathbf{y}$ existuje $i \in \{1, 2, \dots n\}$ také, že $x_i \neq y_i \quad (y_i - x_i)(F_i(\mathbf{x} - F_i(\mathbf{y})) \geq 0$
- monotónne zobrazenie, ak pre $\forall \mathbf{x}, \mathbf{y} \in R^n$ :

$$(\mathbf{x} - \mathbf{y})^T(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})) \geq 0$$

Poznamenávame, že každé monotónne zobrazenie je $P_0$ zobrazenie, navyše Jakobián spojite diferencovateľnej $P_0$ funkcie je $P_0$-matica.

V algoritme regularizačnej newtonovej metódy budeme používať Fischerovu funckiu $\Psi : R^2 \to R$, ktorú ako prvý vo svojej práci uviedol Fischer (r. 1992) [4].

**Definícia 3.** Fischerova funkcia je definovaná nasledovným predpisom:

$$\Psi(a, b) := \sqrt{a^2 + b^2} - (a + b) \tag{4.2}$$

Fischerova funkcia má jednu zaujímavú vlastnosť, ktorá sa dá vyžiť v úlohách o komplementarite.

$$\Psi(a, b) = 0 \quad \Leftrightarrow \quad a \geq 0,\ b \geq 0, \quad a.b = 0 \tag{4.3}$$

Túto vlastnosť sme už uviedli aj v kapitole 1. Prvý krát bola Fischerova funkcia použitá na štúdium regularizačných metód Facchineim a Kanzowom (r. 1999) [3]. V algoritme regularizačnej newtonovej metódy je použitá pre svoju jednoduchosť.

V nasledovnom budeme používať označenie

$$\mathbf{z} := (\varepsilon, \mathbf{x}) \in R_+ \times R^n$$

a $F_{\varepsilon,i}$ bude označovať $i$-ty komponent zobrazenia $\mathbf{F}_\varepsilon$, $i \in N = \{1,\ 2,\ \ldots n\}$.

Zobrazenie $\mathbf{G} : R^{n+1} \to R^n$ definujeme nasledovne:

$$G_i(\varepsilon, \mathbf{x}) := \Psi(x_i, \mathbf{F}_{\varepsilon,i}(\mathbf{x})) \qquad i \in N, \tag{4.4}$$

kde $\Psi$ je (4.2).

Ďalej si definujeme zobrazenie $\mathbf{H} : R^{n+1} \to R^{n+1}$

$$\mathbf{H}(\mathbf{z}) := \begin{pmatrix} \varepsilon \\ \mathbf{G}(\mathbf{z}) \end{pmatrix}, \qquad \mathbf{z} = (\varepsilon, \mathbf{x}) \in R \times R^n \tag{4.5}$$

Ľahko vidno, že

$$\mathbf{H}(\varepsilon, \mathbf{x}^*) = \mathbf{0} \quad \Leftrightarrow \quad \varepsilon = 0 \text{ a } \mathbf{x}^* \text{ je riešenie úlohy NCP}(\mathbf{F}); \quad (\varepsilon, \mathbf{x}^*) \in R \times R^n$$

Definujme ďalej funkciu $\varphi : R^{n+1} \to R_+$

$$\varphi(\mathbf{z}) := \|\mathbf{G}(\mathbf{z})\|^2 \tag{4.6}$$

kde $\|\cdot\|$ je $l_2$ norma.

Keďže $\Psi(\cdot)^2$ je spojite diferencovateľná na $R^2$, $\varphi(\cdot)$ je spojite diferencovateľná na celom $R^{n+1}$. Definujme ešte poslednú funkciu $f : R^{n+1} \to R_+$ :

$$f(\mathbf{z}) := \|\mathbf{H}(\mathbf{z})\|^2 \ = \varepsilon^2 + \varphi(\mathbf{z}). \tag{4.7}$$

Keďže $f(\mathbf{z}) = \varepsilon^2 + \varphi(\mathbf{z})$, $f$ je tiež spojite diferencovateľná na $R^{n+1}$.

## 4.2 Algoritmus Regularizačnej Newtonovej metódy.

Zvolíme si parametre $\overline{\varepsilon} \in (0, \infty)$ a $\gamma \in (0, 1)$ také, že $\gamma.\overline{\varepsilon} < 1$. Označíme $\overline{\mathbf{z}} := (\overline{\varepsilon}, \mathbf{0}) \in R \times R^n$. Definujeme funkciu $\beta : R^{n+1} \to R_+$:

$$\beta(\mathbf{z}) := \gamma \min\{1, f(\mathbf{z})\}, \tag{4.8}$$

kde $f(\mathbf{z})$ je definovaná vzťahom (4.7).

Definujme si množinu $\Omega$

$$\Omega := \{\ \mathbf{z} = (\varepsilon, \mathbf{x}) \in R \times R^n \quad | \quad \varepsilon \geq \beta(\mathbf{z}).\overline{\varepsilon}\ \}.$$

Keďže pre ľubovoľné $\mathbf{z} \in R^{n+1}$ platí $\beta(\mathbf{z}) \leq \gamma$, potom pre ľubovoľné $\mathbf{x} \in R^n$ platí $(\overline{\varepsilon}, \mathbf{x}) \in \Omega$. Keď zvolíme počiatočný bod $\mathbf{z^0} = (\overline{\varepsilon}, \mathbf{x^0})$, kde $\mathbf{x^0} \in R^n$ bude ľubovoľné, tak postupnosť iterácií generovaná nižšie uvedeným algoritmom ostane v $\Omega$. Toto je dôležitá vlastnosť, pretože v tomto okolí možno zamedziť príliš rýchlej konvergencii postupnosti iterácií k bodu, ktorý nie je riešením. ( dôkaz je uvedený v literatúre [2] )

### 4.2.1 Vstup algoritmu.

Vstupom pre algoritmus regularizačnej newtonovej metódy bude zobrazenie $\mathbf{F} : R^n \to R^n$ a začiatočný bod $\mathbf{x^0}$.

### 4.2.2 Parametre algoritmu.

Algoritmus má nasledovné parametre:

| | |
|---|---|
| $\delta \in (0,1)$ | |
| $\sigma \in (0, \frac{1}{2})$ | |
| $\overline{\varepsilon} \in (0, \infty)$ | |
| $\gamma \in (0,1) : \quad \gamma.\overline{\varepsilon} < 1$ | |
| $\overline{\mathbf{z}} := (\overline{\varepsilon}, \mathbf{0_n})$ | |
| $\varepsilon^0 := \overline{\varepsilon}$ | |
| $\mathbf{z^0} := (\varepsilon^0, \mathbf{x^0})$ | počiatočný bod |
| $W$, na začiatku $W = f(\mathbf{z^0})$ | |
| $k$, na začiatku $k = 0$ | cez $k$ iterujeme |

### 4.2.3 Telo algoritmu.

Samotný algoritmus regularizačnej newtonovej metódy vyzerá nasledovne.

Pokiaľ $\mathbf{H}(\mathbf{z^k}) \neq 0$, opakujeme

Začiatok

Krok 1.

$\beta_k := \beta(\mathbf{z^k}) = \gamma \min\{1, f(\mathbf{z^k})\}$

$V_k \in \partial\mathbf{H}(\mathbf{z^k})$

Krok 2.

$V_k$ volíme ako jakobián zobrazenia $\mathbf{H}$

vypočítame $\quad \triangle\mathbf{z^k} = (\triangle\varepsilon^k, \triangle\mathbf{x^k})$

$\mathbf{H}(\mathbf{z^k}) + V_k.\triangle\mathbf{z^k} = \beta_k.\overline{\mathbf{z}}$

Krok 3.

nájdeme $l_k$ : najmenšie prirodzené číslo $l$, ktoré vyhovuje :

$\mathbf{z^k} + \delta^l.\triangle\mathbf{z^k} \in \Omega = \{\ \mathbf{z^k} \quad | \quad \varepsilon^k \geq \beta(\mathbf{z^k}).\overline{\varepsilon}\ \}$

a $f(\mathbf{z^k} + \delta^l\triangle\mathbf{z^k}) \leq W - 2\sigma(1 - \gamma.\overline{\varepsilon}).\delta^l.f(\mathbf{z^k})$,

kde $W$ volíme podľa nasledovných pravidiel :

1. na začiatku $W = f(\mathbf{z^0})$
2. ak $f(\mathbf{z^k}) \leq \min_{j=1,2,\ldots 5} f(\mathbf{z^{k-j}})$ tak $W$ sa nemení
   inak $W = f(\mathbf{z^k})$.

Krok 4.

$\mathbf{z^{k+1}} := \mathbf{z^k} + \delta^{l_k}\triangle\mathbf{z^k}$

$k := k + 1$

Koniec

Výstupom algoritmu bude posledná iterácia $\mathbf{x^k}$ a počet iterácií $k$. Pre bod $\mathbf{x^k}$ môžeme pre kontrolu vypočítať aj $\mathbf{F}(\mathbf{x^k})$.

