Numerické metody optimalizace

Numerical optimization methods

Bc. Miloš Jurek

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Miloš JUREK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Numerické metody optimalizace**

Zásady pro vypracování:

1. Zpracujte literární rešerši dané problematiky.
2. Vytvořte přehled metod hledání extrémů reálných funkcí jedné či více reálných proměnných. V souladu s tématem práce se přitom zaměřte především na numerické (iterační) způsoby.
3. K vybraným algoritmům vytvořte programovou a vizualizační podporu ve zvoleném prostředí (MATLAB, Mathematica, atd.).
4. Proveďte komparativní analýzu naprogramovaných optimalizačních metod.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

[1] Brunovská, A.: Malá optimalizácia. Alfa, Bratislava, 1990.
[2] Kočiča, M.: Iterační metody optimalizace. Diplomová práce, FAI UTB ve Zlíně, 2006.
[3] Maňas, M.: Optimalizační metody, SNTL Praha, 1979.
[4] Prokop, R.: Teória systémov a optimalizácia. CHTF, SVŠT v Bratislavě, 1985.
[5] Štecha, J.: Optimální rozhodování a řízení. Vydavatelství ČVUT, Praha, 2000.
[6] Vítečková, M., Jedlička, D.: Statická optimalizace systémů [online].[cit.1. února 2007]. Dostupné z WWW: http://www.fs.vsb.cz/books/statickaoptimalizace/.

Vedoucí diplomové práce: **Ing. Radek Matušů**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **28. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

Abstrakt

Abstrakt česky

Optimalizační metody představují vyhledávání extrémů reálných funkcí jedné nebo více reálných proměnných s různým omezením na definiční obor. Práce může být použita k programové a vizualizační podpoře předmětu, který se touto problematikou zabývá. Cílem práce bylo vytvořit algoritmy optimalizačních iteračních metod v prostředí Matlab. Důraz je kladen na iterační metody komparativní a gradientní. Dále byly vypracovány vzorové příklady, ve kterých jsou popsány jednotlivé metody, provedeno jejich hodnocení a graficky znázorněny výsledky.

Klíčová slova: Optimalizace, extrém funkce, gradient, iterace, účelová funkce

Abstrakt ve světovém jazyce

Optimization methods constitute the searching of extremes of real functions of one or more real variables with the different limitations per definition range. The thesis can be used for program and visual support of the subject, which deals with this problems. The goal of this thesis was to create algorithms of optimization iterations methods in the Matlab environment. The stress is put on the iteration methods comparative and gradient. The sample examples, in which each method is described, made their classifications and graphically illustrated results, was elaborated.

Keywords: Optimization, function extremes, gradient, iteration, goal function

V rámci této práce bych chtěl poděkovat panu Ing. Radkovi Matušů za zadání zajímavého tématu a za cenné rady a připomínky při odborném vedení mé diplomové práce.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně, 24. 05. 2007 .....

podpis

OBSAH

ÚVOD

Problematika optimalizačních metod zahrnuje vyhledávání extrémů reálných funkcí reálných proměnných. Tato matematická formulace se v průběhu 20. století stala velmi hluboce studovanou a atraktivní disciplínou. S optimalizačními úlohami nejrůznějšího druhu se můžeme velmi často setkat v praxi, a to v širokém spektru odvětví. Lze si představit návrh nákladní trasy tak, že spotřeba pohonných hmot a vzdálenost dosahují svých extrémů, stejně tak k optimalizační úloze vede z oblasti ekonomie problém optimalizace výrobních programů, optimalizace technických konstrukcí ve stavebnictví, strojírenství a vojenství, či hledání optimálního stavu v chemických procesech.

Převedeme-li optimalizační úlohu na matematický tvar, její následné řešení je otázkou matematické rutiny a výpočetní techniky. Teoreticky je možné definovat jakkoli obtížný problém, v praxi se ale následně ukáže, že není možné vymyslet univerzální postup, který by vedl k nejvhodnějšímu řešení. To vedlo ke klasifikaci jednotlivých optimalizačních úloh a k nim příslušným postupům.

Je třeba rozlišovat úlohy bez omezení na definiční obor (volný extrém), omezení typu rovnost (klasický vázaný extrém) a omezení typu nerovnost (neklasický vázaný extrém). Jak bylo řečeno, různé úlohy vyžadují různý matematický aparát a různé metody. V prvé řadě jde o analytické metody, ty jsou nejbližší ke klasické matematické analýze. Využívají prostředky diferenciálního počtu a jsou vhodné zejména pro individuální ruční výpočet. Druhou skupinu tvoří iterační metody, ve kterých se sestavuje posloupnost bodů definičního oboru, která konverguje k extrému. Metody jsou především vhodné k programování na počítačích. Třetí skupinu metod tvoří speciální úlohy, které vyžadují specifické metody. Jedná se o lineární, dynamické, nelineární a jiné programování, které je obecně nazýváno pojmem operační analýza.

V této práci je přiblíženo, co vlastně znamená a obnáší řešení optimalizačních úloh a výpočet extrémů funkcí. Jsou zde zmíněny jednotlivé klasifikace optimalizačních úloh. Dále se v práci pojednává především o iteračních metodách optimalizace. Ve druhé kapitole je podrobně popsáno jejich rozdělení. Jsou zde přiblíženy jednotlivé principy algoritmů a vlastnosti jednotlivých metod. V neposlední řadě je produktem této práce program vytvořený v prostředí Matlab. Hlavní částí programu je deset základních optimalizačních metod, které jsou rozděleny pro úlohy jedné proměnné a pro metody více

proměnných. Pro úlohy jednorozměrné byly vytvořeny čtyři algoritmy a pro vícerozměrné úlohy šest algoritmů. Jednotlivé naprogramované algoritmy uživateli vypíší výsledky do textových polí a zobrazí graf příslušné funkce a znázorní zjištěný extrém. Dále byly naprogramované metody podrobeny testům, po kterých následovalo zhodnocení výsledků a hlavních výhod a nevýhod jednotlivých metod. V deseti přílohách jsou uvedeny hlavní části jednotlivých naprogramovaných algoritmů.

I. TEORETICKÁ ČÁST

1 OPTIMALIZACE

Teorie optimalizace je matematickou disciplínou, která se zabývá určováním minimálních a maximálních hodnot funkcí při určitých omezujících podmínkách, tj. řešením optimalizačních úloh. S optimalizačními úlohami nejrůznějšího druhu se v praxi setkáváme velmi často. Většinou jsou formulovány slovně a řeší se na základě zkušeností a intuice. Takový přístup při současné úrovni rozvoje vědy a techniky již zcela nestačí. Neposkytuje objektivní a vědecké podklady pro řízení a rozhodování.

Z jiného pohledu lze o optimalizaci také říci, že jde o matematickou disciplínu, ve které hledáme minimum respektive maximum dané funkce f(x) na dané množině X. Tato funkce se nazývá účelová, nebo též optimalizační. Množinou se rozumí oblast, pro kterou lze nalezené řešení považovat za korektní, eventuálně fyzikálně realizovatelné a podobně. Obecně bývá tato oblast vymezena soustavami rovnic či nerovnic. Jejich významem je zamezit prohledávání prostoru, pro nějž nalezené řešení nebude akceptovatelné.

Impuls pro rozvoj extremalizačních metod přinesly v minulém století úlohy z oblasti ekonomie, které můžeme označit jako problémy optimalizace výrobních programů. Řada extremalizačních úloh přichází z oblasti techniky. Jsou popsány výpočty směřující k optimalizaci z oblasti stavebnictví, strojírenství, vojenství. Z oblasti chemie je typickou extremalizační úlohou výpočet chemické rovnováhy v plynné soustavě. Také v teorii her se řada úloh řeší převedením na vyhledávání extrému funkce. Praktické aplikace optimalizačních metod jsou přiblíženy např. v [1] či [5].

1.1 Základní pojmy

Účelová, kriteriální či optimalizační funkce jsou v tomto textu synonyma pro jednu a tutéž funkci. Nejčastěji bude užíváno pojmu účelová funkce a za tímto označením se skrývá n-rozměrná reálná funkce reálných proměnných s označením $f(x_1 ,.., x_n)$.

Funkce f(x), která má na množině X pouze jeden extrém (daného druhu), se nazývá unimodální. V opačném případě se jedná o funkci multimodální.

Množinu X budeme nazývat množinou přípustných řešení (bodů, vektorů) a funkci f(x) účelovou funkcí.

Společný název pro minimum a maximum je extrém. Body $x \in X$, ve kterých má funkce f(x) extrém, tj. funkce f(x) nabývá extremální (minimální nebo maximální) hodnoty, se

nazývají extremální body funkce f(x) na množině X. Extrémy mohou být dvojího druhu a to buď lokální či globální.

Přesná definice říká, že funkce f: $X \rightarrow R$ má v bodě $x_0 \in X$

- Lokální minimum – na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $x \in X$, $|x-x_0| < \delta$ platí $f(x) \geq f(x_0)$
- Ostré lokální minimum - na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $x \in X$, $0 < |x-x_0| < \delta$ platí $f(x) > f(x_0)$
- Globální minimum - na $X \subset R^n$, jestliže pro každé $x \in X$ platí $f(x) \geq f(x_0)$
- Ostré globální minimum - na $X \subset R^n$, jestliže pro každé $x \in X$, $x_0 \neq x$ platí $f(x) > f(x_0)$
- Lokální maximum - na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $x \in X$, $|x-x_0| < \delta$ platí $f(x) \leq f(x_0)$
- Ostré lokální maximum - na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $x \in X$, $0 < |x-x_0| < \delta$ platí $f(x) < f(x_0)$
- Globální maximum - na $X \subset R^n$, jestliže pro každé $x \in X$ platí $f(x) \leq f(x_0)$
- Ostré globální maximum - na $X \subset R^n$, jestliže pro každé $x \in X$, $x_0 \neq x$ platí $f(x) < f(x_0)$

1.2 Klasifikace optimalizačních úloh

Optimalizační úlohy se klasifikují dle jejich omezení na definiční obor. Vyšetřujeme-li extrémy funkce n proměnných na celém prostoru R^n mluvíme o tzv. volných extrémech. Zajímají-li nás pouze takové body maxima nebo minima, které splňují nějaké další podmínky, mluvíme o vázaných extrémech. Názvy volné a vázané extrémy nejsou termíny v matematickém smyslu přesně zavedené a podle toho používané. O volných extrémech se například mluví i tehdy, kdy extremalizovaná funkce není všude v R^n definovaná a vyšetřuje se pouze na té množině $X \subset R^n$, kde definována je. Je dohodnuto, že volný extrém funkce f(x) je globální extrém této funkce vzhledem k R^n.

Vázané extrémy se původně systematicky vyšetřovaly pouze při podmínkách tvaru $g_j(x_1,...,x_n)=0$, $j=1,...,n$, kde g_j byly funkce se spojitými prvními parciálními derivacemi.

Po druhé světové válce se objevily důležité úlohy, v nichž se vyskytovaly vedlejší podmínky s nerovnostmi, upravované nejčastěji na tvar

$$g_j(x_1,...,x_n) \geq 0, j = 1,...,m \qquad x_i \geq 0, i = 1,...,n \tag{1}$$

Extremalizační úlohy s omezeními ve tvaru (1) se nazývají úlohy matematického programování. V matematickém programování se extremalizovaná funkce nazývá funkce účelová. Ta $x \in R^n$, která vyhovují omezujícím podmínkám, se nazývají přípustná řešení. Přípustné řešení x, které je globálním extrémem účelové funkce vzhledem k množině přípustných řešení, se nazývá optimální řešení.

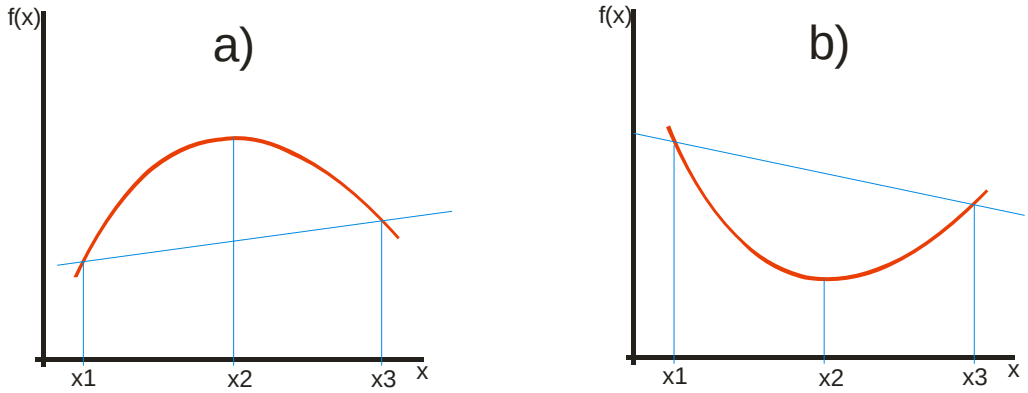
1.2.1 Volný extrém funkce

Volným extrémem funkce rozumíme lokální minimum nebo lokální maximum funkce v libovolném bodě definičního oboru. Metody analytického vyhledávání volných extrémů jsou založené na výpočtech derivací, a proto jsou vhodné pro analyticky zadané funkce.

Jednorozměrný případ

V případech, že je funkce na prohledávaném intervalu hladká a spojitá lze využít známých matematických postupů. Analýza funkcí vychází z jejích základních vlastností.

Funkce definovaná na intervalu X se nazývá konvexní (konkávní), když pro libovolnou trojici bodů z X $x_1<x_2<x_3$ leží bod $(x_2, f(x_2))$ pod úsečkou (nad úsečkou), která spojuje body $(x_1, f(x_1))$ a $(x_3, f(x_3))$ a její rovnice je $y - f(x_1) = \frac{f(x_3) - f(x_1)}{x_3 - x_1}(x - x_1)$.

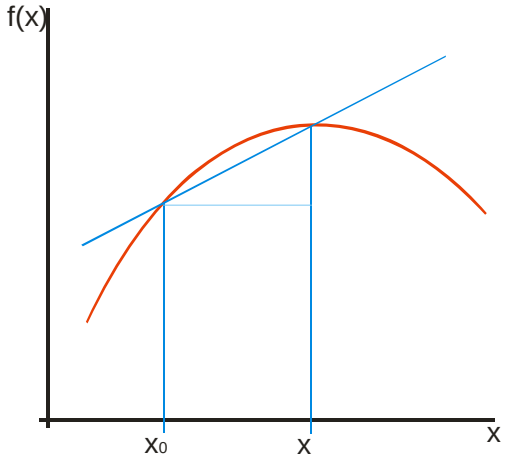


Obr. 1. a) Konkávní funkce, b) Konvexní funkce

Když je funkce f(x) na intervalu X spojitá a má v každém bodě konečnou druhou derivaci, pak je funkce f(x) na intervalu konvexní právě tehdy, když pro každé $x \in X$ je $f''(x) \geq 0$ a funkce f(x) je konkávní právě tehdy, když pro každé $x \in X$ je $f''(x) \leq 0$.

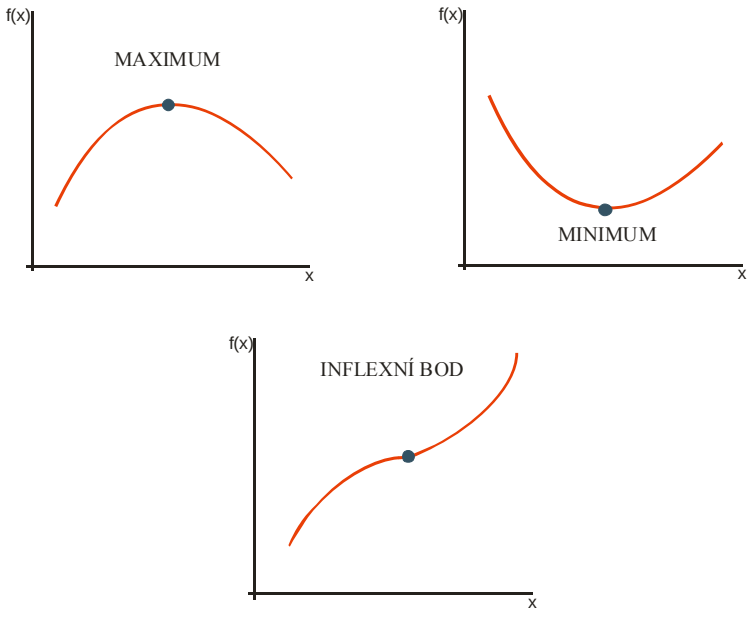
Derivace funkce v bodě x_0 je definována limitou: $f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$

a vyjadřuje směrnici tečny v daném bodě.



Obr. 2. Derivace funkce v bodě

Stacionární bod funkce f(x) je bod x_0 pro který platí, že jeho první derivace je rovna nule $f'(x_0)=0$. V tomto bodě má daná funkce f(x) minimum, maximum nebo jde o tzv. inflexní bod.



Obr. 3. Stacionární body funkcí

Předpokládejme reálnou funkci jedné reálné proměnné. Pojem extrém funkce úzce souvisí s pojmy konkávnosti či konvexnosti.

Funkce definovaná v jistém okolí bodu x_0 má v tomto bodě ostré lokální maximum, jestliže existuje $\delta>0$ tak, že pro každé x, $0<|x-x_0|<\delta$ platí $f(x)<f(x_0)$. O neostrém lokálním maximu hovoříme, jestliže je splněná neostrá nerovnost $f(x)\leq f(x_0)$. Pojem lokálního minima se definuje analogicky, ale při splnění nerovnosti $f(x)>f(x_0)$. Globální maximum (minimum) funkce f(x) na intervalu X je největší (nejmenší) hodnota, kterou funkce f(x) na intervalu X nabude.

Nechť funkce f(x) má první derivaci na intervale X a v bodě x_0 tohoto intervalu nechť existuje druhá derivace.

Potom platí, že

- funkce f(x) má v bodě x_0 ostré lokální maximum, když $f'(x_0)=0$ a $f''(x_0)<0$.
- funkce f(x) má v bodě x_0 ostré lokální minimum, když $f'(x_0)=0$ a $f''(x_0)>0$.

Jestliže druhá derivace funkce f(x) v bodě x_0 neexistuje

- a první derivace $f'(x_0)$ je v levém okolí bodu x_0 kladná a v pravém okolí x_0 je záporná, potom má funkce f(x) v bodě x_0 ostré lokální maximum.
- a první derivace $f'(x_0)$ je v levém okolí bodu x_0 záporná a v pravém okolí x_0 je kladná, potom má funkce f(x) v bodě x_0 ostré lokální minimum.

Může se stát, že první i druhá derivace v bodě x_0 je rovna nule. Potom funkce může mít v tomto bodě inflexi anebo je třeba zkoumat vyšší derivace.

Nechť existuje přirozené číslo n tak, že $f^{(n)}(x_0)\neq 0$ a pro každé $k<n$ je $f^{(k)}(x_0)=0$. Potom platí, že:

- jestliže je n sudé a $f^{(n)}(x_0)>0$, má f(x) v x_0 ostré lokální minimum.
- jestliže je n sudé a $f^{(n)}(x_0)<0$, má f(x) v x_0 ostré lokální maximum.
- jestliže je n liché, jde o inflexní bod.V inflexním bodě se mění konkávnost funkce na konvexnost či naopak. Konkávnost souvisí s maximem funkce a konvexnost s minimem.

Vyhodnocením těchto podmínek identifikujeme stacionární bod na minimum, maximum nebo inflexní bod. Není však zaručeno nalezení všech extrémů, protože není pokryta varianta neexistence první derivace funkce f(x) v daném bodě.

Vícerozměrný případ

Analytické řešení vyhledávání extrémů reálné funkce více reálných proměnných je v zásadě zevšeobecněním jednorozměrného případu. Namísto pojmu první a druhá derivace se používá pojem gradientu a Hessovy matice a namísto inflexního bodu se používá pojem sedlový bod. Gradient funkce se definuje jako vektor parciálních derivací:

$$\text{grad f}(x_1,...,x_n) = \nabla f = \left(\frac{\partial f}{\partial x_1},...,\frac{\partial f}{\partial x_n} \right) \tag{2}$$

Hessova matice se definuje pomocí druhých parciálních derivací:

$$H = \nabla^2 f(x_1,...,x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{3}$$

Při vyšetřování extrémů se musí určit, zda je Hessova matice pozitivně či negativně definitní, což odpovídá kladné nebo záporné druhé derivaci v jednorozměrném případě.

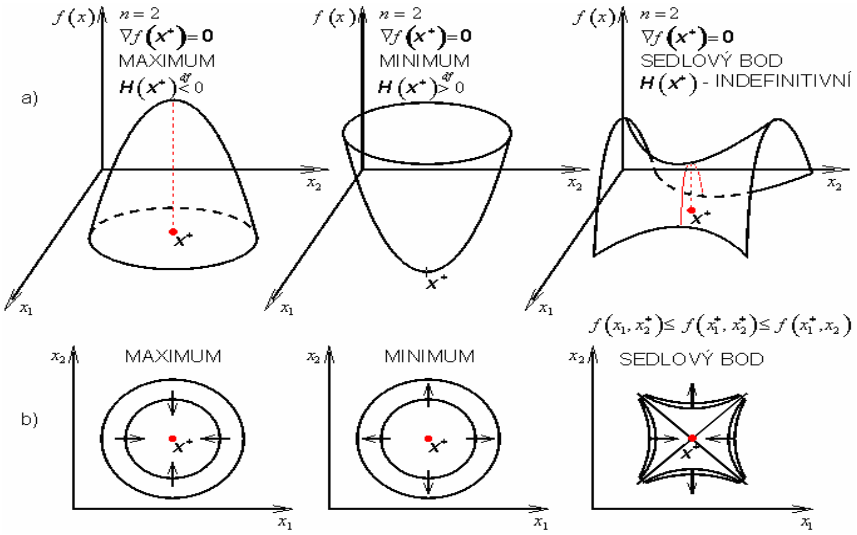
S problematikou definitnosti souvisí tzv. kvadratická forma. Tuto problematiku podrobně popisuje [6].

