

Animace pro předmět Optimalizace

A Set of Animations for the Subject: Optimization

Pavel Reich

Bakalářská práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel REICH**
Osobní číslo: **A10055**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **prezenční**

Téma práce: **Animace pro předmět Optimalizace**

Zásady pro vypracování:

1. Seznamte se s náplní předmětu Optimalizace a zpracujte její sylabus.
2. Zvolte a podrobně uveďte vhodné pojmy, tvrzení, postupy apod., jež doplníte animacemi či videi.
3. Popište program či prostředí, ve kterém budete animace (videa) vytvářet, a práci s ním.
4. Vytvořte scénáře a animace (videa) ke zvoleným tématům.
5. Zhodnoťte vytvořené animace (videa) a další možnosti vylepšení elektronických výukových materiálů z didaktického hlediska.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. MAŇAS, Miroslav. Optimalizační metody. 1. vyd. Praha: SNTL, 1979, 257 s.
2. BARTKO, Róbert. MATLAB II. Vyd. 1. Praha: Vydavatelství VŠCHT Praha, 2008, 226 s. ISBN 9788070806913.
3. VENKATARAMAN, P. Applied Optimization with Matlab Programming. New York: John Wiley & Sons, 2002, 398 s. ISBN 0471349585.
4. PERŮTKA, Karel. MATLAB: základy pro studenty automatizace a informačních technologií. Vyd. 1. Zlín: Ústav řízení procesů, Institut řízení procesů a aplikované informatiky, Fakulta technologická, Univerzita Tomáše Bati ve Zlíně, 2005, 303 s. ISBN 8073183552.
5. CHRAMCOV, Bronislav. Základy práce v prostředí Mathematica. Vyd. 2. Zlín: Univerzita Tomáše Bati, 2006, 122 s. ISBN 8073185105.

Vedoucí bakalářské práce:

Ing. Libor Pekař

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

24. února 2013

Termín odevzdání bakalářské práce:

14. června 2013

Ve Zlíně dne 24. února 2013



prof. Ing. Vladimír Vašek, CSc.

děkan



prof. Ing. Vladimír Vašek, CSc.

Me ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce je zpracování animací pro předmět Optimalizace. V rámci teoretické části je zpracován sylabus předmětu Optimalizace a studenti jsou seznámeni s programy, které obsahují optimalizační toolbox a ve kterých byly dané animace zpracovány. V rámci praktické části jsou popsány jednotlivé animace, které byly vytvořeny a které slouží k osvětlení a lepšímu pochopení probírané látky. Některé animace jsou vytvořeny v prostředí Matlab a některé z nich v programu na vytváření prezentací MS Office PowerPoint.

Klíčová slova: Optimalizace, Analytické metody, Iterační metody, Lineární programování, Dynamické programování, Teorie her

ABSTRACT

The main object of this thesis is to make a set of animations for the subject optimization. In the teorethical part is elaborated syllabus of the course Optimization and students are introduced with software, which contains optimization toolbox and in which the animations was processed. In the practical part are the animations described, which were created and which serve to explain and better understanding of learned subject. Some of animations are created in software Matlab and some of these animations are created in MS Office PowerPoint.

Keywords: Optimization, Analytic methods, Iterative methods, Linear programming, Dynamic programming, Theory of games

Tímto bych chtěl poděkovat Ing. Liboru Pekařovi za rady a čas, který mi věnoval při konzultacích této bakalářské práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 SYLABUS PŘEDMĚTU OPTIMALIZACE	11
1.1 ANALYTICKÉ METODY.....	11
1.1.1 Jednorozměrné případy.....	11
1.1.1.1 Metoda srovnání hodnot funkce.....	11
1.1.1.2 Metoda srovnání znamének první derivace.....	11
1.1.1.3 Metoda vyšších derivací.....	12
1.1.2 Mnohorozměrné případy.....	12
1.1.3 Omezení typu rovnost.....	13
1.1.4 Omezení typu nerovnost.....	13
1.2 ITERAČNÍ METODY.....	14
1.2.1 Komparativní.....	14
1.2.1.1 Jednorozměrné komparativní metody.....	14
1.2.1.2 Mnohorozměrné komparativní metody.....	16
1.2.2 Gradientní.....	18
1.2.2.1 Jednorozměrné gradientní metody.....	18
1.2.2.2 Mnohorozměrné gradientní metody.....	19
1.2.3 Metody náhodného vyhledávání.....	21
1.3 SPECIÁLNÍ METODY.....	22
1.3.1 Lineární programování.....	22
1.3.1.1 Simplexová tabulka.....	23
1.3.1.2 Celočíselné programování.....	24
1.3.2 Dynamické programování.....	24
1.4 TEORIE HER.....	25
2 VÝUKOVÝ SOFTWARE	28
2.1 MATLAB.....	28
2.1.1 Optimalizační Toolbox.....	28
2.2 WOLFRAM MATHEMATICA.....	29
2.2.1 Optimalizace v prostředí Mathematica.....	29
II PRAKTICKÁ ČÁST	31
3 ANIMACE	32
3.1 METODA SIMPLEXŮ.....	32
3.1.1 Pravidelný simplex.....	33
3.1.2 Možnosti zacyklení pravidelného simplexu.....	33
3.1.3 Nepravidelný simplex.....	35
3.2 METODA DLOUHÉHO KROKU.....	36
3.3 METODA PARALELNÍCH TEČEN.....	37
3.4 LINEÁRNÍ PROGRAMOVÁNÍ.....	37

3.5	CELOČÍSELNÉ PROGRAMOVÁNÍ	39
3.6	DYNAMICKÉ PROGRAMOVÁNÍ – SÍŤOVÁ FORMA	39
3.7	DYNAMICKÉ PROGRAMOVÁNÍ – TABULKOVÁ FORMA	40
3.8	TEORIE HER – MATICOVÁ HRA	41
ZÁVĚR		43
ZÁVĚR V ANGLIČTINĚ.....		44
SEZNAM POUŽITÉ LITERATURY.....		45
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....		46
SEZNAM OBRÁZKŮ		47
SEZNAM PŘÍLOH.....		48

ÚVOD

Tato práce se zabývá vypracováním animací pro předmět Optimalizace.

Optimalizaci jako takovou můžeme nadefinovat jako hledání hodnot proměnných, které minimalizují nebo maximalizují cílovou funkci a vyhovují vedlejším podmínkám. [1, s.15]

S optimalizačními úlohami nejrůznějšího druhu se v praxi setkáváme často. Většinou jsou formulovány slovně a řeší se na základě zkušeností a intuice. Takový přístup při současné úrovni rozvoje vědy a techniky již nestačí. Neposkytuje objektivní a vědecké podklady pro řízení a rozhodování. [9]

Jednotlivé animace by měly sloužit k lepšímu pochopení probírané látky studenty. Celá tato práce je koncipována do dvou částí. Teoretická část se zabývá zpracováním sylabu pro tento předmět a popisu jednotlivých softwarů, které můžeme použít pro řešení optimalizačních úloh. V rámci sylabu jsou studenti seznámeni s probíranou látkou v předmětu Optimalizace. Sylabus je rozdělen do 4 částí tak, jak na sebe navazuje právě probíraná látka. V části popisu softwaru budou studenti seznámeni s jednotlivými příkazy, které mohou využít pro řešení optimalizačních úloh v různých oblastech technických aplikací. V praktické části jsou popsány vytvořené animace. Jednotlivé animace byly vytvořeny v programu Matlab, ve kterém jsou zpracovány iterační metody a ostatní animace byly zpracovány v programu na tvorbu prezentací MS Office PowerPoint a demonstrují jak postupovat při výpočtu jednotlivých speciálních metod optimalizace a teorie her.

I. TEORETICKÁ ČÁST

1 SYLABUS PŘEDMĚTU OPTIMALIZACE

1.1 Analytické metody

1.1.1 Jednorozměrné případy

U spojitě funkce jedné proměnné $f(x)$ lze nutné podmínky existence lokálního extrému získat analýzou první derivace $\frac{df}{dx} = f'(x)$. V bodech, kde je derivace prvního řádu nulová, má účelová funkce $f(x)$ lokální extrém (lokální minimum, lokální maximum).

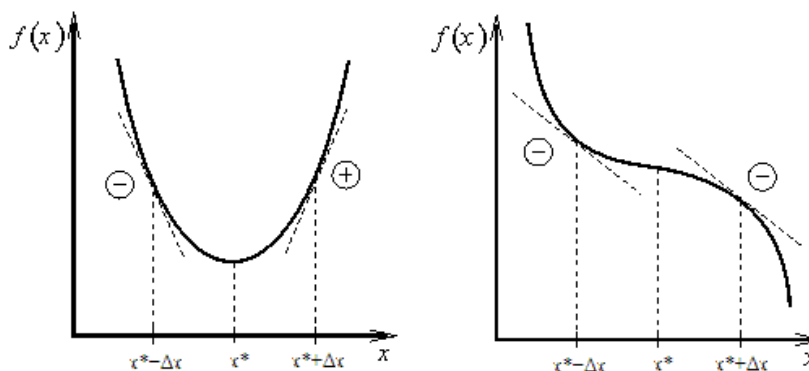
Při určování lokálních extrémů spojitě účelové funkce $f(x)$ nejdříve na základě nutných podmínek určíme „podezřelé body“ a poté pomocí některé z následujících metod zjistíme průběh účelové funkce $f(x)$ v okolí těchto podezřelých bodů. [9]

1.1.1.1 Metoda srovnání hodnot funkce

Tato metoda srovnává hodnoty účelové funkce $f(x)$ v podezřelém bodě x^* s hodnotami v bodech $x^* + \Delta x$ a $x^* - \Delta x$, které jsou tak blízko bodu x^* , že mezi nimi neleží už žádný jiný bod podezřelý z extrému. Platí-li nerovnost $f(x^*) < f(x^* \pm \Delta x)$, pak se v bodě x^* nachází ostré lokální minimum a pokud platí nerovnost $f(x^*) > f(x^* \pm \Delta x)$, tak má účelová funkce $f(x)$ v bodě x^* ostré lokální maximum. [9]

1.1.1.2 Metoda srovnání znamének první derivace

Při této metodě srovnáváme znaménka první derivace $f'(x)$ v bodech $x^* + \Delta x$ a $x^* - \Delta x$. Budou-li znaménka první derivace v těchto bodech stejná, tak v daném bodě x^* není extrém. Pokud se změní znaménko první derivace $f'(x)$ při přechodu z bodu $x^* - \Delta x$ do bodu $x^* + \Delta x$ z (-) na (+), tak má funkce v bodě x^* ostré lokální minimum. Změní-li se znaménko z (+) na (-), má funkce v bodě x^* ostré lokální maximum. [9]



Obr. 1. Metoda srovnání znamének první derivace

1.1.1.3 Metoda vyšších derivací

Tuto metodu lze použít pouze, pokud v podezřelých bodech existují spojité derivace vyšších řádů. Reálné kořeny rovnice $\frac{df(x^*)}{dx} = 0$ vyznačují kritické body, ve kterých vypočítáme druhé derivace. Pokud je ve stacionárním bodě druhá derivace kladná, pak je funkce $f(x)$ ryze konvexní, a má v tomto bodě ostré lokální minimum a pokud je ve stacionárním bodě druhá derivace záporná, pak je funkce $f(x)$ ryze konkávní a má v tomto bodě ostré lokální maximum. Pokud je druhá derivace v nějakém stacionárním bodě nulová a třetí derivace v tomtéž bodě nenulová, tak funkce $f(x)$ nemá v tomto bodě extrém. Takovému stacionárnímu bodu se říká bod inflexní. [9]

1.1.2 Mnohorozměrné případy

Analytické řešení vyhledávání extrému reálné funkce více reálných proměnných je v zásadě zevšeobecněním jednorozměrného případu. Zobecněním první derivace je gradient funkce a zobecněním druhé derivace je Hessova matice

Gradient funkce se definuje jako vektor parciálních derivací:

$$\text{grad } f(x) = \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right) \quad (1)$$

Hessova matice se definuje pomocí druhých parciálních derivací:

$$H = \nabla^2 f(x_1, x_2, \dots, x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \quad (2)$$

Pro určení extrému je třeba spočítat, zda je Hessova matice pozitivně nebo negativně definitní. K tomu je potřeba vyšetřit všechny subdeterminanty Hessovy matice.

