

Optimalizace

Petr Temlík

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

V rámci této práce bych chtěl poděkovat panu RNDr. Ing. Miloši Krčmářovi za zadání zajímavého tématu a za odborné vedení mé bakalářské diplomové práce.

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně, 29. 08. 2005

.....

podpis

ABSTRAKT

Abstrakt česky

Optimalizační metody jsou matematickou disciplínou, která řeší hledání extrému účelové funkce. Cílem mojí práce bylo vytvořit sadu příkladů pro předmět Optimalizace. Zpracoval jsem přehled jednotlivých optimalizačních metod. Z celé skupiny metod jsem se hlavně zaměřil na metody iterační. Podrobně popisuji jednotlivé metody, provádím jejich hodnocení a uvažuji o možnostech využití konkrétní metody na řešení určité problematiky. V příloze přikládám řadu příkladů v programovém prostředí *Mathematica* včetně grafického znázornění řešení.

Abstrakt ve světovém jazyce

Optimization methods come under mathematic discipline which solved searching for extreme of purpose. This thesis deals with the realization of examples set for the subject Optimization. I elaborated overview each one of optimization methods. From the whole group of methods I focused on iterated methods mainly. I characterized each method in detail, evaluated it and considered possibilities of applying concrete method to deal with specific problems. Enclosure includes examples set programmed in *Mathematica* inclusive of graphics demonstration of solution.

OBSAH

ÚVOD	6
1 ZÁKLADNÍ POJMY	7
2 KLASIFIKACE METOD.....	10
3 ANALYTICKÉ METODY	11
3.1 Jednorozměrný případ	11
3.2 Vícerozměrný případ.....	12
4 ITERAČNÍ METODY OPTIMALIZACE	13
4.1 Iterační metody jednorozměrné optimalizace	13
4.1.1 Metody jednorozměrné komparativní.....	14
4.1.2 Metody jednorozměrné gradientní.....	18
4.2 Iterační metody vícerozměrné optimalizace	21
4.2.1 Metody vícerozměrné komparativní	21
4.2.2 Metody vícerozměrné gradientní.....	26
4.3 Iterační metody náhodného hledání.....	31
ZÁVĚR.....	33
SEZNAM POUŽITÉ LITERATURY	35
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	36
SEZNAM OBRÁZKŮ.....	37
SEZNAM PŘÍLOH	38

ÚVOD

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.

Převédeme-li optimalizační úlohu na matematický tvar, její následné řešení je otázkou matematické rutiny a výpočetní techniky. Použitím výpočetní techniky lze značně zkrátit časy řešení optimalizačních úloh, a tím velmi podstatně zvýšit efektivnost rozhodování a řízení.

Impuls pro rozvoj extremalizačních metod přinesly v minulém století úlohy z oblasti ekonomie, které můžeme označit jako problémy optimalizace výrobních programů. Jinou skupinou těchto úloh z ekonomické oblasti jsou dopravní problémy. Řada extremalizačních úloh přichází z oblasti techniky. Jsou popsány výpočty směřující k optimalizaci technických konstrukcí 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.

Ve své práci podávám nejdříve přehled základních pojmů a definic, které používám v dalších kapitolách. Následuje přehled jednotlivých optimalizačních metod. Jejich klasifikace není jednoznačná a v různých odborných literaturách je pohled na jejich rozdělení různý. V další části práce se zaměřuji na jednu skupinu těchto metod – iterační metody optimalizace. Tyto metody vyhledávání extrémů funkcí jsou velice dobře zpracovány. Jsou popsány algoritmy, které jsou dobře programovatelné a nemají velké nároky na tvary účelových funkcí a omezení. Podrobně popisují jednotlivé metody, provádím jejich hodnocení a uvažuji o možnostech využití konkrétní metody na řešení určité problematiky. Okrajově se zmiňuji o konvergenci iteračních výpočetních postupů a o odhadu chyby výpočtu, podrobnější informace jsou uvedeny v [2].

V příloze přikládám řadu příkladů řešených v programovém prostředí *Mathematica* včetně grafického znázornění řešení. Většina algoritmů, které jsem použil, jsou podrobně popsány v [1].

1 ZÁKLADNÍ POJMY

Funkce $f(x)$ definovaná na množině $X \in \mathbb{R}^n$ má v bodě x_0 **ostré lokální maximum**, jestliže existuje okolí $\delta > 0$ tak, že pro každé x z tohoto okolí $0 < |x - x_0| < \delta$ platí $f(x) < f(x_0)$.

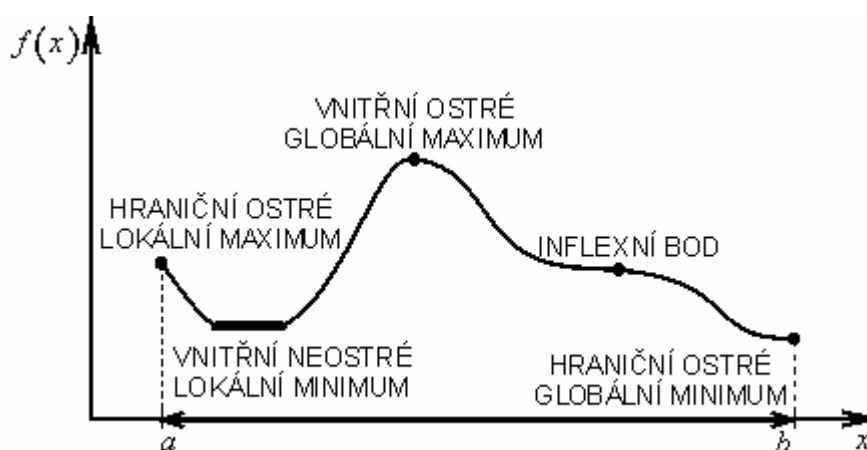
Jestliže platí neostrá nerovnost $f(x) \leq f(x_0)$, hovoříme o neostrém maximum.

Funkce $f(x)$ definovaná na množině $X \in \mathbb{R}^n$ má v bodě x_0 **ostré lokální minimum**, jestliže existuje okolí $\delta > 0$ tak, že pro každé x z tohoto okolí $0 < |x - x_0| < \delta$ platí $f(x) > f(x_0)$.

Jestliže platí neostrá nerovnost $f(x) \geq f(x_0)$, hovoříme o neostrém minimum.