Při vyšetřování pozitivní či negativní definitnosti matice je třeba počítat hlavní subdeterminanty. Podmínky pro definitnost matice udává Sylvestrovo kritérium: „Matice je pozitivně definitní, když všechny hlavní subdeterminanty této matice mají kladnou hodnotu. Matice je negativně definitní, když hodnoty hlavních subdeterminantů střídají znaménka, ale první subdeterminant musí být záporný“.

Pokud je matice pozitivně defintní, má funkce f(x) v daném bodě x_0 lokální minimum.

Pokud je matice negativně defintní, má funkce f(x) v daném bodě x_0 lokální maximum.

Pokud je matice indefinitní, jedná se o sedlový bod.[4]



Obr. 4. Extrémy funkcí více proměnných

1.2.2 Vázaný extrém funkce

V případě volného extrému se hledá extrém na celém definičním oboru. V praxi se však vyhledávají extrémy při různých omezeních. Nejfrekventovanějším omezením bývají rovnost a nerovnost, případně smíšená omezení. Úlohy s omezením typu rovnost se nazývají úlohy klasického vázaného extrému, úlohy s nerovnostmi se nazývají úlohy neklasického vázaného extrému.

Klasický vázaný extrém

Postup řešení klasického vázaného extrému je všeobecně známý pod názvem metoda Lagrangeových multiplikátorů. Je třeba maximalizovat funkci $f(x_1,\ldots,x_n)$ při podmínkách $gj(x_1,\ldots,x_n)=0$, $j=1,\ldots,m < n$

Při řešení této úlohy se definuje Lagrangeova funkce

$$\phi(x,p) = f(x) + \sum_{j=1}^{m} p_j g_j(x) \qquad (4)$$

kde vektor $\lambda=(\lambda_1,\ldots,\lambda_m)$ se nazývá vektor Lagrangeových multiplikátorů. Body vázaného extrému se potom hledají jako body extrému Lagrangeovy funkce, tj. jako nulové body všech n+m parciálních derivací funkce ϕ.[4]

Když označíme:

$$\nabla_x \phi = \left(\frac{\partial \phi}{\partial x_1}, \frac{\partial \phi}{\partial x_2}, ..., \frac{\partial \phi}{\partial x_n} \right)^T$$
$$\nabla_p \phi = \left(\frac{\partial \phi}{\partial p_1}, \frac{\partial \phi}{\partial p_2}, ..., \frac{\partial \phi}{\partial p_m} \right)^T \quad (5)$$

potom existenci extrému udává následující věta: „Nechť funkce $f,g_1,\ldots,g_n$ mají spojité parciální derivace a nechť jsou funkce $\nabla g_j(x)$ lineárně nezávislé. Potom platí, že když je $x_0 \in R^n$ bodem lokálního extrému funkce f při omezeních $g_j(x)=0$, potom existuje $p_0=(p_{01},\ldots,p_{0m})$ tak, že

$$\nabla_x \phi(x_0, p_0) = \nabla_2(x_0, p_0) = 0 \quad (6)$$

Neklasický vázaný extrém

Neklasický vázaný extrém se definuje jako úloha maximalizovat funkci $f(x_1,\ldots,x_n)$ při omezeních tvaru $g_j(x_1,\ldots,x_n)\geq 0$, $j=1,\ldots,m$ $x_i \geq 0$ $i=1,\ldots,n$.

Takto formulovaná úloha se nazývá úloha matematického programování. Jednou z nejdůležitějších teorémů charakterizující takovouto úlohu je Kuhn-Tuckerova věta. Tato věta byla publikována roku 1951 a nazývá se též větou o sedlovém bodě. Podobně jako v případě omezení rovnostmi se definuje Lagrangeova funkce (4).

Kuhn-Tuckerova věta: „Vektor x_0 je optimálním řešením úlohy právě tehdy, když existuje vektor p_0 takový, že $x_0 \geq 0$ a $p_0 \geq 0$ a pro všechny $x \geq 0$, $p \geq 0$ platí:

$$\phi(x, p_0) \leq \phi(x_0, p_0) \leq \phi(x_0, p) \quad (7)$$

Bod (x_0,p_0) se nazývá nezáporný sedlový bod funkce ϕ.

Takto formulovaná věta má pro matematické programování význam jen teoretický, protože sedlový bod se hledá obtížně. Pro výpočetní účely má význam jen její důsledek, který platí pro diferencovatelné funkce f a g_j.

Důsledek Kuhn-Tuckerovy věty: „Nechť f a g_j mají všechny první parciální derivace. Potom podmínka (7) je ekvivalentní tzv. lokálním Kuhn-Tuckerovým podmínkám:

$$\left(\frac{\partial \phi}{\partial x_i} \right)_{(x_0, p_0)} \leq 0 \qquad x_{0i} \left(\frac{\partial \phi}{\partial x_i} \right)_{(x_0, p_0)} = 0 \qquad i = 1, ..., n \qquad x_{0i} \geq 0$$
$$\left(\frac{\partial \phi}{\partial p} \right)_{(x_0, p_0)} \geq 0 \qquad p_{0j} \left(\frac{\partial \phi}{\partial p_j} \right)_{(x_0, p_0)} = 0 \qquad j = 1, ..., n \qquad p_{0j} \geq 0 \quad (8)$$

1.3 Klasifikace optimalizačních metod

Analytické metody

- jednorozměrné případy – derivace
- vícerozměrné případy – gradienty
    - omezení typu = metoda Lagrangeova multiplikátoru
    - omezení typu ≤ Kuhn-Tuckerova věta

Iterační metody:

- komparativní
    - jednorozměrné
    - vícerozměrné
- gradientní
    - bez omezení
    - s omezením
- metody náhodného vyhledávání
    - jednoduché
    - adaptivní
    - s umělou inteligencí

Specielní metody:

- lineární programování
- dynamické programování
- konvexní programování

Teorie her:

- maticové hry
- diferenciální hry

2 ITERAČNÍ METODY OPTIMALIZACE

Tyto metody lze rozdělit na tři typy. Na komparativní optimalizační metody, které nevyžadují výpočet nebo odhad derivací účelové funkce. Iterační charakter těchto metod je založený na deterministickém principu. Metody, které potřebují výpočet či odhad derivací, jsou metody gradientní a třetím typem jsou metody nedeterministické. Iterační přírůstek těchto metod či jeho směr má náhodný charakter.

2.1 Komparativní metody

Základní charakteristikou komparativních metod je vyčíslení diskrétních hodnot účelové funkce v určitých bodech, jejich porovnání s hodnotami vypočtenými v dalším kroku a v určení postupu konstrukce dalších diskrétních bodů. Postupuje se tak dlouho, dokud se dosahuje dalšího zlepšení hodnoty účelové funkce. Jedná se o postupy pro jednorozměrné a mnohorozměrné problémy.[4]

2.1.1 Jednorozměrné komparativní metody

Metody jednorozměrné optimalizace mají v celé teorii optimalizace základní význam. Jsou názorné a mají jednoduchou grafickou interpretaci. Navíc se mohou využívat i ve vícerozměrné optimalizaci.

Nejznámější dvě metody jednorozměrné optimalizace, které nevyžadují výpočet derivací, jsou Fibonacciho metoda a metoda zlatého řezu. Obě metody jsou založené na myšlence zvolit posloupnost bodů x_i z daného intervalu <a;b> tak, abychom našli interval co nejmenší délky, ve kterém se nachází bod extrému.

Není-li daná účelová funkce f(x) na intervalu <a,b> unimodální, pak je třeba přibližně určit polohu bodu globálního minima a interval <a,b> změnit tak, aby na změněném intervalu účelová funkce f(x) unimodální byla.

Fibonacciho metoda

Fibonacciho metoda využívá při zkracování intervalů neurčitosti přímé úměrnosti jejich délek číslům Fibonacciho posloupnosti, tj.

$$l_N = \frac{b-a}{F_N} = \frac{l_1}{F_N} = \frac{l_2}{F_{N-1}} = \ldots = \frac{l_k}{F_{N-(k-1)}} = \ldots = \frac{l_{N-1}}{F_2} = \frac{l_N}{F_1} \tag{9}$$

kde čísla Fibonacciho posloupnosti $\{F_k\}$ jsou dána vztahy

$$F_0 = F_1 = 1, \qquad F_k = F_{k-1} + F_{k-2}, \qquad k = 2,3,..., \tag{10}$$

resp.

$$F_k = \frac{1}{\sqrt{5}}\left[z^{k+1} - \left(-\frac{1}{z} \right)^{k+1} \right], \qquad k = 0,1,2,..., \qquad z = \frac{1+\sqrt{5}}{2} \tag{11}$$

Pro velké k lze v rovnici (11) výraz $\left(-\frac{1}{z} \right)^{k+1}$ zanedbat a dostaneme přibližný vzorec pro výpočet čísel Fibonacciho posloupnosti

$$F_k \approx \frac{1}{\sqrt{5}} z^{k+1}, \qquad k = 0,1,2,... \tag{12}$$

Fibonacciho metoda pro vyhledání maxima může být popsána následujícím algoritmem:

1. $\alpha_1 = a \qquad \beta_1 = b \qquad$ *zvolíme počet kroků* N
2. $\tilde{\alpha}_{i+1} = \beta_i - \frac{F_{N-i+1}}{F_{N-i+2}} |\beta_i - \alpha_i|$

   $\tilde{\beta}_{i+1} = \alpha_i + \frac{F_{N-i+1}}{F_{N-i+2}} |\beta_i - \alpha_i|$
3. $f(\tilde{\alpha}_{i+1}), \qquad f(\tilde{\beta}_{i+1})$
4. $f(\tilde{\alpha}_{i+1}) \le f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \tilde{\alpha}_{i+1} \qquad \beta_{i+1} = \beta_i$
5. $f(\tilde{\alpha}_{i+1}) > f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \alpha_i \qquad \beta_{i+1} = \tilde{\beta}_{i+1}$
6. $i = i+1 \qquad$ *skok na bod* 2

Po N-tém kroku leží optimální bod x v intervalu $\langle a_N, b_N \rangle$. Pro dosažení přesnosti ε při lokalizaci optimálního bodu x lze použít vztah

$$F_N \ge \frac{b-a}{2\varepsilon} \tag{13}$$

Musíme tedy najít takové číslo Fibonacciho posloupnosti, které vyhovuje nerovnosti (13). Jeho index N udává počet potřebných kroků a zároveň počet iterací.[6]

Např. pro funkci f(x) definovanou na intervalu <-1,1> hledáme řešení s přesností ε = 0,1.

Počet kroků N zjistíme ze vztahu (13): $F_N \geq \frac{b-a}{2\varepsilon} = \frac{1+1}{2.0,1} = 10 \Rightarrow F_7 = 13 \Rightarrow N = 7$

Metoda zlatého řezu

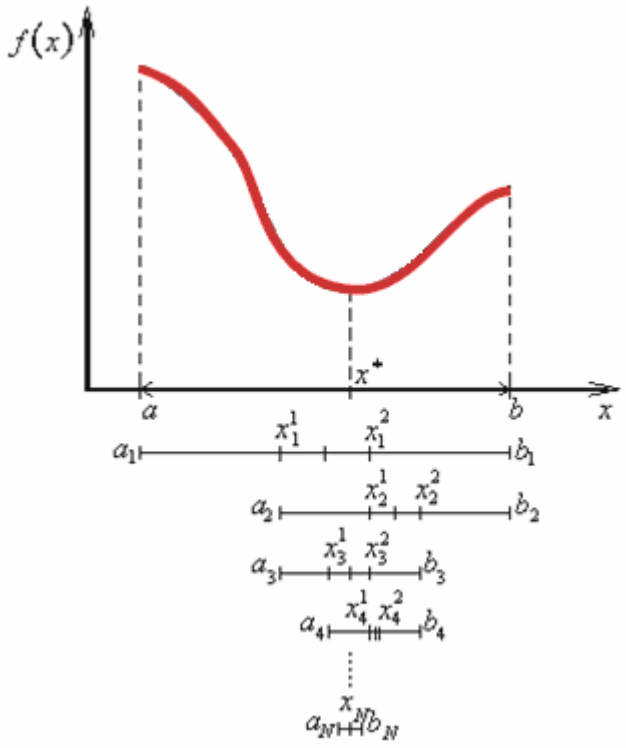
Metoda zlatého řezu patří mezi postupné komparativní metody hledání minima libovolné spojité unimodální účelové funkce f(x).

Metoda zlatého řezu je odvozením Fibonacciho metody. Místo podílu $\frac{F_{N-i+1}}{F_{N-i+2}}$ se používá limitní podíl:

$$\lim_{n\to\infty} \frac{F_{N-i+1}}{F_{N-i+2}} = 0,618 \qquad (14)$$

Hodnota 0,618 se nazývá poměr zlatého řezu (obdélník se stranami v poměru $1:\frac{1}{0,618}$ byl považován ve starém Řecku za nejestetičtější).

Při metodě zlatého řezu ve všech krocích (kromě prvního) interval neurčitosti obsahuje spolu s krajními body jeden vnitřní bod. Proto je třeba určit hodnotu účelové funkce f(x) pouze v jednom novém bodě umístěném symetricky k již známému bodu .



Obr. 5. Princip metody zlatého řezu

Popis algoritmu pro vyhledání maxima:

1. $\alpha_1 = a \qquad \beta_1 = b \qquad zvolíme\ N$
2. $\widetilde{\alpha}_{i+1} = \beta_i - 0{,}618.|\beta_i - \alpha_i|$
   $\widetilde{\beta}_{i+1} = \alpha_i + 0{,}618.|\beta_i - \alpha_i|$
3. $f(\widetilde{\alpha}_{i+1}), \qquad f(\widetilde{\beta}_{i+1})$
4. $f(\widetilde{\alpha}_{i+1}) \le f(\widetilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \widetilde{\alpha}_{i+1} \qquad \beta_{i+1} = \beta_i$
5. $f(\widetilde{\alpha}_{i+1}) > f(\widetilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \alpha_i \qquad \beta_{i+1} = \widetilde{\beta}_{i+1}$
6. $i = i + 1 \qquad skok\ na\ bod\ 2$

Počet kroků není explicitně omezený, výpočet končí testem na přesnost argumentu.
Počet kroků N je dán požadovanou přesností ε, pro kterou platí vztah:

$$b_N - a_N \le 2\varepsilon \tag{15}$$

2.1.2 Metody vícerozměrné optimalizace

Mezi těmito komparativními iteračními metodami jsou nejznámější Box-Wilsonova metoda, simplexové metody, metoda mapování kriteriální plochy a metoda cyklické záměny parametrů. Jejich principem hledání extrémů je vyčíslení hodnoty účelové funkce v určitých bodech a vzájemným porovnáváním těchto hodnot v jednotlivých iteracích dle jejich algoritmů, konvergují k extrému.

Box-Wilsonova metoda

Jedná se o jednu z nejzákladnějších komparativních optimalizačních metod. Byla publikována již v roce 1951, avšak i v současnosti zůstává vhodným nástrojem pro vyhledávání extrémů účelových funkcí.
Základními rysy a výhodami této metody jsou snadná algoritmizovatelnost, jednoduchý princip vyhledávání extrému a nezávislost na počtu proměnných.
Pro n=2 se účelová funkce vyčísluje ve vrcholech čtverce a v jeho geometrickém středu. V dalším kroku se vrchol s největší či nejmenší hodnotou bere za střed nového čtverce a postup se opakuje. Postup se na n-rozměrný případ zevšeobecní tak, že místo čtverce se konstruuje nadkvádr s 2^N vrcholy. Nevýhodou je výpočet $1 + 2^N$ funkčních hodnot funkce.
Přiblížit funkci algoritmu lze nejlépe na funkci se dvěma proměnnými.
Jako první je zvolen počáteční bod S se souřadnicemi [sx1;sx2], od kterého se dále odvíjí celý proces výpočtů. Dalším vstupním parametrem je délka hrany čtverce ω, který se

konstruuje okolo počátečního bodu. Počáteční bod leží v geometrickém středu čtverce. Při nevhodné volbě počátečního bodu se prodlužuje doba a počet výpočtů. Proto by měl být za počátek zvolen takový bod, který dle odhadu povede nejrychleji k řešení. Totéž platí pro volbu hrany čtverce. Při nevhodné velikosti naroste počet výpočtů nebo se sníží přesnost konečného výsledku. Lze tedy říci, že doba a počet výpočtů je nepřímo úměrná délce hrany a přímo úměrná vzdálenosti počátečního bodu od hledaného extrému.

Po zadání těchto vstupních parametrů se provede konstrukce vrcholů čtverce. Souřadnice jednotlivých vrcholů získáme tak, že k oběma souřadnicím středu přičteme polovinu délky strany čtverce. Pro ostatní souřadnice zbylých vrcholů je postup obdobný s tím rozdílem, že prostřídáme všechny kombinace znamének.

Podoba jednotlivých souřadnic je pak následující:

$$M_1 : \left[sx1 + \frac{\omega}{2}; sx2 + \frac{\omega}{2} \right]$$

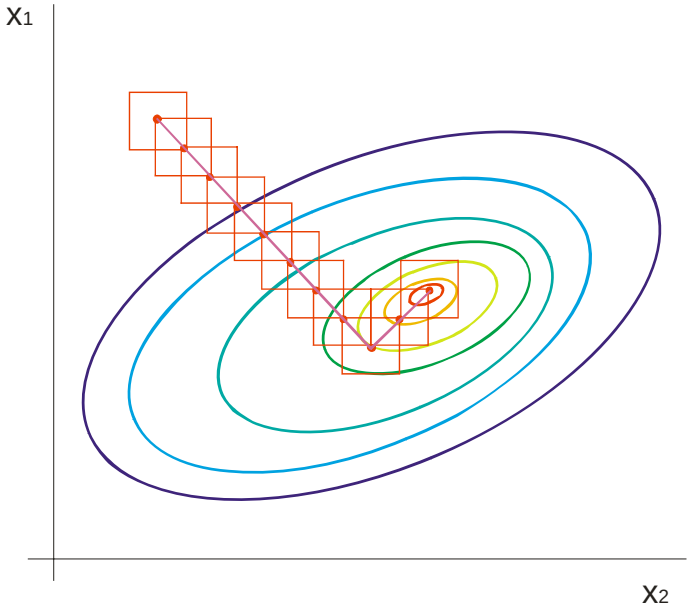
$$M_2 : \left[sx1 - \frac{\omega}{2}; sx2 + \frac{\omega}{2} \right]$$

$$M_3 : \left[sx1 + \frac{\omega}{2}; sx2 - \frac{\omega}{2} \right]$$

$$M_4 : \left[sx1 - \frac{\omega}{2}; sx2 - \frac{\omega}{2} \right]$$

Poté následuje vypočítání hodnot účelové funkce v daných bodech (S,M_1,M_2,M_3,M_4) a z této pětice se vybere ten, který má největší hodnotu. Pokud je maximem jeden z vrcholů čtverce, stává se novým středem v další iteraci a pokračuje se v ohodnocování jednotlivých vrcholů. Pokud má maximální hodnotu střed čtverce, je vyhodnocen jako extrém a algoritmus je ukončen.

Z vyhledávání bodu s největší hodnotou pramení, že algoritmus vyhledává maximum funkce. Při potřebě nalezení minima funkce se musí změnit vyhledávání na bod s nejmenším ohodnocením nebo hledat extrém namísto f(x1,x2) na funkci –f(x1,x2).



Obr. 6. Princip Box-Wilsonovy metody

Zápis algoritmu pro vyhledávání maxima (n=2):

- Volba - střed krychle S a délka strany ω
- Generování vrcholů krychle $M_i=S\pm \omega\, \delta_i$, $i=1,\ldots,2^N$
- Výpočet f(S), $f(M_1),\ldots,f(M_4)$ v jednotlivých bodech čtverce S, M_1, M_2, M_3, M_4
- Výběr extremální hodnoty $f(M_e)$ a M_e
- Je-li $M_e=S$ => STOP
- $S=M_e$ a skok na generování vrcholů

Simplexové metody

Simplexové metody patří mezi další komparativní metody, které nepotřebují výpočet derivací účelové funkce. Je zde určitá podobnost s Box-Wilsonovou metodou, avšak oproti předcházející metodě se počet bodů výpočtu hodnot účelové funkce redukuje na minimum. Pojmem simplex je označován nejmenší konvexní polyedr v daném prostoru. Jinými slovy, v n-dimenzionálním prostoru definujeme n+1 bodů jež tvoří simplex. Ve dvojrozměrném prostoru je simplexem rovnostranný trojúhelník, v třírozměrném prostoru je to pravidelný čtyřstěn.