Poznamenajme, že pre regularizačnú newtonovu metódu možno za predpokladu, že $\mathbf{F}$ je $P_0$ zobrazenie dokázať, že každý hromadný bod postupnosti iterácií je riešením NCP(F) a že postupnosť iterácií je ohraničená, ak je množina riešení NCP(F) neprázdna a ohraničená. Naviac, ak zobrazenie $\mathbf{F}$ je monotónne a Lipschitzovsky spojité, možno dokázať, že postupnosť iterácií je ohraničená práve vtedy, keď množina riešení úlohy NCP(F) je neprázdna. Dôkazy týchto tvrdení sú uvedené v literatúre [2].

# Kapitola 5

# Numerické experimenty.

V tejto kapitole popíšeme výsledky, ktoré sme získali pri testovaní nami naprogramovanej regularizačnej newtonovej metódy na riešenie úloh o komplementarite. Túto metódu sme testovali na úlohách s vopred známym riešením, kvôli kontrole správnosti riešenia.

## 5.1 Vytváranie úloh.

Na vygenerovanie úlohy o komplementarite so známym riešením sme potrebovali vygenerovať najprv zobrazenie $\mathbf{F} : R^n \to R^n$, riešenie úlohy $\mathbf{x}^*$ a následne upraviť zobrazenie o vektor $d$ tak, aby úloha o komplementarite ( $\mathbf{x} \geq 0,\ \mathbf{F}(\mathbf{x}) \geq 0,\ \mathbf{x}^T\mathbf{F}(\mathbf{x}) = 0$ ) mala riešenie $\mathbf{x}^*$.

Zobrazenie $\mathbf{F} : R^n \to R^n$ sme generovali v nasledujúcom tvare :

$$\mathbf{F}(\mathbf{x}) = \frac{1}{\|A\|_F^2}(\mathbf{x}^T A\mathbf{x})A\mathbf{x} + B\mathbf{x} + \mathbf{d} \tag{5.1}$$

kde $A \in R^{n\times n}$, $B \in R^{n\times n}$ sú kladne definitné, regulárne matice, $\|\cdot\|_F$ je Frobeniova norma matice ( $\|A\|_F^2 = \sum_{i,j=1}^{n} (a_{ij})^2$ ) a $\mathbf{d}$ je vektor, ktorým zabezpečíme, že $\mathbf{x}^*$ bude riešením úlohy o komplementarite. (Poznamenávame, že takto vytvorené zobrazenie $\mathbf{F}$ je $P_0$ zobrazenie.)

Vektor $\mathbf{x}^*$ sme generovali ako kombináciu núl a jednotiek tak, aby bol ich počet približne rovnaký. Keď už máme známe riešenie $\mathbf{x}^*$, vektor $\mathbf{d}$ dopočítavame nasledovným spôsobom:

a) ak $x_i^* = 0$ potom $d_i = 1 - \frac{1}{\|A\|_F^2}(\mathbf{x}^{*T} A\mathbf{x}^*)(A\mathbf{x}^*)_i - (B\mathbf{x}^*)_i$

b) ak $x_i^* > 0$ potom $d_i = -\frac{1}{\|A\|_F^2}(\mathbf{x}^{*T} A\mathbf{x}^*)(A\mathbf{x}^*)_i - (B\mathbf{x}^*)_i$

pre všetky $i = 1, 2, \ldots n$.

Keď $\mathbf{F}$ vytvoríme takýmto spôsobom, tak pre každé $i = 1, 2, \ldots n$ bude platiť buď $x_i^* = 0$ a $F_i(x^*) = 1$, alebo $x_i^* = 1$ a $F_i(\mathbf{x}^*) = 0$.

Poznamenávame, že takto vytvorené riešenie úlohy o komplementarite neuberá na všeobecnosti, pretože v týchto úlohách nám ide len o to, či jednotlivé zložky sú, alebo nie sú nulové a keď nie sú nulové, nezáleží nám na tom, aká je ich hodnota.

Ďalej poznamenávame, že takto vytvorená úloha má jednoznačné riešenie, pretože pôvodne vznikla ako nutné a postačujúce podmienky optimality pre úlohu

$$\min\left\{ \frac{1}{\|A\|_F^2}\frac{1}{4}(\mathbf{x}^T A\mathbf{x})^2 + \frac{1}{2}\mathbf{x}^T B\mathbf{x} + C \mid \mathbf{x} \geq 0 \right\}$$

Táto má jednoznačné riešenie.

## 5.2 Úprava systému lineárnych rovníc.

V kroku 2. Algoritmu regularizačnej newtonovej metódy potrebujeme vyriešiť systém $n+1$ lineárnych rovníc o $n+1$ neznámych $V_k \triangle \mathbf{z}^k = \beta_k \overline{\mathbf{z}} - \mathbf{H}(\mathbf{z^k})$, kde $V_k$ je Jakobián zobrazenia $\mathbf{G}$ (4.3). Tento systém má nasledovný tvar

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \frac{\partial G_1(\mathbf{z^k})}{\partial \varepsilon} & \frac{\partial G_1(\mathbf{z^k})}{\partial x_1} & \cdots & \frac{\partial G_1(\mathbf{z^k})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial G_n}{\partial \varepsilon}(\mathbf{z^k}) & \frac{\partial G_n(\mathbf{z^k})}{\partial x_1} & \cdots & \frac{\partial G_n(\mathbf{z^k})}{\partial x_n} \end{pmatrix} \begin{pmatrix} \triangle\varepsilon^k \\ \triangle x_1^k \\ \vdots \\ \triangle x_n^k \end{pmatrix} = \begin{pmatrix} \beta_k \cdot \overline{\varepsilon} \; - \quad \varepsilon^k \\ \beta_k \cdot 0 - G_1(\mathbf{z^k}) \\ \vdots \\ \beta_k \cdot 0 - G_n(\mathbf{z^k}) \end{pmatrix} \tag{5.2}$$

Vektor $\mathbf{H}(\mathbf{z^k})$ sme nahradili jeho konkrétnym tvarom ( $\mathbf{H}(\mathbf{z^k}) = (\varepsilon, G_1(\mathbf{z^k}), \ldots G_n(\mathbf{z^k}))$ ), takisto aj vektor $\overline{\mathbf{z}}$ ( $\overline{\mathbf{z}} = (\overline{\varepsilon}, 0, \ldots, 0)$ ). Číslo $\beta_k$ je dané vzťahom $\beta_k = \gamma \cdot \min\{1, f(\mathbf{x^k})^t\}$, definovaným v kapitole 4.

Na prvý pohľad si možno všimnúť, že neznáma $\triangle\varepsilon^k$ sa dá zo systému jednoducho eliminovať. Je jasné, že riešenie je

$$\triangle\varepsilon^k = \beta_k\overline{\varepsilon} - \varepsilon^k.$$

Týmto sa zníži rozmer systému na $n$ a prejde na nasledovný tvar:

$$\begin{pmatrix} \frac{\partial G_1(\mathbf{z^k})}{\partial x_1} & \cdots & \frac{\partial G_1(\mathbf{z^k})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial G_n(\mathbf{z^k})}{\partial x_1} & \cdots & \frac{\partial G_n(\mathbf{z^k})}{\partial x_n} \end{pmatrix} \begin{pmatrix} \triangle x_1^k \\ \vdots \\ \triangle x_n^k \end{pmatrix} = \begin{pmatrix} -G_1(\mathbf{z^k}) - \frac{\partial G_1(\mathbf{z^k})}{\partial \varepsilon} \cdot \triangle\varepsilon^k \\ \vdots \\ -G_n(\mathbf{z^k}) - \frac{\partial G_n(\mathbf{z^k})}{\partial \varepsilon} \cdot \triangle\varepsilon^k \end{pmatrix} \tag{5.3}$$

Označme si vektor pravej strany ako $\mathbf{r^k}$ ($r_i^k = -G_i(\mathbf{z^k}) - \frac{\partial G_i(\mathbf{z^k})}{\partial \varepsilon} \cdot \triangle\varepsilon^k$, $i = 1, 2, \ldots n$) a maticu na ľavej strane ako $U_k$. Potom (5.3) možno zapísať nasledovne

$$U_k \triangle \mathbf{x^k} = \mathbf{r^k} \tag{5.3a}$$

Zobrazenie $\mathbf{G}$ je definované po zložkách: $G_i(\mathbf{x}) = \Psi(x_i, F_i(\mathbf{x}+\varepsilon x_i)),\ i = 1, 2, \ldots n$, kde $\Psi$ je Fischerova funkcia (4.1). Jednotlivé parciálne derivácie sú potom nasledovné:

$$\frac{\partial G_i(\mathbf{z})}{\partial \varepsilon} = \left[\frac{F_i(\mathbf{x}) + \varepsilon x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] \cdot x_i \qquad i = 1,\ 2,\ \ldots n \tag{5.4}$$

$$\frac{\partial G_i(\mathbf{z})}{\partial x_j} = \left[\frac{F_i(\mathbf{x}) + \varepsilon x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] \cdot \frac{\partial F_i(\mathbf{x})}{\partial x_j} \qquad i \neq j \tag{5.5}$$

$$\frac{\partial G_i(\mathbf{z})}{\partial x_i} = \left[\frac{F_i(\mathbf{x}) + \varepsilon x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] \cdot \left[\frac{\partial F_i(\mathbf{x})}{\partial x_i} + \varepsilon\right] + \left[\frac{x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] \tag{5.6}$$