Platí-li neostrá nerovnost $f(x) \leq f(x_0)$, resp. ostrá nerovnost $f(x) < f(x_0)$ pro všechny body $x \in X$, pak funkce $f(x)$ má v bodě x **globální**, resp. **ostré globální maximum**. Ostré globální maximum je vždy jednoznačné. Je zřejmé, že každé globální maximum je zároveň lokálním maximum.

Platí-li neostrá nerovnost $f(x) \geq f(x_0)$, resp. ostrá nerovnost $f(x) > f(x_0)$ pro všechny body $x \in X$, pak funkce $f(x)$ má v bodě x **globální**, resp. **ostré globální minimum**. Ostré globální minimum je vždy jednoznačné. Je zřejmé, že každé globální minimum je zároveň lokálním minimum.



Obr. 1: Extrémy funkce

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

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

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

Platí tato tvrzení:

Nechť funkce $f(x)$ má první derivaci na intervalu X a v bodě x_0 tohoto intervalu nechť existuje druhá derivace. Pak platí, že funkce $f(x)$ má v bodě x_0 ostré lokální maximum, když $f'(x_0)=0$ a $f''(x_0)<0$.

Jestliže $f'(x_0)=0$ a $f''(x_0)>0$, potom má funkce $f(x)$ v bodě x_0 ostré lokální minimum.

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

Jestliže je první derivace $f'(x_0)$ v levém okolí x_0 záporná a v pravém okolí x_0 kladná, potom má f v x_0 ostré lokální minimum.

Jestliže se $f'(x_0)=0$ a $f''(x_0)=0$, potom může mít funkce $f(x)$ v bodě x_0 **inflexní bod** nebo je nutné zkoumat vyšší derivace.

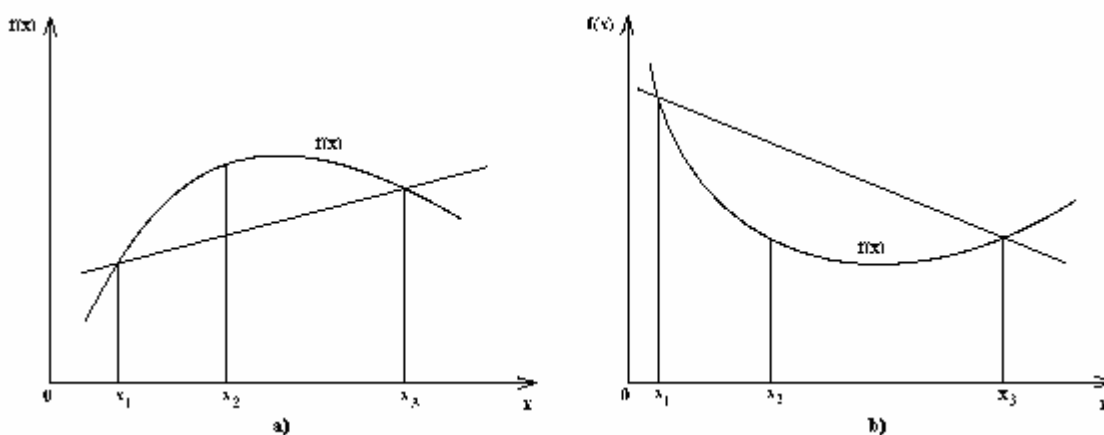
Nechť existuje přirozené číslo n tak, že $f^{(n)}(x_0) \neq 0$ a pro každé $k < n$ je $f^{(k)}(x_0)=0$. Potom platí, že jestliže je n sudé a $f^{(n)}(x_0)>0$, má $f(x)$ v x_0 ostré lokální minimum.

Jestliže n je sudé a $f^{(n)}(x_0)<0$, má $f(x)$ v x_0 ostré lokální maximum.

Jestliže je n liché, jde o inflexní bod.

Funkce definovaná na intervalu J se nazývá **konvexní (konkávní)**, když pro libovolnou trojici bodů z J $x_1 < x_2 < x_3$ 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. 2: a) konkávní funkce b) konvexní funkce

V inflexním bodě se mění konkávnost funkce na konvexnost nebo naopak. Konkávnost souvisí s maximem funkce a konvexnost s minimem.

Jestliže je funkce 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 konvexní právě tehdy, když pro každé $x \in X$ je $f''(x) \leq 0$.

Volný extrém funkce je lokální maximum nebo lokální minimum v libovolném bodě definičního oboru.

Vázaný extrém funkce – hledání extrému je omezeno omezeními typu rovnost (klasický vázaný extrém) nebo omezeními typu nerovnost (neklasický vázaný extrém). Některá omezení mohou být smíšená.

Gradient funkce, který je nutné použít při řešení vícerozměrných případů, se definuje jako vektor prvních parciálních derivací

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

Hessova matice se definuje pomocí druhých parciálních derivací

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

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

Při vyšetřování pozitivní či negativní definitnosti matice je třeba počítat hlavní subdeterminanty. Podmínky pro definitnost matice udává **Silvestrovo 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. Další definice viz. [1].

2 KLASIFIKACE METOD

Klasifikace metod řešení úloh optimalizace je velmi složitým problémem a má vždy charakter určité dohody. Proto také každá klasifikace má své nedostatky.

Základní rozdělení metod optimalizace:

- **Analytické metody** – využívají výsledků klasických i neklasických metod diferenciálního a variačního počtu.
- **Iterační metody** (numerické, krokové) – využívají každou předcházející informaci v iteračním procesu ke zlepšení řešení, přičemž se pracuje s konkrétními numerickými hodnotami. Konvergence se ve všeobecných případech nedá zaručit.
- **Speciální metody** – jedná se zejména o lineární, dynamické a konvexní programování
- **Grafické metody** – jsou založeny na grafickém zobrazení dané optimalizační úlohy a na její grafické analýze.
- **Experimentální metody** – experimentuje se přímo s reálnými objekty (veličinami), přičemž výsledky předcházejícího experimentu jsou využívány k plánování následujícího experimentu, což umožňuje dosáhnout zlepšení řešení.