V R^n jsou body pravidelného simplexu určené následujícími vztahy

$$x_1 = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} \quad x_2 = \begin{pmatrix} a_1 + d + e \\ a_2 + d \\ \dots \\ a_n + d \end{pmatrix}, \quad \dots, \quad x_{n+1} = \begin{pmatrix} a_1 + d \\ a_2 + d \\ \dots \\ a_n + d + e \end{pmatrix} \tag{16}$$

$$d = \rho \frac{\sqrt{n+1} - 1}{\sqrt[n]{2}} \qquad e = d + \frac{\rho}{2} \tag{17}$$

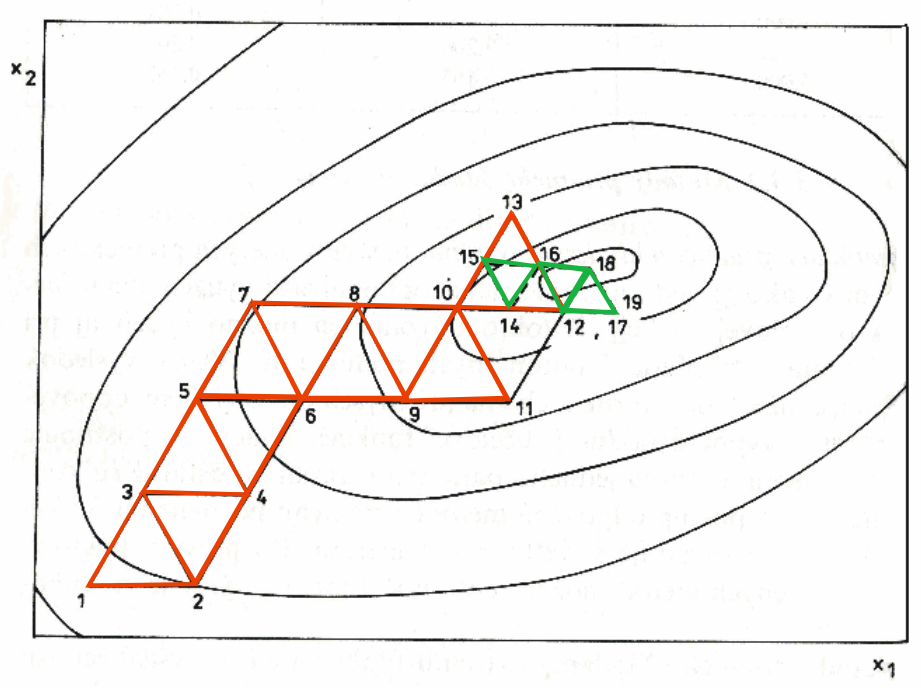
kde ρ je délka hrany simplexu a n je dimenze prostoru.[3]

Základní strategií pro maximalizaci účelové funkce v R^n je spočítání hodnot účelové funkce ve vrcholech simplexu a určení bodu s nejmenší hodnotou. Tento bod se překlopí přes těžiště simplexu do směru největšího růstu tak daleko, aby tento bod tvořil se základnou opět pravidelný simplex. Nový vrchol simplexu se sestrojí podle vztahu

$$x = x_m + 2(c - x_m) = 2c - x_m \tag{18}$$

$$c = \frac{1}{n} \sum_{j=1}^{n+1} x_j \tag{19}$$

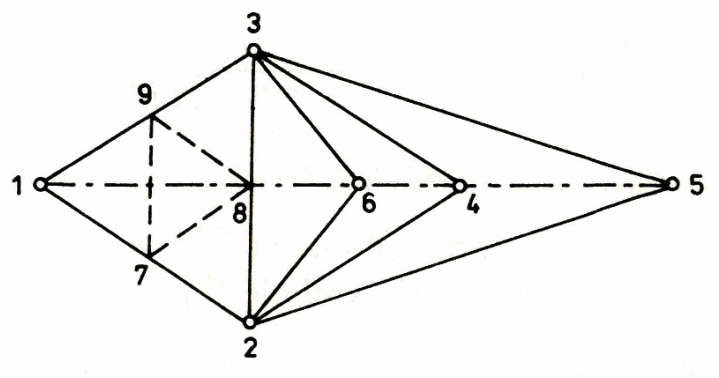
Bod x_m je bod s nejnižší hodnotou účelové funkce.



Obr. 7. Hledání extrému funkce $f(x_1,x_2)$ simplexovou metodou

Ponecháním konstantní délky hrany simplexu ρ po celou dobu výpočtu stejnou, konverguje metoda zpočátku velmi pomalu. Při nevhodné volbě délky hrany simplexu ρ dochází k oscilaci kolem extremálního bodu.

Metodu pravidelného simplexu lze vylepšit tak, že délka hrany simplexu se v průběhu výpočtu zmenšuje.



Obr. 8. Princip flexibilní simplexové metody

Tím se zlepší konvergence a nedochází k oscilacím okolo extremálního bodu. Upravená metoda se nazývá metodou flexibilního simplexu nebo též metoda Nelder-Mead dle tvůrců.

Zkonstruují se vrcholy simplexu x(1),...,x(n+1) s maximální hodnotou v x(i). K dalšímu bodu se postupuje podle vztahů:

$$\textit{jestliže} \quad f(x(i)) > \max\{f(x(1)), f(x(2)),..., f(x(n+1))\}, \ \textit{pak}$$

$$x(i+1) = c + \alpha(x(i) - c) \qquad \alpha \in (1;\infty)$$

1. expanze:

$$\textit{jestliže} \quad f(x(i+1)) > \max\{f(x(1)),..., f(x(n+1))\}, \ \textit{pak}$$

$$\textit{se pokračuje v postupu úspěšným směrem};$$

$$\textit{jestliže} \quad f(x(i)) < f(x(j)) \qquad j \neq i, \textit{pak}$$

2. kontrakce:

$$x(i+1) = c + \beta(x(i) - c) \qquad \beta \in (1;1)$$

$$\textit{jestliže} \quad f(x(i+1)) < f(x(k)), \textit{pak se zmenší délka simplexu}$$

3. redukce:

$$\textit{podle vztahu} \quad x(j) = x(k) + \frac{1}{2}(x(j) - x(k))$$

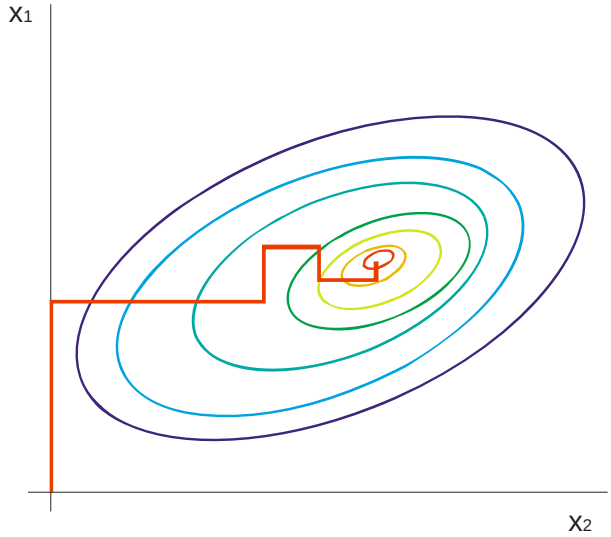
Jedná se o jednu z nejdokonalejších komparativních metod.

Metoda cyklické záměny parametrů

Tato metoda se někdy nazývá Gauss-Seidlova metoda. Při optimalizaci v R^n se zvolí jedna proměnná x_k a ostatní se nechají konstantní. Najde se částečné optimum nějakou metodou jednorozměrné optimalizace a proměnná x_k se na této hodnotě zafixuje. Přejde se k další proměnné x_j a opět se hledá jednorozměrný extrém. Tímto způsobem se vystřídají všechny proměnné a cyklus se opakuje. Uvedený postup však nemusí být vždy konvergentní a konvergence závisí na pořadí proměnných. Pro všeobecný tvar účelové funkce se metoda programuje obtížně.

I v této metodě se začíná definováním počátečního bodu S. Nejlépe je znovu vycházet ze znalosti problému a snažit se zvolit bod co nejblíže extrému. Dalším parametrem, který je zapotřebí je přesnost ε a používá se pro ukončení algoritmu. Posledním parametrem je počet proměnných účelové funkce.

V prvním kroku výpočtů je do funkce dosazen počáteční bod, přičemž k hodnotě dosazované za první proměnnou je přičten parametr ξ_1. Tímto vznikne parametrizovaná funkce a následuje vyhledání extrému funkce jedné proměnné pomocí derivace. Položením derivace rovno nule získáme hodnotu parametru ξ_1. Nový bod M_1, který se přiblíží k hledanému extrému dostaneme součtem parametru ξ_1 a hodnotou proměnné počátečního bodu - $M_1[sx1+\xi_1;sx2]$. Nyní následuje analogický postup pro další proměnnou s tím rozdílem, že bude použito nového bodu M_1. Opět nalezneme extrém a provedeme součet hodnot. Dostaneme nový bod $M_2[sx1+\xi_1; sx2+\xi_2]$. Bod M_2 je konečným stavem první iterace. Další iterace je provedena pokud platí podmínka $\xi_1^2+\xi_2^2 \geq \varepsilon$.



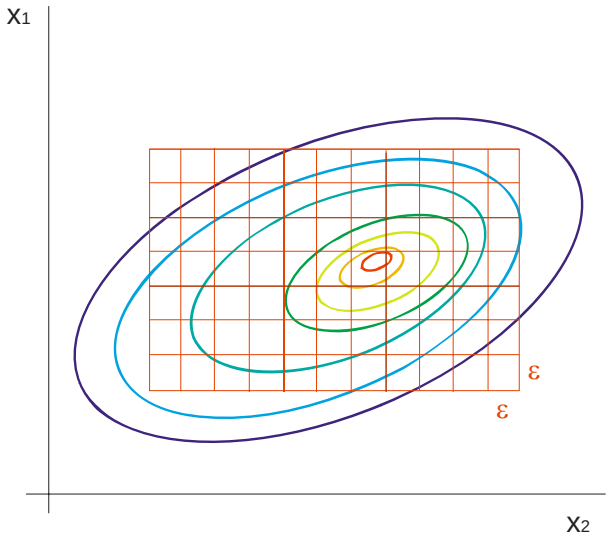
Obr. 9. Princip metody Gauss-Seidl

Zápis algoritmu:

- i=0 poč. cyklů; def. přesnost ε>0, $x^{(0)}=(x_1^{(0)},\ldots,x_N^{(0)})$
- k=0 poč. proměnných
- k=k+1 a def. fce $p_k(\xi k)=f(x_1^{(e)},\ldots,x_k^{(e)}+\xi_k,\ldots x_N^{(e)})$
- Nalezení optima pro $p_k(\xi_k)$
- Změna k-té souřadnice v $x^{(e)}$ na $x_k+\tilde{\xi}_k$
- Je-li k<N=> skok na 3
- (tj. k=N): je-li $\sum_{k=1}^{N}\xi_k^2<\varepsilon \Rightarrow STOP$, jinak i=i+1 a skok na 2

Metoda mapování kriteriální plochy

V diskretizovaných bodech povoleného podprostoru se zkoumají hodnoty účelové funkce. Postupuje se tak, že všechny nezávislé proměnné jsou konstantní a mění se jen jedna od dolní meze po horní. Potom se změní o krok další nezávislá proměnná a postup se opakuje. Nevýhodou je, že při zvyšování počtu nezávislých proměnných se neúměrně zvyšuje počet vyčíslení účelové funkce. Krok diskretizace je třeba předem stanovit. Programová realizace je velmi jednoduchá. Výpočet hodnot účelové funkce se realizuje ve vnořených cyklech. Po vyčíslení všech hodnot je třeba jednoduchým porovnávacím algoritmem najít extremální prvek a k němu hodnotu nezávislých proměnných. Toto porovnávání lze provádět přímo ve vnořených cyklech.



Obr. 10. Princip metody mapování kriteriální plochy

Zápis algoritmu:

- Volba hranic prohledávané plochy
- Volba přesnosti ε
- Rozdělení vymezujících intervalů na podintervaly o velikosti ε
- Nalezení všech průsečíků
- Výpočet funkčních hodnot ve všech průsečících
- Nalezení maximální (minimální) hodnoty a její pozice

2.2 Gradientní metody

Gradientními metodami jednorozměrné a vícerozměrné optimalizace budeme rozumět iterační algoritmus, ve kterém přírůstek vektoru parametrů je úměrný gradientu účelové funkce. Iterační vztah lze všeobecně vyjádřit jako

$$x(k+1) = x(k) + \lambda_k \nabla f(x(k)). \tag{20}$$

2.2.1 Jednorozměrné gradientní metody

Nejpoužívanější metody jednorozměrné gradientní optimalizace jsou Newtonova metoda a metoda regula-falsi. Spíše než praktický význam mají tyto metody význam teoretický, protože jsou východiskem pro odvození složitějších a lépe konvergujících postupů. Pro hladké kvadratické funkce dávají obě metody hodnotu extrému hned v prvním kroku. Rychlejší konvergenci lze očekávat u metod, které využívají druhé derivace funkce.

Newtonova metoda

Metoda spočívá v sestrojení tečny v bodě $[x_k, f'(x_k)]^T$ funkce $f'(x)$, viz (Obr.11.). Průsečík tečny s osou x, tj. bod x_{k+1} je (k+1)-ní iterace optimálního bodu x^*.

Účelová funkce se aproximuje prvními třemi členy Taylorova rozvoje

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2} f''(x_i)(x - x_i)^2 \tag{21}$$

Derivováním (21) a vyřešením $f'(x_{i+1})=0$ se získá

$$f'(x_i) + f''(x_i)(x_{i+1} - x_i) = 0 \tag{22}$$

z čehož plyne iterační vztah pro x_{i+1} ve tvaru

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)} \tag{23}$$

Metoda vyžaduje existenci druhé derivace.

Za počáteční aproximaci je vhodné volit takový bod x_1, který vyhovuje nerovnosti

$$f'(x_1) f'''(x_1) > 0 \tag{24}$$

Konvergence Newtonovy metody je tím rychlejší, čím více se účelová funkce $f(x)$ blíží kvadratické parabole, pro kterou je přesné řešení hned po jednom kroku.

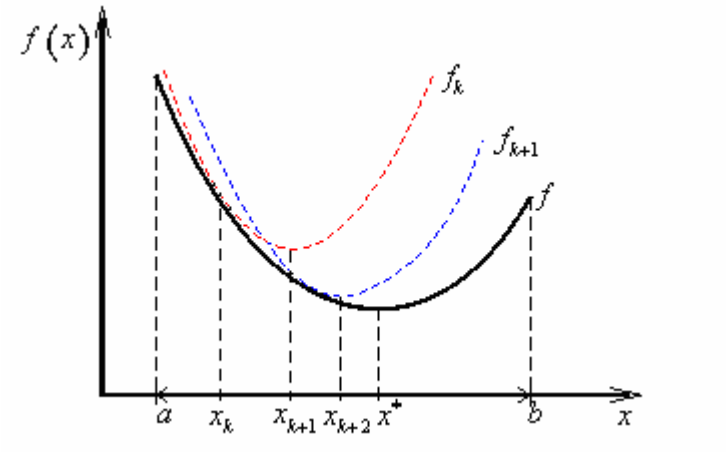
Nevýhodou Newtonovy metody je nutnost počítat první a druhou derivaci v každé iteraci.

Newtonova metoda je vhodná pro ryze konvexní (konkávní) účelové funkce, u kterých lze snadno vypočítat první a druhou derivaci (derivace musí být spojité). Konvergence je velmi rychlá, ale požadavky na vlastnosti účelové funkce jsou velmi ostré.[6]

U Newtonovy metody se nepočítá počet kroků iterace, výpočet se ukončí, když platí:

$$\left| x_{i+1} - x_i \right| \le \varepsilon \tag{25}$$

V praxi je možno použít modifikaci Newtonovy metody spočívající v tom, že když druhá derivace $f''(x)$ se již mnoho nemění, lze ji ponechat beze změny i pro následující iterace, tj. druhá derivace se nahradí konstantou (tzv. Whittakerova metoda – zmínka na [6]).



Obr. 11. Princip Newtonovy metody

Regula-falsi

Při metodě regula-falsi se účelová funkce aproximuje následujícím polynomem

$$p(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{f'(x_{i-1}) - f'(x_i)}{x_{i-1} - x_i} \cdot \frac{(x - x_i)^2}{2} \tag{26}$$

Další bod iterace se najde z podmínky extrému $p'(x_{i+1})=0$. Derivováním předchozího vztahu dostaneme

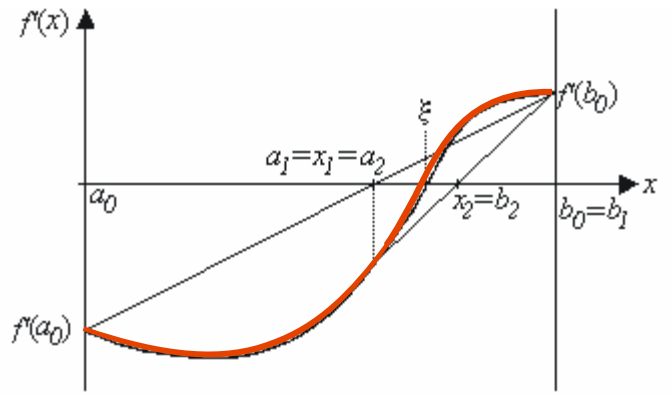
$$p'(x)/_{x=x_i} = f'(x_i) + \frac{f'(x_{i-1}) - f'(x_i)}{x_{i-1} - x_i}(x_{i+1} - x_i) = 0 \tag{27}$$

Z (27) vyplývá iterační předpis ve tvaru

$$x_{i+1} = x_i - f'(x_i)\frac{x_{i-1} - x_i}{f'(x_{i-1}) - f'(x_i)} \tag{28}$$

Druhá derivace z Newtonovy metody je zde nahrazena numerickým odhadem:

$$f''(x_i) = \frac{f'(x_i) - f'(x_{i-1})}{x_i - x_{i-1}} \tag{29}$$



Obr. 12. Princip metody regula-falsi

Metoda regula falsi potřebuje dva inicializační body x_1 a x_2 . Metoda vyžaduje existenci první derivace na celém intervalu, ve kterém hledáme extrém.

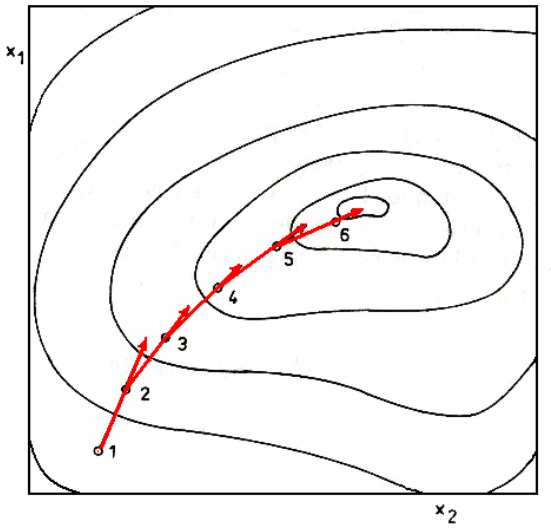
2.2.2 Vícerozměrné gradientní metody

Gradientní metoda s krátkým krokem

Algoritmus výpočtu se realizuje podle vztahu

$$x_{i+1} = x_i + \lambda_i \nabla f(x_i) \tag{30}$$

Tato metoda používá k výpočtu konstantní λ_i . Takovýto výpočet je poměrně jednoduchý, ale úspěšná konvergence posloupnosti není vždy zaručena. Při konstantním λ_k je možné přeskočení hledaného extrému. Ani volba $\lambda_i = \lambda_0 * 2^{-i}$ tento problém neodstraňuje. Pro výpočet minima funkce musí být λ_i záporné.



Obr.13. Hledání extrému gradientní metodou

Gradientní metoda s dlouhým krokem

Byly vypracovány metody, které mění λ_k v každém kroku tak, aby se dosáhlo co největšího možného zlepšení účelové funkce směrem k minimu anebo k maximu. Nejběžnější z těchto metod je gradientní metoda s dlouhým krokem.

Při této metodě se volba uskutečňuje podle vztahu

$$\lambda(i) = \max\{\lambda; \frac{d}{d\lambda} f(x(i) + \lambda \nabla f(x(i)) = 0\} \tag{31}$$

Tento postup zabezpečuje řešení v menším počtu kroků, ale vyžaduje větší počet výpočtů. V každém kroku iterace je zapotřebí řešit úlohu jednorozměrné optimalizace.

Gradientní metody lze použít i v případech, kdy nejsou známy derivace. Používají se numerické odhady derivací, které mohou být symetrické anebo nesymetrické. Nejjednodušší vzorce tzv. dvojbodové mají následující tvary:

symetrický vzorec

$$\frac{\partial f}{\partial x_i} = \frac{f(x_1,...,x_i + \Delta,...,x_n) - f(x_1,...,x_i - \Delta,...,x_n)}{2\Delta} \tag{32}$$

levý nesymetrický vzorec

$$\frac{\partial f}{\partial x_i} = \frac{f(x_1,...,x_i + \Delta,...,x_n) - f(x_1,...,x_i,...,x_n)}{\Delta} \tag{33}$$

nebo pravý nesymetrický vzorec

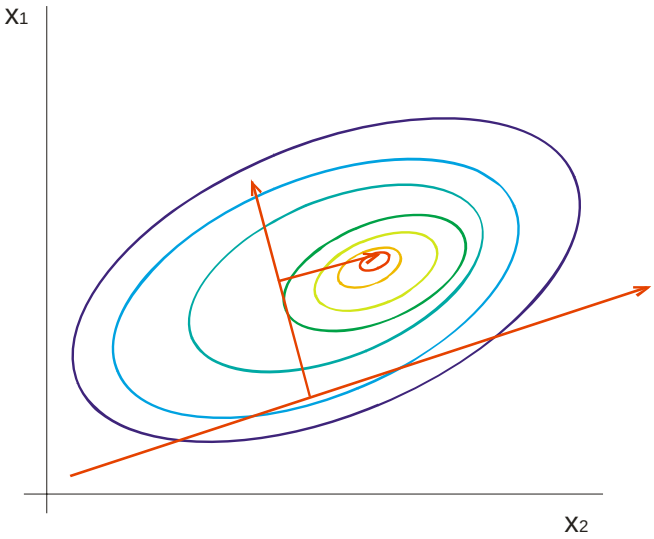
$$\frac{\partial f}{\partial x_i} = \frac{f(x_1,...,x_i,...,x_n) - f(x_1,...,x_i - \Delta,...,x_n)}{\Delta} \tag{34}$$

Pro druhou derivaci

$$\frac{\partial^2 f}{\partial x_i^2} = \frac{f(x_1,...,x_i + \Delta,...,x_n) - 2f(x_1,...,x_n) + f(x_1,...,x_i - \Delta,...,x_n)}{2\Delta} \tag{35}$$

kde Δ je malé kladné číslo. Jiné vzorce pro více bodů výpočtu udává numerická matematika.