Vidíme, že výrazy (5.4), (5.5) a (5.6) obsahujú rovnaký člen,

$$Q_i(\mathbf{x}) = \left[\frac{F_i(\mathbf{x}) + \varepsilon x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] < 0 \qquad i = 1,\ 2,\ \ldots n.$$

Podobne zavedieme označenie $\overline{Q}_i$ pre komplementárny výraz

$$\overline{Q}_i(\mathbf{x}) = \left[\frac{x_i}{\sqrt{x_i^2 + (F_i(\mathbf{x}) + \varepsilon x_i)^2}} - 1\right] < 0 \qquad i = 1,\ 2,\ \ldots n.$$

Keď tieto označenia dosadíme do výrazov (5.4 - 5.6), dostaneme nasledovné vzťahy.

$$\begin{aligned}
\frac{\partial G_i(\mathbf{z})}{\partial \varepsilon} &= Q_i(\mathbf{x}) \cdot x_i && i = 1,\ 2,\ \ldots n \\
\frac{\partial G_i(\mathbf{z})}{\partial x_j} &= Q_i(\mathbf{x}) \cdot \frac{\partial F_i(\mathbf{x})}{\partial x_j} && i \neq j \\
\frac{\partial G_i(\mathbf{z})}{\partial x_i} &= Q_i(\mathbf{x}) \cdot \left[\frac{\partial F_i(\mathbf{x})}{\partial x_i} + \varepsilon\right] + \overline{Q}_i(\mathbf{x})
\end{aligned}$$

S použitím tohto prepisu je potom $i$-ty riadok matice $U_k$ nasledovný

$$\underset{j = 1,\ 2,\ \ldots i-1}{Q_i(\mathbf{x}^\mathbf{k})\tfrac{\partial F_i(\mathbf{x}^\mathbf{k})}{\partial x_j},} \qquad Q_i(\mathbf{x}^\mathbf{k})\left[\tfrac{\partial F_i(\mathbf{x}^\mathbf{k})}{\partial x_i} + \varepsilon^k\right] + \overline{Q}_i(\mathbf{x}^\mathbf{k}), \qquad \underset{j = i+1,\ \ldots n}{Q_i(\mathbf{x}^\mathbf{k})\tfrac{\partial F_i(\mathbf{x}^\mathbf{k})}{\partial x_j}}$$

a pravá strana $r_i^k$ bude

$$r_i^k = -G_i(\mathbf{x}^\mathbf{k}) - Q_i(\mathbf{x}^\mathbf{k}).x_i^k \triangle \varepsilon^k \qquad i = 1,\ 2,\ \ldots n.$$

Z tohto zápisu vidno, že z $i$-teho riadku matice $U_k$ sa dá vyňať $Q_i = Q_i(\mathbf{x}^\mathbf{k})$, potom dostávame

$$U_k = diag\{Q_1,\ Q_2, \ldots Q_n\} \begin{pmatrix} \frac{\partial F_1(\mathbf{x}^\mathbf{k})}{x_1} + \varepsilon^k + q_1 & \cdots & \frac{\partial F_1(\mathbf{x}^\mathbf{k})}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n(\mathbf{x}^\mathbf{k})}{x_1} & \cdots & \frac{\partial F_n(\mathbf{x}^\mathbf{k})}{x_n} + \varepsilon^k + q_n \end{pmatrix} \tag{5.7}$$

kde $q_i = \overline{Q}_i/Q_i,\ i = 1, 2, \ldots n$. A teda systém (5.3a) možno prenásobiť zľava maticou $diag\{Q_1^{-1}, \ldots Q_n^{-1}\}$, ktorá je inverzná k matici $diag\{Q_1, \ldots Q_n\}$, čím ho prevedieme na ekvivalentný systém (5.8)

$$\left[\partial\mathbf{F} + diag\{\varepsilon^k + q_1, \ldots, \varepsilon^k + q_n\}\right] \triangle\mathbf{x^k} = \overline{\mathbf{r}}^{\mathbf{k}}, \tag{5.8}$$

kde $\partial\mathbf{F}$ je Jakobián zobrazenia $\mathbf{F}$ a $\overline{r}_i^k = -G_i(\mathbf{x^k})/Q_i(\mathbf{x^k}) - x_i^k \triangle\varepsilon^k$ $(i = 1,\ 2,\ \ldots n)$, presnejšie

$$\overline{r}_i^k = \sqrt{1 + \left(\frac{F_i(x^k)}{x_i^k} + \varepsilon^k\right)^2} \left\{\sqrt{(x_i^k)^2 + (F_i(\mathbf{x^k}) + \varepsilon^k x_i^k)^2} + (F_i(\mathbf{x^k}) + \varepsilon^k x_i^k) - x_i^k\right\}$$

Označme

$$\overline{U}_k = \left[\partial\mathbf{F} + diag\{\varepsilon^k + q_1, \ldots, \varepsilon^k + q_1\}\right]$$

Potom (5.8) možno zapísať v nasledovne:

$$\overline{U}_k \triangle\mathbf{x^k} = \overline{\mathbf{r}}^{\mathbf{k}} \tag{5.8a}$$

Zobrazenie $\mathbf{F}$ sme generovali v nasledovnom tvare (4.1):

$$\mathbf{F}(\mathbf{x}) = \frac{1}{\|A\|_F^2}(\mathbf{x}^T A\mathbf{x})A\mathbf{x} + B\mathbf{x} + \mathbf{d}$$

a tak Jakobián $\partial\mathbf{F}$ tohto zobrazenia je:

$$\partial\mathbf{F}(\mathbf{x}) = \frac{1}{\|A\|_F^2}\left(2(A\mathbf{x})(A\mathbf{x})^T + (\mathbf{x}^T A\mathbf{x})A\right) + B \tag{5.9}$$

Matice $A$, $B$ sú kladne definitné, a teda aj Jakobián $\partial\mathbf{F}$ je kladne definitná matica. Z toho máme, že aj matica $\overline{U}_k$ je regulárna, pretože je súčtom kladne definitnej matice $\partial\mathbf{F}$ a kladnej diagonálnej matice $diag\{\varepsilon^k + q_1, \ldots, \varepsilon^k + q_n\}$, kde $q_i = \overline{Q}_i/Q_i > 0,\ i = 1, 2, \ldots n$.

Namiesto systému (5.2), ktorý má rozmer $n+1$ budeme riešiť menší systém (5.8a), ktorý má rozmer $n$ a je aj stabilnejší, než pôvodný systém $V_k \triangle\mathbf{z}^k = \beta_k\overline{\mathbf{z}} - \mathbf{H}(\mathbf{z^k})$. Na riešenie tohto systému lineárnych rovníc sme použili metódu QR rozkladu.

## 5.3 Hodnoty parametrov použitých v programe.

Algoritmus Regularizačnej newtonovej metódy sme naprogramovali v jazyku C++ s použitím vývojového prostredia Microsoft Visual C++. Hodnoty parametrov, ktoré obsahuje algoritmus sme (podobne ako autori [2]) zvolili nasledovné:

$$\begin{aligned} \delta &= 0.5 \\ \sigma &= 0.00005 \\ \overline{\varepsilon} &= 1 \\ \gamma &= 0.1 \end{aligned}$$

Ako koncovú podmienku algoritmu sme zvolili

$$f(\mathbf{z}^k) < tol_\varepsilon \ , \quad tol_\varepsilon = 10^{-10}$$

Uvedieme aj hodnoty ďalších parametrov, ktoré sme použili v našom programe (pokiaľ neuvedieme iné):

| | | |
|---|---|---|
| $N$ | $= 5$ | rozmer riešených úloh, zadávaný užívateľom |
| $PP$ | $= 1$ | minimálny počet 1 vo vektore $\mathbf{x}^*$ |
| $ODCH$ | $= 0.1$ | štartovací bod sme volili $x_i^0 = x_i^* + ODCH$ |
| $MAX$ | $= 1000$ | označuje počet riešených úloh |
| $K\_MAX$ | $= 500$ | horné ohraničenie pre počet iterácií |
| $L\_MAX$ | $= 10$ | horné ohraničenie pre veľkosť $l_k$ |