3.2 Vícerozměrný případ

Příklad 2.: Určete lokální extrémy funkce $f(x,y)=x + y - xy - x^2 - y^2$

a) výpočet první a druhé parciální derivace

$$\frac{\partial f}{\partial x} = 1 - y - 2x \qquad \frac{\partial f}{\partial y} = 1 - x - 2y$$

$$\frac{\partial^2 f}{\partial x \partial y} = -1 \qquad \frac{\partial^2 f}{\partial y \partial x} = -1$$

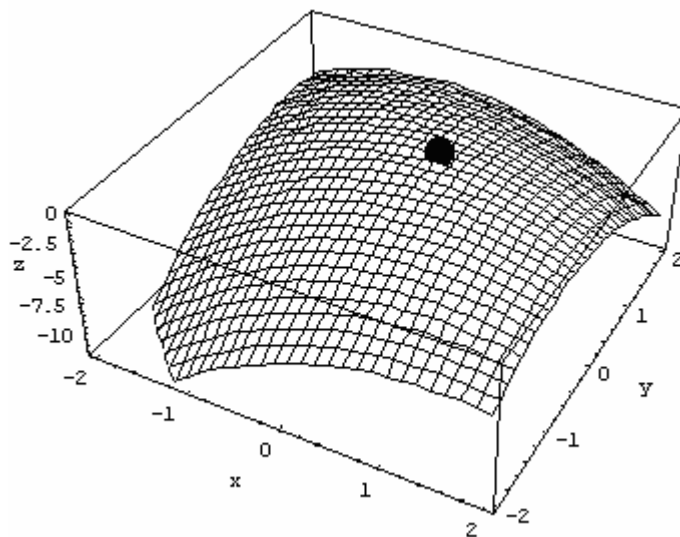
$$\frac{\partial^2 f}{\partial x^2} = -2 \qquad \frac{\partial^2 f}{\partial y^2} = -2$$

b) $\text{grad } f(x,y) = (1-y-2x; 1-x-2y)$

c) Hessova matice $H = \begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix}$ je negativně definitní

d) $\text{grad } f(x,y)=0 \Rightarrow x^0 = 1/3, y^0 = 1/3$

e) $f\left(\frac{1}{3}, \frac{1}{3}\right) = \frac{1}{3}$



Obr. 4: Průběh funkce-příklad 2.

Funkce $f(x,y)$ nabývá v bodě $(\frac{1}{3}, \frac{1}{3})$ lokální maximum $\frac{1}{3}$.

4 ITERAČNÍ METODY OPTIMALIZACE

Iterační metody optimalizace lze rozdělit na tři základní skupiny:

- ***komparativní metody*** – nevyžadují výpočet derivací, určují se pouze hodnoty účelové funkce. Jejich princip spočívá ve výpočtu diskrétních hodnot účelové funkce, jejich porovnání (komparace) a v určení postupu konstrukce dalších diskrétních bodů. Postupuje se tak dlouho, dokud se dosahuje dalšího zlepšení hodnoty účelové funkce.
- ***gradientní metody*** – vyžadují určování hodnot účelové funkce a její první, resp. druhé derivace. Gradientními metodami obecně budeme rozumět iterační algoritmus, ve kterém je přírůstek vektoru parametrů úměrný gradientu účelové funkce. Iterační vztah lze vyjádřit:

$$x_{i+1} = x_i + \lambda_i \nabla f(x_i) \quad (3)$$

Hodnota $\lambda_i < 0$ znamená minimalizaci funkce $f(x)$, hodnota $\lambda_i > 0$ se volí při maximalizaci. Koeficient λ_i nemusí být konstantní.

- ***metody náhodného hledání*** – představují speciální třídu iteračních metod. Iterační přírůstek proměnných je závislý na náhodně generované veličině.

4.1 Iterační metody jednorozměrné optimalizace

Tato část se zabývá iteračními metodami jednorozměrné optimalizace spojitých unimodálních funkcí (jsou to takové funkce, které mají jeden extrém na daném intervalu).

Úlohy jednorozměrné optimalizace řeší úlohu

$$x^* = \arg \min \{ f(x) \}, \text{ pro } x \in X \quad \text{nebo} \quad x^* = \arg \max \{ f(x) \}, \text{ pro } x \in X \quad (4)$$

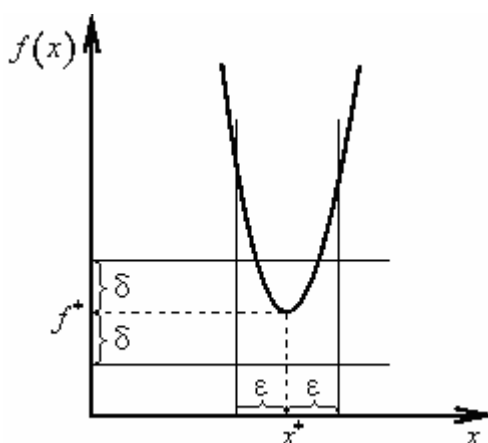
Přitom platí:

$$\arg \max \{ f(x) \} = \arg \min \{ -f(x) \} \quad (5)$$

Není-li daná účelová funkce $f(x)$ na intervalu $\langle a, b \rangle$ unimodální, pak je třeba přibližně určit polohu bodu globálního minima a interval $\langle a, b \rangle$ zmenšit tak, aby na zmenšeném intervalu účelová funkce $f(x)$ byla unimodální.

U iteračních metod je nutno předem zadat požadovanou přesnost optimálního řešení x^* , tj. nezáporné číslo ε , pro které N -tá aproximace řešení x_N musí vyhovovat nerovnosti

$$|x_N - x^*| \leq \varepsilon \quad (6)$$



Obr. 5: Přesnost řešení

U některých iteračních metod je kontrola splnění podmínky (6) problematická. Jde především o ty případy, kdy informace o vlastnostech účelové funkce $f(x)$ jsou velmi nedostatečné, proto se rovněž používají vztahy

$$|x_N - x_{N-1}| \leq \varepsilon_N \quad (7)$$

4.1.1 Metody jednorozměrné komparativní

Metody jednorozměrné optimalizace mají v celé teorii optimalizace základní význam. Jsou velmi názorné, mají jednoduchou grafickou interpretaci a hlavně – využívají se 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 – jak zvolit posloupnost x_i bodů z daného intervalu $\langle a, b \rangle$, aby byl nalezen interval co nejmenší délky, ve kterém se nachází bod extrému.

Obě metody nepředpokládají kromě unimodality žádné další vlastnosti funkce.