Existují i další varianty gradientních metod s dlouhým krokem, které se snaží jiným, efektivnějším způsobem nahradit jednorozměrnou extremalizaci podle λ v iteračním postupu. Tyto metody jsou potom vhodné pro programování na počítačích.[4]



Obr. 14. Princip gradientní metody s dlouhým krokem

Metoda konjugovaných gradientů - CONGRA

Iterační algoritmus lze popsat následujícími vztahy

$$d(0) = \nabla f(x(0)) \qquad x(0) - počáteční \quad bod \quad postupu$$

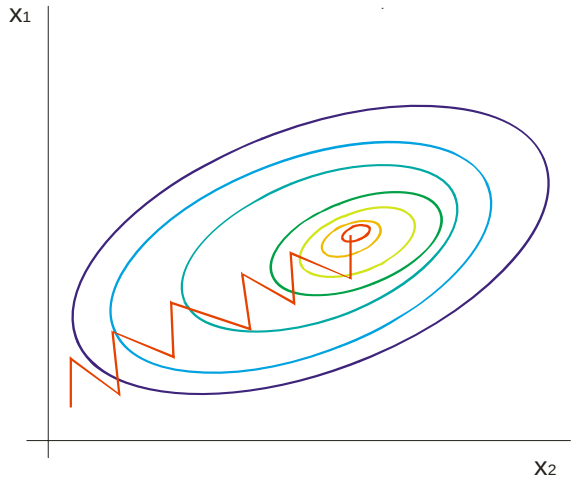
$$x(i+1) = x(i) + \lambda(i)d(i)$$

$$\lambda(i) = -\frac{\nabla f^T(i)d(i)}{d^T(i)\nabla^2 f(i)d(i)}$$

$$d(i+1) = \nabla f(x(i+1)) + \delta(i)d(i)$$

$$\delta(i) = -\frac{\nabla f(i+1) \quad \nabla f(i)}{d^T(i) \quad \nabla f(i) \quad d(i)}d(i)$$

Negativní vlastností metody je, že vyžaduje v každém kroku výpočet matice $\nabla^2 f_i$.



Obr. 15. Princip metody konjugovaných gradientů

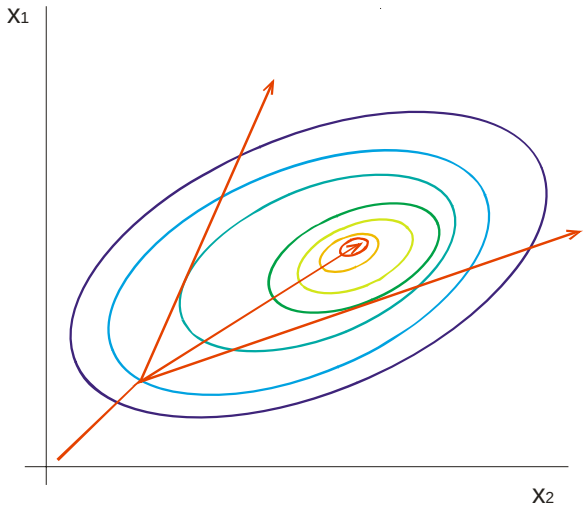
Metoda paralelních tečen – PARTAN

Metoda je podobná metodě konjugovaných gradientů, kdy se vytvoří dva paralelní směry, ze kterých se vybere jistý průměr.

Zápis algoritmu:

1. $volba \quad počátečního \quad bodu \quad x(0); \quad volba \quad N$
2. $i = 1, \quad x(1) = x(0) + \lambda \nabla f(0) \quad pro \quad \lambda, které \quad \max imalizuje \quad f(x(0) + \tilde{\lambda} \nabla f(0))$
3. $y(i) = x(i) + \tilde{\lambda}(i) \nabla f(i), kde \quad \tilde{\lambda}(i) \quad \max imalizuje \quad f(x(i) + \tilde{\lambda} \nabla f(x(i)))$
4. $x(i+1) = x(i-1) + \delta(i)[x(i) - x(i-1)], kde \quad \delta(i) \quad \max imalizuje$

   $f(x(i+1)) = f(x(i-1) + \delta(i)[y(i) - x(i-1)])$
5. $jestliže \quad i < N, \quad pak \quad i = i+1 \quad a \quad skok \quad na \quad 3,$
6. $jestliže \quad i = N, \quad pak \quad x(0) = x(N) \quad a \quad skok \quad na \quad 2$



Obr. 16. Princip metody PARTAN

DFP (Davidon, Fletcher,Powell)

Tato metoda se opírá o algoritmus Newtonovy metody, ve které se namísto Hessovy matice druhých parciálních derivací konstruuje pouze její odhad. Kombinuje se zde jednoduchost gradientní metody s dobrou konvergencí metody Newtonovy.
Princip vychází z iteračního vztahu

$$x(i+1) = x(i) + \lambda(i)S(i)\nabla f(i) \tag{36}$$

kde S(i) je matice, která aproximuje inverzi Hessovy matice. Výpočet DFP metodou lze popsat následujícím algoritmem:

1. *volba počátečního bodu* $x(0)$ *a pozitivně definitní matice* $S(0); i = 0$
2. $d(i) = S(i)\nabla f(i)$
3. $x(i+1) = x(i) + \lambda(i)d(i), kde \quad \lambda(i) \quad \max imalizuje \quad f(x(i) + \lambda(i)d(i))$

   $a \quad výpočet \quad f(x(i+1))$
4. $v(i) = \lambda(i)d(i) \qquad w(i) = \nabla f(i) - \nabla f(i+1)$

   $$S(i+1) = S(i) + \frac{v(i)v^T(i)}{v^T(i)w(i)} - \frac{S(i)w(i) \quad w^T(i)S(i)}{w^T(i) \quad w(i)}$$
5. $i = i+1 \quad a \quad skok \quad na \quad 2.$

2.2.3 Gradientní metody s omezením

Použití gradientních metod pří vícerozměrné optimalizaci naráží na problémy ve dvou případech. V první řadě jde o případy neanalyticky zadaných funkcí, kde vyjádření parciálních derivací je problematické, protože se musí použít numerických odhadů. V druhé řadě je to v případě omezení kladených na nezávislé proměnné.

Metoda projekce gradientu

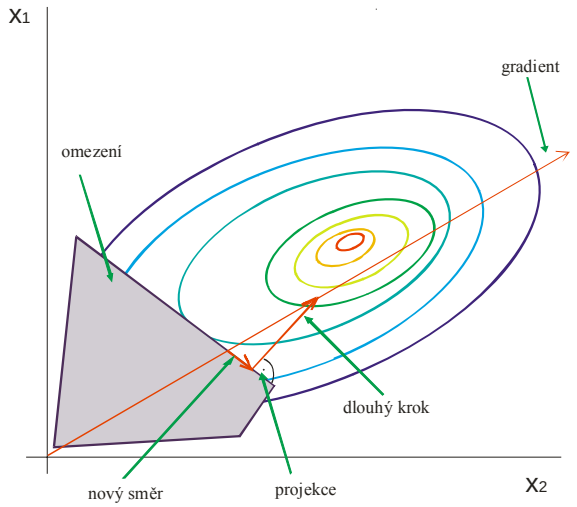
Metoda je schopná akceptovat omezení ve tvaru konvexních množin (nerovnic).
Algoritmus vychází z běžného gradientního postupu, při kterém se kontroluje, jestli nedochází k překročení omezení. Jestliže k překročení dojde, vykoná se geometrický průmět tohoto bodu na hranici omezení. Z tohoto bodu se potom opět postupuje gradientní metodou s dlouhým krokem.

Zápis algoritmu:

1. *sestavení* Q
2. $q = (QQ^T)^{-1}Q \quad \nabla f(x(k))$
3. $test: \quad jestliže \quad q_k < 0 \quad k > 1, pak \quad vynechat \quad k-tý \quad řádek, \quad sestrojit \quad Q^*$

   $a \quad skok \quad na \quad 2;$

   $jestliže \quad q_k > 0 \quad pro \quad všechny \quad k \geq 1, pak \quad konec$
4. $P = (I - Q^T(QQ^T)^{-1}Q) \ldots projekční \quad matice$
5. $s = P\nabla f(x(k))$
6. $\tilde{\lambda} = \max\{\lambda, x(k) + \lambda s \quad je \quad přípustné\}$
7. $x(k+1) = x(k) + \tilde{\lambda}s$
8. $skok \quad na \quad bod \quad 1.$



Obr. 17. Princip metody projekce gradientu

2.3 Metody náhodného vyhledávání

Při těchto metodách je iterační přírůstek proměnných závislý na náhodně generované veličině. V zásadě jde tyto metody rozdělit do dvou tříd:

- jednoduché (striktně náhodné) metody, při kterých se pouze vyhodnocuje, jestli je nový bod lépe položený než předcházející. Jestliže to tak není, strategie se vrací do předcházejícího bodu a generuje se nový směr a velikost kroku.

- metody s adaptací, při kterých se nějakou strategií vyhodnocuje, jestli se jde správným směrem anebo ne. Potom se mění velikost a směr kroku. Jde o kombinaci stochastické a deterministické strategie.

Konvergence algoritmů náhodného vyhledávání se všeobecně zaručit nedá a některá odvození jsou značně heuristická a intuitivní. Napříč tomu se ukazuje, že někdy jsou tyto metody efektivnější než deterministické. Hlavně v případech skutečně mnohorozměrných a v případech, kdy účelová funkce není zadána analyticky.

Uvažujme problém maximalizace účelové funkce f(x) n reálných proměnných, která je definovaná na podmnožině R^n pomocí systému nerovností. Všeobecně lze algoritmus náhodného vyhledávání popsat vztahy

$$\begin{aligned} x(i+1) &= x(i) + \Delta x(i) \\ \Delta x(i) &= a_i \psi[i; \Delta F(i), x(i)] \xi_i \\ \Delta F(i) &= F(i) - F(i-1) \end{aligned} \tag{37}$$

První rovnice charakterizuje iterační postup metod, třetí rovnice se nazývá zpětná diference účelové funkce. Podle její hodnoty se určuje „zlepšení“ anebo „zhoršení“ účelové funkce. Prostřední rovnice charakterizuje vlastní strategii postupu. V této rovnici a_i udává velikost kroku, ψ je adaptační funkce, která nějakým nenáhodným způsobem hodnotí předcházející zkušenosti iteračního postupu a ξ_i je n-rozměrný náhodný vektor.

II. PRAKTICKÁ ČÁST

3 REALIZACE PROGRAMŮ

Cílem při realizování programů bylo navrhnout a zrealizovat jednotlivé algoritmy metod a formou grafického uživatelského rozhraní zajistit jednoduché a přehledné ovládání. Prostředím pro realizaci byl vybrán software Matlab od společnosti The MathWorks. Důvodem proč byl vybrán tento software je, že na rozdíl od jiných programovacích jazyků byl Matlab vyvinut pro matematické účely a má možnost přehledné tvorby GUI (grafického uživatelského rozhraní).

3.1 Popis programového prostředí

Matlab je programové prostředí a skriptovací programovací jazyk pro vědeckotechnické numerické výpočty, modelování, návrhy algoritmů, počítačové simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. Matlab je nástroj, jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací. Tento výpočetní systém se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací nejen ve sféře vědy, výzkumu a průmyslu, ale i v oblasti vzdělávání. Matlab 6 je považován za přelom nejen z hlediska rozsahu, integrace a kvality produktu, ale především z hlediska vztahu k uživateli a jeho pohodlí při práci.
Vlastní Matlab není jen v jedné linii základního programu, ale používá se spousta rozšíření (toolbox). Nejznámější a asi nejpoužívanější je Simulink. Je to program pro simulaci a modelování dynamických systémů, který využívá algoritmy Matlabu pro numerické řešení nelineárních diferenciálních rovnic. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic.
Název MATLAB vznikl zkrácením slov MATrix LABoratory (volně přeloženo „laboratoř s maticemi“), což odpovídá skutečnosti, že klíčovou datovou strukturou při výpočtech v Matlabu jsou matice. Vlastní programovací jazyk vychází z jazyka Fortran.

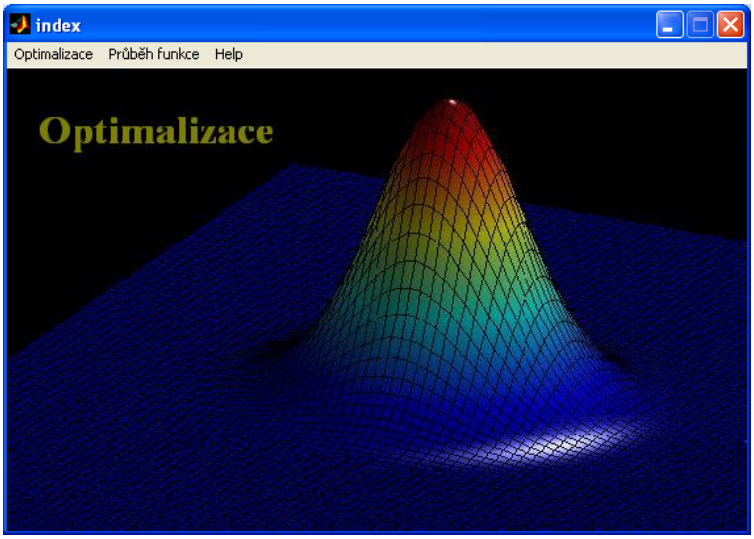
Výhoda Matlabu je nejen v jeho velkých možnostech, ale i v tom, jak je široce rozšířen v průmyslu a jeho verze existují pro řadu operačních systémů (Unix, Linux, Windows, Open VMS, IRIX, Solaris, Macintosh, HP-UX a další). [7]

Matlab je velmi mocný nástroj pro řešení a analýzu technické problematiky. Integruje výpočty, vizualizaci a programování do jednoduše ovladatelného prostředí, kde problémy a řešení jsou vyjádřeny pomocí dobře známých matematických vztahů. Typické použití zahrnuje:

- Matematiku a výpočty
- Tvorba algoritmů
- Získávání dat
- Modelování a simulace
- Analýza dat, výzkum a vizualizace
- Vědecká a inženýrská grafika
- Tvorba aplikací, včetně grafického rozhraní

Nejsilnější vlastností Matlabu je práce s datovými poli, které není potřeba dimenzovat. To umožňuje řešit mnoho technických problémů s použitím formulací pomocí vektorů a matic.

Při výběru metod, které byly naprogramovány, padla volba na Fibonacciho metodu, metodu zlatého řezu, Newtonovu metodu a metodu regula-falsi jako zástupce jednorozměrných optimalizačních metod a Box-Wilsonovu metodu, metodu flexibilního simplexu (Nelder-Mead), Newtonovu metodu, gradientní metody s krátkým a dlouhým krokem a metodu DFP jako zástupce vícerozměrných optimalizačních metod.
Kvůli lepší názornosti a možnosti grafického znázornění extrému, byly vícerozměrné metody naprogramovány pro dvě proměnné. Pro násilné ukončení a ochranu algoritmu proti zacyklení u metod, u kterých toto hrozí, nebylo zvoleno počtu iterací, ale čas po který má algoritmus běžet.
Program se spouští pomocí souboru index.m. Po jeho spuštění se nám otevře úvodní okno se třemi Menu nabídkami (Optimalizace, Průběh funkce, Help). V Menu Optimalizace máme možnost volby ze tří submenu (2D, 3D, Zavřít). 2D slouží k otevření okna pro výběr optimalizačních metod pro jednu proměnnou, 3D pro metody dvou proměnných a Zavřít slouží k zavření okna. Menu Průběh funkce slouží čistě jen pro vykreslení grafu námi definované funkce na zvoleném intervalu. Dělení je opět provedeno pro funkce jedné a dvou proměnných. V Menu Help jsou uvedeny popisy jednotlivých polí, použitých v programech.



Obr. 18. Úvodní okno programu

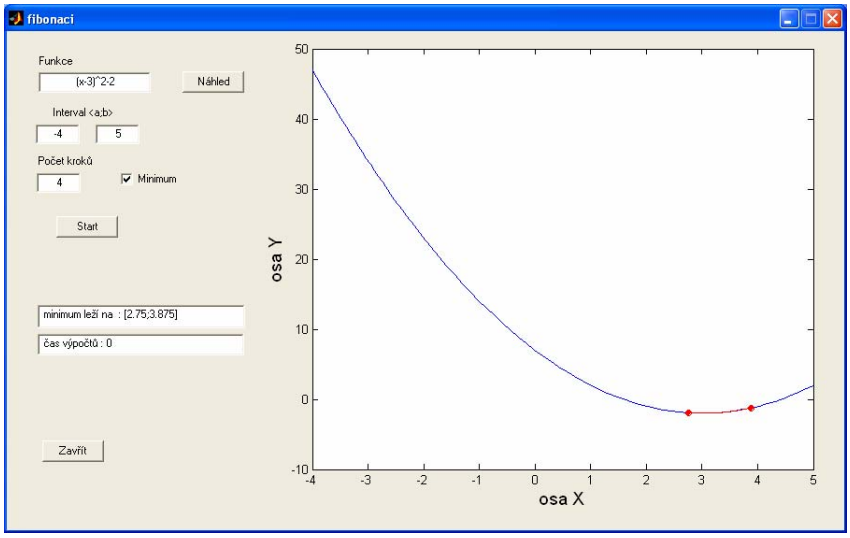
Obr. 19. Ukázka oken pro výběr optimalizačních metod

V oknech jednotlivých optimalizačních metod jsou pole pro zadávání vstupních parametrů, které daná metoda potřebuje. Dále tlačítko Náhled, které vykreslí graf námi zadané účelové funkce na námi zadaném intervalu bez provedení výpočtů. Výpočty se spustí tlačítkem Start a zavření okna se provede tlačítkem Zavřít.

3.1.1 Fibonacciho metoda

Tato metoda je naprogramována tak, aby hledala maximum funkce na zadaném intervalu, avšak pomocí zaškrtávacího políčka (checkbox) je možno změnit vyhledávání na minimum. Před spuštěním je nutno zadat účelovou funkci, interval na kterém se bude

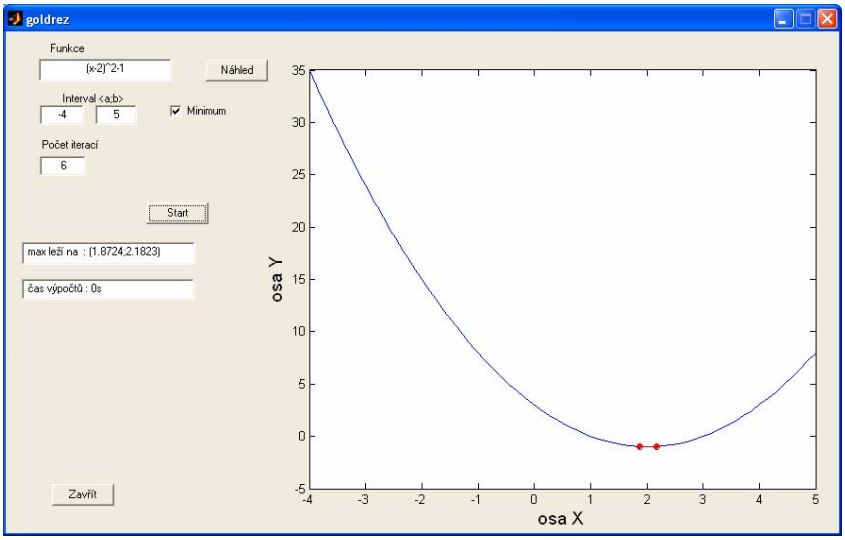
hledat daný extrém a počet iterací. Po provedení výpočtů je výsledek vypsán do textových polí. Je zde uvedeno o jaký extrém se jedná, na jakém intervalu leží a čas běhu výpočtů a dále je vykreslen graf účelové funkce se znázorněním výsledného intervalu.



Obr. 20. Okno Fibonacciho metody

3.1.2 Metoda zlatého řezu

Tato metoda je opět naprogramována tak, aby hledala maximum funkce na zadaném intervalu, avšak volbu extrému je opět možno změnit zaškrtávacím políčkem. Před spuštěním je nutno zadat účelovou funkci, interval na kterém se bude hledat daný extrém a počet iterací. Po provedení výpočtů je výsledek prezentován stejně jako u Fibonacciho metody.