Pokud jsou všechny subdeterminanty kladné, tak je pozitivně definitní a v daném bodě je lokální minimum. Pokud subdeterminanty střídají znaménka a první subdeterminant je záporný, tak je Hessova matice negativně definitní, což znamená, že se v daném bodě nachází lokální maximum. Pokud je některý ze subdeterminantů nulový, je Hessova matice semidefinitní a nelze rozhodnout. V ostatních případech extrém neexistuje. [7]

1.1.3 Omezení typu rovnost

Postup řešení je známý jako metoda Lagrangeových multiplikátorů. Úlohu omezení typu rovnost definujeme jako nalezení extrému funkce $f(x_1, x_2, \dots, x_n)$ při podmínkách

$$g_j(x_1, x_2, \dots, x_n) = 0, j=1, 2, \dots, m < n.$$

Pro řešení je nadefinována Lagrangeova funkce:

$$\Phi(x, \lambda) = f(x) + \sum_{j=1}^m \lambda_j g_j \quad (3)$$

kde vektor $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$ se nazývá vektor Lagrangeových multiplikátorů. Body extrému poté nalezneme jako extrémy Lagrangeovy funkce, tj. jako nulové body všech $n+m$ parciálních derivací funkce Φ .

Pokud označíme:

$$\nabla_x \Phi = \left(\frac{\partial \Phi}{\partial x_1}, \frac{\partial \Phi}{\partial x_2}, \dots, \frac{\partial \Phi}{\partial x_n} \right)^T \quad (4)$$

$$\nabla_\lambda \Phi = \left(\frac{\partial \Phi}{\partial \lambda_1}, \frac{\partial \Phi}{\partial \lambda_2}, \dots, \frac{\partial \Phi}{\partial \lambda_m} \right)^T \quad (5)$$

tak existenci extrému udává věta: Necht' funkce f, g_1, g_2, \dots, g_m mají spojité parciální derivace a necht' jsou funkce $\nabla g_j(x)$ lineárně nezávislé. Potom platí, je-li x_0 extrém funkce f při omezení $g_j(x) = 0$, potom existuje $\lambda_0 = (\lambda_{01}, \lambda_{02}, \dots, \lambda_{0m})$ tak, že platí: dle [7]

$$\nabla_x \Phi(x_0, \lambda_0) = \nabla_\lambda \Phi(x_0, \lambda_0) = 0 \quad (6)$$

1.1.4 Omezení typu nerovnost

Úlohu omezení typu nerovnost definujeme jako nalezení extrému funkce $f(x_1, x_2, \dots, x_n)$ při omezeních ve tvaru $g_j(x_1, x_2, \dots, x_n) \geq 0, j=1, 2, \dots, m$ $x_i \geq 0, i=1, 2, \dots, n$.

Stejně jako v případě omezení rovností se definuje Lagrangeova funkce (3), která je doplněna o Kuhn-Tuckerovu větu. Tato věta se též nazývá větou o sedlovém bodě.

Kuhn-Tuckerova věta: Vektor x_0 je optimálním řešením úlohy právě tehdy, když existuje vektor λ_0 takový, že $x_0 \geq 0$ a $\lambda_0 \geq 0$ a pro všechny $x_0 \geq 0, \lambda_0 \geq 0$ platí:

$$\Phi(x, \lambda_0) \leq \Phi(x_0, \lambda_0) \leq \Phi(x_0, \lambda) \quad (7)$$

Věta nemá konstruktivní charakter a pro výpočet se nehodí. Pro diferencovatelné funkce f , g_j se výpočet redukuje na tzv. Kuhn-Tuckerovy lokální podmínky:

Jestliže existují první parciální derivace f a g_j (všechny), pak nalezení maxima je ekvivalentní podmínkám: dle [7]

$$\left(\frac{\partial \Phi}{\partial x_i}\right)_{(x_0, \lambda_0)} \leq 0 \quad x_{0i} \left(\frac{\partial \Phi}{\partial x_i}\right)_{(x_0, \lambda_0)} = 0 \quad i = 1, 2, \dots, n \quad (8)$$

$$\left(\frac{\partial \Phi}{\partial \lambda_j}\right)_{(x_0, \lambda_0)} \geq 0 \quad \lambda_{0j} \left(\frac{\partial \Phi}{\partial \lambda_j}\right)_{(x_0, \lambda_0)} = 0 \quad j = 1, 2, \dots, n \quad (9)$$

1.2 Iterační metody

Iterační metody optimalizace dělíme na tři typy. Jsou to metody komparativní, gradientní a metody náhodného vyhledávání. Matematicky lze iterační metody popsat následujícím vztahem: $x(k+1) = x(k) + \Delta(k)$, kde $x(k)$ značí současný stav, $x(k+1)$ označuje následující vztah a $\Delta(k)$ reprezentuje změnu, ke které dojde během jednoho iteračního kroku. U iteračních metod nám figuruje podmínka $\lim_{n \rightarrow \infty} x(k) = x_{opt}$, což nám zaručuje, že pro n kroků jdoucím k nekonečnu dospějeme k optimálnímu řešení.[7]

1.2.1 Komparativní

Komparativní metody optimalizace se opírají o srovnávání hodnot funkce v různých bodech a nevyžadují tak výpočet derivací. [4, s.54]

1.2.1.1 Jednorozměrné komparativní metody

Fibonacciho metoda

Tato metoda spočívá v hledání minima jednorozměrné funkce $f(x)$ na intervale $\langle a, b \rangle$ pomocí postupného dělení intervalu tak, aby závěrečný interval byl co nejmenší délky a obsahoval minimum funkce. Fibonacciho algoritmus je optimální v tom, že nejúčinněji redukuje velikost intervalu. V každém kroku nám stačí vypočítat jednu funkční hodnotu a v závěrečném kroku je interval rozdělený do dvou stejně velikých intervalů neurčitosti dle [1, s. 31-32]

Základem metody je Fibonacciho posloupnost zadaná pomocí vztahu $F_i = F_{i-1} + F_{i-2}$, kde $F_0 = F_1 = 1$. Několik prvních členů Fibonacciho posloupnosti jsou:

$F_i = \{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \dots\}$, dle [1, s. 32]

Algoritmus Fibonacciho metody je následovný:

1. Volba počtu kroků N , volba intervalu $\langle a, b \rangle$
2. $i=1, \alpha_1=a, \beta_1=b$
3. $\bar{\alpha}_{i+1} = \beta_i - \frac{F_{N-i+1}}{F_{N-i+2}} |\beta_i - \alpha_i|$
4. $\bar{\beta}_{i+1} = \alpha_i + \frac{F_{N-i+1}}{F_{N-i+2}} |\beta_i - \alpha_i|$
5. Výpočet: $f(\bar{\alpha}_{i+1}), f(\bar{\beta}_{i+1})$
6. Porovnání:
 Je-li $f(\bar{\alpha}_{i+1}) < f(\bar{\beta}_{i+1})$, tak $\alpha_{i+1} = \bar{\alpha}_{i+1}, \beta_{i+1} = \beta_i$
 Je-li $f(\bar{\alpha}_{i+1}) \geq f(\bar{\beta}_{i+1})$, tak $\alpha_{i+1} = \alpha_i, \beta_{i+1} = \bar{\beta}_{i+1}$
7. Je-li $i = N$, tak algoritmus ukončíme
8. $i=i+1$ a skočíme na bod 3

Metoda zlatého řezu

Metoda zlatého řezu je velice podobná Fibonacciho metodě. Zlatým řezem je nazývána hodnota limity podílu dvou sousedních členů Fibonacciho posloupnosti:

$$\lim_{i \rightarrow \infty} \frac{F_i}{F_{i+1}} \cong 0.618. \quad (10)$$

Výpočet se oproti Fibonacciho metodě liší v bodě 3 a 4, kde není interval rozdělen dvěma po sobě jdoucími čísly Fibonacciho posloupnosti, ale právě hodnotou zlatého řezu. Druhým rozdílem je volba přesnosti ε namísto počtu kroků N , z čehož vyplývá podmínka: pokud je rozdíl nově vzniklého intervalu menší jako přesnost, tak algoritmus ukončíme.[7]

1.2.1.2 Mnohorozměrné komparativní metody

Metoda pravidelného simplexu

Metoda pravidelného simplexu využívá pro hledání minima nebo maxima funkce simplex. V prostoru R^n je simplex tvořený $n+1$ body, které jsou od sebe stejně vzdálené. V prostoru R^2 je to rovnostranný trojúhelník a v prostoru R^3 je to pravidelný čtyřstěn. [1, s. 45].

Princip metody je jednoduchý. V každém iteračním kroku je potřeba spočítat funkční hodnoty v bodech pravidelného simplexu. Principem je překlápění bodů s „nejhorší“ funkční hodnotou postupně až do minima nebo maxima funkce. „Nejhorší“ funkční hodnotou se rozumí při hledání minima bod s největší funkční hodnotou a v případě hledání maxima bod s nejmenší funkční hodnotou. V případě hledání minima tedy vybereme bod s největší funkční hodnotou a překlápíme ho podle strany, která je mezi ostatními dvěma body. Z nově překlápeného bodu a dvou stávajících bodů se nám vytvoří nový simplex. Takto simplex překlápíme postupně, až dokud nenalezneme hledaný extrém.