Fibonacciho metoda – zpracován příklad v programu *Mathematica* - viz. P I :

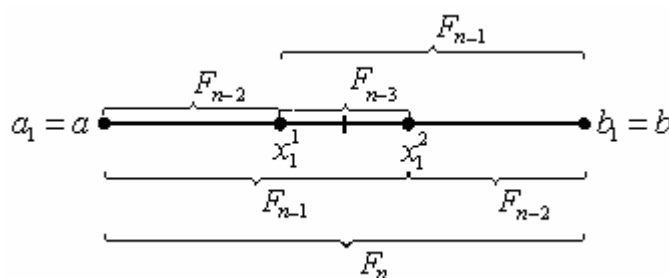
Princip metody:

Fibonacciho metoda se opírá o výpočet délky hledaného podintervalu podle vzorce

$$d_n = 2 \frac{F_1}{F_{n+1}} d_1 \quad (8)$$

kde $d_1 = b - a$ a F_i jsou členy tzv. Fibonacciho posloupnosti, definované vztahem

$$F_0 = F_1 = 1, \quad F_i = F_{i-1} + F_{i-2}, \quad i = 2, 3, \dots, \quad (9)$$



Obr. 6: Dělení intervalu u Fibonacciho metody

Algoritmus metody pro hledání maxima funkce:

1. $\alpha_1 = a \quad \beta_1 = b \quad \text{zvolíme } N$
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. $f(\tilde{\alpha}_{i+1}), \quad f(\tilde{\beta}_{i+1})$
4. $f(\tilde{\alpha}_{i+1}) \leq f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \tilde{\alpha}_{i+1} \quad \beta_{i+1} = \beta_i$
5. $f(\tilde{\alpha}_{i+1}) > f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \alpha_i \quad \beta_{i+1} = \tilde{\beta}_{i+1}$
6. $i = i + 1 \quad \text{skok na bod 2}$

(10)

Pro výpočet minima funkce je nutné změnit nerovnost v kroku 5.

Hodnocení metody:

Při výpočtu se vždy jedno z dvojic čísel α_i, β_i rovná jednomu z dvojice čísel $\alpha_{i+1}, \beta_{i+1}$,

takže skutečně se provádí výpočet hodnoty účelové funkce $f(x)$ jen v N bodech. Nevýhodou Fibonacciovy metody je, že je třeba určit počet iterací N před zahájením výpočtu. Platí vztah:

$$F_N \geq \frac{b-a}{2\varepsilon} \quad (11)$$

tj. musíme najít takové číslo Fibonacciovy posloupnosti, které vyhovuje této nerovnosti. Index N udává počet potřebných kroků iterace.

Např. pro funkci $f(x)$ definovanou na intervalu $\langle 0,1 \rangle$ hledáme řešení s přesností $\varepsilon = 0,01$.

Počet kroků N zjistíme ze vztahu (11) $F_N \geq \frac{b-a}{2\varepsilon} = \frac{1-0}{2 \cdot 0,01} = 50 \Rightarrow F_9 = 55 \Rightarrow N = 9$

Metoda zlatého řezu – zpracován příklad v programu *Mathematica* - viz. P V :

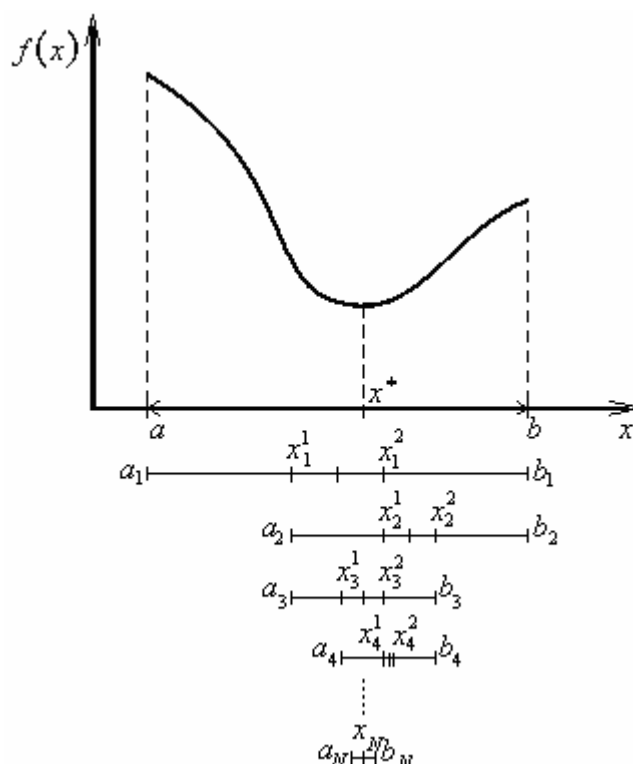
Princip metody:

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

limitní podíl:

$$\lim_{n \rightarrow \infty} \frac{F_{N-i+1}}{F_{N-i+2}} = 0,618 \quad (12)$$

Přesné odvození limitní hodnoty je popsán v [2].



Obr. 7: Princip metody zlatého řezu

Algoritmus metody pro hledání maxima funkce:

1. $\alpha_1 = a \quad \beta_1 = b \quad \text{zvolíme } N$
 2. $\tilde{\alpha}_{i+1} = \beta_i - 0,618 \cdot |\beta_i - \alpha_i|$
 $\tilde{\beta}_{i+1} = \alpha_i + 0,618 \cdot |\beta_i - \alpha_i|$
 3. $f(\tilde{\alpha}_{i+1}), \quad f(\tilde{\beta}_{i+1})$
 4. $f(\tilde{\alpha}_{i+1}) \leq f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \tilde{\alpha}_{i+1} \quad \beta_{i+1} = \beta_i$
 5. $f(\tilde{\alpha}_{i+1}) > f(\tilde{\beta}_{i+1}) \Rightarrow \alpha_{i+1} = \alpha_i \quad \beta_{i+1} = \tilde{\beta}_{i+1}$
 6. $i = i + 1 \quad \text{skok na bod 2}$
- (13)

Pro výpočet minima funkce je nutné změnit nerovnost v kroku 5.

Hodnocení metody:

Výpočet se zjednoduší, ale rychlost konvergence je nižší.

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 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. Počet kroků není explicitně omezený, výpočet končí testem na přesnost argumentu.

Počet kroků N je dán požadovanou přesností ε , pro kterou platí vztah:

$$b_N - a_N \leq 2\varepsilon \quad (14)$$

U obou výše uvedených metod počet kroků N nemůže být příliš veliký, protože díky zaokrouhlování může dojít k numerické nestabilitě. Doporučuje se volit $N \leq 20$.