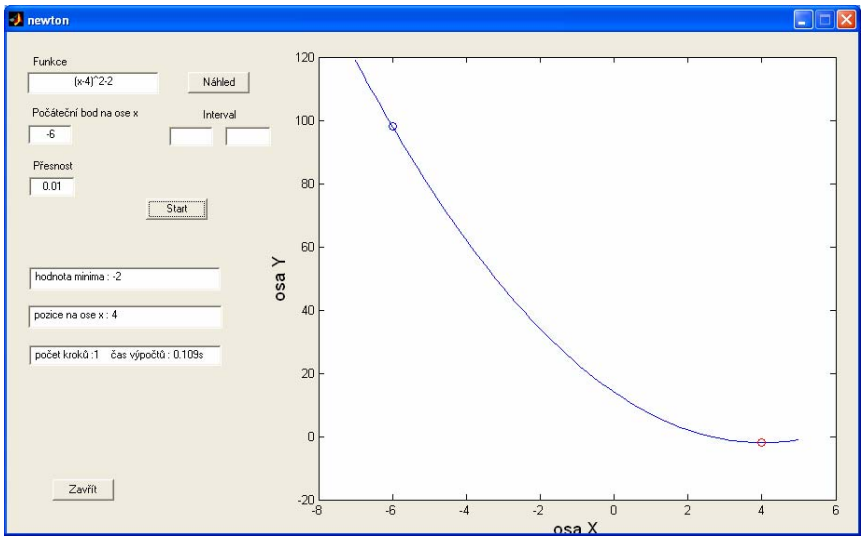


Obr. 21. Okno metody zlatého řezu

3.1.3 Newtonova metoda

Jedná se o metodu, která hledá extrém pomocí derivace účelové funkce, tudíž není třeba zvlášť rozlišovat, jaký extrém se má vyhledávat. Zjištění typu extrému se provede dosazením bodů z pravého a levého okolí nalezeného extrému. Před spuštěním výpočtů je nutno zadat počáteční bod na ose x a požadovanou přesnost, po které bude algoritmus ukončen. Tato přesnost je nejmenší možná vzdálenost mezi dvěma mezivýsledky ve dvou po sobě následujících iteracích.

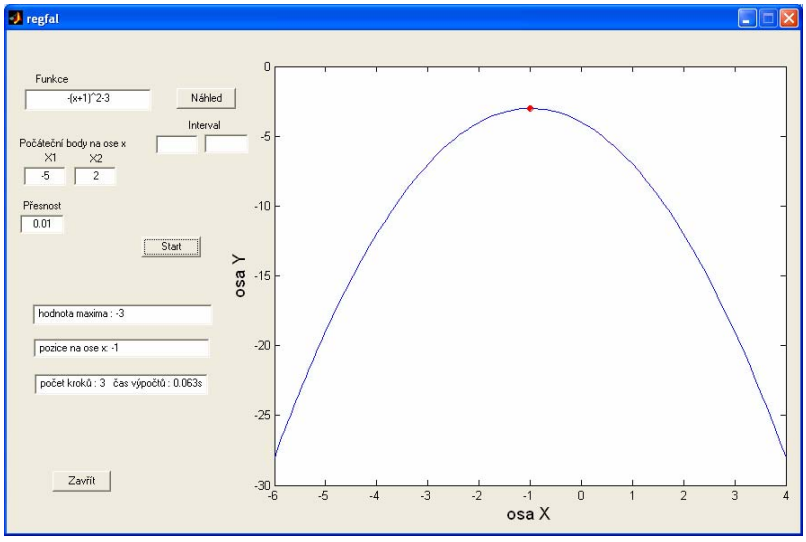
Výsledky jsou pak prezentovány výpisem typu extrému, jeho hodnotou, pozicí na ose x, počtem iterací a časem výpočtů. Dále je vykreslen graf účelové funkce se znázorněním počátečního bodu a výsledného bodu.



Obr. 22. Okno Newtonovy metody

3.1.4 Regula-Falsi

Postup při zadávání vstupních parametrů je stejný jako u Newtonovy metody. Rozdíl je však v tom, že zde zadáváme dva počáteční body na ose x. Po provedení výpočtů jsou výsledky prezentovány opět jako u Newtonovy metody a v grafu znázorněn výsledný extrém.

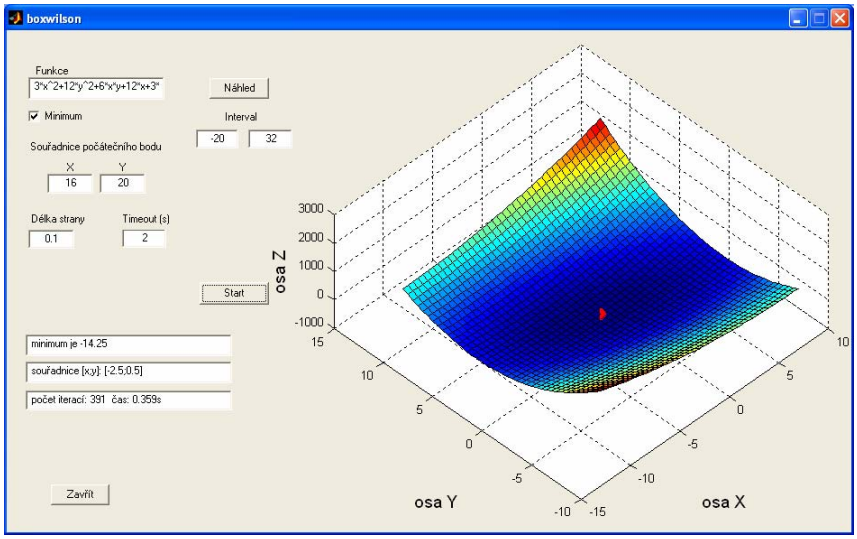


Obr. 23. Okno metody regula-falsi

3.1.5 Box-Wilsonova metoda

Metoda je naprogramována pro vyhledání minima i maxima účelové funkce a tuto volbu určujeme opět pomocí zaškrtávacího políčka. Dále pak zadáváme souřadnice počátečního bodu a délku hrany čtverce, který se bude dle algoritmu konstruovat. Argument Timeout slouží k násilnému přerušení běhu algoritmu, a to z důvodu možnosti špatné volby hledaného extrému. Pokud neznáme přibližný tvar účelové funkce, je možno užít náhledu na zvoleném intervalu a poté zvolit vyhledávání maxima či minima.

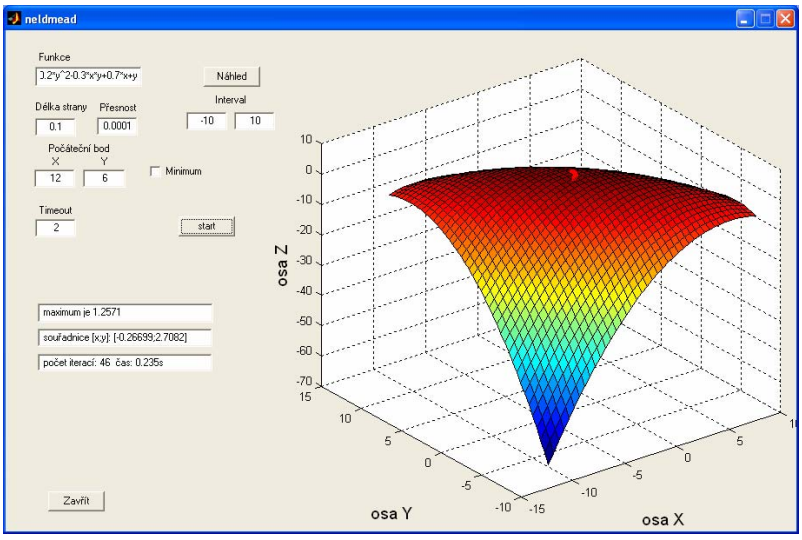
Po provedení výpočtů jsou výsledky prezentovány v textových polích, a to výpisem druhu extrému a jeho hodnotou, souřadnicemi extrému, počtem iterací a časem výpočtů. Ve vykresleném grafu je znázorněn extrém červeným bodem.



Obr. 24. Okno Box-Wilsonovy metody

3.1.6 Metoda flexibilního simplexu

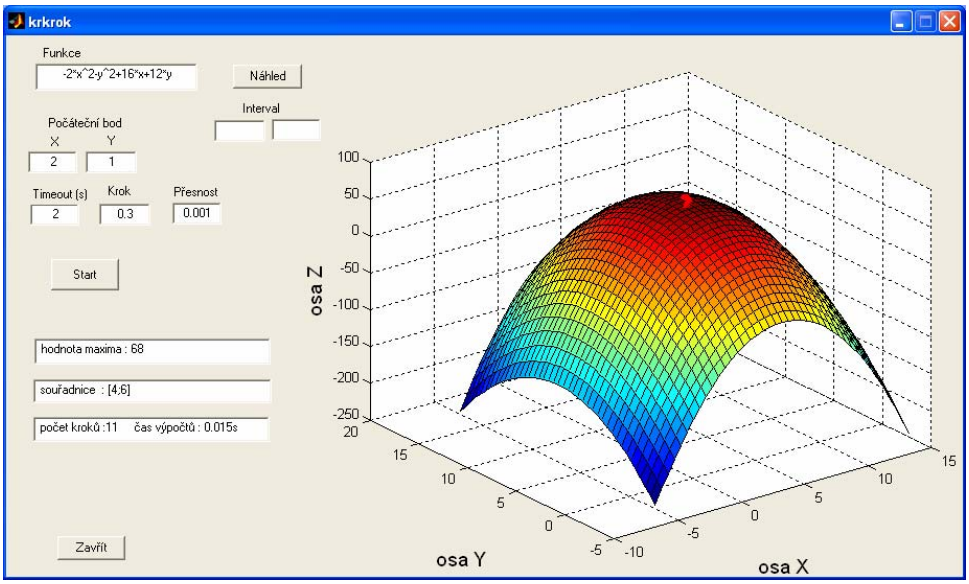
Tato metoda je v zadávání vstupních argumentů stejná jako Box-Wilsonova metoda. Zde je navíc vstupní parametr přesnost, který slouží k ukončení výpočtů. Je to nejmenší rozdíl mezi hodnotami účelové funkce ve vrcholech simplexu. Pokud je rozdíl hodnot účelové funkce ve všech vrcholem menší než zadaná přesnost, je algoritmus ukončen a jsou vypsány výsledky. Konstrukce simplexu je provedena dle (16) a (17). Parametry pro redukci a kontrakci byly zvoleny hodnoty α=2,5 a β=0,5 jak je doporučeno v [3].



Obr. 25. Okno metody Nelder-Mead

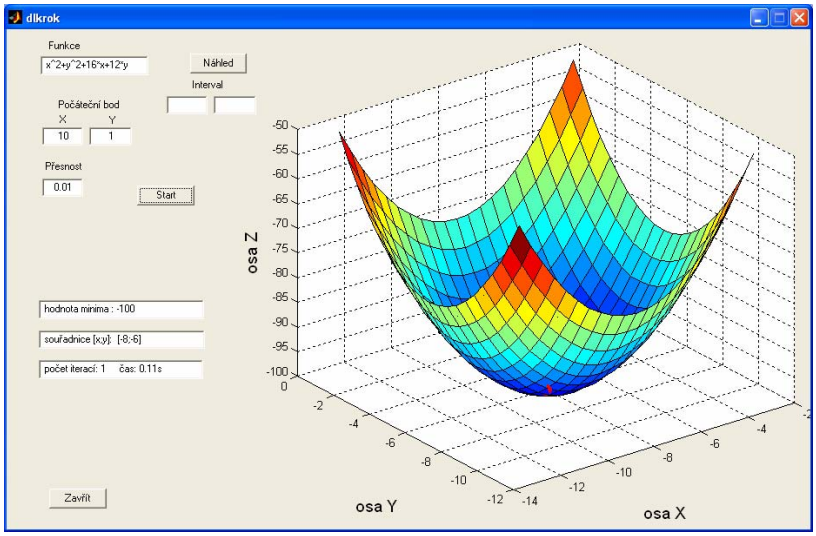
3.1.7 Gradientní metoda s krátkým a dlouhým krokem

U gradientní metody s krátkým krokem musíme zadat počáteční bod, velikost parametru λ (Krok) a Timeout, který slouží k násilnému ukončení programu, aby bylo ošetřeno možné přeskočení extrému a tím k zacyklení běhu algoritmu. Při této události je vypsána hláška s upozorněním k volbě jiné velikosti kroku. Jinou možností je také změnit počáteční bod. Při požadavku vyhledávat minimum účelové funkce musíme zadat parametr λ<0.



Obr. 26. Okno gradientní metody s krátkým krokem

U gradientní metody s dlouhým krokem se nezadává parametr λ a není zapotřebí zadávat parametr Timeout, protože za normálních podmínek by nemělo dojít k zacyklení běhu algoritmu. Avšak uvnitř kódu je tato doba omezena na 10s.

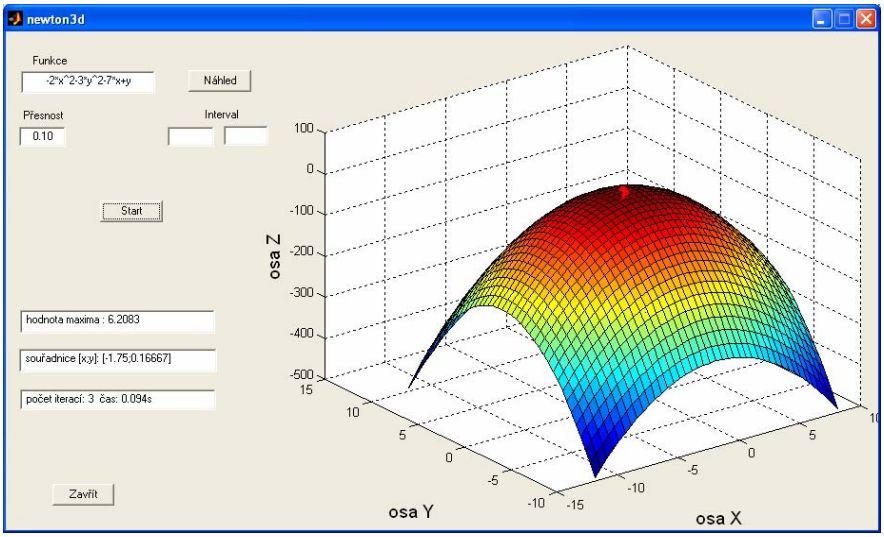


Obr. 27. Okno gradientní metody s dlouhým krokem

3.1.8 Newtonova metoda

Při této metodě je nutné invertovat Hessovu matici druhých parciálních derivací, což nemusí být vždy jednoduchá záležitost. Praktické výpočty potvrzují rychlou konvergenci metody.

Vstupním parametrem je Přesnost pro ukončení a počáteční bod.

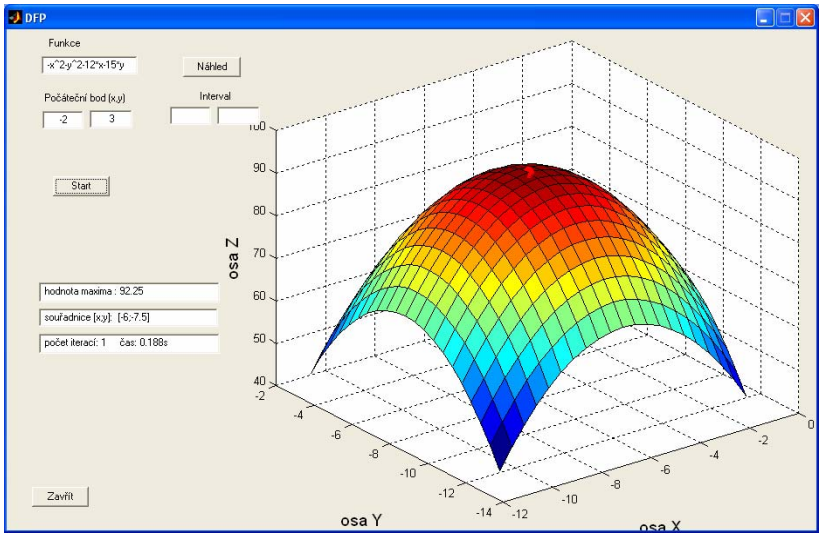


Obr. 28. Okno Newtonovy metody

3.1.9 DFP

DFP je zástupce kvazinewtonských metod. Jde o metody, které se snaží spojit jednoduchost gradientní metody a dobrou konvergenci metody Newtonovy. DFP je základní metoda toho typu.

Vstupním parametrem je pouze počáteční bod. Za symetrickou pozitivně definitní matici S, která je zmíněna v (36) byla zvolena matice $S_i=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$.



Obr. 29. Okno DFP metody

3.2 Testování metod

Funkce: $f(x) = \frac{3}{4}x - (x-1)^2$

Analytické řešení:

$f'(x) = \frac{11}{4} - 2x$

$f'(x) = 0 \quad \Rightarrow \quad x_0 = 1,375$

$f(x_0) = 0,8906$

Zápis funkce ve tvaru pro Matlab: 3/4*x-(x-1)^2

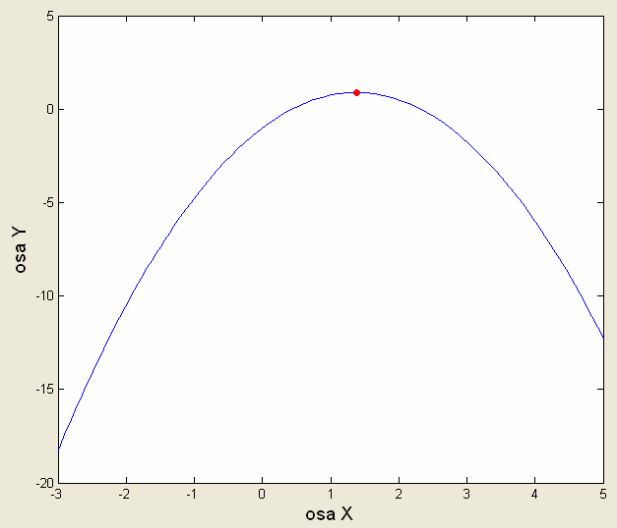
Tab. 1: Výsledky pro funkci $f(x) = \frac{3}{4}x - (x-1)^2$

$f(x) = \frac{3}{4}x - (x-1)^2$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Fibonacci	Interval <-5;5> Počet kroků: 6	Hledáno maximum Interval <1,1905; 1,6667> Čas: 0,001s	Interval <-6;9> Počet kroků: 17	Hledáno maximum Interval <1,3762;1,3798> Čas: 0,001s
Zlatý řez	Interval <-5;5> Počet kroků: 6	Interval <1,1805; 1,5248> Čas: 0,001s	Interval <-6;9> Počet kroků: 17	Interval <1,3739;1,3765> Čas: 0,015s
Newton	Počáteční bod: 5 Přesnost: 0,001	Bod - minimum [1,375; 0,89063] Čas: 0,109s Počet kroků: 1	Počáteční bod: 15 Přesnost: 0,01	Bod - minimum [1,375; 0,89063] Čas: 0,109s Počet kroků: 1
Regula-Falsi	Počáteční body: X1=-12 X2=-7 Přesnost: 0,01	Bod - minimum [1,375; 0,89063] Čas: 0,063s Počet kroků: 3	Počáteční body: X1=15 X2=5 Přesnost: 0,001	Bod - minimum [1,375; 0,89063] Čas: 0,062s Počet kroků: 3

Na tomto příkladu je ukázáno, že u komparativních metod přesnost zjištěné polohy extrému závisí na velikosti počátečního intervalu a počtu iterací které se provedou. Pro předem požadovanou přesnost, lze počet iterací získat dle vztahu (13).

Při stejných podmínkách výsledky ukazují, že metoda zlatého řezu je o něco přesnější než Fibonacciho metoda.

Naproti tomu u gradientních metod se ukázalo, že u tohoto typu účelové funkce téměř nezáleží na poloze počátečních bodů. Newtonova metoda našla extrém v prvním kroku a regula-falsi ve třetím.