## 5.4 Výsledky z experimentov.

Samotný experiment sme zostavili ako sériu riešení 1000 (MAX) náhodných úloh, pre ktoré boli hodnoty parametrov rovnaké. Pre každú úlohu sa najprv vygenerovali príslušné matice $A$, $B$ pre funkciu $F$, potom bod riešenia $\mathbf{x}^*$. Následne sa ešte dogeneroval vektor $\mathbf{d}$, ktorý zabezpečuje, že $\mathbf{x}^*$ je riešením úlohy o komplementarite (1.1). Bod $\mathbf{x}^*$ sme vytvárali tak, aby počet jednotiek (aj núl) bol minimálne $PP$ a maximálne $N - PP$. Týmto sme sa vyhli takým riešeniam, ktoré by obsahovali len samé nuly alebo jednotky (vtedy je komplementárny vektor $\mathbf{F}(\mathbf{x}^*) = \mathbf{0_N}$)

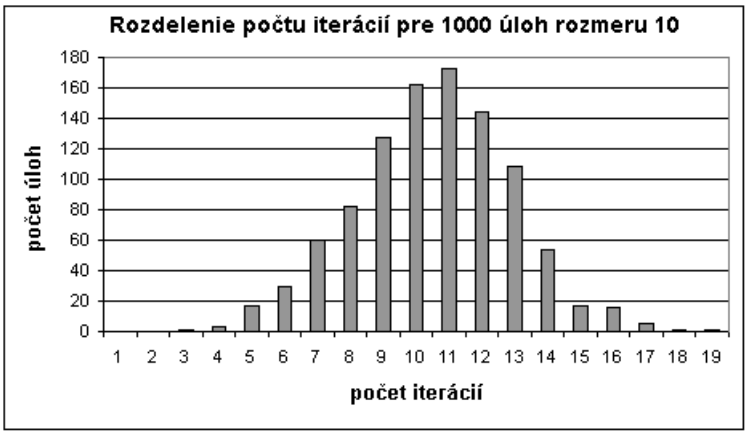
Štartovací bod $\mathbf{x^0}$ sme volili nasledovne:

$$x_i^0 = x_i^* + ODCH$$

Parametrom ODCH sme menili vzdialenosť štartovacieho bodu od optimálneho riešenia. Po vyriešení všetkých 1000 úloh sme zaznamenali priemerný počet iterácií potrebných na vyriešenie daných úloh, minimálny a maximálny počet iterácií a celkový čas potrebný na vyriešenie všetkých 1000 úloh.

Uskutočnili sme dva typy experimentov. V jednom sme menili len rozmer úloh, v druhom sme pre vybrané rozmery menili vzdialenosť štartovacieho bodu od optimálneho riešenia.

Ako prvé sme si všimli rozdelenie počtu iterácií potrebných na vyriešenie 1000 úloh. Uvedieme histogram len pre rozmer úloh $N = 10$.




Obr.5.1

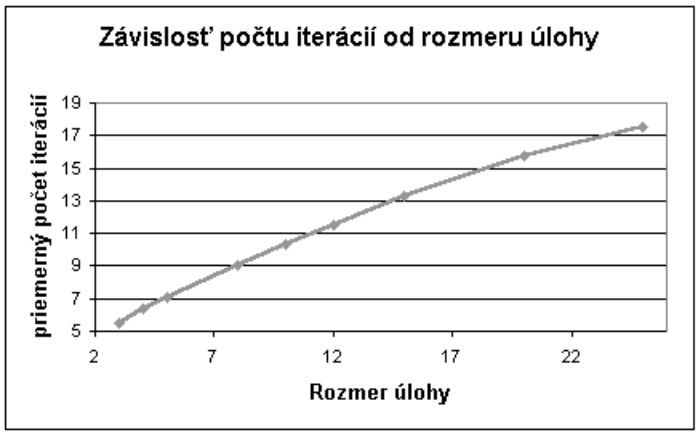
Z (Obr.5.1) vidno, že toto rozdelenie je blízke normálnemu rozdeleniu. Preto môžeme používať strednú hodnotu počtu iterácií a bez toho, aby boli výsledky zavádzajúce.

V tabuľke (Tab.5.1) uvádzame prehľad číselných výsledkov pre rôzne rozmery úloh. Menil sa len parameter $N$ a $PP$, ostatné ostávali nezmenené. V prvom riadku sú uvedené hodnoty $N$, v zátvorke je $PP$.

| rozmer | 3 (1) | 4 (1) | 5 (1) | 8 (1) | 10 (2) | 12 (2) | 15 (3) | 20 (4) | 25 (5) |
|---|---|---|---|---|---|---|---|---|---|
| priem. Iter. | 5,502 | 6,336 | 7,114 | 9,042 | 10,331 | 11,511 | 13,273 | 15,716 | 17,542 |
| k_min | 2 | 2 | 2 | 2 | 3 | 4 | 4 | 5 | 6 |
| k_max | 11 | 13 | 13 | 17 | 19 | 21 | 24 | 18 | 38 |
| cas | 7,85 | 10,16 | 12,97 | 26,09 | 39,38 | 58,11 | 90,84 | 179,06 | 309,01 |

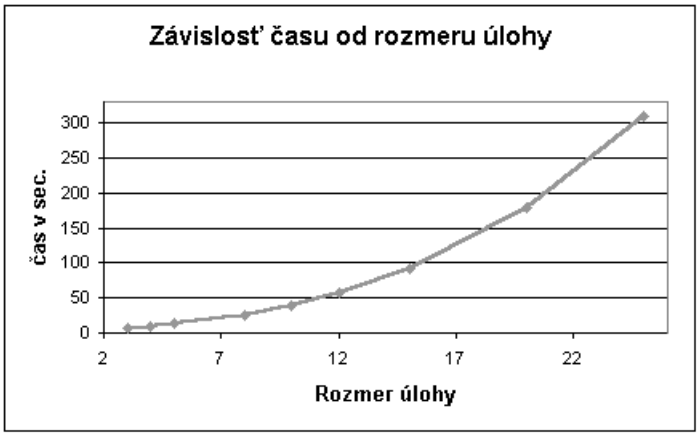
Tab.5.1

Riadky k_min a k_max obsahujú minimálny a maximálny počet iterácií potrebných na vyriešenie úloh v testovanej sérii. Ako vidno, regularizačná newtonova metóda je schopná veľmi rýchlo skonvergovať a najhoršie prípady tiež nie sú veľmi zlé. Údaje z Tab.5.1 sú znázornené na nasledovných grafoch.




Obr.5.2

Z obr.5.2 vidno, že priemerný počet iterácií rastie s rozmerom riešených úloh, no tento rast sa spomaľuje.




Obr.5.3

Obr. 5.3 zobrazuje závislosť času potrebného na vyriešenie celej série úloh od ich rozmeru. Čas je uvedený v sekundách. Možno si všimnúť, že rastie kubicky. Toto je spôsobené tým, že s rozmerom úlohy rastie aj rozmer lineárneho systému, ktorý treba riešiť, čím stúpa aj počet operácií potrebných na vyriešenie tohto systému, a teda aj čas.

V ďalších experimentoch sme rozmer úloh volili $N = 5,\ 10,\ 15$ a pre tieto rozmery sme menili odchýlku $ODCH$. Výsledky sú uvedené tabuľkách Tab.5.2 a Tab.5.3. Opäť sme sledovali priemerný počet iterácií a čas potrebný na vyriešenie celej série úloh. V prvom riadku tabuliek sú uvedené vybrané rozmery úloh, v prvom stĺpci sú hodnoty parametra $ODCH$.