4.1.2 Metody jednorozměrné gradientní

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 – zpracován příklad v programu *Mathematica* - viz. P II :

Princip metody:

Metoda spočívá v sestrojení tečny v bodě $[x_k, f'(x_k)]^T$ funkce $f'(x)$, viz obr. 8. Průsečík tečny s osou x , tj. bod x_{k+1} je $(k+1)$ -ní iterace optimálního bodu x^* .

Podrobnější popis metody a její modifikace včetně řešených příkladů lze nalézt na [5].

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

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

$$|x_{i+1} - x_i| \leq \varepsilon \quad (18)$$

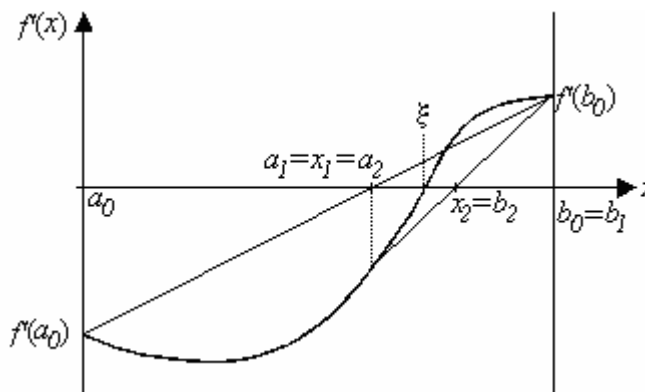
V praxi je možno použít modifikaci Newtonovy metody spočívající v tom, že když druhá derivace $f''(x)$ se již mnoho nemění, lze ji ponechat beze změny i pro následující iterace, tj. druhá derivace se nahradí konstantou (tzv. Whittakerova metoda – zmínka na [5]).

Regula falsi – zpracován příklad v programu *Mathematica* - viz. P III :

Princip metody:

Druhá derivace z Newtonovy metody je zde nahrazena numerickým odhadem:

$$f''(x_i) = \frac{f'(x_i) - f'(x_{i-1})}{x_i - x_{i-1}} \quad (19)$$



Obr. 9: Princip metody regula falsi

Algoritmus metody:

Účelová funkce se aproximuje polynomem:

$$p(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{f'(x_{i-1}) - f'(x_i)}{x_{i-1} - x_i} \cdot \frac{(x - x_i)^2}{2} \quad (20)$$

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

Derivováním vztahu pro polynom $p(x)$ se získá

$$p'(x) /_{x=x_i} = f'(x_i) + \frac{f'(x_{i-1}) - f'(x_i)}{x_{i-1} - x_i} (x_{i+1} - x_i) = 0 \quad (21)$$

a odtud vyplývá iterační vztah metody ve tvaru:

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

Hodnocení metody:

Metoda regula falsi potřebuje dva inicializační body x_0 a x_1 . Metoda vyžaduje existenci první derivace na celém intervalu, ve kterém hledáme extrém.

4.2 Iterační metody vícerozměrné optimalizace

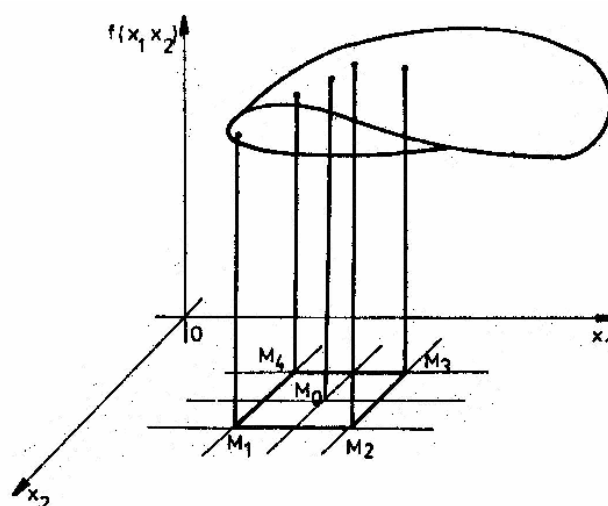
4.2.1 Metody vícerozměrné komparativní

Mezi nejznámější metody vícerozměrné komparace patří metoda Boxova-Wilsonova, simplexové metody a metoda cyklické záměny parametrů.

Boxova - Wilsonova metoda – zpracován příklad v programu *Mathematica* - viz. P IV :

Princip metody:

Účelová funkce se vypočítá 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.



Obr. 10: Boxova-Wilsonova metoda ve dvourozměrném případě

Algoritmus metody:Metoda pro $N=2$:

1. zvolit střed M_0 libovolně, stanovit hodnotu Δ
2. okolo středu M_0 sestrojít vrcholy čtverce M_1, M_2, M_3, M_4
3. vypočítat hodnoty y_0, y_1, y_2, y_3, y_4 v jednotlivých bodech čtverce M_0, M_1, M_2, M_3, M_4
4. určit extrémální bod z y_1, y_2, y_3, y_4 a označit ho y_e
5. jestliže je funkční hodnota y_0 v bodě M_0 bližší extrému, výpočet končí
6. v opačném případě $y_0 = y_e$, tj. novým středem se stává bod s hodnotou nejbližší extrému

Hodnocení metody:

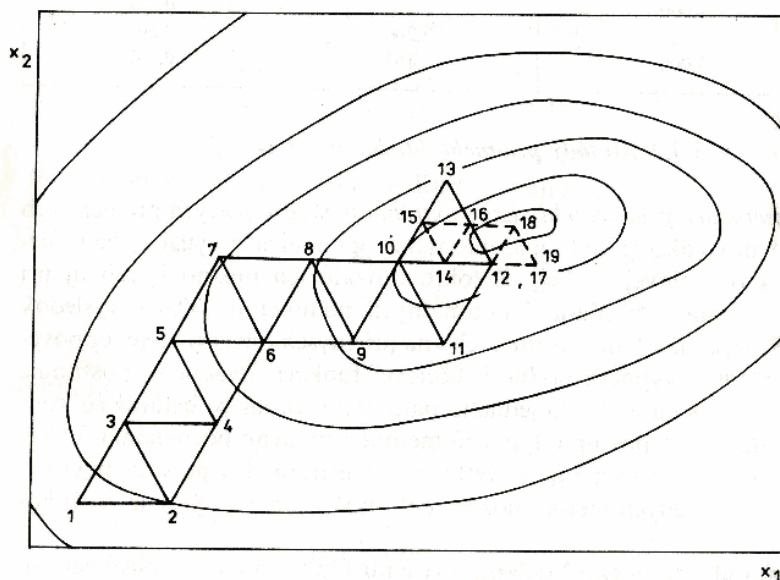
Postup se velmi lehce zevšeobecní na N -rozměrný případ tak, že místo čtverce se konstruuje nadkvádr s 2^N vrcholy. Nevýhodou je výpočet $1 + 2^N$ funkčních hodnot funkce. Rychlost konvergence je hodně závislá na výběru souřadnic středu a délky strany čtverce.