Obecně je simplex zadán takto:

$$x^{(1)} = \begin{pmatrix} a_1 \\ \dots \\ a_N \end{pmatrix} \quad x^{(2)} = x^{(1)} + \begin{pmatrix} d+e \\ d \\ \dots \\ d \end{pmatrix} \quad \dots; \quad x^{(N+1)} = \begin{pmatrix} d \\ \dots \\ d \\ d+e \end{pmatrix} + x^{(1)} \quad (11)$$

kde $d = \rho \frac{\sqrt{N+1}-1}{N\sqrt{2}}$, $e = \frac{\rho}{2}$, ρ = délka hrany simplexu dle [7]

Překlápený vrchol má souřadnice:

$$x^{(p)} = x^{(m)} + 2(c - x^{(m)}) = 2c - x^{(m)} \quad (12)$$

kde $c = \frac{1}{N} \sum_{j=1}^{N+1} x^{(j)}$, $x^{(m)}$ je překlápějící se vrchol simplexu dle [7]

Metoda nepravidelného simplexu

Jedná se o upravení metody pravidelného simplexu. Pomocí dalšího výpočtu upravíme velikost simplexu a tak dosáhneme toho, že docílíme výsledku dříve a také přesněji. [7]

Pomocí algoritmu výpočtu lze dosáhnout čtyř možností překlápení simplexu. Jde o reflexi, expanzi, kontrakci a redukci. Princip této metody je podobný jako u pravidelného simplexu. Po sestavení simplexu jsou vypočítány funkční hodnoty v jednotlivých bodech. Daný bod s nejhorší funkční hodnotou je překlápen (vznikne nám bod $x^{(eI)}$).

Vytvoříme nový bod $x^{(e2)}$, který bude novým vrcholem simplexu a podle následujících kritérií rozhodneme o jaký případ se bude jednat:

- a) Pokud je funkční hodnota překlopeného bodu $x^{(e1)}$ lepší jako funkční hodnoty bodů původního simplexu, tak dochází k expanzi a nový bod $x^{(e2)}$ vypočítáme:

$$x^{(e2)} = c + \alpha[x^{(m)} - c], \alpha > 1, x^{(m)} \text{ je bod s "nejhorší" funkční hodnotou}$$

- b) Pokud je funkční hodnota bodu $x^{(e1)}$ menší jako funkční hodnota alespoň jednoho bodu simplexu, pak dochází ke kontrakci a nový bod vypočítáme:

$$x^{(e2)} = c + \beta[x^{(m)} - c], \beta \in \langle 0,1 \rangle, x^{(m)} \text{ je bod s "nejhorší" funkční hodnotou}$$

- c) Pokud je funkční hodnota bodu $x^{(e1)}$ menší jako funkční hodnota bodu s „nejhorší“ funkční hodnotou, tak dochází k redukci (zmenší se celková velikost simplexu) podle vztahu:

$$x^{(j)} = x^{(k)} + \frac{1}{2}[x^{(j)} - x^{(k)}], \forall j \neq k, x^{(k)} \text{ je bod s nejvyšší hodnotou účelové funkce}$$

- d) V případě že není splněna ani jedna s podmínek, dochází k reflexi (překlopení) a $x^{(e2)} = x^{(e1)}$

Pokud je splněna podmínka: $\sum_{j=1}^{N+1} [f(x^{(j)}) - f(x^{(e1)})]^2 < \varepsilon$, kde $\varepsilon > 0$ je předem definovaná tolerance, tak dochází k ukončení výpočtu. Pokud tomu tak není, celý výpočet opakujeme. [7]

Metoda cyklické záměny parametrů

Tato metoda je taktéž nazývána Gauss-Seidlovou metodou. Pro funkci více proměnných $f(x_1, x_2, \dots, x_n)$ se zvolí jedna proměnná a pro tu se jednorozměrnou optimalizací najde částečné optimum. Při tomto kroku jsou ostatní proměnné zafixovány. V dalším kroku se zvolí druhá proměnná a celý postup se opakuje jen pro tuto proměnnou. Tímto způsobem se vypočítá částečné optimum pro všechny proměnné a postup se opakuje znovu. [7]

1.2.2 Gradientní

Při počítání gradientních metod vycházíme z nějakého bodu $x_{(0)}$ a další členy posloupnosti $x_{(1)}$, $x_{(2)}$ vypočítáme podle vzorce:

$$x_{(k+1)} = x_k + \lambda_k \text{grad}f(x_k) \quad (13)$$

Pokud je λ_k konstantní, tak jde o gradientní metody s krátkým krokem a pokud je λ_k proměnné tak jde o gradientní metody s dlouhým krokem [7].

1.2.2.1 Jednorozměrné gradientní metody

Newtonova metoda

U této metody se účelová funkce $f(x)$ v okolí bodu x_k rozvine do Taylorova rozvoje:

$$p(x) \approx f(x_k) + f'(x_k)(x - x_k) + f''(x_k) \frac{(x - x_k)^2}{2} \quad (14)$$

a nový iterační krok se hledá z podmínky $p'(x) = 0$.

Výpočet nového bodu tedy vypočítáme z rovnice:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \quad (15)$$

ze vztahu vyplývá, že na celém rozsahu účelové funkce musí existovat druhá derivace. Celý iterační cyklus se opakuje do té doby, dokud nesplníme předem daný počet iterací nebo si zvolíme citlivost $\varepsilon = x_{k+1} - x_k$. [7]

Metoda regula falsi

Tato metoda je odvozena od Newtonovy metody. Pokud je druhá derivace nahrazena číselným odhadem:

$$f''(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} \quad (16)$$

a pokud tento odhad dosadíme do Newtonovy metody tak dostáváme vztah pro metodu regula falsi:

$$x_{k+1} = x_k - f'(x_k) \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})} \quad (17)$$

Pro zahájení metody potřebuje regula falsi dva startovací body x_0, x_1 dle [7]

1.2.2.2 Mnohorozměrné gradientní metody

Metoda krátkého kroku

Metoda krátkého kroku využívá pro výpočet dalšího kroku vztah:

$$x_{k+1} = x_k + \lambda_k \nabla f(x_k) \quad (18)$$

Tato metoda využívá pro výpočet konstantní velikost λ_k . Výpočet této metody není složitý, ale díky malému kroku je metoda pomalejší oproti metodě dlouhého kroku.

Metoda dlouhého kroku

Metoda dlouhého kroku se liší od metody krátkého kroku tím, že se v každém iteračním kroku mění velikost λ_k . Tato metoda vyžaduje sice více výpočtů, ale vyhledá extrém v menším počtu kroků.

Výpočet λ_k se provádí podle následujícího vztahu:

$$\lambda_k = \max_{\lambda} \left\{ \frac{d}{d\lambda} f(x_k + \lambda \nabla f(x_k)) = 0 \right\} \quad (19)$$

Metoda konjugovaných směrů

Metoda konjugovaných směrů je malou modifikací metody největšího spádu (Metoda dlouhého a krátkého kroku). Nejlepší výsledek získáme pro kvadratickou funkci. Pro kvadratickou funkci o n proměnných nebudeme potřebovat více jak n iterací. [8, s.244]

Algoritmus metody:

1. Je zadána $f(x)$, počáteční bod x_1 , počet iteračních kroků N nebo minimální velikost skoku $\varepsilon > 0$.
2. $i = 1$, označíme $x_i = x_1$, vypočteme gradient $\nabla f(x_i)$ a Hessián $Hf(x_i)$. Označíme $d_i = \nabla f(x_i)$.

3. Výpočet optimální velikosti $\lambda_{opt,i}$ směru d_i podle:

$$\lambda_{opt,i} = \frac{[\nabla f(x_i)]^T d_i}{[d_i]^T Hf(x_i) d_i}$$

4. Nový odhad extrému je:

$$x_{i+1} = x_i + \lambda_{opt,i} d_i$$

5. Vypočteme gradient $\nabla f(x_{i+1})$ a Hessián $Hf(x_{i+1})$

6. Výpočet nového konjugovaného směru:

$$d_{i+1} = \nabla f(x_i) - \frac{[\nabla f(x_{i+1})]^T \text{Hf}(x_{i+1}) d_i}{[d_i]^T \text{Hf}(x_i) d_i} d_i$$

7. Pokud $i=N$ nebo $\|x_{i+1} - x_i\| < \varepsilon \Rightarrow$ Konec. Jinak $i = i + 1$ a skok na 3.

Nevýhodou této metody je výpočet druhých derivací a také, že metoda špatně konverguje pro protáhlé křivky. Tuto vlastnost ale odstraňuje metoda paralelních tečen. [7]

Metoda paralelních tečen

Metoda paralelních tečen vytváří 2 paralelní směry, ze kterých se vybere průměr.

Algoritmus metody: dle [7]

1. $j=0, i=1$, volba počátečního bodu x_0 , volba N
2. $x_1 = x_0 + \lambda \nabla f(x_0)$ a $\max p(\lambda) = f(x_0 + \lambda \nabla f(x_0))$
3. pomocná iterace $y_i = x_i + \lambda_i \nabla f(x_i)$ a $\max q(\lambda_i) = f(x_i + \lambda_i \nabla f(x_i))$
4. $x_{i+1} = x_{i-1} + \delta_i [y_i - x_{i-1}]$ a $\max f[x_{i-1} + \delta_i (y_i - x_{i-1})]$
5. je-li $i < N$, pak $i=i+1$ a skok na krok 3
je-li $i=N$, pak $x_0=x_N, i=1$ a skok na 2

Metoda DFP

Tato metoda Davidon-Fletcher-Powell má základ v algoritmu Newtonovy metody, ve které se namísto Hessovy matice druhých derivací vytváří jen její odhad. Metoda vychází z rovnice:

$$x_{i+1} = x_i + \lambda_i S_i \nabla f(x_i), \text{ kde } S_i \text{ je aproximace Hessovy matice} \quad (20)$$

Výpočet metody DFP lze popsat následujícím algoritmem: dle [7]

1. $i=0$, volba startovacího bodu x_0 a pozitivně definitní matice S_0 , volba tolerance $\varepsilon > 0$
2. Výpočet: $d_i = S_i \nabla f(x_i)$
3. Výpočet: $x_{i+1} = x_i + \lambda_i d_i$, kde λ_i maximalizuje $f(x_i + \lambda_i d_i)$
4. Výpočet $\nabla f(x_{i+1})$
je-li $\sum_{j=1}^N [\nabla f(x_j)]^2 < \varepsilon$, tak algoritmus ukončíme

5. Označíme $v_i = \lambda_i d_i$; $w_i = \nabla f(x_i) - \nabla f(x_{i+1})$ a vypočítáme:

$$S_{i+1} = S_i + \frac{v_i v_i^T}{v_i^T w_i} - \frac{S_i w_i w_i^T S_i}{w_i^T S_i w_i}$$

6. $i=i+1$, skok na 2

Metoda projekce gradientu

Jedná se o úlohu, která bývá formulovaná ekonomicky a analogicky souvisí s lineárním programováním. Tato úloha bývá formulovaná takto:

hledání maxima $f(x_1, x_2, \dots, x_n)$ při omezení $Ax \leq b$ a podmínkou nezápornosti $x \geq 0$ dle [7]

Algoritmus:

1. sestavení projekční matice Q
2. výpočet $q = (QQ^T)^{-1}Q\nabla f(x_k)$
3. zkouška:
 - je-li $q_k < 0$ a $k > 1$, pak vynecháme k -tý řádek matice, sestrojíme matici Q^x a přejdeme na krok 2
 - je-li $q_k \geq 0$ pro všechny k , tak ukončíme algoritmus
4. sestrojíme $P = [I - Q^T(QQ^T)^{-1}Q]$
5. vypočítáme směr $s = P\nabla f(x_k)$
6. výpočet $\tilde{\lambda} = \left\{ \max \lambda; \frac{\partial p}{\partial \lambda} = 0 \text{ a } x_k + \lambda_s \text{ je přípustné} \right\}$
7. $x_{k+1} = x_k + \tilde{\lambda}_s$
8. skok na 1

1.2.3 Metody náhodného vyhledávání

Při metodě náhodného vyhledávání je iterační přírůstek závislý na náhodné veličině. Tyto metody lze rozdělit do 3 skupin:

1. Jednoduché metody – tyto metody jsou prosté a s konstantní strategií
2. Adaptační – určení směru a velikosti se vyhodnocuje podle určité strategie
3. S prvky umělé inteligence – tyto prvky mohou být například genetické, evoluční nebo jiné.