Obr. 30. Graf funkce $f(x)=\frac{3}{4}x-(x-1)^2$

Funkce: $f(x)=2x^3+4x^2-8x+5$

Analytické řešení:

$f'(x)=6x^2+8x-8$

$f'(x)=0 \quad \Rightarrow \quad 1) \quad x_0=-2 \qquad f(x_0)=21$

$\qquad\qquad\qquad\quad 2) \quad x_0=2/3 \qquad f(x_0)=2,037$

Zápis funkce ve tvaru pro Matlab: 2*x^3+4*x^2-8*x+5

Tab. 2: Výsledky pro funkci $f(x)=2x^3+4x^2-8x+5$

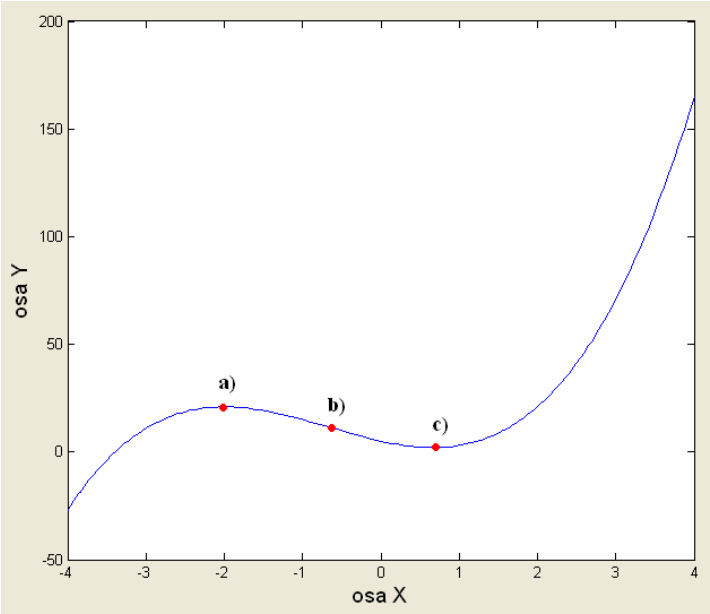
$f(x)=2x^3+4x^2-8x+5$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Fibonacci	Interval <-2;4>	Hledáno minimum Interval	Interval <-6;9>	Hledáno minimum Interval
	Počet kroků: 11	<0,6781; 0,7038> Čas: 0,016s	Počet kroků: 17	<0,6658; 0,6694> Čas: 0,015s
Zlatý řez	Interval <-5;5>	Hledáno minimum Interval	Interval <-6;9>	Hledáno minimum Interval
	Počet kroků: 6	<0,4916; 0,8359> Čas: 0,014s	Počet kroků: 17	<0,6652; 0,6678> Čas: 0,015s

Newton	Počáteční bod: -2 Přesnost: 0,01	Nalezeno maximum Bod [-2; 21] Čas: 0,12s Počet kroků: 1	Počáteční bod: 3 Přesnost: 0,001	Nalezeno minimum Bod [0,6666; 2,037] Čas: 0,125s Počet kroků: 4
Regula-Falsi	Počáteční body X1=-5 X2=3 Přesnost: 0,01	Nalezeno minimum Bod [0,6666; 2,037] Čas: 0,063s Počet kroků: 15	Počáteční body X1=-6 X2=-1 Přesnost: 0,001	Nalezeno maximum Bod [-2; 21] Čas: 0,077s Počet kroků: 17

Tento typ funkce byl zvolen z toho důvodu, aby se zjistilo co se stane, když účelová funkce nebude unimodální. Jednotlivé extrémy jsou zobrazeny na (Obr.31.). V bodě a) má funkce lokální maximum, v b) inflexní bod a v c) lokální minimum.

Jak ukazují výsledky, tak komparativní metody si s tímto problémem za námi daných podmínek poradily a na zadaném intervalu vyhledaly požadovaný extrém. I při změně vyhledávání na maximum požadovaný extrém nalezly.

U gradientních metod typ extrému který vyhledaly, záležel na poloze počátečního bodu. Pokud počáteční bod ležel v levém okolí inflexního bodu, bylo nalezeno maximum. Pokud ležel v pravém okolí inflexního bodu, bylo nalezeno minimum.



Obr. 31. Graf funkce $f(x) = 2x^3 + 4x^2 - 8x + 5$

Funkce: $f(x) = x*\sin(x)^2 + x^2 + 3$

Analytické řešení:
$f'(x) = \sin(x)^2 + 2x\sin(x)\cos(x) + 2x$

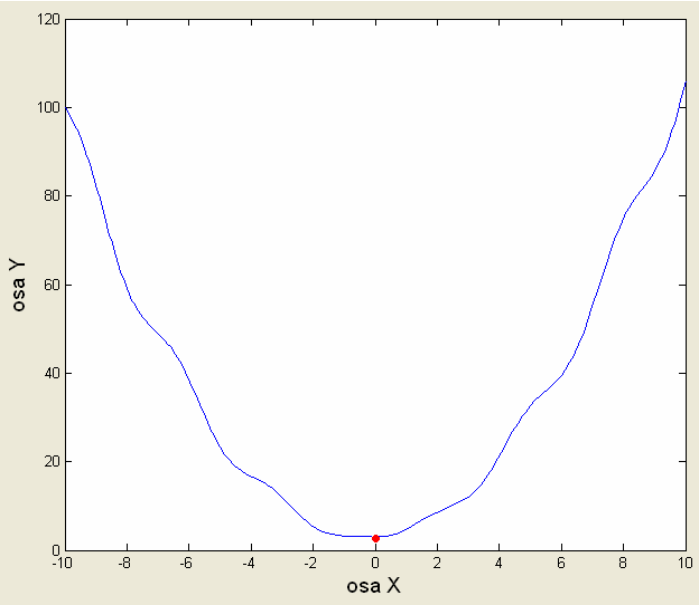
$f'(x) = 0 \quad \Rightarrow \quad x_0 = 0 \qquad f(x_0) = 3$

Zápis funkce ve tvaru pro Matlab: x*sin(x)^2+x^2+3

Tab. 3: Výsledky pro funkci $f(x) = x*\sin(x)^2 + x^2 + 3$

$f(x) = x*\sin(x)^2 + x^2 + 3$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Fibonacci	Interval <-10;10> Počet kroků: 8	Hledáno minimum Interval <-0,1818; 0,1818> Čas: 0,001s	Interval <-3;7> Počet kroků: 10	Hledáno minimum Interval <-0,0138; 0,0555> Čas: 0,001s
Zlatý řez	Interval <-10;10> Počet kroků: 8	Hledáno minimum Interval <-0,1318; 0,1311>	Interval <-3;7> Počet kroků: 10	Hledáno minimum Interval <-0,0320; 0,0181>
Newton	Počáteční bod: 10 Přesnost: 0,01	Extrém nenalezen	Počáteční bod: 2 Přesnost: 0,001	Nalezeno minimum Bod [0,00005; 3] Čas: 0,312s Počet kroků: 8
Regula-Falsi	Počáteční body X1=-5 X2=4 Přesnost: 0,01	Nalezeno minimum Bod [0,00005; 3] Čas: 0,062s Počet kroků: 19	Počáteční body X1=28 X2=15 Přesnost: 0,001	Nalezeno minimum Bod [0,00001; 3] Čas: 0,093s Počet kroků: 35

Na této funkci bylo testováno, jak si jednotlivé metody poradí se zakomponovanou periodickou funkcí sinus. Všechny metody nalezly žádaný extrém, až na Newtonovu metodu, u které byl běh násilně přerušen po uplynutí Timeoutu (10s), který je definován uvnitř kódu. Dobu zřejmě ovlivnil vzdálený počáteční bod a výpočty derivací periodické funkce. Při změně počátečního bodu, byl požadovaný extrém nalezen.



Obr. 32. Graf funkce $f(x) = x * \sin(x)^2 + x^2 + 3$

Funkce: $f(x,y) = -2x^2 - y^2 + 16x + 12y$

Analytické řešení:

$$\frac{\partial f}{\partial x} = -4x + 16$$

$$\Rightarrow \quad x_0 = 0 \quad y_0 = 0 \quad \Rightarrow \quad f(x_0, y_0) = 0$$

$$\frac{\partial f}{\partial y} = -2y + 12$$

Zápis funkce ve tvaru pro Matlab: -2*x^2-y^2+16*x+12*y

Tab. 4: Výsledky pro funkci $f(x,y) = -2x^2 - y^2 + 16x + 12y$

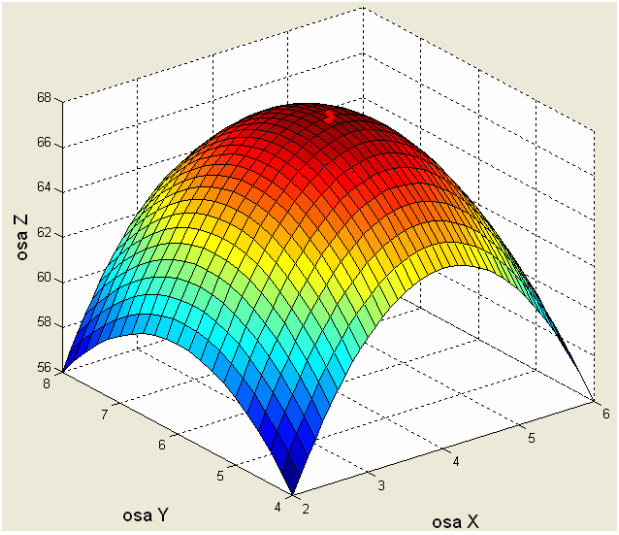
$f(x,y) = -2x^2 - y^2 + 16x + 12y$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Box-Wilson	Počáteční bod: [-12,3;3] Délka strany: 0,5	Hledáno maximum Hodnota: 67,9325 Souřadnice: [3,95;6,25] Počet iterací: 66 Čas: 0,062s	Počáteční bod: [-12,3;10] Délka strany: 0,01	Hledáno maximum Hodnota: 68 Souřadnice: [4;6] Počet iterací: 3261 Čas: 2,75s
Nelder-Mead	Počáteční bod: [-12,3;3] Délka strany 0,5 Přesnost 0,1	Hledáno maximum Hodnota: 67,902 Souřadnice: [4,03;5,9101] Počet iterací: 24 Čas: 0,141s	Počáteční bod: [-12,3;3] Délka strany: 0,5 Přesnost: 0,001	Hledáno maximum Hodnota: 67,9997 Souřadnice [4,002;5,997] Počet iterací: 41 Čas: 0,219s

Krátký krok	Počáteční bod: [-12,3;3] Délka kroku: 0,5 Přesnost: 0,01	Extrém nenalezen	Počáteční bod: [-12,3;3] Délka kroku: 0,2 Přesnost: 0,01	Nalezeno maximum Hodnota: 67,9999 Souřadnice [4;5,989] Počet iterací: 11 Čas: 0,016s
Dlouhý krok	Počáteční bod: [-12,3;3] Přesnost: 0,1	Nalezeno maximum Hodnota: 68 Souřadnice [4,019;5,999] Počet iterací: 4 Čas: 0,218s	Počáteční bod: [-12,3;3] Přesnost: 0,001	Nalezeno maximum Hodnota: 68 Souřadnice [4;6] Počet iterací: 6 Čas: 0,406s
Newton	Počáteční bod: [-12,3;3] Přesnost: 0,1	Nalezeno maximum Hodnota: 68 Souřadnice: [4;6] Počet iterací: 2 Čas: 0,078s	Počáteční bod: [-12,3;3] Přesnost: 0,001	Nalezeno maximum Hodnota: 68 Souřadnice: [4;6] Počet iterací: 2 Čas: 0,094s
DFP	Počáteční bod: [-12,3;3]	Nalezeno maximum Hodnota: 68 Souřadnice: [4;6] Počet iterací: 2 Čas: 0,328s	Počáteční bod: [50;30]	Nalezeno maximum Hodnota: 68 Souřadnice: [4;6] Počet iterací: 2 Čas: 0,313s

Na této účelové funkci jsou ukázány základní vlastnosti optimalizačních metod pro více proměnných. Výsledky opět ukázaly, že u komparativních metod záleží na nastavení počátečních parametrů a že přesnost s dobou výpočtů jsou v přímé úměrnosti. Dobu běhu také ovlivní vzdálenost počátečního bodu od extrému. Proto je dobré použít více opakování vyhledávání a postupně zvyšovat požadavky na přesnost.

U gradientních metod ovlivňuje výsledky požadovaná přesnost. U metody s krátkým krokem se potvrdilo, že mohou nastat problémy s nastavením kroku a tím přeskočení extrému a zacyklení programu.

Nejlepší výsledky podaly Newtonova metoda a metoda DFP, které nalezly extrém už ve druhém kroku. Z toho Newtonova metoda podstatně rychleji. Delší doba běhu metody DFP je způsobena násobením a invertováním matic uvnitř algoritmu.



Obr. 33. Graf funkce $f(x,y) = -2x^2 - y^2 + 16x + 12y$

Funkce: $f(x,y) = 0,3x^2 + 0,2y^2 + 0,3x + 2y + 0,1xy$

Analytické řešení:

$$\frac{\partial f}{\partial x} = \frac{3}{5}x + \frac{3}{10} + \frac{1}{10}y$$

$$\Rightarrow \quad x_0 = 0 \quad y_0 = 0 \quad \Rightarrow \quad f(x_0, y_0) = 0$$

$$\frac{\partial f}{\partial y} = \frac{2}{5}y + 2 + \frac{1}{10}x$$

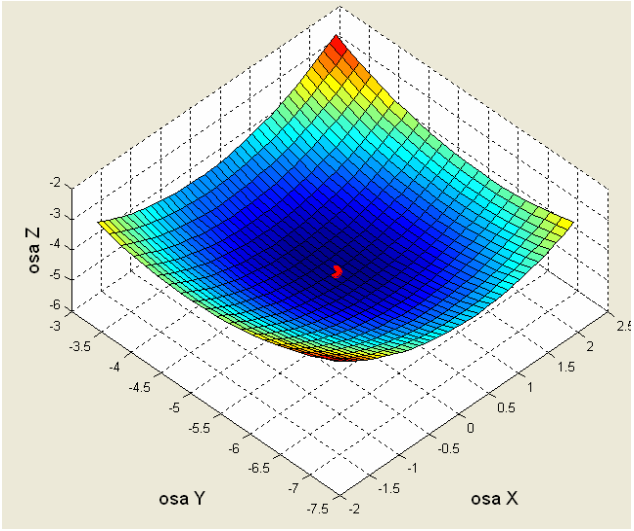
Zápis funkce ve tvaru pro Matlab: 0.3*x^2+0.2*y^2+0.3*x+2*y+0.1*x*y

Tab. 5: Výsledky pro funkci $f(x,y) = 0,3x^2 + 0,2y^2 + 0,3x + 2y + 0,1xy$

$f(x,y) = 0,3x^2 + 0,2y^2 + 0,3x + 2y + 0,1xy$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Box-Wilson	Počáteční bod: [15;6,8] Délka strany: 0,5	Hledáno minimum Hodnota: -5,029 Souřadnice: [0,25;-4,95] Počet iterací: 60 Čas: 0,063s	Počáteční bod: [0,5;-5] Délka strany: 0,0001	Hledáno minimum Hodnota: -5,0348 Souřadnice: [0,3478;-5,0869] Počet iterací: 3044 Čas: 2,985s
Nealder-Mead	Počáteční bod: [-14,8;16] Délka strany: 0,5 Přesnost: 0,1	Hledáno minimum Hodnota: -4,7478 Souřadnice: [0,19;-5,6461] Počet iterací: 15 Čas: 0,11s	Počáteční bod: [0,5;-5] Délka strany: 0,001 Přesnost: 0,00001	Hledáno minimum Hodnota: -5,0335 Souřadnice: [0,326;-5,005] Počet iterací: 12 Čas: 0,094s

Krátký krok	Počáteční bod: [-6;8] Délka kroku: -0,6 Přesnost: 0,1	Nalezeno minimum Hodnota: -5,008 Souřadnice: [0,2;-4,7329] Počet iterací: 15 Čas: 0,062s	Počáteční bod: [0,5;-5] Délka kroku: -0,1 Přesnost: 0,001	Nalezeno maximum Hodnota: -5,0347 Souřadnice: [0,3604;-5,0754] Počet iterací: 36 Čas: 0,125s
Dlouhý krok	Počáteční bod: [-17,2;22,5] Přesnost: 0,1	Nalezeno minimum Hodnota: -5,0347 Souřadnice: [0,3344;-5,0659] Počet iterací: 4 Čas: 0,375s	Počáteční bod: [20;20] Přesnost: 0,001	Nalezeno minimum Hodnota: -5,0348 Souřadnice: [0,3477;-5,0868] Počet iterací: 7 Čas: 0,578s
Newton	Počáteční bod: [-8;7] Přesnost: 0,1	Nalezeno minimum Hodnota: -5,0348 Souřadnice: [0,3478;-5,087] Počet iterací: 2 Čas: 0,087s	Počáteční bod: [-44;17] Přesnost: 0,001	Nalezeno minimum Hodnota: -5,0348 Souřadnice: [0,3478;-5,087] Počet iterací: 2 Čas: 0,078s
DFP	Počáteční bod: [16,5;13]	Nalezeno minimum Hodnota: -5,0348 Souřadnice: [0,3478;-5,087] Počet iterací: 2 Čas: 0,359s	Počáteční bod: [-40;-2,3]	Nalezeno minimum Hodnota: -5,0348 Souřadnice: [0,3478;-5,087] Počet iterací: 2 Čas: 0,375s

Zde bylo testováno jak si jednotlivé metody povedou u funkce, která má hladké okolí extrému. U komparativních metod byly provedeny výpočty s nižší přesností a poté se pokračovalo z výsledného bodu s přesností vyšší. V tomto srovnání dopadla lépe metoda Nelder-Mead, která dosáhla extrému za zlomek doby, kterou potřebovala metoda Box-Wilson. U gradientní metody s krátkým krokem, musel být zadán záporný parametr kroku, aby algoritmus hledal minimum. Nejlépe opět dopadly Newtonova metoda a metoda DFP, u kterých se ani nijak neprojevila větší vzdálenost počátečního bodu.



Obr. 34. Graf funkce $f(x,y) = 0,3x^2 + 0,2y^2 + 0,3x + 2y + 0,1xy$

Funkce: $f(x,y) = e^{-(x-1)^2-(y-5)^2}$

Analytické řešení:

$$\frac{\partial f}{\partial x} = (-2x+2) * e^{(-(x-1)^2-(y-5)^2)}$$

$$\Rightarrow \quad x_0 = 1 \quad y_0 = 5 \quad \Rightarrow \quad f(x_0, y_0) = 1$$

$$\frac{\partial f}{\partial y} = (-2y+10) * e^{(-(x-1)^2-(y-5)^2)}$$

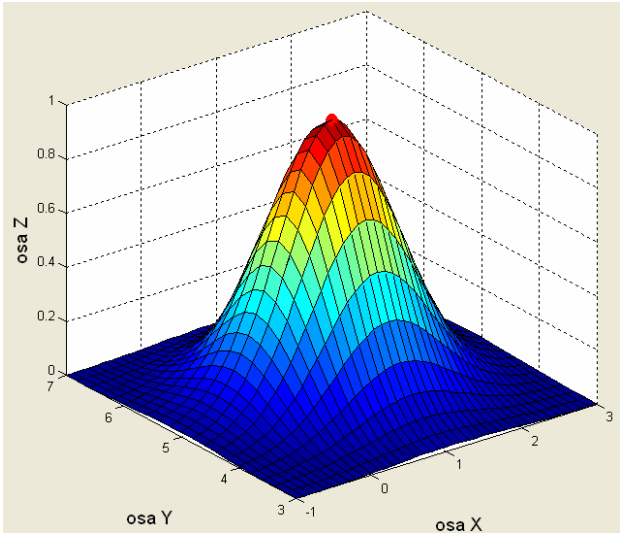
Zápis funkce ve tvaru pro Matlab: exp(-(x-1)^2-(y-5)^2)

Tab. 6: Výsledky pro funkci $f(x,y) = e^{-(x-1)^2-(y-5)^2}$

$f(x,y) = e^{-(x-1)^2-(y-5)^2}$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Box-Wilson	Počáteční bod: [17,6;-16,1] Délka strany: 0,5	Hledáno maximum Hodnota: 0,9802 Souřadnice: [1,1;4,9] Počet iterací: 85 Čas: 0,062s	Počáteční bod: [0,2;0,3] Délka strany: 0,01	Hledáno maximum Hodnota: 1 Souřadnice: [1;5] Počet iterací: 941 Čas: 0,719s
Nelder-Mead	Počáteční bod: [4,7;9] Délka strany: 0,5 Přesnost: 10^{-13}	Hledáno maximum Hodnota: 1 Souřadnice: [1;5] Počet iterací: 43 Čas: 0,187s	Počáteční bod: [0,5;3] Délka strany: 0,1 Přesnost: 0,0001	Hledáno maximum Hodnota: 0,9999 Souřadnice: [0,9984;5,0032] Počet iterací: 24 Čas: 0,125s

Krátký krok	Počáteční bod: [1;1] Délka kroku: 0,3 Přesnost: 0,001	Špatný výsledek	Počáteční bod: [0,5;3] Délka kroku: 0,3 Přesnost: 0,001	Nalezeno maximum Hodnota: 1 Souřadnice: [0,9998;4,9994] Počet iterací: 29 Čas: 0,094s
Dlouhý krok	Počáteční bod: [6;-3] Přesnost: 1	Nalezeno maximum Hodnota: 1 Souřadnice: [1;5] Počet iterací: 2 Čas: 0,125s	Počáteční bod: [-9;-7] Přesnost: 0,001	Nalezeno maximum Hodnota: 1 Souřadnice: [1;5] Počet iterací: 2 Čas: 0,156s
Newton	Počáteční bod: [-4;7] Přesnost: 0,001	Špatný výsledek Nalezeno maximum Hodnota: 0 Souřadnice: [-4,087;7,035] Počet iterací: 1 Čas: 0,078s	Počáteční bod: [0,7;2,5] Přesnost: 0,001	Špatný výsledek Nalezeno maximum Hodnota – 0 Souřadnice: [0,67;2,28] Počet iterací: 1 Čas: 0,093s
DFP	Počáteční bod: [0;0]	Extrém nenalezen	Počáteční bod: [0,9;4,5]	Extrém nenalezen

Tato exponenciální účelová funkce se ukázala jako velice kritická. Jediné metody, které bez problémů nalezly extrém jsou Box-Wilsonova metoda a gradientní metoda s dlouhým krokem, která extrém nalezla už ve druhém kroce. U metody Nelder-Mead se ukázal jako kritický parametr Přesnost, čili nejmenší rozdíl mezi hodnotami účelové funkce v jednotlivých vrcholech simplexu. Extrém byl nalezen až při dostatečném zmenšení tohoto parametru na 10^{-13} nebo při zadání počátečního bodu v blízkosti extrému. Gradientní metoda s krátkým krokem nalezla správný extrém jen při zadání počátečního bodu poblíž extrému. Newtonova metoda a metoda DFP nedokázaly extrém najít ani při zadání počátečního bodu téměř přímo do samotného extrému. Tuto skutečnost zřejmě způsobilo to, že se algoritmu nepodařilo nalézt kořeny derivované funkce.