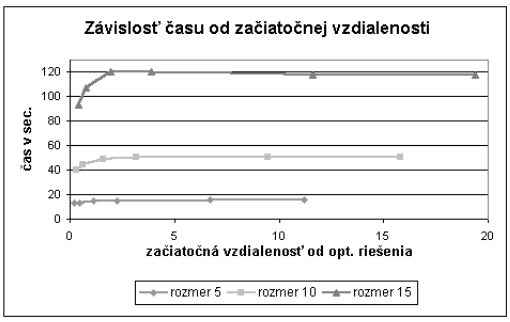
| | 5 | 10 | 15 |
|---|---|---|---|
| 0,1 | 7,053 | 10,503 | 13,331 |
| 0,2 | 7,873 | 12,257 | 15,787 |
| 0,5 | 8,748 | 14,093 | 18,306 |
| 1 | 9,416 | 15,243 | 19,941 |
| 3 | 10,056 | 15,486 | 20,29 |
| 5 | 10,607 | 15,93 | 20,32 |

Tab.5.2 : priemerné iteráce

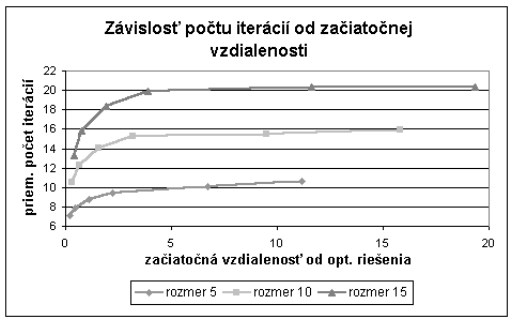
| | 5 | 10 | 15 |
|---|---|---|---|
| 0,1 | 12,85 | 39,82 | 93,38 |
| 0,2 | 13,52 | 44,93 | 107,27 |
| 0,5 | 15,1 | 48,82 | 120,56 |
| 1 | 15,16 | 50,64 | 120,34 |
| 3 | 15,93 | 50,58 | 117,87 |
| 5 | 15,76 | 50,37 | 118,04 |

Tab.5.3 : celkový čas

Pre lepšiu názornosť zobrazíme údaje z tabuliek Tab.5.2 a Tab.5.3 na grafoch:




Obr.5.4




Obr.5.5

Na osi $x$ je vzdialenosť štartovacieho bodu $\mathbf{x^0}$ od $\mathbf{x}^*$ (určí sa nasledovne $\|\mathbf{x^0} - \mathbf{x}^*\| = ODCH\sqrt{N}$ ). Údaje pre každý rozmer sú na grafoch znázornené jednou čiarou. Vidno, že aj priemerné iterácie, aj čas rastú so vzdialenosťou, no od určitého okamihu sa tento rast veľmi spomalí.

## 5.5 Záver

Algoritmus regularizačnej newtonovej metódy sme testovali len na jednom type úloh o komplementarite (s konkrétnym zobrazením $\mathbf{F}$). Tento algoritmus je možné testovať aj pre iné typy zobrazenia $\mathbf{F}$. V našom programe stačí len zmeniť príslušné funkcie a generátor úloh prispôsobiť novému typu zobrazenia $\mathbf{F}$.

Výsledky našich numerických experimentov potvrdili, že uvedený algoritmus pracuje dobre pre testovaný typ zobrazenia $\mathbf{F}$. Toto zobrazenie bolo $P_0$ zobrazenie, a teda podľa teórie uvedenej v [2] mal algoritmus nájsť riešenie v každom prípade, čo sa potvrdilo.

# Literatúra


[1] Ferris, M. C., Pang J.S., *Engineering and Economic Applications of Complementarity Problems*, SIAM Rev. Vol. 39, No. 4, pp. 669 - 713, December 1997.

[2] D. Sun, *A Regularizaton Newton Method for Solving Nonlinear Complementarity Problems*, Applied Mathematics & Optimization Vol. 40, pp. 315 - 339, 1999.

[3] C. Facchinei a F. Kanzow, *Beyond Monotonicity in Regularization Methods for Nonlinear Complementarity Problems*, SIAM Journal on Control and Oprimization Vol. 37, No. 4, pp. 1150 - 1161, 1999.

[4] A. Fischer, *A Special Newton-type Optimization Method*, Optimization Vol. 24, pp. 269 - 284, 1992.

# Príloha 1

## Ako spustiť program ?
## Užívateľská príručka.

Pred spustením programu je potrebné súbor *riesitel.exe* nakopírovať na disk. Počas svojho behu si program ukladá vygenerované matice na disk ( *ulohy_1.txt* ), do toho istého adresára v ktorom sa nachádza.

Program sa spúšťa z príkazového riadku príkazom *riesitel.exe N*, kde miesto parametra *N* napíšeme prirodzené číslo v rozsahu od 2 do 30 (napríklad *riesitel.exe 8*). Toto číslo udáva rozmer úloh, ktoré bude program riešiť. Ak parameter *N* nezadáme, použije sa prednastavená hodnota N = 5. Po spustení sa vypočíta 1000 úloh daného rozmeru a počas behu programu sa po každej vyriešenej úlohe objaví na obrazovke jedna bodka. Na koniec sa na obrazovku vypíšu výsledky a program čaká na stlačenie klávesy *Enter*.

# Stručný popis jednotlivých modulov programu.

Náš program sa skladá z niekoľkých modulov: *generátor*, *funkcie*, *RegNewt* a *riešiteľ*, z ktorých každý pozostáva z niekoľkých funkcií. Modul *generátor.cpp* obsahuje funkcie potrebné na generovanie matíc a vektorov a základnú prácu s nimi. Modul *funkcie.cpp* obsahuje všetky pomocné funkcie a funkcie definované v kapitole 4. Modul *RegNewt.cpp* obsahuje samotnú regularizačnú newtonovu metódu na riešenie úloh o komplementarite. Modul *riesitel.cpp* obsahuje hlavné telo programu, ktorý vyrieši sériu 1000 náhodne vygenerovaných úloh a vypíše výsledky. Nasleduje kompletný zoznam funkcií uvedených modulov s ich krátkymi popismi:

## Funkcie modulu *generator.cpp*

- *vytvor_maticu* - funkcia alokuje miesto pre maticu daných rozmerov
- *zrus_maticu* - uvoľní miesto po danej matici
- *generuj_matice* - vygeneruje dve náhodné, kladne definitné matice o rozmeroch $N \times N$
- *matica_kladne_def* - z matice $A$ vytvorí kladne definitnú maticu $(A^TA + diag\{10, 10, \ldots 10\})$
- *vytvor_x_opt* - vygeneruje vektor optimálneho riešenia
- *zapis_matice* - zapíše na disk do suboru *uloha_1.txt* dve matice
- *citaj_matice* - načíta matice zo súboru

## Funkcie modulu *funkcie.cpp*

- *funkcia_F* - zobrazenie (5.1) bez vektora $\mathbf{d}$

$$\mathbf{F}(\mathbf{x}) = \frac{1}{\|A\|_F^2}(\mathbf{x}^TA\mathbf{x})A\mathbf{x} + B\mathbf{x}$$

- *funkcia_F_eps* - zobrazenie $\mathbf{F}_\varepsilon$ definované vzťahom (4.1)

$$F_{\varepsilon,i}(\mathbf{x}) = \frac{1}{\|A\|_F^2}(\mathbf{x}^TA\mathbf{x})(A\mathbf{x})_i + (B\mathbf{x})_i + d_i + \varepsilon x_i$$

- *jakobian_F* - vytvorí Jakobián zobrazenia $\mathbf{F}$ (5.9)

$$\partial\mathbf{F}(\mathbf{x}) = \frac{1}{\|A\|_F^2}\left(2(A\mathbf{x})(A\mathbf{x})^T + (\mathbf{x}^TA\mathbf{x})A\right) + B$$

- *funkcia_Fi* - Fisherova funkcia (4.2)

$$\Psi(a,b) = \sqrt{a^2+b^2} - (a+b)$$

- *funkcia_G* - zobrazenie (4.4)

$$G_i(\varepsilon, \mathbf{x}) := \Psi(x_i, \mathbf{F}_{\varepsilon,i}(\mathbf{x}))$$

- *funkcia_f* - funkcia (4.7)

$$f(\mathbf{x}) = \sum_{i=1}^{n} G_i^2(\mathbf{x}) + \varepsilon^2$$

- *funkcia_beta* - funkcia $\beta$ (4.8)

$$\beta(\mathbf{z}) := \gamma \min\{1, f(\mathbf{z})\}$$

## Funkcie modulu *RegNewt.cpp*

- *QRrozklad* - vyrieši zadanú sústavu lineárnych rovníc metódou QR rozkladu
- *find_W* - určí $W_k$
- *find_delta_z* - určí smer $\triangle z^k$
- *find_l* - určí $l_k$ pre úpravu dĺžky kroku
- *find_f_k* - podporná funkcia pre *find_W*
- *RegularizationNewton* - funkcia, ktorá obsahuje celý algoritmus regularizačnej newtonovej metódy a rieši úlohu o komplementarite

Hlavičkový súbor *parametre.h* obsahuje všetky parametre použité v našom programe.

## Modul *riesitel.cpp*

Tento modul obsahuje len hlavné telo programu a nie je ďalej delený na funkcie. Ako prílohu prikladáme aj CD médium s programom *riesitel.exe*, takisto aj všetkými uvedenými modulmi.