Tyto metody jsou efektivní hlavně u skutečně mnohorozměrných případů a v případech, kdy funkce není zadána analyticky. Jde o maximalizování účelové funkce v n rozměrném prostoru, která je zadána pomocí systému nerovností. Obecné je tento systém zadán následujícími vztahy: $x_{k+1} = x_k + \Delta x_k$ (21)

$$\Delta x_k = a_k \psi[k, \Delta f_k, x_k] \xi_k \quad (22)$$

$$\Delta f_k = f(x_k) - f(x_{k-1}) \quad (23)$$

, kde ψ je funkce vyhodnocení (úspěšnost, neúspěšnost)

ξ je rovnoměrně rozložená náhodná veličina na intervalu $\langle -1; 1 \rangle$

Iterační postup metod popisuje první rovnice. Druhá rovnice popisuje strategii postupu a třetí rovnice se nazývá zpětná diference účelové funkce [3, s.36-37]

1.3 Speciální metody

1.3.1 Lineární programování

Jde o speciální úlohu neklasického vázaného extrému. Charakteristickou vlastností této metody je, že účelová funkce i omezení jsou lineární. Základní formulace této úlohy je :

$$f(x_1, x_2, \dots, x_n) = c^T x \quad (24)$$

$$Ax \leq b \quad (25)$$

Základní klasifikace úloh lineárního programování je pak:

$$\max c^T x \quad Ax \leq b \quad (26)$$

$$\min c^T x \quad Ax \leq b \quad (27)$$

$$\max c^T x \quad Ax \geq b \quad (28)$$

$$\min c^T x \quad Ax \geq b \quad (29)$$

Další variantou může být kombinovaná úloha lineárního programování, kde jsou v omezeních všechny tvary:

$$\sum_{r=1}^n a_{ir} x_r \leq b_i \quad i = 1, \dots, p_1 \quad (30)$$

$$\sum_{r=1}^n a_{jr} x_r = b_j \quad j = p_1 + 1, \dots, p_2 \quad (31)$$

$$\sum_{r=1}^n a_{kr} x_r \geq b_k \quad k = p_2 + 1, \dots, m \quad (32)$$

Postup řešení základní úlohy lineárního programování:

Nejdříve musíme doplnit nerovnice na rovnice tak, že přidáme k nerovnici přídavné proměnné podle schématu:

$$\begin{pmatrix} a_{11} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \dots \\ x_n \\ x_{n+1} \\ \dots \\ x_{n+m} \end{pmatrix} = \begin{pmatrix} b_1 \\ \dots \\ b_m \end{pmatrix} \quad (33)$$

V daném schématu jsou proměnné x_1, \dots, x_n základní proměnné a x_{n+1}, \dots, x_{n+m} jsou přídavné proměnné. Soustava obsahuje m rovnic s $(n+m)$ neznámými a je v kanonickém tvaru. Základním řešením jsou čísla, kde jsou hodnoty základních proměnných rovny hodnotám pravých stran. Další metoda vyhledání řešení je úplná eliminační metoda, kde nalezneme všechny základní řešení pomocí ekvivalentních úprav matice. O řešení úloh platí tyto věty:

1. Vektor je základním přípustným řešením právě tehdy, je-li vrcholem množiny omezení.
2. Existuje-li optimální řešení úlohy lineárního programování, pak existuje i základní přípustné řešení se stejnou hodnotou účelové funkce. [3, s.39-40]

1.3.1.1 Simplexová tabulka

Simplexová tabulka je efektivní způsob jak docílit optimálního řešení. Soustavu rovnic zapíšeme do tabulky podle obrázku č. 2:

x_1	x_2	\dots	x_n	x_{n+1}	\dots	x_{n+m}	omezení
a_{11}	a_{12}	\dots	a_{1n}	1	\dots	0	b_1
\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots
a_{m1}	a_{m2}	\dots	a_{mn}	0	\dots	1	b_m
$-c_1$	$-c_2$	\dots	$-c_m$	0	\dots	0	f

Obr. 2. Simplexová tabulka

Algoritmus řešení úlohy:

1. sestavení simplexové tabulky
2. test optimality, je-li pro všechna $i(-c_i) \geq 0$ pak algoritmus ukončíme
3. výběr klíčového sloupce tak, že vybereme maximální c_j se záporným znaménkem
4. výběr klíčového řádku tak, že vybereme nejmenší nezáporný podíl $\beta_i = \frac{b_i}{a_{ij}}$
5. eliminujeme klíčový prvek
6. skok na bod 2

1.3.1.2 Celočíselné programování

Celočíselné programování navazuje na lineární programování. Navíc obsahuje podmínku, že proměnné nabývají pouze celočíselných hodnot. Mezi nejznámější algoritmus řešení celočíselných úloh patří tzv. metoda sečných nadrovin (Gomoryho metoda rozkladu).

Algoritmus řešení celočíselného programování: dle [7]

1. sestavení simplexové tabulky a řešení úlohy lineárního programování bez ohledu na celočíselné řešení.
2. Ověření, zda je řešení celočíselné. Pokud ano, tak je výpočet ukončen.
3. Rozklad řešení na celé a necelé části. $a_i = [a_i] + r_i$, $b_i = [b_i] + r_i$
4. Výběr největšího neceločíselného zbytku
5. Sestavení nového omezení pomocí zbytků nebázických proměnných

$$-r_1x_1 - r_2x_2 - \dots - r_nx_n + x_{n+1} = -r$$
6. Doplnění simplexové tabulky o omezení
7. Řešení rozšířené simplexové tabulky
8. Odečtení řešení a skok na 2

1.3.2 Dynamické programování

Hlavní myšlenkou dynamického programování je rozklad úloh na podúlohy, které jsou řešeny a jejich dílčí výsledky ukládány pro potencionální použití. Tato metoda je vhodná pro typ úloh, kde se podúlohy opakují a jsou si podobné. Dynamické programování tak patří do skupiny separovatelného programování. [7]

Dynamické programování dělíme na síťovou formu a tabulkovou formu. Řešení takovýchto úloh pomocí Bellmanova principu má vždy dva chody. Jeden přímý a jeden zpětný. [7]

Analytická formulace dynamického programování:

$$\text{extr. } f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n f_i(x_i) \quad (34)$$

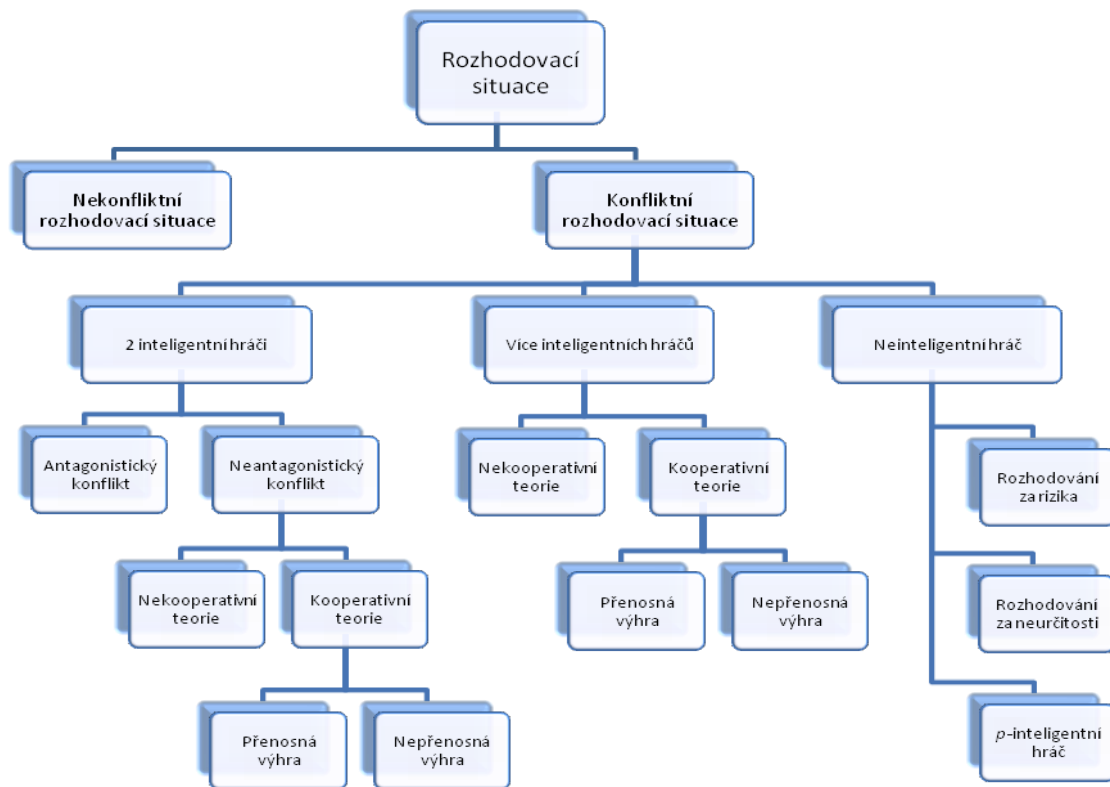
$$\text{při omezeních: } \sum_{i=1}^n x_i \leq b; \quad x_i \geq 0 \quad (35)$$

Řešení úloh dynamického programování se řeší následujícím způsobem:

1. Konstantně krokem h musíme rozdělit b na b_k
2. Def.: $F_1(x_1) = f_1(x_1)$
3. Def.: $F_2(x_2) = \max_{0 \leq x_2 \leq b} \{f_2(x_2) + F_1(b - x_2)\}$
4. Def.: $F_3(x_3) = \max_{0 \leq x_3 \leq b} \{f_3(x_3) + F_2(b - x_3)\}$
5. ...
6. Def.: $F_n(x_n) = \max_{0 \leq x_n \leq b} \{f_n(x_n) + F_{n-1}(b - x_n)\}$
7. Zpětné odečtení optimálních řešení x_n, x_{n-1}, x_1
8. Všechny průběžné výsledky se zapisují do tabulky dynamického programování