Obr. 35. Graf funkce $f(x,y) = e^{-(x-1)^2-(y-5)^2}$

Funkce: $f(x,y) = (1-x)^2 + 100(y-x^2)^2$ - Rosenbrockova funkce

Analytické řešení:

$$\frac{\partial f}{\partial x} = -2 + 2x - 400(y - x^2)x$$

$$\Rightarrow \quad x_0 = 1 \quad y_0 = 1 \quad \Rightarrow \quad f(x_0, y_0) = 0$$

$$\frac{\partial f}{\partial y} = 200y - 200x^2$$

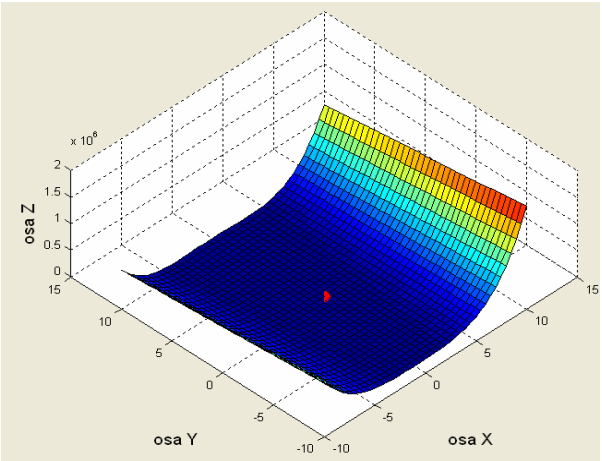
Zápis funkce ve tvaru pro Matlab: (1-x)^2+100*(y-x^2)^2

Tab. 7: Výsledky pro funkci $f(x,y) = (1-x)^2 + 100(y-x^2)^2$ (Rosenbrockova funkce)

$f(x,y) = (1-x)^2 + 100(y-x^2)^2$				
Metoda	**Parametry**	**Výsledek**	**Parametry**	**Výsledek**
Box-Wilson	Počáteční bod: [3;8,6] Délka strany: 0,001	Hledáno minimum Hodnota: 3,7711 Souřadnice: [2,9415;8,6565] Počet iterací: 118 Čas: 0,094s	Počáteční bod: [-3;-9,4] Délka strany: 0,001	Hledáno minimum Hodnota: 0,00961 Souřadnice: [0,9025;0,8135] Počet iterací: 20666 Čas: 15,094s
Nealder -Mead	Počáteční bod: [8;4] Délka strany: 0,5 Přesnost: 10^{-4}	Hledáno minimum Hodnota: 0,66 Souřadnice: [1,8124;3,2871] Počet iterací: 28 Čas: 0,156s	Počáteční bod: [2;3] Délka strany: 0,1 Přesnost: 0,0001	Hledáno minimum Hodnota: 7.10^{-6} Souřadnice: [0,9995;0,9997] Počet iterací: 88 Čas: 0,39s

Krátký krok	Počáteční bod: [1;0,9] Délka kroku: 0,01 Přesnost: 0,5	Nalezeno minimum Hodnota: 158,92 Souřadnice: [1,4;0,7] Počet iterací: 1 Čas: 0,016s	Počáteční bod: [0;0,9] Délka kroku: 0,01 Přesnost: 0,1	Extrém nenalezen
Dlouhý krok	Počáteční bod: [-3,3;-1] Přesnost: 0,001	Nalezeno minimum Hodnota: 0 Souřadnice: [1;1] Počet iterací: 3 Čas: 0,421s	Počáteční bod: [20;36] Přesnost: 0,001	Nalezeno minimum Hodnota: 0 Souřadnice: [1;1] Počet iterací: 3 Čas: 0,297s
Newton	Počáteční bod: [2;3] Přesnost: 10^{-4}	Nalezeno minimum Hodnota: 0,99 Souřadnice: [1,995;3,98] Počet iterací: 2 Čas: 0,078s	Počáteční bod: [0,5;0,3] Přesnost: 10^{-4}	Nalezeno minimum Hodnota: 0,309 Souřadnice: [0,4444;0,1944] Počet iterací: 2 Čas: 0,078s
DFP	Počáteční bod: [0;0]	Extrém nenalezen	Počáteční bod: [0,9;0,9]	Extrém nenalezen

S touto funkcí si nejlépe poradila gradientní metoda s dlouhým krokem. I při volbě vzdálenějšího počátečního bodu nalezla extrém ve třetí iteraci. U komparativních metod se opět ukázalo, že výsledek velmi záleží na vstupních parametrech. U většího počtu opakování použití jednotlivých metod by však extrému spolehlivě dosáhly. U gradientní metody s krátkým krokem se opět ukázal problém s nastavením délky kroku, ale ani tak se nepodařilo dosáhnout správného výsledku. Newtonova metoda by potřebovala více opakovaní použití stejně jako komparativní metody. U metody DFP se nepodařilo výsledek zjistit ani při zadání počátečního bodu téměř do extrému.



Obr. 36. Graf Rosenbrockovy funkce

ZÁVĚR

Teorie optimalizace je matematickou disciplínou zabývající se určováním minimálních a maximálních hodnot funkcí při určitých omezujících podmínkách. S optimalizačními problémy se v každodenním životě setkáváme neustále. V minulém století docházelo k rozvoji optimalizačních metod a začaly nacházet uplatnění v řadě průmyslových a ekonomických aplikací, a to jak v projektové tak v provozní oblasti.

Původně bylo vyšetřování extrémů klasickým tématem matematické analýzy.

Rozvoj počítačů posunul řešení další řady úloh, které do té doby nebylo možné řešit na analytické úrovni. K řešení extremalizačních úloh existuje velké množství postupů, které jsou dle základních charakteristik členěny do několika skupin.

V této práci jsou uvedeny jednotlivé klasifikace optimalizačních úloh a dále jsou podrobněji rozebrány iterační optimalizační metody. Rozvoj výpočetní techniky znamená také nárůst výpočetní výkonnosti. Tato skutečnost umožňuje širší uplatnění iteračních metod, které jsou svým principem nevhodné pro ruční výpočet, ale téměř výlučně připravené pro používání na počítači.

Iterační metody lze rozdělit na tři kategorie, na metody komparativní, gradientní a metody náhodného vyhledávání. Komparativní metody nevyžadují výpočet ani odhad derivací účelové funkce. Jejich jediným požadavkem je unimodalita (existuje pouze jeden extrém) a spojitost účelové funkce. Tyto metody jsou postaveny na principu výpočtu hodnot účelové funkce v určitých bodech pomocí daného algoritmu, jejich vzájemné komparace a tím nalezení extrému. Gradientní iterační metody používají diferenciálního počtu. V případech, kdy derivace nejsou známy, se pomocí numerické matematiky vypočítají jejich symetrické či nesymetrické odhady. Algoritmy pak využívají hodnot gradientů a Hessovy matice, z nichž stanovují hodnotu účelové funkce v daném bodě. Metody náhodného vyhledávání jsou v optimalizačních metodách specifickou třídou. Jejich algoritmus je založený tak, že iterační přírůstek v jednotlivých krocích je závislý na náhodně generované veličině.

Z teoretické části bylo čerpáno při programování jednotlivých iteračních metod v prostředí Matlab. Pomocí M-file souborů byl vytvořen program, jehož hlavní částí je deset optimalizačních metod. Pro jednorozměrné úlohy byly vypracovány Fibonacciho metoda, metoda zlatého řezu, metoda regula-falsi a Newtonova metoda. Pro vícerozměrné úlohy byly vypracovány Box-Wilsonova metoda, metoda flexibilního simplexu (Nelder-Mead), gradientní metoda s krátkým a dlouhým krokem, Newtonova metoda pro vícerozměrný případ a metoda DFP. Kvůli lepší názornosti a možnosti grafického znázornění extrému

byly vícerozměrné metody naprogramovány pro dvě proměnné. Metody jsou obecně naprogramovány pro vyhledávání maxima, ale pomocí zaškrtávacího políčka (checkbox) je možno toto změnit na minimum.

Správná funkčnost algoritmů a vlastnosti jednotlivých metod byly testovány na praktických příkladech. Výsledky jsou vypsány v tabulkách. Po uskutečnění výpočtů bylo provedeno zhodnocení a přiblíženy výhody a nevýhody jednotlivých algoritmů.

Hlavním přínosem této práce je sada naprogramovaných metod, kterou lze dále rozšiřovat a spolu s teoretickou částí může sloužit jako studijní materiál.

ZÁVĚR V ANGLIČTINĚ

Theory of optimization is mathematical discipline dealing with determining of minimal and maximal values of functions at given limiting conditions. In the past century there has been evolution of optimization methods, which started to be used in the number of industrial and economical applications, both in the project and operational area. In the real life we encounter the optimization problems all the time.

Originally the examination of extremes was the classical subject of mathematical analysis. Expansion of computers advanced solving of another number of tasks, which was not solvable on the analytical level. There are a lots of ways for solving extremalization tasks, which are, according to basic characteristics divided into several groups.

There are shown individual classifications of the optimization in this thesis and also there are iteration optimization methods parsed in detail. The evolution of computers means also the growth of computer power. This fact enables wider use of iteration methods, which are improper by their principles to use for hand calculation, but almost exclusively ready for using on computer.

Iteration methods can be divided into three categories – comparative methods, gradient and methods of random search. Comparative methods does not require calculation nor estimation of derivations of target function. Their only requirement is unimodality (only one extreme exists). These methods are based on the principle of calculation of the values of dedicated function in particular points with the help of given algorithm, their mutual comparison thereby finding of an extreme. Gradient iteration methods uses differential calculation. In the cases, when derivations are not known, their symmetrical and non symmetrical are calculated by numerical mathematics. Algorithms then uses values of the gradients and Hess matrix, of which they determine the value of target function in the given point. Methods of random search are the specific class in the optimization methods. Their algorithm is based on the increment it each step, which depends on the random generated value.

When programming each iteration methods in the Matlab, it was drawn from the theoretical part. The program was created with the help of m-files. Its main part is ten optimization methods. For one dimensional tasks the Fibonacci, golden cut, regula-falsi and Newton's methods was created. For n-dimensional tasks the Box-Wilson, flexible simplex (Nelder-Mead), gradient with short and long step, Newton's method of n-dimensional case and DFP methods was elaborated. For better clearness and possibility of

graphical illustration of the extreme, the n-dimensional methods was coded for 2 variables. Methods are generally programmed for searching the maximum, but by checking the checkbox one can change this to minimum. Right functionality of the algorithms and properties of each method were tested on practical examples. Results are listed in the tables. After realization of the calculations, the estimation and approximation of pros and cons of each algorithms were explained. The main gain of this thesis is the set of programmed methods, which can be expanded furthermore and together with theoretical part it can serve as educational material.

SEZNAM POUŽITÝCH ZDROJŮ

[1] BRUNOVSKÁ, A.: *Malá optimalizácia*, Alfa, Bratislava, 1990

[2] KOČIČA, M.: *Iterační metody optimalizace.* Diplomová práce, FAI UTB ve Zlíně, 2006

[3] MAŇAS, M.: *Optimalizační metody*, SNTL Praha, 1979

[4] PROKOP, R. : *Teória systémov a optimalizácia*, CHTF, SVŠT v Bratislavě, 1985

[5] ŠTECHA, J.: *Optimální rozhodování a řízení*. Vydavatelství ČVUT, Praha, 2000

[6] VÍTEČKOVÁ, M., JEDLIČKA, D.: *Statická optimalizace* systémů [online]. [cit.1. února 2007], Dostupný z URL: < www.fs.vsb.cz/books/ statickaoptimalizace >

[7] *Matlab a Simulink* [online]. [cit. 1. května 2007] Dostupný z URL:< www.humusoft.com >

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

b	vektor omezujících konstant
DFP	označení metody Davidon-Fletcher-Powell
f(x)	účelová funkce v bodě
∇ f(x)	gradient funkce
F_i	číslo Fibonacciho posloupnosti
g(x)	vektor omezujících funkcí
GUI	grafické uživatelské rozhraní
H, $\nabla^2 f(x_1,\ldots,x_n)$	Hessova matice
$\Phi(x,p)$	Lagrangeova funkce
λ	označení parametru krok u gradientních metod
ε	označení přesnosti
ω	označení délky hrany čtverce u metody Box-Wilson
ξ	označení parametru u metody cyklické záměny parametrů
N	počet kroků u Fibonacciho metody a metody zlatého řezu
p	Lagrangeův multiplikátor
R^n	n-rozměrný prostor
S_i	označení pozitivně definitní matice u metody DFP
x_0	stacionární bod
X	množina přípustných řešení

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

P I : FIBONACCIHO METODA

Zdrojový kód Matlab:

```
Fi={1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28
657,46368,75025};
syms x;syms f; f=handles.fce;
ret=1;
try
x=0; y=0; lk=eval(f); clear x;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0; i=2;
end
syms x;
a=handles.DM;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Interval (a)');
   ret=0;
end
c=0; b=handles.HM;
if(isnan(b)==1)
   errordlg('Musí být zadáno číslo!', 'Error Interval (b)');
   ret=0;
end
kr=handles.krok;
if(isnan(kr)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počet Kroků');
   ret=0;
   kr=1;
end
if (kr>21)
   set(handles.vyp,'String',['nepoužívá se více než 21 kroků']);
else
if (a>b)
   a=c;a=b;b=c;
else
   a=a;b=b;
end
tic
for i=1:kr
   A=b-(Fi{kr-i+1}/Fi{kr-i+2})*abs(b-a);
   B=a+(Fi{kr-i+1}/Fi{kr-i+2})*abs(b-a);
   x=A;
   VA=eval(f);
   x=B;
   VB=eval(f);
   if(handles.checkbox1==0)
```

```
   if (VA>VB)
      a=a; b=B;
   else
      a=A; b=b;
   end
else
   if (VA<VB)
      a=a; b=B;
   else
      a=A; b=b;
   end
end
   i=i+1;

end
t=toc;
Int={a,b};
if (ret==1)
if(handles.checkbox1==1)
set(handles.vyp,'String',['minimum leží na  : [',num2str(a),';',num2str(b),']']);
set(handles.vypiscas,'String',['čas výpočtů : ',num2str(t),'s']);
else
   set(handles.vyp,'String',['maximum leží na  : [',num2str(a),';',num2str(b),']']);
   set(handles.vypiscas,'String',['čas výpočtů : ',num2str(t),'s']);
end
[xx,yy]=fplot(f,[handles.DM handles.HM]);
h1=a;
x=a;
hh1=eval(f);
h2=b; x=b;
hh2=eval(f);
[kk,ll]=fplot(f,[a b]);
plot(xx,yy,kk,ll,'-r',h1,hh1,'or',h2,hh2,'or','MarkerFaceColor','r','MarkerSize',5);
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
end
end
clear all;
```

P II : METODA ZLATÉHO ŘEZU

Zdrojový kód Matlab:

```
syms x f;
f=handles.fce;
ret=1;
try
x=0;
lk=eval(f);
clear x;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0; i=2;
end
syms x;
aa=handles.dint;
if(isnan(aa)==1)
   errordlg('Musí být zadáno číslo!', 'Error Interval (a)');
   ret=0;
end
bb=handles.hint;
if(isnan(bb)==1)
   errordlg('Musí být zadáno číslo!', 'Error Interval (b)');
   ret=0;
end
n=handles.krok;
if(isnan(n)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počet Iterací');
   ret=0;
   n=0;
end
zl=0.618;
a=aa;b=bb;c=0;
if (a>b)
   a=c;a=b;b=c;
else
   a=a;b=b;
end
tic
for i=1:n+1
   A=b-zl*abs(b-a);
   B=a+zl*abs(b-a);
   x=A;FA=eval(f);
   x=B;FB=eval(f);
   if(handles.chck==0)
   if (FA>FB)
      a=a; b=B;
   else
      a=A; b=b;
   end
```

```
else
   if (FA<FB)
      a=a; b=B;
   else
      a=A; b=b;
   end
end
   i=i+1;
end
t=toc;
if (ret==1)
set(handles.vypint,'String',['max leží na  : (',num2str(a),';',num2str(b),')']);
set(handles.vypiscas,'String',['čas výpočtů : ',num2str(t),'s']);
 [xx,yy]=fplot(f,[handles.dint handles.hint]);
h1=a;
x=a;
hh1=eval(f); h2=b; x=b; hh2=eval(f);
[kk,ll]=fplot(f,[a b]);
plot(xx,yy,kk,ll,'-r',h1,hh1,'or',h2,hh2,'or','MarkerFaceColor','r','MarkerSize',5);
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
end
clear all;
```

P III : METODA REGULA-FALSI

Zdrojový kód Matlab:

```
syms x f;
f=handles.fce;ret=1;i=0;
try
x=0; y=0;
lk=eval(f);
clear x;clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=2;
end
syms x;
a=handles.edit2;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X1');
   ret=0;i=2;
end
b=handles.edit3;
if(isnan(b)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X2');
   ret=0;i=2;
end
e=handles.presnost;
if(isnan(e)==1)
   errordlg('Musí být zadáno číslo!', 'Error Přesnost');
   ret=0;i=2;
end
c=0;
if (a>b)
   c=a;a=b;b=c;
else
   a=a;b=b;
end
tic
krok=0;
x1=a;x2=b; Y1=diff(f,x); sd=0;
while i<1
   t=toc;
   if(t>10)
   i=3;
   sd=12;
   end
   krok=krok+1;
   x=x1;y1=eval(Y1);
   x=x2;y2=eval(Y1);
   x3=x2-(y2*((x1-x2)/(y1-y2)));
   if (abs(x3-x2)>e)
      x1=x2;x2=x3; krok=krok+1;
   else
      i=1;
```

```
   end
end
t=toc;
x=x3; ff=eval(f);
if(ret==1)
   x=x3+1;  m1=eval(f);
   x=x3-1;   m2=eval(f); x=x3;
   if(m1>ff)&&(m2>ff)
   set(handles.extrem,'String',['hodnota minima : ',num2str(ff)]);
end
if (m1<ff)&&(m2<ff)
set(handles.extrem,'String',['hodnota maxima : ',num2str(ff)]);
end
if (m1<ff)&&(m2>ff)||(m1>ff)&&(m2<ff)
set(handles.extrem,'String',['inflexní bod : y->',num2str(ff)]);
end
set(handles.souradnice,'String',['pozice na ose x: ',num2str(x)]);
set(handles.cas,'String',['počet kroků : ',num2str(krok),'   čas výpočtů : ',num2str(t),'s']);
if(sd==12)
set(handles.extrem,'String',['Timeout - změnte počáteční body']);
set(handles.souradnice,'String',['Timeout - změnte počáteční body']);
set(handles.cas,'String',['Timeout - změnte počáteční body']);
end
hh=x; jj=ff;
[xx,yy]=fplot(f,[x-5 x+5]);
plot(xx,yy,hh,jj,'or','MarkerFaceColor','r','MarkerSize',5);
xlabel('osa X','FontSize',14); ylabel('osa Y','FontSize',14);
end
clear all;
```

P IV : NEWTONOVA METODA

Zdrojový kód Matlab:

```
syms x f;
f=handles.fce;
```

```
ret=1;i=0;
try
x=0; y=0;
lk=eval(f);
clear x; clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=2;
end
syms x; b=handles.hint;
if(isnan(b)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X');
   ret=0;i=2;
end
e=handles.presnost;
if(isnan(e)==1)
   errordlg('Musí být zadáno číslo!', 'Error Přesnost');
   ret=0;i=2;
end
krok=1; x2=b; sd=0;
tic
while i<1
   t=toc;
   if(t>10)
      i=3;   sd=12;
   end
   y1=diff(f,x); y2=diff(f,x,2); x=x2;Y1=eval(y1); x=x2;Y2=eval(y2); x3=x2-(Y1/Y2);
   if (abs(x3-x2)>e)
      x2=x3;
      krok=krok+1;
   else
     i=1;
  end
end
t=toc; x=b; bb=eval(f); x=x3; xa3=eval(f);
if (ret==1)
   x=x3+1; m1=eval(f); x=x3-1; m2=eval(f); x=x3;
   if(m1>xa3)&&(m2>xa3)
   set(handles.vzpismax,'String',['hodnota minima : ',num2str(xa3)]);
end
if (m1<xa3)&&(m2<xa3)
set(handles.vzpismax,'String',['hodnota maxima : ',num2str(xa3)]);
end
if (m1<xa3)&&(m2>xa3)||(m1>xa3)&&(m2<xa3)
set(handles.vzpismax,'String',['inflexní bod : y->',num2str(xa3)]);
end
set(handles.vypiscas,'String',['počet kroků :',num2str(krok-1),'    čas výpočtů :
',num2str(t),'s']);
set(handles.interval,'String',['pozice na ose x : ',num2str(x)]);
```

```
if(sd==12)
set(handles.vzpismax,'String',['Timeout- změnte počáteční bod']);
set(handles.vypiscas,'String',['Timeout- změnte počáteční bod']);
set(handles.interval,'String',['Timeout- změnte počáteční bod']);
end
gg=max(b,x);
ggg=min(b,x);
[xx,yy]=fplot(f,[ggg-1 gg+1]);
plot(xx,yy,x,xa3,'or',b,bb,'ob');
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
end
clear all;
```