# Zdrojový kód programu.

## Súbor *funkcie.cpp*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include "generator.h"
#include "parametre.h"
#include "funkcie.h"
#include "Regnewt.h"

/*****************************************/
/*                                       */
/*         POMOCNE FUNKCIE               */
/*                                       */
/*****************************************/

/*   prva cast funkcie F (bez vektora D)
void funkcia_F(double ***mat,double *x,double *y)
{
    int i,j;
    double frob=0, frob1, a0=0, *a1, *a2, a_max, a_max1;

    a1=(double*)malloc(sizeof(double)*N);
    a2=(double*)malloc(sizeof(double)*N);

    a_max = 0;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            if(fabs(mat[0][i][j]) > a_max) a_max= fabs(mat[0][i][j]);

    a_max1=1/a_max;

    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            frob+= (mat[0][i][j]*a_max1)*(mat[0][i][j]*a_max1);
    frob = a_max*sqrt(frob);
    frob1 = 1/frob;

    for(i=0;i<N;i++)
    {
        a1[i] = 0;
        a2[i] = 0;
```

```
    }

    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
        {
           a1[i] += mat[0][i][j]*x[j];
           a2[i] += mat[1][i][j]*x[j];
        }

    for(i=0;i<N;i++) a1[i] *= frob1;
    a0=0;

    for(i=0;i<N;i++)
        a0+=x[i]*a1[i];

    for(i=0;i<N;i++)
        y[i] = a0*a1[i] +a2[i];

    free(a1);
    free(a2);
}

/*
 *        funkcia_F_eps vysledok vlozi do y
 *         F_eps(x) := F(x) + D + epsilon*x
*/
void funkcia_F_eps(double *x, double epsilon,
                   double ***uloha, double *D, double *y)
{
    int i;
    double *y_temp;

    y_temp=(double*)malloc(N*sizeof(double));
    funkcia_F(uloha,x,y_temp);

    for(i=0;i<N;i++)
        y[i] = y_temp[i]+D[i]+ epsilon*x[i];

    free(y_temp);
}

/*
 *   jakobian_F vrati vysledok do matice y
 *   derivacia funkcie F podla x
*/
```

```
void jakobian_F( double *x, double ***uloha, double *D, double **y)
{
   int i,j;

   double *a_i,a=0,frob=0, frob1, a_max, a_max1;
   a_i=(double*)malloc(sizeof(double)*N);

   a_max = 0;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            if(fabs(uloha[0][i][j]) > a_max) a_max= fabs(uloha[0][i][j]);
    a_max1=1/a_max;

    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            frob+= (uloha[0][i][j]*a_max1)*(uloha[0][i][j]*a_max1);
    frob = a_max*sqrt(frob);
    frob1 = 1/frob;

    for(i=0;i<N;i++)
      a_i[i] = 0;

    for(i=0;i<N;i++)
      for(j=0;j<N;j++)
         a_i[i]+=x[j]*uloha[0][i][j];
    for(i=0;i<N;i++)
      a_i[i]*=frob1;

   a=0;
   for(i=0;i<N;i++)
      a+=x[i]*a_i[i];

   for(i=0;i<N;i++)
      for(j=0;j<N;j++)
         y[i][j] = 2*a_i[i]*a_i[j] + a*uloha[0][i][j]*frob1 +
                   uloha[1][i][j];
   free(a_i);
}

/*
 *      funkcia_Fi    vysledok vlozi do y
 *
 *      Fi(a,b) pouziva  Fischerovu transformaciu,
 *       je definovana nasledovne
 *
```

```
 *      Fi(a,b) =  sqrt(a^2+b^2) - (a+b)
*/
void funkcia_Fi(double *a, double *b, double *y)
{
    int i;

    for( i=0; i<N; i++)
      y[i] = sqrt(a[i]*a[i] +b[i]*b[i]) - (a[i]+b[i]);
}

/*
 *     funkcia_G   vysledok vlozi do y
 *     G_i(epsilon,x) := Fi(x_i, F_epsilon,i(x) )
 *     vstupom je x a epsilon ( netreba F_eps(x) )
*/
void funkcia_G( double *x, double epsilon, double ***uloha,
                double *D, double *y)
{
    double *z;
    z = (double*)malloc(N*sizeof(double));
    funkcia_F_eps(x,epsilon,uloha,D,z);
    funkcia_Fi( x, z, y);
    free(z);
}

/*
 *             funkcia_f vrati sum(G_k^2)+epsilon^2
 *          vstup staci x_k, epsilon_k  (nie z_k)
 *
 *          ||H(z)||^2  = epsilon^2 + ||G(z)||^2
 */
double funkcia_f( double *x, double epsilon, double ***uloha,
                  double *D)
{
    int i;
    double temp;
    double *z;
    z= (double*)malloc(N*sizeof(double));
    temp = 0;
    funkcia_G(x,epsilon,uloha,D,z);
    for( i=0; i<N; i++)
        temp+=z[i]*z[i];
    temp+=epsilon*epsilon;
    free(z);
    return(temp);
```

```
}

/*
 *       funkcia_beta vrati GAMA*min(1,f(z_k)^T)
*/
double funkcia_beta(double *x_k,double epsilon_k, double ***uloha,
                    double *D)
{
    double temp,f_k;
    f_k=funkcia_f(x_k,epsilon_k,uloha,D);
    temp = 1<f_k?1:f_k;
    temp*=GAMA;
    return(temp);
}
```

## Súbor *funkcie.h*

```
#ifndef _FUNKCIE_H
#define _FUNKCIE_H

void funkcia_F(double ***mat,double *x,double *y);
void funkcia_F_eps(double *x, double epsilon,
                   double ***uloha, double *D, double *y);
void jakobian_F( double *x, double ***uloha, double *D, double **y);

void funkcia_Fi(double *a, double *b, double *y);
void funkcia_G( double *x, double epsilon, double ***uloha,
                double *D, double *y);
double funkcia_f( double *x, double epsilon, double ***uloha,
                  double *D);
double funkcia_beta(double *x_k,double epsilon_k, double ***uloha,
                    double *D);

#endif //_FUNKCIE_H
```

**Súbor** *generator.cpp*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

#include "generator.h"
#include "parametre.h"
#include "funkcie.h"
#include "Regnewt.h"


/*****************************************************/
/*                                                   */
/*               GENERATOR ULOH                      */
/*                                                   */
/*****************************************************/

double **vytvor_maticu(int m,int n)
{
 int i,j;
 double **mat;
 mat=(double**)malloc(sizeof(double*)*m);
 for(i=0;i<m;i++) mat[i]=(double*)malloc(sizeof(double)*n);
 for(i=0;i<m;i++)
    for(j=0;j<n;j++) mat[i][j]=0;
 return mat;
}

void zrus_maticu(double **mat,int m,int n)
{
 int i;
 for(i=0;i<m;i++) free(mat[i]);
 i=n;
 free(mat);
}

void citaj_matice(int n, int num, double ***mat)
{
    int i,j,k;
    FILE *file;
    char name[40];

    int ***mat_temp;
```

```
    mat_temp = (int***)malloc(sizeof(int**)*2);
    mat_temp[0] = (int**)malloc(sizeof(int*)*n);
    for(i=0;i<n;i++)
        mat_temp[0][i] = (int*)malloc(sizeof(int)*n);
    mat_temp[1] = (int**)malloc(sizeof(int*)*n);
    for(i=0;i<n;i++)
        mat_temp[1][i] = (int*)malloc(sizeof(int)*n);

    sprintf(name,"uloha_%d.txt",num);

    if((file = fopen(name,"r"))!= NULL)
    {
     for(i=0;i<2;i++)
     {
      for(j=0;j<n;j++)
      {
        for(k=0;k<n;k++)  fscanf(file,"%5d ",&mat_temp[i][j][k]);
        fscanf(file,"\n");
      }
      fscanf(file,"\n");
     }
     fclose(file);
    }
    else printf("nepodarilo sa otvorit subor %s pre citanie",name);

    for(i=0;i<2;i++)
       for(j=0;j<n;j++)
          for(k=0;k<n;k++)
              mat[i][j][k]=(double)mat_temp[i][j][k];

    for(i=0;i<n;i++)
    {
        free(mat_temp[0][i]);
        free(mat_temp[1][i]);
    }
    free(mat_temp[0]);
    free(mat_temp[1]);
}

/*      zapise 2 matice o rozmere n x n do suboru
*
*        s nazvom     uloha_num.txt
*/
void zapis_matice(double ***mat,int n, int num)
```

```
{
    int i, j,k;
    FILE *file;
    char name[20];
    int ***mat_temp;

    mat_temp = (int***)malloc(sizeof(int**)*2);
    mat_temp[0] = (int**)malloc(sizeof(int*)*n);
    for(i=0;i<n;i++)
        mat_temp[0][i] = (int*)malloc(sizeof(int)*n);
    mat_temp[1] = (int**)malloc(sizeof(int*)*n);
    for(i=0;i<n;i++)
        mat_temp[1][i] = (int*)malloc(sizeof(int)*n);

    sprintf(name,"uloha_%d.txt",num);

    for(i=0;i<2;i++)
       for(j=0;j<n;j++)
          for(k=0;k<n;k++)
              mat_temp[i][j][k]=(int)mat[i][j][k];

    if( (file = fopen( name,"w" )) != NULL)
    {
     for(i=0;i<2;i++)
     {
       for(j=0;j<n;j++)
       {
         for(k=0;k<n;k++)
           fprintf(file,"%5d ",mat_temp[i][j][k]);
         fprintf(file,"\n");
       }
       fprintf(file,"\n");
     }
     fclose(file);
    }
    else printf("nepodarilo sa otvorit subor %s pre zapis",name);

    for(i=0;i<n;i++)
    {
        free(mat_temp[0][i]);
        free(mat_temp[1][i]);
    }
    free(mat_temp[0]);
    free(mat_temp[1]);
}
```

```
/*
*       vytvori kladne definitnu maticu
*    (z uz nadefinovanej matice A -> A^T A
*/
void matica_kladne_def(double **mat,int n)
{
    int i,j,k;
    double **m_vys;

    m_vys = vytvor_maticu(n,n);
    for(i=0;i<n;i++)
      for(j=0;j<n;j++) m_vys[i][j]=0;
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
        for(k=0;k<n;k++)
           m_vys[i][j]+=mat[k][i]*mat[k][j];

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            mat[i][j]=m_vys[i][j]+10;

    zrus_maticu(m_vys,n,n);
}

/*
*   vygeneruje dve kladne definitne matice o rozmeroch n x n
*           a zapise do suboru uloha_cislo.txt
*/
void generuj_matice(int n,int cislo)
{
    int i,j,k;
    double ***uloha;
    double znamienko;

    uloha=(double***)malloc(2*sizeof(double**));
    for(i=0;i<2;i++)
       uloha[i]=vytvor_maticu(n,n);

    for(i=0;i<2;i++)
        for(j=0;j<n;j++)
            for(k=0;k<n;k++)
            {
                if(k<j) znamienko = -1;
                else znamienko = 1;
```

```
                uloha[i][j][k]=znamienko*(rand()%20);
            }
    for(i=0;i<2;i++)
        matica_kladne_def(uloha[i],n);

    zapis_matice(uloha,n,cislo);
    for(i=0;i<2;i++)
        zrus_maticu(uloha[i],n,n);
    free(uloha);
}

/*
*        dogeneruje vektor D, tak, aby uloha mala riesenie x_opt
*/
void vytvor_D(int n, double ***mat,double *x_opt,
              double *D,double *y_opt)
{
    int i;
    double *y;

    y = (double *)malloc(n*sizeof(double));
    funkcia_F(mat,x_opt,y);

    for(i=0;i<n;i++)
    {
        if(x_opt[i]>0)
        {
            y_opt[i]=0;
            D[i]=-y[i];
        }
        else
        {
            y_opt[i] = 1;
            D[i] = 1 - y[i];
        }
    }
    free(y);
}

/*
* vygeneruje vektor, ktory obsahuje nuly a jednotky
*/
void vytvor_x_opt(double *x_opt, int n)
{
  int i;
```

```
    for(i=0;i<n;i++)
        x_opt[i] = rand()%2;
}
```