1.4 Teorie her

Základní pojetí teorie her spočívá v modelování a studiu konfliktních rozhodovacích situací. Podmínkou je dva a více hráčů, kteří se snaží maximalizovat svou výhru. Rozhodování takových hráčů je buď individuální nebo kolektivní. Výhrou se rozumí např. zisk nebo užitek. Cílem teorie her je nalezení optimální strategie, která zajistí nejvyšší výhru nezávisle na tom, jakou strategii zvolí protihráč. [7]



Obr. 3. Rozhodovací situace teorie her

Je několik pohledů na to, jak rozdělit hry. Hry lze dělit podle počtu hráčů, charakteru výher, podle počtu strategií, inteligenci účastníků nebo typu modelu hry. [7]

Základní matematický model hry v normálním tvaru:

$$\{Q; X_1, X_2, \dots, X_N; M_1(x), M_2(x), \dots, M_N(x)\}$$

kde Q je množina hráčů, X_i je prostor strategií i -tého hráče a funkci M_i nazveme výplatní funkcí i -tého hráče. Jsou to hry, kde první hráč usiluje o co nejvyšší zisk na úkor toho druhého. Matici hry sestavíme z m řádků a n sloupců, kde m je počet strategií prvního hráče a n je počet strategií druhého hráče. Čísla v matici pak udávají, kolik hráč získá nebo popřípadě ztrácí. [7]

Dalším typem her jsou dvoumaticové hry. Při neantagonistickém konfliktu každý z hráčů sleduje své zájmy, které ale nemusí být v konfliktu se zájmy soupeře. Jedná se o hry s nekonztantním součtem a nemůžeme proto odvodit výplatu druhého hráče. Z tohoto důvodu je potřeba pro každého hráče napsat zvlášť matici výplat. [7]

2 VÝUKOVÝ SOFTWARE

V této teoretické části jsou popsány prostředí, se kterými jsou studenti seznámeni ve výuce na FAI, a ve kterých můžeme využívat optimalizační toolbox. První polovina animací je vytvořena v prostředí Matlab a druhá polovina v programu MS Office PowerPoint. Je zde popsán také program Wolfram Mathematica, jelikož byl použit pro srovnání výsledků. Animace, které byly vytvořeny, jsou naprogramovány vykreslováním jednotlivých bodů pomocí jednoduchých funkcí, které jsou popsány v příloze. [P I] MS Office PowerPoint zde není popsán, jelikož je to jednoduchý program na vytváření prezentací a animace v nich jsou vytvořeny pomocí časových přechodů.

2.1 Matlab

Prostředí Matlab lze popsat jako integrované prostředí pro vědeckotechnické výpočty, analýzu a prezentaci dat, návrhy algoritmů, modelování, simulace, měření a zpracování signálů a návrhy řídicích a komunikačních systémů. Tento výpočetní systém se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací ve sféře vědy, výzkumu, průmyslu i vzdělávání. [6, s.8]

Matlab poskytuje nejen grafické a výpočetní nástroje, ale i rozsáhlé knihovny funkcí, které jsou svým rozsahem využitelné prakticky ve všech oblastech lidské činnosti.

Optimalizace je v Matlabu rozčleněna do několika nástrojů, kde základním z nich je Optimalizační Toolbox, který si popíšeme. [1, s.9]

2.1.1 Optimalizační Toolbox

Optimalizační Toolbox přináší funkce a algoritmy pro standardní i rozsáhlé optimalizační úlohy. Tyto algoritmy zahrnují funkce pro lineární i kvadratické programování, nelineární optimalizaci, nelineární metodu nejmenších čtverců, řešení soustav nelineárních rovnic nebo víceúlohovou optimalizaci. [1, s.10]

Zde si popíšeme některé základní funkce, které nám pomůžou při řešení optimalizačních úloh:

Funkce `fminbnd` – vyhledá minimum funkce $F(x)$ jedné proměnné na určitém intervalu

$$\langle x_1, x_2 \rangle$$

Funkce `fmincon` – vyhledá minimum nelineární funkce $F(x)$ s vedlejšími podmínkami

Funkce `fminsearch` – vyhledá minimum nelineární funkce více proměnných bez vedlejších podmínek

Funkce `fsolve` – vyhledá řešení (kořeny) soustavy nelineárních algebraických rovnic ve tvaru $F_{(x)} = 0$

Funkce `fzero` – vyhledá řešení nelineární algebraické rovnice ve tvaru $F_{(x)} = 0$

Funkce `linprog` – řeší problémy lineárního programování s vedlejšími podmínkami

Funkce `lsqlin` – řeší lineární problém nejmenších čtverců s vedlejšími podmínkami

Funkce `lsqnonlin` – řeší nelineární problém nejmenších čtverců (proložení bodů nelineární funkcí)

Funkce `optimget` – vrátí hodnoty volitelných nastavení pro optimalizační funkce

Funkce `optimset` – vytváří nebo edituje volitelnou strukturu parametrů optimalizačních funkcí

Funkce `quadprog` – řeší problémy kvadratického programování s vedlejšími podmínkami

2.2 Wolfram Mathematica

Software Mathematica je prostředí, které zahrnuje nástroje pro numerickou a symbolickou matematiku, grafický a dokumentační systém a zajišťuje pokročilé propojení s dalšími aplikacemi. Klíčovou myšlenkou pro vytvoření tohoto softwaru bylo objevení nového symbolického jazyka. Symbolické programování umožnilo reprezentovat data, funkce, grafy, programy a dokonce i celý dokument jednotně jako symbolický výraz. [2,s.7]

Využití software Mathematica je obrovské. Můžeme jej použít pro zpracování komplexních symbolických výpočtů, výzkum chování modelů a simulací, ilustraci matematických nebo vědeckých konceptů pro studenty nebo také pro provádění odborných prezentací a seminářů. [2, s.10]

2.2.1 Optimalizace v prostředí Mathematica

Prostředí Wolfram Mathematica obsahuje několik funkcí, které nám pomáhají řešit optimalizační metody. Tyto funkce nám pomáhají řešit standardní optimalizační úlohy, lineární programování, nelineární metodu nejmenších čtverců nebo také nelineární optimalizaci [5]

Mezi tyto funkce patří:

Funkce *FindMinimum* – vyhledání lokálního minima funkce

Funkce *FindMaximum* – vyhledání lokálního maxima funkce

Funkce *NMinimize* – vyhledání globálního minima funkce

Funkce *NMaximize* – vyhledání globálního maxima funkce

Funkce *Minimize*, *Maximize* – vyhledání globálního minima, maxima

Funkce *LinearProgramming* – řeší lineární programování s vedlejšími podmínkami

Funkce *LeastSquares* – řeší problém nejmenších čtverců.

Pro naprogramování jednotlivých algoritmů nejsou používány jen tyto optimalizační funkce, ale také jednotlivé funkce pro práci s maticemi, grafy atd. [5]

II. PRAKTICKÁ ČÁST

3 ANIMACE

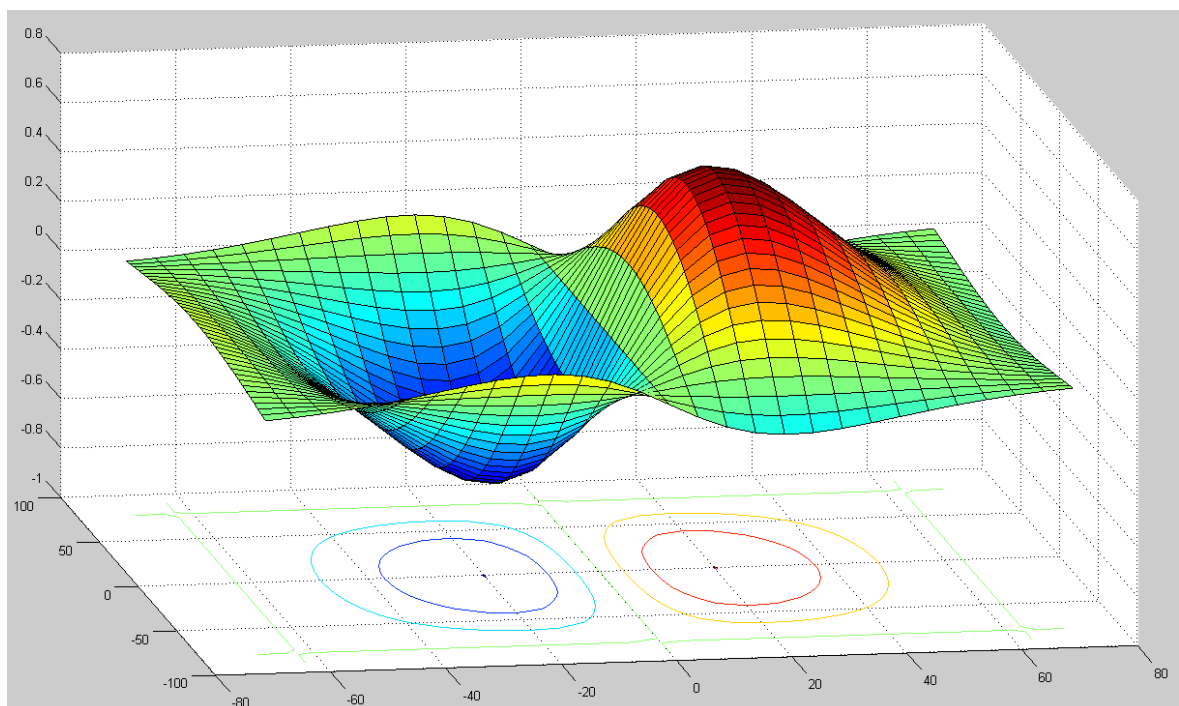
Praktická část se zabývá jednotlivými animacemi, které byly vytvořeny jako podpora k jednotlivým pojmům optimalizace. Animací je celkem 10 a byly vybrány tak, aby osvětlovaly probíranou látku. Prvních 5 animací se zabývají hledáním extrémů funkcí a popisují, jak metody postupují při hledání těchto extrémů. Mezi tyto metody jsem zařadil metodu pravidelného a nepravidelného simplexu, metodu dlouhého kroku, metodu paralelních tečen a popis jak může dojít k zacyklení simplexu. Další animace popisují, jak správně postupovat při výpočtu lineárního a celočíselného programování, dynamického programování a teorie her.

3.1 Metoda simplexů

Metoda pravidelného a nepravidelného simplexu, jak už bylo zmíněno v teoretické části, patří mezi komparativní iterační metody. Obě tyto metody jsem použil na stejné funkci, abych mohl zhodnotit jednotlivé výsledky.