P V : BOX-WILSONOVA METODA

Zdrojový kód Matlab:

```
syms x; syms y; syms f;
f=handles.fce;i=0;
try
x=0; y=0; lk=eval(f);
```

```
clear x; clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=10;
end
ret=1;
sx=handles.iks; sy=handles.yps; a=handles.str; ttm=handles.timeout;
if(isnan(sx)==1)
   errordlg('Musí být zadáno číslo!', 'Error Souřadnice X');
   ret=0;i=10;
end
if(isnan(sy)==1)
   errordlg('Musí být zadáno číslo!', 'Error Souřadnice Y');
   ret=0;i=10;
end
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Délka Strany');
   ret=0;i=10;
end
if(isnan(ttm)==1)
   errordlg('Musí být zadáno číslo!', 'Error Timeout');
   ret=0;i=10;
end
k=0;
tic
while i<1
   t=toc;
   if t>ttm
      i=1;
      if(handles.checkbox1==0)
set(handles.vypis,'String',['timeout - algoritmus hledá maximum']);
set(handles.vypisxy,'String',['timeout - algoritmus hledá maximum']);
set(handles.vypiskrok,'String',['timeout - algoritmus hledá maximum']);
else
set(handles.vypis,'String',['timeout - algoritmus hledá minimum']);
set(handles.vypisxy,'String',['timeout - algoritmus hledá minimum']);
set(handles.vypiskrok,'String',['timeout - algoritmus hledá minimum']);
end
   else
   xx(1)=sx+0.5*a;
   yy(1)=sy+0.5*a;
   xx(2)=sx+0.5*a;
   yy(2)=sy-0.5*a;
   xx(3)=sx-0.5*a;
   yy(3)=sy+0.5*a;
   xx(4)=sx-0.5*a;    yy(4)=sy-0.5*a;
   x=sx; y=sy; FS=eval(f); x=xx(1); y=yy(1); F1=eval(f); x=xx(2); y=yy(2); F2=eval(f);
   x=xx(3); y=yy(3); F3=eval(f); x=xx(4); y=yy(4); F4=eval(f); FX=[F1,F2,F3,F4];
m=find(FX==max(FX)); mm=find(FX==min(FX)); k=k+1;
```

```
if (handles.checkbox1==0)
    if (max(FX)>FS)
    FS=max(FX); sx=xx(m(1)); sy=yy(m(1));
    else
    i=99; FS=roundn(FS,-5); sx=roundn(sx,-5); sy=roundn(sy,-5);
  set(handles.vypis,'String',['maximum je ',num2str(FS)]);
    set(handles.vypisxy,'String',['souřadnice [x;y]:
[',num2str(sx),';',num2str(sy),']']);%,';',num2str(sx),']']);
    set(handles.vypiskrok,'String',['počet iterací: ',num2str(k),'  čas: ',num2str(t),'s']);
    end
else
    if (min(FX)<FS)
    FS=min(FX); sx=xx(mm(1)); sy=yy(mm(1));
    else
    i=99; FS=roundn(FS,-5); sx=roundn(sx,-5); sy=roundn(sy,-5);
   set(handles.vypis,'String',['minimum je ',num2str(FS)]);
    set(handles.vypisxy,'String',['souřadnice [x;y]:
[',num2str(sx),';',num2str(sy),']']);%,';',num2str(sx),']']);
    set(handles.vypiskrok,'String',['počet iterací: ',num2str(k),'  čas: ',num2str(t),'s']);
    end
end
end
end
if (ret==1)
fff=handles.fce; f1=strrep(fff,'^','.^'); f2=strrep(f1,'*','.*'); f3=strrep(f2,'/','./');
[x,y]=meshgrid(sx-10:0.5:sx+10 ,sy-10:0.5:sy+10);
z=eval(f3);
if(handles.checkbox1==0)
plot3(sx,sy,FS,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off;
xlabel('osa X','FontSize',14); ylabel('osa Y','FontSize',14); zlabel('osa Z','FontSize',14);
else
plot3(sx,sy,FS,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off; view ([-5 -5 15])
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
end
clear all;
```

P VI : METODA FLEXIBILNÍHO SIMPLEXU (NELDER-MEAD)

Zdrojový kód Matlab:

```
syms x; syms y; syms f; ret=1;
f=handles.fce;
i=0;
try
x=0;y=0;lk=eval(f);
```

```
clear x;clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=2;
end
syms x;syms y;syms a;
if(handles.check==0)
   f=handles.fce;
else
   ff=eval(handles.fce*a); a=-1; f=eval(ff);
end
sx=handles.iks;
if(isnan(sx)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X');
   ret=0;i=2;
end
sy=handles.yps;
if(isnan(sy)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod Y');
   ret=0;i=2;
end
a=handles.strana;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Délka Strany');
   ret=0;i=2;
end
p=handles.timeout;
if(isnan(p)==1)
   errordlg('Musí být zadáno číslo!', 'Error Přesnost');
   ret=0;i=2;
end
ttm=handles.timeou;
if(isnan(ttm)==1)
   errordlg('Musí být zadáno číslo!', 'Error Timeout');
   ret=0;i=2;
end
k=0; d=a*((sqrt(3)-1)/(2*sqrt(2))); e=d+(a/sqrt(2));
Y1=[sx, sx+e, sx+d]; Y2=[sy, sy+d, sy+e];
tic
while i<1
   t=toc;
   if (t>ttm)
      i=2;
      if(handles.check==0)
         set(handles.edit13,'String',['timeout - algoritmus hledá maximum']);
set(handles.edit14,'String',['timeout - algoritmus hledá maximum']);
set(handles.edit15,'String',['timeout - algoritmus hledá maximum']);
Y1(mm)=0; Y2(mm)=0;
      else
```

```
            set(handles.edit13,'String',['timeout - algoritmus hledá minimum']);
set(handles.edit14,'String',['timeout - algoritmus hledá minimum']);
set(handles.edit15,'String',['timeout - algoritmus hledá minimum']);
Y1(mm)=0; Y2(mm)=0;
        end
    else
    x=Y1(1); y=Y2(1); F1=eval(f); x=Y1(2); y=Y2(2); F2=eval(f); x=Y1(3); y=Y2(3);
    F3=eval(f); FX=[F1,F2,F3]; c=abs(F1-F2); v=abs(F2-F3); h=abs(F3-F1);
    if (c<p) && (v<p) && (h<p)
       i=2;
    end
    o=find(FX==min(FX)); m=o(1); oo=find(FX==max(FX)); mm=oo(1); xx=Y1(m);
    yy=Y2(m); y1=Y1; y2=Y2; y1(m)=0; y2(m)=0;
    C1=0.5*(y1(1)+y1(2)+y1(3)); C2=0.5*(y2(1)+y2(2)+y2(3));
    xc=2*C1-xx; yc=2*C2-yy; x=xc; y=yc; FF=eval(f);
    if(FF>max(FX))
       x=C1+2.5*(xc-C1); y=C2+2.5*(yc-C2); ST=eval(f);
       if (ST>max(FX))
       Y1(m)=C1+2.5*(xc-C1); Y2(m)=C2+2.5*(yc-C2);
    else
       Y1(m)=xc;Y2(m)=yc;
    end
    else
      if(FF<min(FX))
       if (mm==1)
          Y1(2)=Y1(1)+0.5*(Y1(2)-Y1(1)); Y2(2)=Y2(1)+0.5*(Y2(2)-Y2(1));
          Y1(3)=Y1(1)+0.5*(Y1(3)-Y1(1)); Y2(3)=Y2(1)+0.5*(Y2(3)-Y2(1));
       end
       if(mm==2)
          Y1(1)=Y1(2)+0.5*(Y1(1)-Y1(2)); Y2(1)=Y2(2)+0.5*(Y2(1)-Y2(2));
          Y1(3)=Y1(2)+0.5*(Y1(3)-Y1(2)); Y2(3)=Y2(2)+0.5*(Y2(3)-Y2(2));
       end
       if(mm==3)
          Y1(1)=Y1(3)+0.5*(Y1(1)-Y1(3)); Y2(1)=Y2(3)+0.5*(Y2(1)-Y2(3));
          Y1(2)=Y1(3)+0.5*(Y1(2)-Y1(3)); Y2(2)=Y2(3)+0.5*(Y2(2)-Y2(3));
       end
    else
       Y1(m)=xc;Y2(m)=yc;
        end
end
    k=k+1;
t=toc;
if(handles.check==0)
set(handles.edit13,'String',['maximum je ',num2str(FF)]);
    set(handles.edit14,'String',['souřadnice [x;y]:
[',num2str(Y1(mm)),';',num2str(Y2(mm)),']']);%,';',num2str(sx),']']);
    set(handles.edit15,'String',['počet iterací: ',num2str(k),'  čas: ',num2str(t),'s']);
else
set(handles.edit13,'String',['minimum je ',num2str(-1*FF)]);
```

```
   set(handles.edit14,'String',['souřadnice [x;y]:
[',num2str(Y1(mm)),';',num2str(Y2(mm)),']']);%,';',num2str(sx),']']);
   set(handles.edit15,'String',['počet iterací: ',num2str(k),'  čas: ',num2str(t),'s']);
end
end
end
if (ret==1)
fff=handles.fce;
f1=strrep(fff,'^','.^');f2=strrep(f1,'*','.*');f3=strrep(f2,'/','./');
[x,y]=meshgrid(Y1(mm)-10:0.5:Y1(mm)+10 ,Y2(mm)-10:0.5:Y2(mm)+10);
z=eval(f3);
if(handles.check==0)
plot3(Y1(mm),Y2(mm),FF,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off;
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
else
plot3(Y1(mm),Y2(mm),FF,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off; view ([-5 -5 15])
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
end
clear all;
```

P VII : GRADIENTNÍ METODA S KRÁTKÝM KROKEM

Zdrojový kód Matlab:

```
syms x; syms y; syms f;
f=handles.fce;
ret=1;i=1;
try
x=0;y=0;lk=eval(f);
clear x; clear y;
catch
```

```
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0; i=11;
end
syms x; syms y; pp=handles.pres;
if(isnan(pp)==1)
   errordlg('Musí být zadáno číslo!', 'Error Přesnost');
   ret=0;i=11;
end
a=handles.krok;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Krok');
   ret=0;i=11;
end
ttm=handles.timeout;
if(isnan(ttm)==1)
   errordlg('Musí být zadáno číslo!', 'Error Timeout');
   ret=0;i=11;
end
yy=diff(f,y); xx=diff(f,x);q=0;c=0;sx=handles.iks;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční bod X');
   ret=0;i=11;
end
sy=handles.yps;
if(isnan(a)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční bod Y');
   ret=0;i=11;
end
sd=0;
tic
r=0;
while i<10
   t=toc; r=r+1;
   if t>ttm
      i=1001;sd=3;
      set(handles.vypismax,'String',['timeout - změnit hodnotu kroku ']);
      set(handles.souradnice,'String',['timeout - změnit hodnotu kroku']);
      set(handles.vypiscas,'String',['timeout - změnit hodnotu kroku']);
   else
  x=sx; y=sy; sdx=sx; dfx=eval(xx);  y=sy; sdy=sy; dfy=eval(yy);
  sx=x+a*dfx;sy=y+a*dfy;sx=roundn(sx,-4); sy=roundn(sy,-4); ssx=abs(sx-sdx);
  ssy=abs(sy-sdy);
  if (ssx<pp)&&(ssy<pp)
      i=1000; x=sx; y=sy;  c=eval(f);
   else
      i=i;
   end

end
```

```
end
t=toc;
if (ret==1)
if t>ttm
       i=1001;
       set(handles.vypismax,'String',['timeout - změnit hodnotu kroku ']);
       set(handles.souradnice,'String',['timeout - změnit hodnotu kroku']);
       set(handles.vypiscas,'String',['timeout - změnit hodnotu kroku']);
   else
       xx3=x;  yy3=y; x=x+50; y=y+5; m1=eval(f); x=xx3; y=yy3; x=x-5;y=y-50;
   m2=eval(f);x=xx3; y=yy3;
   if(m1>c)&&(m2>c)
   set(handles.vypismax,'String',['hodnota minima : ',num2str(c)]);
end
if (m1<c)&&(m2<c)
set(handles.vypismax,'String',['hodnota maxima : ',num2str(c)]);
end
if (m1<c)&&(m2>c)||(m1>c)&&(m2<c)
set(handles.vypismax,'String',['sedlový bod :',num2str(c)]);
end
set(handles.souradnice,'String',['souřadnice  : [',num2str(x),';',num2str(y),']']);
set(handles.vypiscas,'String',['počet kroků :',num2str(r),'     čas výpočtů : ',num2str(t),'s']);
end
fff=handles.fce; f1=strrep(fff,'^','.^'); f2=strrep(f1,'*','.*'); f3=strrep(f2,'/','./'); xx=x;
yy=y;
if(sd==0)
[x,y]=meshgrid(x-10:0.5:x+10, y-10:0.5:y+10);
z=eval(f3);
zz=c;
plot3(xx,yy,zz,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); hold off
grid on
xlabel('osa X','FontSize',14); ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
end
clear all;
```

P VIII : GRADIENTNÍ METODA S DLOUHÝM KROKEM

Zdrojový kód Matlab:

```
syms x; syms y; syms f;
f=handles.fce; dx=diff(f,x); dy=diff(f,y); i=0; ret=1;
try
x=0;y=0;lk=eval(f);clear x;clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;    i=2;
```

```
end
syms n; sx=handles.iks; sy=handles.yps; p=handles.pe;
if(isnan(sx)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X');
   ret=0;
end
if(isnan(sy)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod Y');
   ret=0;
end
if(isnan(p)==1)
   errordlg('Musí být zadáno číslo!', 'Error Přesnost');
   ret=0;
end
k=0;sd=1;
tic
while(i<1)
   t=toc;
   if(t>20)
   i=2;sd=0;

   end
k=k+1; x=sx; y=sy; ssx=sx; ssy=sy; dfx=eval(dx); dfy=eval(dy); x=sx+n*dfx;
y=sy+n*dfy; ff=diff(eval(f),n); N=solve(ff);
sx=sx+N(1)*dfx; sy=sy+N(1)*dfy; sx=eval(sx); sy=eval(sy);
SX=abs(sx-ssx); SY=abs(sy-ssy);  x=sx; y=sy; dfx=eval(dx); dfy=eval(dy);
if(SX<p)&&(SY<p)||(dfx==0)&&(dfy==0)
i=2;
end;
end
t=toc; x=sx; y=sy; FS=eval(f);
if(ret==1)
   c=FS; xx3=x; yy3=y; x=x+50;  y=y+5;  m1=eval(f); x=xx3; y=yy3; x=x-5; y=y-50;
   m2=eval(f); x=xx3; y=yy3;
   if(m1>c)&&(m2>c)
   set(handles.edit6,'String',['hodnota minima : ',num2str(FS)]);
end
if (m1<c)&&(m2<c)
set(handles.edit6,'String',['hodnota maxima : ',num2str(FS)]);
end
if (m1<c)&&(m2>c)||(m1>c)&&(m2<c)
set(handles.edit6,'String',['sedlový bod :',num2str(FS)]);
end
set(handles.edit7,'String',['souřadnice [x;y]:
[',num2str(sx),';',num2str(sy),']']);%,';',num2str(sx),']']);
set(handles.edit8,'String',['počet iterací: ',num2str(k),'     čas: ',num2str(t),'s']);
if(sd==0)
set(handles.edit6,'String',['Error algoritmu nebo funkce']);
set(handles.edit7,'String',['Zvolte jiný počátek nebo přesnost']);%,';',num2str(sx),']']);
set(handles.edit8,'String',['Error algoritmu nebo funkce']);
```

```
end
fff=handles.fce;
f1=strrep(fff,'^','.^');
f2=strrep(f1,'*','.*');
f3=strrep(f2,'/','./');
[x,y]=meshgrid(sx-5:0.5:sx+5 ,sy-5:0.5:sy+5);
z=eval(f3);
plot3(sx,sy,FS,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
clear all;
```

P IX : NEWTONOVA METODA

Zdrojový kód Matlab: - vícerozměrný případ

```
syms x y f; f=handles.fce; ret=1;i=0;
try
x=0; y=0; lk=eval(f); clear x; clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=2;
end
syms y x; e=handles.presnost;
if(isnan(e)==1)
```

```
    errordlg('Musí být zadáno číslo!', 'Error Přesnost');
    ret=0;i=2;
end
tic
xx=diff(f,x); yy=diff(f,y); xxx=diff(xx,y); yyy=diff(yy,x);
q=diff(f,x,2);r=diff(f,y,2);
x=2;y=3;krok=0;c=eval(f); H=[[eval(q) eval(xxx)];[eval(yyy) eval(r)]];
H1=inv(H); H2=[[eval(xx)];[eval(yy)]]; p=[[x];[y]];
while i<1
    x3=p-H1*H2; x=x3(1);
y=x3(2); x2=x3; ff=eval(f);
    if (abs(c-ff)>e)
        c=ff;
    else
        i=1;
    end
    krok=krok+1;
end
t=toc;
if(ret==1)
    c=ff; xx3=x; yy3=y; x=x+50; y=y+5; m1=eval(f);
 x=xx3; y=yy3; x=x-5; y=y-50;
    m2=eval(f); x=xx3; y=yy3;
    if(m1>c)&&(m2>c)
    set(handles.extrem,'String',['hodnota minima : ',num2str(ff)]);
end
if (m1<c)&&(m2<c)
set(handles.extrem,'String',['hodnota maxima : ',num2str(ff)]);
end
if (m1<c)&&(m2>c)||(m1>c)&&(m2<c)
set(handles.extrem,'String',['sedlový bod :',num2str(ff)]);
end
set(handles.souradnice,'String',['souřadnice [x;y]: [',num2str(x),';',num2str(y),']']);
set(handles.cas,'String',['počet iterací: ',num2str(krok),'  čas: ',num2str(t),'s']);
xx=x; yy=y; k=x; l=y; fff=handles.fce;
 f1=strrep(fff,'^','.^');f2=strrep(f1,'*','.*');
f3=strrep(f2,'/','./');  [x,y]=meshgrid(k-10:0.5:k+10,l-10:0.5:l+10); z=eval(f3);
plot3(xx,yy,ff,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z);hold off; grid on;
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
clear all;
```

P X : METODA DFP

Zdrojový kód Matlab:

```
syms x n; syms y; syms f; ret=1; f=handles.fce; dx=diff(f,x); dy=diff(f,y);i=0;
try
x=0; y=0; lk=eval(f); clear x; clear y;
catch
   errordlg('Špatně zadané proměné ve funkci! Podporovány jsou pouze x,y !!!', 'Error
Funkce');
   ret=0;i=2;
end
sd=0; sx=handles.iks; sy=handles.yps;
if(isnan(sx)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod X');
```

```
   ret=0;i=2;
end
if(isnan(sy)==1)
   errordlg('Musí být zadáno číslo!', 'Error Počáteční Bod Y');
   ret=0;i=2;
end
k=0;
S=([1 0; 0 1]);
tic
while (i<1)
   k=k+1; x=sx; y=sy; ssx=sx; ssy=sy; dfx=eval(dx); dfy=eval(dy); d=S*([dfx; dfy]);
   x=sx+n*d(1); y=sy+n*d(2); ff=diff(eval(f),n); N=solve(ff); sx=sx+N*d(1);
   sy=sy+N*d(2);v=N*d; x=sx; y=sy; dfx2=eval(dx); dfy2=eval(dy); df1=([dfx;dfy]);
   df2=([dfx2;dfy2]); w=df1-df2;
   S=S+((v*transpose(v))/(transpose(v)*w))-((S*w*transpose(w)*S)/(transpose(w)*S*w));
   dfxa=eval(dx); dfya=eval(dy);
   if(dfxa==0)&&(dfya==0)
   i=1;
   end
   t=toc;
   if(t>3)
   i=2;  sd=12;ret=0;
   end
end
t=toc; sx=eval(sx); sy=eval(sy); x=sx; y=sy; FS=eval(f);
if(sd==12)
set(handles.vypis1,'String',['Error algoritmu nebo funkce']);
set(handles.edit7,'String',['Zvolte jiný počáteční bod']);%,';',num2str(sx),']']);
set(handles.edit8,'String',['Error algoritmu nebo funkce']);
end
if(ret==1)
   c=FS; xx3=x;yy3=y; x=x+50; y=y+5; m1=eval(f); x=xx3; y=yy3; x=x-5; y=y-50;
   m2=eval(f);x=xx3; y=yy3;
   if(m1>c)&&(m2>c)
   set(handles.vypis1,'String',['hodnota minima : ',num2str(FS)]);
end
if (m1<c)&&(m2<c)
set(handles.vypis1,'String',['hodnota maxima : ',num2str(FS)]);
end
if (m1<c)&&(m2>c)||(m1>c)&&(m2<c)
set(handles.vypis1,'String',['sedlový bod :',num2str(FS)]);
end
%set(handles.vypis1,'String',['hodnota extrému je ',num2str(FS)]);
set(handles.edit7,'String',['souřadnice [x;y]:
[',num2str(sx),';',num2str(sy),']']);%,';',num2str(sx),']']);
set(handles.edit8,'String',['počet iterací: ',num2str(k),'     čas: ',num2str(t),'s']);
fff=handles.fce;
f1=strrep(fff,'^','.^');
f2=strrep(f1,'*','.*');
f3=strrep(f2,'/','./');
```

```
[x,y]=meshgrid(sx-5:0.5:sx+5 ,sy-5:0.5:sy+5);
z=eval(f3);
plot3(sx,sy,FS,'or','MarkerSize',10,'MarkerFaceColor','r');hold on
surf(x,y,z); grid on;hold off;
xlabel('osa X','FontSize',14);
ylabel('osa Y','FontSize',14);
zlabel('osa Z','FontSize',14);
end
clear all;
```