### Súbor *generator.h*

```
#ifndef _GENERATOR_H
#define _GENERATOR_H

void generuj_matice(int n,int cislo_ulohy);
void matica_kladne_def(double **mat,int n);

void zapis_matice(double ***mat,int n, int num);
void citaj_matice(int n, int num, double ***mat);
void vytvor_D(int n, double ***mat,double *x_opt,
              double *D,double *y_opt);

void vytvor_x_opt(double *x_opt, int n);

double **vytvor_maticu(int m,int n);
void zrus_maticu(double **mat,int m,int n);

#endif //_GENERATOR_H
```

**Súbor** *RegNewt.cpp*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#include "RegNewt.h"
#include "parametre.h"
#include "funkcie.h"


/***************************************/
/*  QR rozklad - riesenie systemu      */
/***************************************/

void QRrozklad(double **mat, int n, double *y)
{
    int i,j,k, n1 = n+1;
    double aa, s, s_inv, alfa, alfa_inv, sum, xx;

    for(j=0; j<n;j++)
    {
        s=0;
        for(i=j;i<n;i++)
        {
            aa = fabs(mat[i][j]);
            if(aa>s) s=aa;
        }
        s_inv = 1/s;
        sum=0;
        for(i=j;i<n;i++)
            sum+=(mat[i][j]*s_inv)*(mat[i][j]*s_inv);
        s = s* sqrt(sum);
        if (mat[j][j] >0) s= -s;
        mat[j][j] = mat[j][j] - s;
        alfa = s* mat[j][j];
        alfa_inv = 1/alfa;
        for(k=j+1; k<n1; k++)
        {
            sum=0;
            for(i=j;i<n;i++)
                sum += mat[i][j]*mat[i][k];
            sum*= alfa_inv;
```

```
            for(i=j;i<n;i++)
                mat[i][k]= mat[i][k] + sum*mat[i][j];
        }
        mat[j][j] = s;
    }

// riesenie sustavy danej trojuholnikovou maticou

    y[n-1] = mat[n-1][n] / mat[n-1][n-1];
    for(i=n-2;i>=0; i--)
    {
        xx = mat[i][n1-1];
        for(j = i+1; j<n; j++)
            xx = xx - mat[i][j]*y[j];
        y[i] = xx/mat[i][i];
    }

}

/*****************************************/
/*                                       */
/*   REGULARIZACNA NEWTONOVA METODA      */
/*                                       */
/*****************************************/

double find_W(double W, double *f_k )
{
    double f_min;

    f_min=f_k[0]<f_k[1]?f_k[0]:f_k[1];
    f_min=f_min<f_k[2]?f_min:f_k[2];
    f_min=f_min<f_k[3]?f_min:f_k[3];
    f_min=f_min<f_k[4]?f_min:f_k[4];
    f_min=f_min<f_k[5]?f_min:f_k[5];
    if(f_k[5] > f_min)
        return(f_k[5]);
    else
        return(W);
}

/*
*       krok 3 z algoritmu -  urcovanie dlzky l_k
*/

int find_l( double eps_k, double *x_k, double ***uloha,
```

```
       double *D, double d_eps_k, double *d_x_k,double *f, double W)
{
    int l=-1, yes, i;
    double eps_l,f_l,f_k;
    double *x_l;

    x_l=(double*)malloc(N*sizeof(double));

    do{
        yes = 0;
        l=l+1;
        eps_l=eps_k + (double)pow(DELTA,l) * d_eps_k;
        for( i=0;i<N;i++)
            x_l[i]=x_k[i] + (double)pow(DELTA,l)*d_x_k[i];
        f_l=funkcia_f(x_l,eps_l,uloha,D);
        f_k=funkcia_f(x_k,eps_k,uloha,D);

        if( eps_l > funkcia_beta(x_l,eps_l,uloha,D)*EPS )
            yes += 1;
        if( f_l < W-2*SIGMA*(1-GAMA*EPS)*(double)pow(DELTA,l)*f_k )
            yes += 2;

        if(l==L_STOP)
        {
//          printf(" vela iteracii  \n");
            yes=4;
        }
    }while(yes < 3);
    free(x_l);
 return(l);
}

/*
*        hladanie smeru delta_z
*/

void find_delta_z( double *x_k, double epsilon, double ***uloha,
          double *D, double *delta_x_k, double &delta_epsilon_k,
          int k)
{
    int i,j;
    double **V_k, **sustava;
    double beta_k,  delta_eps_k;
    double *x, *r, *q, *base1, *base2;
```

```
V_k = (double**)malloc((N)*sizeof(double*));
for(i=0;i<N; i++)
  V_k[i] = (double*)malloc((N)*sizeof(double));
sustava = (double**)malloc((N)*sizeof(double*));
for( i=0;i<N; i++)
  sustava[i] = (double*)malloc((N+1)*sizeof(double));
q = (double*)malloc(N*sizeof(double));
x = (double*)malloc((N)*sizeof(double));
r = (double*)malloc((N)*sizeof(double));
base1 = (double*)malloc(N*sizeof(double));
base2 = (double*)malloc(N*sizeof(double));

             // riesenie delta_epsilon_k
beta_k= funkcia_beta(x_k,epsilon,uloha,D);
delta_eps_k = beta_k * (double)EPS - epsilon;

funkcia_F_eps(x_k,epsilon,uloha,D,base1);

for(i=0;i<N;i++)
{
   base2[i] = x_k[i]*x_k[i] + base1[i]*base1[i];
   q[i]=(base2[i] - x_k[i]*base1[i]+
        (base1[i]-x_k[i])*sqrt(base2[i]))/(x_k[i]*x_k[i]);
}

for(i=0;i<N;i++)
{
   r[i]= ( x_k[i]*sqrt(base2[i]) -
           sqrt(base2[i])*(base1[i]+sqrt(base2[i]) ) )
          / (x_k[i]*x_k[i]) - x_k[i]*delta_eps_k;
}
        // U_k - lava strana rovnice
jakobian_F(x_k,uloha,D,V_k);

for(i=0; i<N; i++)
    V_k[i][i] += epsilon + q[i];