Metoda pravidelného simplexu – zpracován příklad v programu *Mathematica* - viz. P VI :

Princip metody:

Simplexové metody patří mezi další komparativní metody, které nepotřebují výpočet derivací účelové funkce. Oproti předcházející metodě se počet bodů výpočtu hodnot účelové funkce redukuje na minimum. Ve dvojrozměrném prostoru je simplexem rovnostranný trojúhelník, v třírozměrném prostoru je to pravidelný čtyřstěn.

Základní strategií pro maximalizaci účelové funkce v R^n je výpočet hodnot účelové funkce ve vrcholech simplexu a určení bodu s nejmenší funkční hodnotou. Tento bod se překlopí přes těžiště simplexu do směru největšího růstu tak daleko, aby tento bod tvořil se základnou opět pravidelný simplex. Modifikace této metody je popsána v [4] – viz. Obr.11.



Obr. 11: Hledání extrému funkce $f(x_1, x_2)$ simplexovou metodou

Algoritmus metody:

V R^n jsou body pravidelného simplexu určené následujícími vztahy:

$$x_1 = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} \quad x_2 = \begin{pmatrix} a_1 + d + e \\ a_2 + d \\ \dots \\ a_n + d \end{pmatrix}, \quad \dots, \quad x_{n+1} = \begin{pmatrix} a_1 + d \\ a_2 + d \\ \dots \\ a_n + d + e \end{pmatrix} \quad (23)$$

$$d = \rho \frac{\sqrt{n+1}-1}{\sqrt[2]{2}} \quad e = d + \frac{\rho}{2} \quad (24)$$

kde ρ je délka hrany simplexu a n je dimenze prostoru.

Nový vrchol simplexu se sestrojí podle vztahu:

$$x = x_m + 2(c - x_m) = 2c - x_m \quad (25)$$

$$c = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_j \quad (26)$$

Bod x_m je bod s nejnižší hodnotou účelové funkce.

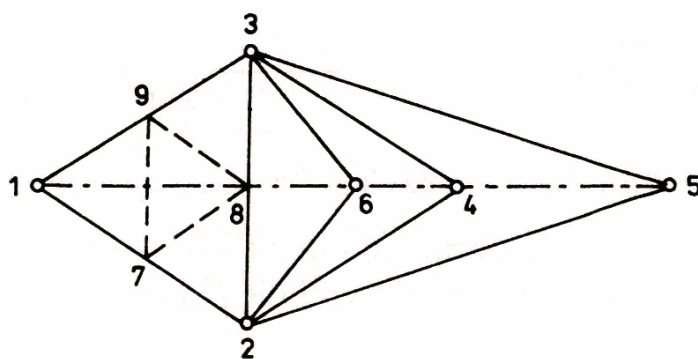
Hodnocení metody:

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 extrémálního bodu.

Metoda flexibilního simplexu (Nelder-Mead)

Princip metody:

Metodu pravidelného simplexu lze vylepšit tak, že délka hrany simplexu se v průběhu výpočtu mění, např. při větší vzdálenosti se krok prodlužuje a naopak v blízkosti extrému se krok zkracuje. Tím se zlepší konvergence a nedochází k oscilacím okolo extrémálního bodu.



Obr. 12: Flexibilní simplexová metoda

Algoritmus metody:

Zkonstruují se vrcholy simplexu $x_1, \dots, x_{n+1}$ s maximální hodnotou v x_i .

K dalšímu bodu se postupuje podle vztahů:

1. expanze : jestliže $f(x_i) > \max\{f(x_1), f(x_2), \dots, f(x_{n+1})\}$, potom
 $x(x_{i+1}) = c + \alpha(x_i) - c \quad \alpha \in (1, \infty)$

jestliže $f(x_{i+1}) > \max\{f(x_1), \dots, f(x_{n+1})\}$,

pak se pokračuje v postupu úspěšným směrem;

2. kontrakce : jestliže $f(x_i) < f(x_j) \quad j \neq i$, potom
 $x_{i+1} = c + \beta(x_i - c) \quad \beta \in (0, 1)$

3. redukce : jestliže $f(x_{i+1}) < f(x_k)$, potom se zmenší délka hrany simplexu
 podle vztahu $x_j = x_k + \frac{1}{2}(x_j - x_k)$

Hodnocení metody:

Jedná se o nejdokonalejší komparativní metodu.

Metoda cyklické záměny parametrů – zpracován příklad v programu *Mathematica* – viz. P VII :

Princip metody:

Při optimalizaci v R^n se zvolí jedna proměnná x_i a ostatní se nechají konstantní. Najde se částečné optimum nějakou metodou jednorozměrné optimalizace a proměnná x_i 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.

Algoritmus metody:

Algoritmus je vhodné ukázat na konkrétním příkladu:

Příklad 3.: Najděte maximum funkce $f(x,y)=16x + 12y - 2x^2 - y^2$ metodou cyklické záměny parametrů.

a) v 1. kroku se extremalizuje funkce podle proměnné x

$$f(0 + \eta_1, 0) = p(\eta_1) = 16\eta_1 - 2\eta_1^2$$

$$\frac{dp}{d\eta_1} = 16 - 4\eta_1 = 0 \Rightarrow \eta_1 = 4$$

b) v 2. kroku se extremalizuje funkce podle proměnné y

$$f(4, 0 + \eta_2) = q(\eta_2) = 32 + 12\eta_2 - \eta_2^2$$

$$\frac{dq}{d\eta_2} = 12 - 2\eta_2 = 0 \Rightarrow \eta_2 = 6$$

c) postup se zopakuje z nového bodu (4,6):

$$p(\eta_1) = f(4 + \eta_1, 6) = 68 - \eta_1^2$$

$$\frac{dp}{d\eta_1} = -4\eta_1 = 0 \Rightarrow \eta_1 = 0$$

$$q(\eta_2) = f(4, 6 + \eta_2) = 68 - \eta_2^2$$

$$\frac{dq}{d\eta_2} = -2\eta_2 = 0 \Rightarrow \eta_2 = 0$$

V druhém cyklu je přírůstek v obou souřadnicích nulový, proto je zřejmý bod extrému $x^0 = (4, 6)$.