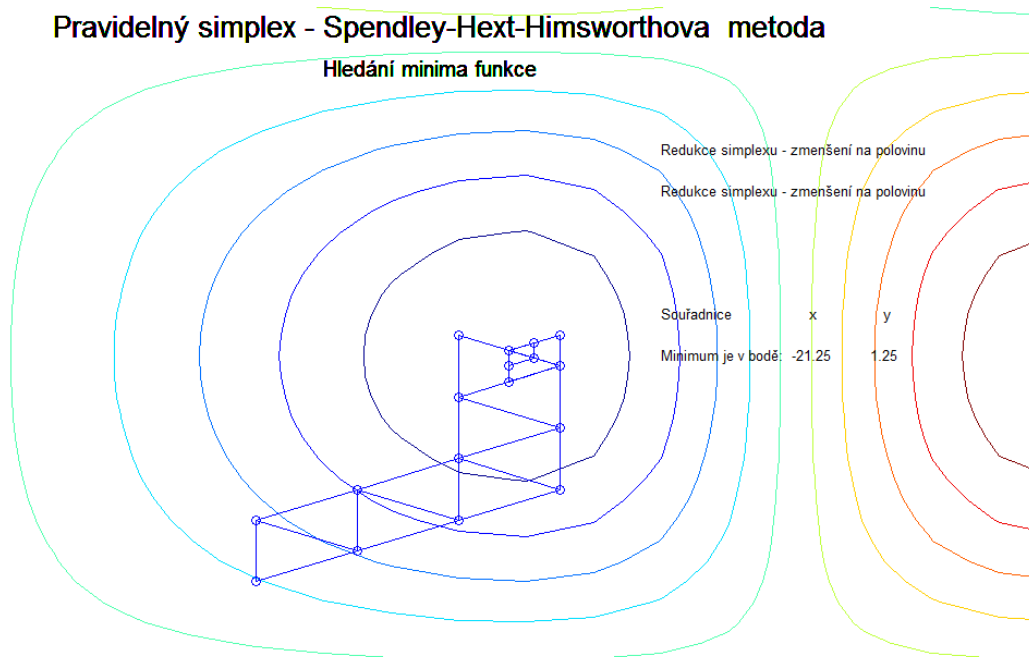
$$f = \frac{\sin\left(\frac{x}{20}\right) * \cos\left(\frac{y}{50}\right)}{\frac{x^2}{1000} + 1} \quad (36)$$

Tato funkce byla zvolena záměrně, protože má opakující se průběh a lze na ní názorně předvést obě simplexové metody. Průběh funkce lze vidět Obr. 4.



Obr. 4. Účelová funkce pro vytvoření animací simplexů

3.1.1 Pravidelný simplex



Obr. 5. Pravidelný simplex

Metoda pravidelného simplexu, kterou jsem naprogramoval je nastavena na 13 iteračních kroků. Na začátku animace je vytvořen počáteční simplex. V dalších krocích se simplex překlápí postupně k hledanému minimu. Tento počet iteračních kroků je zvolen hlavně pro přehlednost. Pravidelný simplex konverguje k hledanému minimu a větší počet iteračních kroků by udělal výslednou animaci nepřehlednou. Výsledné minimum pro tuto funkci o 13-ti iteračních krocích bylo nalezeno v bodě $[-21,25; 1,25]$.

3.1.2 Možnosti zacyklení pravidelného simplexu

U pravidelného simplexu nám můžou nastat dvě situace, které zkomplikují vyhledání extrému. První situace je taková, že jeden z bodů nebyl dlouho vybrán a může dojít k uzavření pravidelného šestiúhelníku. Proto je nutné zmenšit daný simplex na polovinu.

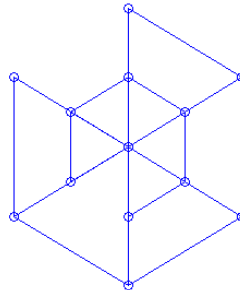
Pravidelný simplex - Spendley-Hext-Himsworthova metoda

Možnosti zacyklení simplexu

1. Jeden z bodů nebyl dlouho vybrán

Jelikož by v dalším kroku simplex uzavřel pravidelný šestiúhelník, tak musíme simplex zmenšit na polovinu

V dalším kroku by se opakoval stejný průběh jako v prvním případě, musíme tedy simplex opět zmenšit na polovinu



Obr. 6. Možnosti zacyklení simplexu – jeden z bodů nebyl dlouho vybrán

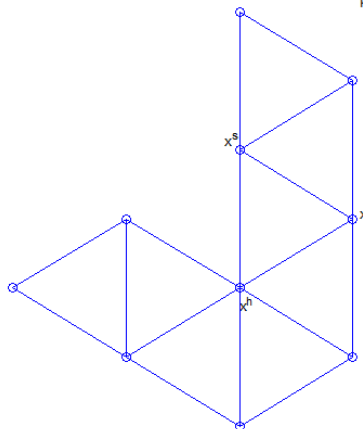
Druhá možnost zacyklení pravidelného simplexu nastane, když se daný simplex začne překlápět zpět do původní polohy. Pro odstranění tohoto problému musíme vybrat bod s druhou „nejhorší“ funkční hodnotou a překlápět simplex podle něj.

Pravidelný simplex - Spendley-Hext-Himsworthova metoda

Možnosti zacyklení simplexu

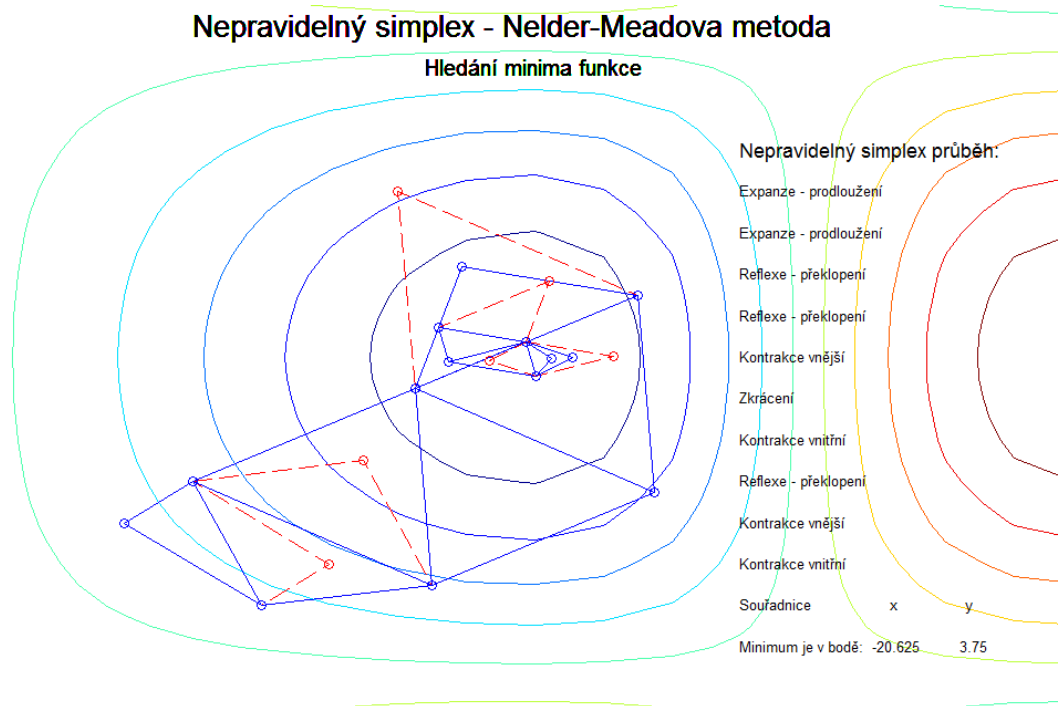
2. Zrcadlení simplexu

V dalším kroku by se simplex překlápěl zpět do původní polohy a celá situace překlápění by se zacyklila, tak musíme vybrat bod, který má druhou nejvyšší funkční hodnotu



Obr. 7. Možnosti zacyklení simplexu – zrcadlení simplexu

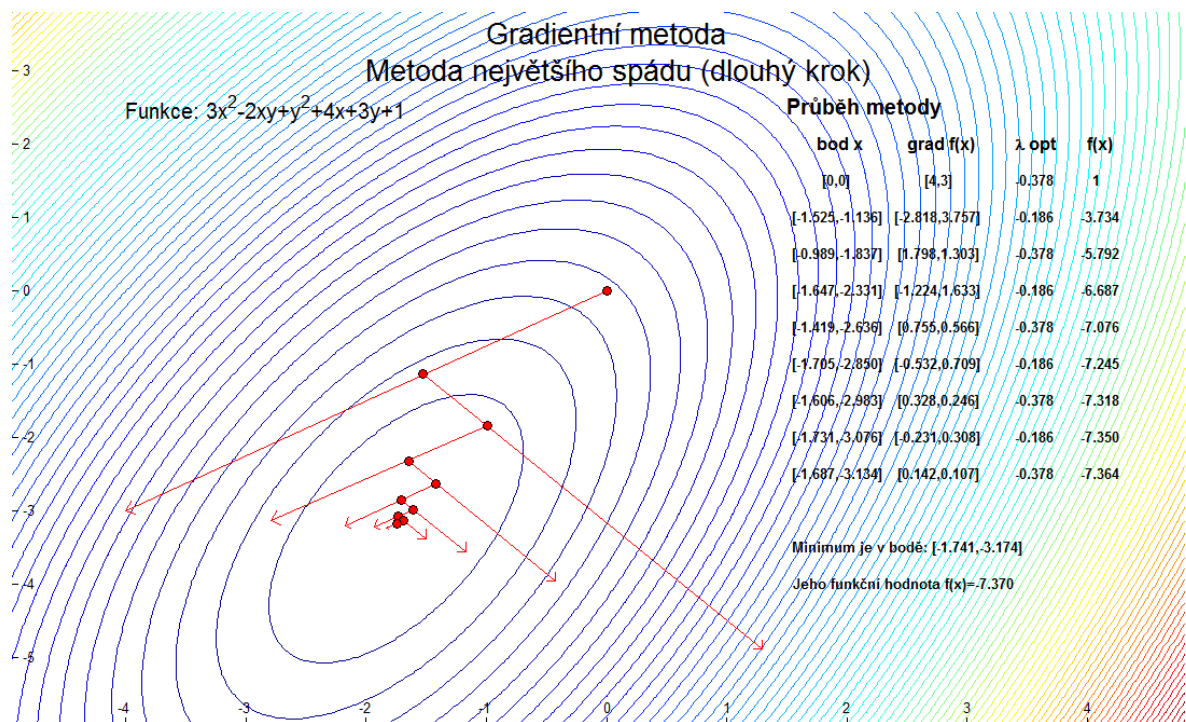
3.1.3 Nepravidelný simplex



Obr. 8. Nepravidelný simplex

Metoda nepravidelného simplexu byla zpracována pouze pro 11 kroků, jelikož by se animace stala opět nepřehlednou. Pokud bychom ale měli srovnat metodu pravidelného a nepravidelného simplexu, tak dojdeme k závěru, že metoda nepravidelného simplexu konverguje k extrému rychleji. Pro stejný počet iteračních kroků jako u pravidelného simplexu (13 iterací), dosáhla metoda nepravidelného simplexu extrému v bodě $[-21,762; -0,517]$. Hledaný extrém je v bodě $[-21,038; -0,405]$. Z tohoto výsledku je patrné, že po třinácti iteracích je blíže metoda nepravidelného simplexu. Funkce, na které je hledaný extrém byla zvolena tak, aby byly vidět všechny možnosti výpočtu simplexu. V prvním kroku animace je vytvořen počáteční simplex, vypočítá se nový (překlopený) bod a podle něj se zvolí jedna ze čtyř strategií. Pokud se nejedná o reflexi (překlopení), tak je reflexe znázorněna červenou přerušovanou čarou, a dochází k výpočtu jiné strategie, která je popsána v pravé části animace. Jak metoda pravidelného, tak metoda nepravidelného simplexu byly vytvořeny pro hledání minima funkce.