        // zostavenie rozsirenej matice sustavy
for(i = 0; i< N; i++)
    for(j= 0; j<N; j++)
        sustava[i][j] = V_k[i][j];
for( i = 0; i<N ; i++)
    sustava[i][N] = r[i];

        // riesenie systemu -> delta z_k
```

```
    QRrozklad(sustava, N, x);

            // rozlozenie na d_epsilon a d_x_k
    delta_epsilon_k=delta_eps_k;
    for( i = 0; i<N; i++)
        delta_x_k[i] = x[i];

    for(i=0;i<N;i++)
       free(V_k[i]);
    free(V_k);
    for(i=0;i<N;i++)
       free(sustava[i]);
    free(sustava);
    free(q);
    free(base1);
    free(base2);
    free(x);
    free(r);
}

void posun_f_k(double *f_k, double *x_k, double eps_k,
                            double ***uloha, double *D)
{
    int i;
    double f=funkcia_f(x_k,eps_k,uloha,D);

    for(i=0; i<5; i++)
        f_k[i] = f_k[i+1];
    f_k[5] = f;
}

/*********************************************************/
/*                                                       */
/*      SAMOTNY ALGORITMUS                               */
/*                                                       */
/*      REGULARIZATION NEWTON METHOD FOR SOLVING NCP     */
/*                                                       */
/*********************************************************/

int RegularizationNewton(double *start_point, double ***uloha,
                         double *D, double *vysledok)
{
   int k=0, l_k, i;
   double W = funkcia_f(start_point,EPS,uloha,D);
   double *x_k, *y_k, eps_k = EPS;
```

```
   int END=0;
   double *delta_x_k, delta_eps_k;
   double *f_k;

   x_k=(double*)malloc(N*sizeof(double));
   y_k=(double*)malloc(N*sizeof(double));
   delta_x_k=(double*)malloc(N*sizeof(double));
   f_k = (double*)malloc(6*sizeof(double));

   for(i=0; i<6; i++)
       f_k[i] = W;

   for(i =0; i<N; i++)
       x_k[i] = start_point[i];

   while(1)
   {

//      kontrola, ci uz sme blizko nuly         ..... KROK 1
//  vypis_vektor(x_k,N);
      funkcia_F_eps(x_k,eps_k,uloha,D,y_k);

      if( funkcia_f(x_k,eps_k,uloha,D) < tol_epsilon ) END=1;
      if(k==K_STOP)
      {
//        printf("vysoke k \n");
          END=1;
      }
      if(k!=0)
         for(i=0;i<N;i++)
             if( fabs(x_k[i])< 1e-10 ) END =1;
      if(END==1) break;  // ak sme dostatocne blizko nuly, alebo
                         // je prilis vela iteracii, ukoncime

//       vypocet smeru kroku delta_z_k          ..... KROK 2

      find_delta_z(x_k,eps_k,uloha,D,delta_x_k,delta_eps_k,k);

//       vypocet "upravy" dlzky kroku -> l_k        ..... KROK 3

      l_k = find_l(eps_k,x_k,uloha,D,delta_eps_k,delta_x_k,f_k,W);
      eps_k = eps_k + (double)pow(DELTA,l_k)*delta_eps_k;

//       x^{k+1} := x^{k} + delta^{l_k} * delta_{x_k}
```

```
      for(i =0; i< N; i++)
          x_k[i] = x_k[i] +(double)pow(DELTA,l_k)*delta_x_k[i];

      posun_f_k(f_k, x_k, eps_k,uloha,D);

      k += 1;
   }

   for(i=0;i<N;i++)
       vysledok[i]=x_k[i];
   free(x_k);
   free(delta_x_k);
   free(f_k);
   free(y_k);
   return k;
}
```

## Súbor *RegNewt.h*

```
#ifndef _REGNEWT_H
#define _REGNEWT_H

void find_delta_z( double *x_k, double epsilon, double ***uloha,
          double *D, double *delta_x_k, double &delta_epsilon_k,
          int k);
int find_l( double eps_k, double *x_k, double ***uloha, double *D,
          double d_eps_k, double *d_x_k, double *f, double W);
double find_W(double W, double *f_k );
void posun_f_k(double *f_k, double *x_k, double eps_k,
          double ***uloha, double *D);

void QRrozklad(double **mat, int n, double *y);

int RegularizationNewton(double *start_point, double ***uloha,
          double *D, double *vysledok);

#endif //_REGNEWT_H
```

**Súbor** *riesitel.cpp*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#include "parametre.h"
#include "RegNewt.h"
#include "funkcie.h"
#include "generator.h"

int N=5;

/*********************************************************/
/*                                                       */
/*                    RIESITEL ULOH                      */
/*                                                       */
/*********************************************************/

void main(int argc, char **argv)
{
  int i, kroky;
  double *D;
  double ***uloha;
  double *x_opt, *y_opt, *y_vysl;
  double *x_start, *vysledok;

  if (argc > 1) {
    if (!strcmp(argv[1], "/?")) {
      printf("Pouzitie: riesitel.exe [N] , kde N je rozmer ulohy.\n");
      printf("Prednastavene N (v pripade ze nie ");
      printf("je specifikovane) je 5\n\n");
    }
    else {
      int temp_N = atoi(argv[1]);
      if ((temp_N>=2) && (temp_N<=30)) {
        N = temp_N;
        printf("Uzivatelom zvoleny rozmer ulohy : N=%d\n\n", temp_N);
      }
      else {
        printf("Nepovoleny rozmer ulohy (povoleny rozsah 2 - 30).\n");
        printf("Pouzije sa prednastavena hodnota N=5\n\n");
      }
```

```
    }
  }

  uloha=(double***)malloc(2*sizeof(double**));
  for(i=0;i<2;i++)
     uloha[i]=vytvor_maticu(N,N);
  D=(double*)malloc(N*sizeof(double));
  x_opt=(double*)malloc(N*sizeof(double));
  y_opt=(double*)malloc(N*sizeof(double));
  y_vysl=(double*)malloc(N*sizeof(double));
  x_start=(double*)malloc(N*sizeof(double));
  vysledok=(double*)malloc(N*sizeof(double));

 int pocet, k_min=K_STOP, k_max=0;
 int *kkk;
 double suma, priem_odchylka=0, priem_kroky=0;

 kkk=(int*)malloc(MAX*sizeof(int));
 for(i=0;i<MAX;i++)
    kkk[i]=0;

 srand( (unsigned)time( NULL ) );

 clock_t start, end;
 double cas;

 start = clock();

for(pocet=0;pocet<MAX; pocet++)
{
   generuj_matice(N,1);
   citaj_matice(N,1,uloha);

   do{
     vytvor_x_opt(x_opt,N);
     suma=0;
     for(i=0;i<N;i++)
       suma+=x_opt[i];
   }while((suma<PP)||(suma>N-PP));
   printf(".");

   vytvor_D(N,uloha,x_opt,D,y_opt);

   for(i=0;i<N;i++)
      x_start[i] = x_opt[i]+ ODCHYLKA;
```

```
   kroky=RegularizationNewton(x_start,uloha,D, vysledok);
   funkcia_F_eps(vysledok,0,uloha,D,y_vysl);
   kkk[pocet]=kroky;
}

end = clock();
cas = (double)(end-start)/CLOCKS_PER_SEC;

priem_kroky = 0;
for(i=0;i<MAX;i++)
{
    if(kkk[i]>k_max) k_max = kkk[i];
    if(kkk[i]<k_min) k_min = kkk[i];
    priem_kroky += (double)kkk[i];
}
priem_kroky= (double)priem_kroky/MAX;

 printf("\n\nVYSLEDKY riesenia %d uloh \n", MAX);
 printf("\nROZMER riesenej ulohy: %d",N);
 printf("\ncelkovy CAS : %f sekund",cas);
 printf("\npriem pocet iteracii : %6.3Lf\n", priem_kroky);
 printf("k_min : %d \nk_max : %d\n\n",k_min,k_max);
 printf("\nStlacte ENTER");

 getchar();

 for(i=0;i<2;i++)
    zrus_maticu(uloha[i],N,N);
 free(uloha);
 free(D);
 free(x_opt);
 free(y_opt);
 free(y_vysl);
 free(x_start);
 free(vysledok);
 free(kkk);
}
```

**Súbor** *parametre.h*

```
#ifndef _PARAMETRE_H
#define _PARAMETRE_H

extern int N;                   //  rozmer ulohy
#define PP 1                    // pocet 0 (1) je medzi PP a N - PP
#define ODCHYLKA 0.1

#define MAX 1000

#define K_STOP 500              // aby nebolo vela iteracii pre k
#define L_STOP 10               // aby nebolo vela iteracii pre l

#define tol_epsilon 1e-10       // tolerancia pre porovnavanie s 0

#define DELTA 0.5               // parametre algoritmu1
#define SIGMA 0.00005
#define T 1
#define EPS 1
#define GAMA 0.1

#endif //_PARAMETRE_H
```