Hodnocení metody:

Uvedený postup však nemusí být vždy konvergentní a konvergence závisí na pořadí proměnných. Pro všeobecný tvar účelové funkce se metoda programuje obtížně. V případě ryze konkávní funkce metoda skutečně konverguje k bodu globálního maxima funkce.

4.2.2 Metody vícerozměrné gradientní

Gradientní iterační postupy se realizují všeobecně podle vztahu (3).

Newtonova metoda – zpracován příklad v programu *Mathematica* - viz. P VIII :

Princip metody:

Je rozšířením Newtonovy metody z kapitoly [4.1.2](#) na vícerozměrný případ.

Algoritmus metody:

Tato metoda je dána iteračním vztahem:

$$x_{i+1} = x_i - [\nabla^2 f(x_i)]^{-1} \nabla f(x_i) \quad (27)$$

kde $\nabla^2 f(x_i)$ je Hessova matice druhých parciálních derivací,

$\nabla f(x_i)$ je gradient funkce

Hodnocení metody:

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

Gradientní metoda s krátkým krokem – zpracován příklad v programu *Mathematica* - viz. P IX :

Algoritmus metody:

Algoritmus výpočtu se realizuje podle vztahu

$$x_{i+1} = x_i + \lambda_i \nabla f(x_i) \quad (28)$$

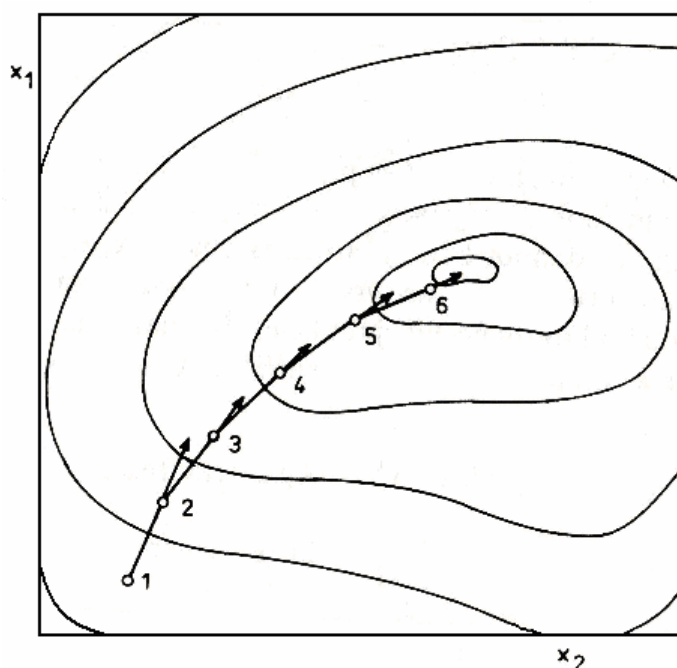
Tato metoda používá k výpočtu konstantní λ_i .

Hodnocení metody:

Výpočet je poměrně jednoduchý, ale úspěšná konvergence postupu není vždy zaručená. Snadno se hledaný extrém může přeskočit. Ani volba $\lambda_i = \lambda_0 \cdot 2^{-i}$ tento problém neodstraňuje. Pro výpočet minima funkce musí být λ_i záporné.

Gradientní metoda s dlouhým krokem**Princip metody:**

Byly vypracovány metody, které místo konstantního λ_i u gradientní metody s krátkým krokem mění λ_i v každém kroku tak, aby se dosáhlo co největšího možného zlepšení účelové funkce směrem k minimu anebo k maximu. Více v [4].

Obr. 13: Hledání extrému funkce $f(x,y)$ gradientní metodou**Algoritmus metody:**

Při této metodě se výpočet λ_i uskutečňuje podle vztahu:

$$\lambda_i = \max\left\{\lambda; \frac{d}{d\lambda} f(x_i + \lambda \nabla f(x_i)) = 0\right\} \quad (29)$$

Hodnocení metody:

Je to nejběžnější z gradientních metod. 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 není možné získat první a druhé derivace přímo v analytické formě a použijí se numerické odhady derivací pro vhodně zvolené malé kladné Δ . Nejjednodušší vzorce tzv. dvojbodové mají následující tvary symetrický vzorec

$$\frac{\partial f}{\partial x_i} = \frac{f(x_1, \dots, x_i + \Delta, \dots, x_n) - f(x_1, \dots, x_i - \Delta, \dots, x_n)}{2\Delta} \quad (30)$$

levý nesymetrický vzorec

$$\frac{\partial f}{\partial x_i} = \frac{f(x_1, \dots, x_i + \Delta, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{\Delta} \quad (31)$$

nebo pravý nesymetrický vzorec

$$\frac{\partial f}{\partial x_i} = \frac{f(x_1, \dots, x_i, \dots, x_n) - f(x_1, \dots, x_i - \Delta, \dots, x_n)}{\Delta} \quad (32)$$

Pro druhou derivaci

$$\frac{\partial^2 f}{\partial x_i^2} = \frac{f(x_1, \dots, x_i + \Delta, \dots, x_n) - 2f(x_1, \dots, x_i, \dots, x_n) + f(x_1, \dots, x_i - \Delta, \dots, x_n)}{2\Delta} \quad (33)$$

Numerickým derivováním se do výpočtu zanáší těžko kontrolovatelné chyby metody – viz. [2].

Metoda konjugovaných gradientů

Princip metody:

Existují i další varianty gradientních metod s dlouhým krokem, které se snaží jiným, efektivnějším způsobem nahradit jednorozměrnou extremalizaci podle λ v iteračním postupu.