3.2 Metoda dlouhého kroku



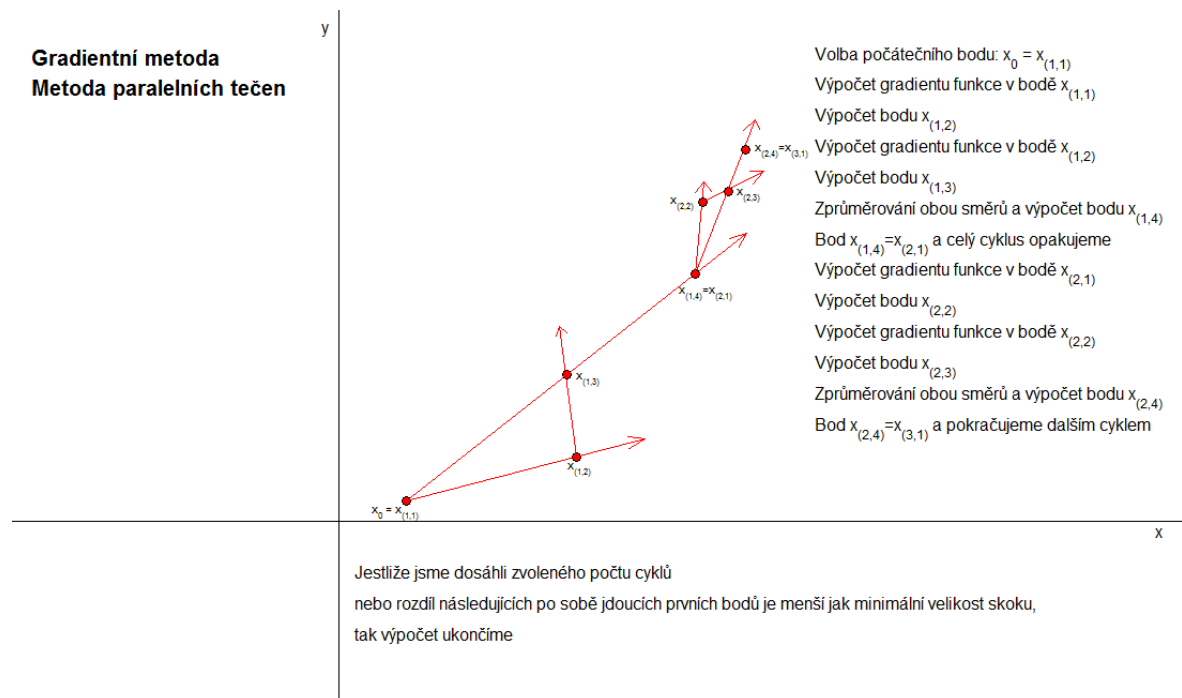
Obr. 9. Metoda největšího spádu (dlouhý krok)

Metoda dlouhého kroku patří mezi iterační gradientní metody. Tato metoda byla vytvořena pro funkci:

$$f = 3x^2 - 2xy + y^2 + 4x + 3y + 1. \quad (37)$$

V každém kroku je spočítán gradient, což je určitý přírůstek a bod ležící na tomto gradientu. Tato animace je zpracována tak, aby v každém kroku bylo patrné, jak tato metoda postupuje k hledanému extrému. Jednotlivé vypočítané hodnoty se navíc zobrazují v pravé části grafu. Animace postupuje vždy vytvořením bodu a popisem, jaké má daný bod souřadnice a jakou má funkční hodnotu v daném bodě. V dalším kroku vypočítá gradient a vypočítaný gradient zaznačí do animace. Tento postup se opakuje, dokud metoda dlouhého kroku nenalezne v tomto případě minimum. Pro přehlednost je tato metoda vytvořena pro 9 iteračních kroků. Pokud bychom chtěli minima dosáhnout co nejpřesněji, použili bychom více iteračních kroků, ale výsledná animace by se v okolí minima pak stala nepřehlednou.

3.3 Metoda paralelních tečen



Obr. 10. Metoda paralelních tečen

Výpočet této metody není demonstrován na žádném konkrétním příkladě, ale je zpracován názorně, jak metoda paralelních tečen hledá daný extrém. Jak hledá metoda paralelních tečen extrém je popsáno v teoretické části. Metoda zvolí počáteční bod a vypočítá gradient funkce v tomto bodě. Tento postup se opakuje tolikrát, kolik rozměrů má daný prostor. Následně se tyto směry průměrují a je vypočítán nový bod. V této animaci jsou naznačeny dva iterační kroky této metody. Jedná se o dvourozměrný prostor, proto metoda vytváří dva paralelní směry, z nichž se udělá průměr. V pravé části této animace je slovně popsáno, jak daná metoda postupuje. Ve spodní části je na konci animace zobrazen popis, jak metoda svůj iterační postup ukončí.

3.4 Lineární programování

Animace na lineární programování je zpracována v programu MS Office PowerPoint a demonstruje krok po kroku jednotlivé postupy při výpočtu lineárního programování pomocí simplexové tabulky. Celá animace obsahuje postup jak matematicky zformulovat zadanou úlohu, jak sestavit simplexovou tabulku, jak přesně postupovat při výpočtu a také jak odečíst výsledky.



Obr. 11. Lineární programování

Výpočet:

x_1	x_2	x_3	x_4	x_5	x_6	b	
2	3	1	1	0	0	20	$\beta=6,6$
1	2	1	0	1	0	12	$\beta=6$
10	12	6	0	0	1	100	$\beta=8,3$
-1200	-1800	-800	0	0	0	0	

- Výběr klíčového sloupce: nejzápornější číslo v posledním řádku
- Výběr klíčového řádku: nejmenší nezáporný poměr $\beta = b_i/a_{ij}$
- Eliminace klíčového prvku: Klíčový prvek musí být = 1 a ostatní prvky ve sloupci = 0. Docílíme toho řádkovými úpravami

Obr. 12. Lineární programování - výpočet

3.5 Celočíslné programování

Celočíslné programování navazuje na lineární programování pomocí simplexové tabulky. V této animaci je přesně popsán výpočet na ukázkovém příkladě. Jelikož tato animace navazuje na předešlou animaci, není zde dopodrobna vysvětlen postup pro výpočet klasického lineárního programování. Student je seznámen jak úlohu sestavit a jak postupovat, abychom docílili celočíselného výsledku. Celá animace je přesně popsána pro výpočet pomocí metody řezných nadrovin, která je popsána v teoretické části.



Obr. 13. Celočíslné programování

3.6 Dynamické programování – síťová forma

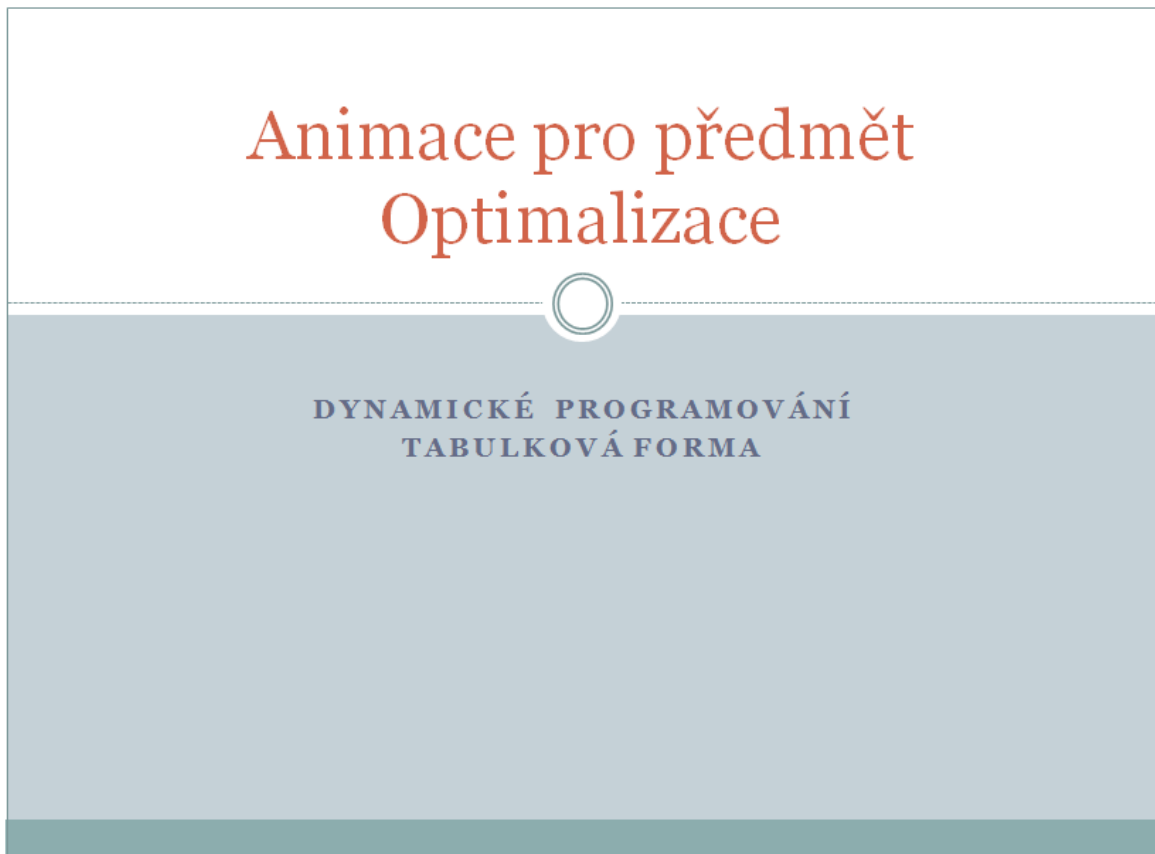
Animace na dynamické programování síťovou formu je zpracována pro příklad čtvercové sítě o 16-ti bodech. V animaci je vysvětleno jak zjistit nejkratší cestu z jednoho bodu do druhého. Dále je tu vysvětleno jak postupovat ve výpočtu, jak zařazovat body do množiny navštívených a nenavštívených bodů, ale také jak odečíst řešení.



Obr. 14. Dynamické programování – síťová forma

3.7 Dynamické programování – tabulková forma

Tabulková forma je druhý způsob výpočtu dynamického programování. Tato metoda dynamického programování využívá principu dělení zdrojů. Student je v této animaci seznámen, jak úlohu sestavit, vypočítat a odečíst výsledky. Opět byl použit ukázkový příklad, který je v animaci postupně vypočítán. Podle podrobného postupu, který je v animaci zvolen by neměl být problém pak navázat i třeba na obtížnější příklady.



Obr. 15. Dynamické programování – tabulková forma

3.8 Teorie her – maticová hra

V rámci animace Teorie her jsou studenti seznámeni se základními pojmy teorie her. Mezi tyto pojmy patří ryzí strategie, sedlový bod a převod hry na lineární programování. První pojem ryzí strategie je vysvětlen na konkrétním příkladu. Přesně je popsáno podle jaké strategie vybírá prvky matice 1. hráč a podle jaké strategie je vybírá 2. hráč tak, aby výsledkem byla ryzí strategie. Pojem sedlový bod navazuje na příklad ryzí strategie a v tomto konkrétním příkladě je naznačeno, kde se sedlový prvek nachází. Popis převodu hry na lineární programování je vysvětlen krok po kroku. Úprava matice na kladný tvar, sestavení omezení, výpočet lineárního programování a odečet výsledků ze simplexové tabulky.



Obr. 16. Teorie her – maticová hra

ZÁVĚR

V rámci této Bakalářské práce bylo za úkol vypracovat sadu animací pro předmět Optimalizace. Optimalizaci nechápeme už jen jako vyhledávání extrémů funkcí s omezujícími podmínkami, ale začíná pronikat do všech odvětví nejenom vědy a techniky. Tato práce je rozdělena do dvou částí. V teoretické části jsou zpracovány jednotlivé metody optimalizace. Tyto metody se rozdělují na čtyři části. Jsou to analytické metody, iterační metody, speciální metody a teorie her. Metody jsou rozděleny do části, kam patří a je zde popsán postup nebo algoritmus, jak daná metoda postupuje při vyhledávání extrému. Mezi speciální metody optimalizace můžeme zařadit lineární programování, celočíselné programování a dynamické programování. Praktická část je zaměřena na zpracování jednotlivých animací. Animace byly vytvářeny v programu Matlab a MS Office PowerPoint. Jednotlivé části kódu jsou popsány v příloze. Animace, které jsou zpracovány v prostředí Matlab, jsou metody iterační. V animaci je znázorněno pro určitý příklad, jak metoda nalezne extrém funkce, a jak k danému extrému postupuje. Pro animace byly použity funkce, na kterých mají metody názorný průběh. Metody pravidelného a nepravidelného simplexu jsou zpracovány pro stejný příklad funkce a porovnány. Ověřili jsme, že metoda nepravidelného simplexu konverguje k extrému v menším počtu iteračních kroků. Mezi další iterační metody, které byly zpracovány patří metoda dlouhého kroku, metoda paralelních tečen a animace, jak může dojít k zacyklení pravidelného simplexu. Další část tvoří animace, které byly vytvořeny v programu MS Office PowerPoint. Mezi tyto animace byly zařazeny speciální metody a teorie her. U těchto animací je zvolen vždy konkrétní příklad, na kterém je znázorněno, jak daným postupem dojít k výsledkům. Všechny animace byly vytvořeny názorně a jsou doplněny o komentáře tak, že by neměl být problém danou látku pochopit. Další částí by mohl být vytvořen výukový web pro tento předmět, na který by byly všechny animace a skripta k látce vloženy a byly by přístupné všem studentům. Usnadnilo by to studentům výuku tohoto předmětu a všechny přístupné materiály by byly zpracovány na jednom místě.

ZÁVĚR V ANGLIČTINĚ

In the context of this thesis was to make a set of animations for the subject Optimization. We don't see Optimization as a search extremes of functions with constraints, but we can find it in all sectors, not just science and technology. This thesis is divided into two parts. In the theoretical part are elaborated individual methods of Optimization. These methods are divided into four parts. These are analytical methods, iterative methods, special methods and theory of game. The methods are divided into parts and they are described by procedure or algorithm, how the methods search extremes. Among special methods of Optimization we include linear programming, integer programming and dynamic programming. The practical part is focused on create a set of animations. Animations were created in Matlab and MS Office PowerPoint. Parts of the code are described in the annex. Animations that are processed in Matlab, are iterative methods. In animations is shown for particular example, how find extreme of function, and how the method progresses to a extreme. For animations were used functions, on which the methods has illustrative behavior of a function. The simplex methods are created for the same example of function and they were compared. We have verified, that the method of flexible simplex converges to the extreme in a smaller number of iteration steps. Other iterative methods, that have been made, include long step method, the method of parallel tangents and animation of simplex looping. Another animations were create in MS Office Powerpoint. These animations includes special methods and theory of game. For these animations are chosen specific examples in which is shown, how to find the results. All animations were created visually and they are supplemented by comments, so that should not be a problem to understand the subject. Another part could be created educational site for this subject to put all animations and scripts for free access to all students. This make it easier for all students to learning Optimization, and all accessible materials should be handled in one place.

SEZNAM POUŽITÉ LITERATURY

- [1] BARTKO, Róbert. *MATLAB II*. Vyd. 1. Praha: Vydavatelství VŠCHT Praha, 2008, 226 s. ISBN 9788070806913.
- [2] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Vyd. 2. Zlín: Univerzita Tomáše Bati, 2006, 122 s. ISBN 8073185105.
- [3] JUREK, Miloš. *Programová podpora předmětu Optimalizace*. Zlín, 2005. Bakalářská práce (Bc.). Univerzita Tomáše Bati, Fakulta technologická. Vedoucí práce prof. Ing. Roman Prokop, CSc.
- [4] MAŇAS, Miroslav. *Optimalizační metody*. 1. vyd. Praha: SNTL, 1979, 257 s.
- [5] Optimalizace. *Wolfram Research* [online]. ©2013 [cit. 2013-03-25]. Dostupné z: <http://reference.wolfram.com/mathematica/guide/Optimization.html>
- [6] PERŮTKA, Karel. *MATLAB: základy pro studenty automatizace a informačních technologií*. Vyd. 1. Zlín: Ústav řízení procesů, Institut řízení procesů a aplikované informatiky, Fakulta technologická, Univerzita Tomáše Bati ve Zlíně, 2005, 303 s. ISBN 8073183552.
- [7] PROKOP, Roman. *Optimalizace*. In: <http://vyuka.fai.utb.cz> [online]. 2010 [cit. 2013-04-21]. Dostupné po přihlášení z: <http://vyuka.fai.utb.cz/mod/resource/view.php?id=8198>
- [8] VENKATARAMAN, P. *Applied Optimization with Matlab Programming*. New York: John Wiley & Sons, 2002, 398 s. ISBN 0471349585.
- [9] VÍTEČKOVÁ, Miluše, JEDLIČKA, David. *Statická optimalizace systémů* [online]. VŠB-TU Ostrava, 2003 [cit. 2013-03-25]. Dostupný z WWW: <http://books.fs.vsb.cz/StatickaOptimalizace/index.htm>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

$f(x)$	Funkce jedné proměnné
$\frac{df(x)}{dx}$	Derivace funkce $f(x)$ podle x
Δx	Přírůstek k bodu x
$\text{grad } f(x), \nabla f(x)$	Gradient funkce
∂	Parciální derivace
$f(x_1, x_2, \dots, x_N)$	Funkce více proměnných
$H, \nabla^2 f(x_1, x_2, \dots, x_n)$	Hessova matice
$\Phi(x, \lambda)$	Lagrangeova funkce
λ	Vektor Lagrangeových multiplikátorů
F_i	Člen Fibonacciho posloupnosti
R^N	N-rozměrný prostor
x_0	Stacionární bod
ρ	Délka hrany simplexu
λ_k	Označení kroku u gradientních metod
ε	Citlivost, tolerance
N	Počet kroků u metod
ψ	Funkce vyhodnocení
ξ	Náhodná veličina
b	Vektor omezujících konstant
DFP	Označení metody Davidon-Fletcher-Powell
Q	Množina hráčů
X_i	Prostor strategií i-tého hráče
M_i	Výplatní funkce i-tého hráče

SEZNAM OBRÁZKŮ

Obr. 1. Metoda srovnání znamének první derivace	11
Obr. 2. Simplexová tabulka	23
Obr. 3. Rozhodovací situace teorie her	26
Obr. 4. Účelová funkce pro vytvoření animací simplexů	32
Obr. 5. Pravidelný simplex	33
Obr. 6. Možnosti zacyklení simplexu – jeden z bodů nebyl dlouho vybrán.....	34
Obr. 7. Možnosti zacyklení simplexu – zrcadlení simplexu.....	34
Obr. 8. Nepravidelný simplex.....	35
Obr. 9. Metoda největšího spádu (dlouhý krok)	36
Obr. 10. Metoda paralelních tečen.....	37
Obr. 11. Lineární programování	38
Obr. 12. Lineární programování - výpočet	38
Obr. 13. Celočíslné programování	39
Obr. 14. Dynamické programování – síťová forma.....	40
Obr. 15. Dynamické programování – tabulková forma.....	41
Obr. 16. Teorie her – maticová hra.....	42

SEZNAM PŘÍLOH

PŘÍLOHA P I: POPIS KÓDŮ PRO VYTVÁŘENÍ ANIMACÍ

PŘÍLOHA P I: POPIS KÓDŮ PRO VYTVÁŘENÍ ANIMACÍ

Animace, které patří do iteračních metod byly vytvořeny v programu Matlab. Metody, které byly vytvářeny pro vyhledávání minima určité funkce, jsou zpracovány pomocí vykreslování bodů do vrstevnicového grafu.

```
% zadání funkce a vykreslení vrstevnicového grafu
f = @(x,y) 3*x^2-2*x*y+y^2+4*x+3*y+1;
[X Y]=meshgrid(-5:0.05:5,-6:0.05:4);
Z=3*X.^2-2*X.*Y+Y.^2+4*X+3*Y+1;
contour(X,Y,Z,80);
```

Animace jsou zpracovány vždy pro určitý počet iteračních kroků, použit cyklus for, a v každém iteračním kroku jsou zadány souřadnice bodu, který je vykreslen do grafu pomocí funkce plot. Veškeré texty, které animace obsahuje, jsou vykresleny pomocí funkce text.

```
A=[0.4,0.2];
B=[1.8,0.8];
C=[1.7,0.82];
D=[1.7,0.7];
xline=[A(1),B(1),C(1),B(1),D(1)];
yline=[A(2),B(2),C(2),B(2),D(2)];
plot(xline,yline,'-r');
text(2.8,4.2,'Výpočet gradientu funkce v bodě x_{(1,1)}','FontSize',12)
```

Postup po jednotlivých krocích nám zajišťuje funkce getframe, která nám po každém kroku vytvoří snímek obrazovky. Tyto snímky poté zobrazujeme pomocí funkce movie a výsledné video vytvoří funkce movie2avi.

```
F(i) = getframe;
movie(F,1,0.5);
movie2avi(F,'Dlouhykrok.avi','fps',0.333,'quality',100);
```

Animace, které jsou vytvořeny na lineární programování, celočíselné programování, dynamické programování a teorii her jsou vytvořeny v programu MS Office PowerPoint a nastaveny tak, aby se po určitých časových úsecích zobrazovaly jednotlivé slajdy. Animaci stačí spustit pomocí tlačítka F5.