Algoritmus metody:

Iterační algoritmus lze popsat následujícími vztahy:

$$\begin{aligned} d_0 &= \nabla f(x_0) & x_0 \dots \text{počáteční bod postupu} \\ x_{i+1} &= x_i + \lambda_i d_i \\ \lambda_i &= -\frac{\nabla f_i^T d_i}{d_i^T \nabla^2 f_i d_i} \\ d_{i+1} &= \nabla f(x_{i+1}) + \delta_i d_i \\ \delta_i &= -\frac{\nabla f_{i+1}^T \nabla f_i}{d_i^T \nabla^2 f_i d_i} d_i \end{aligned} \quad (34)$$

Hodnocení metody:

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

Metoda paralelních tečen - PARTAN

Princip metody:

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

Algoritmus metody:

Základní algoritmus pro hledání maxima této metody je:

1. volba počátečního bodu x_0 ; volba N
2. $i = 1, x_1 = x_0 + \lambda \nabla f_0$ pro λ , které maximalizuje $f(x_0 + \tilde{\lambda} \nabla f_0)$
3. $y_i = x_i + \tilde{\lambda}_i \nabla f_i$, kde $\tilde{\lambda}_i$ maximalizuje $f(x_i + \tilde{\lambda} \nabla f(x_i))$
4. $x_{i+1} = x_{i-1} + \delta_i [y_i - x_{i-1}]$, kde δ_i maximalizuje $f(x_{i+1}) = f(x_{i-1} + \delta_i [y_i - x_{i-1}])$ (35)
5. jestliže $i < N$, potom $i = i + 1$ a skok na 3,
6. jestliže $i = N$, potom $x_0 = x_N$ a skok na 2.

Hodnocení metody:

Nepříjemnou vlastností metody PARTAN je, že v jednom kroku se provádí dvakrát jednorozměrná extremalizace.

Metoda DFP (Davidon, Fletcher, Powell)**Princip metody:**

Další metoda se opírá o algoritmus Newtonovy metody, ve které se namísto Hessiany matice druhých parciálních derivací konstruuje pouze její odhad. Kombinuje se zde jednoduchost gradientní metody s dobrou konvergencí metody Newtonovy.

Algoritmus metody:

Metoda vychází z iteračního vztahu:

$$x_{i+1} = x_i + \lambda_i S_i \nabla f_i, \quad (36)$$

kde S_i je matice, která aproximuje inverzi Hessiany 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 λ_i maximalizuje $f(x_i + \lambda_i d_i)$
a výpočet $f(x_{i+1})$
4. $v_i = \lambda_i d_i$ $w_i = \nabla f_i - \nabla f_{i+1}$
$$S_{i+1} = S_i + \frac{v_i v_i^T}{v_i^T w_i} - \frac{S_i w_i w_i^T S_i}{w_i^T S_i w_i}$$
5. $i = i + 1$ a skok na 2.

(37)

4.3 Iterační metody náhodného hledání

Metody lze rozdělit na dvě třídy:

jednoduché algoritmy hledání - pouze se vyhodnocuje, zda je nový bod lepší než předcházející. Jestliže tomu tak není, strategie se vrací do předcházejícího bodu a generuje se nový směr a velikost kroku.

metody s adaptací (s učením), při kterých se nějakou strategií vyhodnocuje, jestli se jde správným směrem anebo ne. Potom se mění velikost a směr kroku.

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

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

$$\begin{aligned}
 x_{i+1} &= x_i + \Delta x_i \\
 \Delta x_i &= a_i \psi[i; \Delta f_i, x_i] \xi_i \\
 \Delta f_i &= f_i - f_{i-1}
 \end{aligned}
 \tag{38}$$

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. Podrobnější popis algoritmů a způsobů řešení lze nalézt v [1].

ZÁVĚR

Optimalizace je proces interaktivního zlepšování řešení vzhledem k požadavkům, které má řešení splňovat. Optimalizační metody nalézají uplatnění ve všech různých oborech lidské činnosti.

V minulém století docházelo k rozvoji optimalizačních metod a začaly nacházet uplatnění v řadě praktických, průmyslových a ekonomických aplikací. S rozvojem výpočetní techniky se řešení optimalizačních metod stává jednodušším. Přesto je ale důležité znát principy jednotlivých metod pro snadnější rozhodování, kterou z metod pro řešení dané problematiky použít. Ne každá metoda je vhodná pro všechny druhy problémů. Vždy je nutné zvážit, o jakou funkci se jedná, zda splňuje požadavky metody, složitost dalších pomocných výpočtů apod. Použitím výpočetní techniky lze značně zkrátit časy řešení optimalizačních úloh.

Ve své práci jsem nejdříve zpracoval přehled různých typů optimalizačních metod. Stěžejní část mojí bakalářské práce se zabývá jednou skupinou těchto metod, a to iteračními metodami. Tyto metody jsou velice dobře propracovány. Jsou popsány algoritmy, které jsou dobře programovatelné. Iterační metody lze rozdělit do tří kategorií: metody komparativní, gradientní a metody náhodného vyhledávání. Pro přehlednost jsem provedl další rozdělení na jednorozměrné a vícerozměrné metody. V jednotlivých kapitolách jsem popsal princip, algoritmus daných metod a provedl jejich hodnocení.

Komparativní metody nevyžadují výpočet derivací, pro určení extrému účelové funkce se provádí pouze výpočet hodnot funkce v určitých bodech a jejich hodnocení (komparace). Komparativní metody se poměrně snadno programují vzhledem ke slabým požadavkům na účelovou funkci (unimodalita a spojitost). Tyto metody ale většinou konvergují pomaleji.

Gradientní metody vyžadují výpočet derivací. Rychlejší konvergenci prokázali metody, které využívají druhé derivace funkce.

Z této teoretické části jsem pak čerpal při programování iteračních metod v prostředí *Mathematica*. Z jednorozměrných komparativních metod jsem programově zpracoval metodu Fibonacciho a metodu zlatého řezu. Z vícerozměrných pak Boxovu-Wilsonovu metodu, metodu pravidelného simplexu a metodu cyklické záměny parametrů. Z jednorozměrných gradientních metod jsem odladil algoritmus metody Newtonovy a metody regula falsi, z vícerozměrných metodu gradientní s krátkým krokem.

Správnost výpočtu a vlastnosti metod jsem si důkladně ověřil na praktických příkladech. Na začátku programů jednotlivých metod jsou stručně popsány hlavní výhody a nevýhody metody a vstupní parametry. Protože nalezení minima účelové funkce $f(x)$ je ekvivalentní nalezení maxima účelové funkce $-f(x)$, stačí pouhým obrácením znaménka funkce a změnou některých vstupních parametrů provést výpočet maxima nebo minima dané funkce. Výsledky výpočtů jsem převedl do grafické podoby, ve kterých je vidět postup k řešení v jednotlivých krocích.

Hlavním přínosem této práce je sada programů, která bude použita ve výuce předmětu Optimalizace, který se touto problematikou zabývá v magisterském studiu. Doufám, že i teoretická část mé bakalářské práce může posloužit studentům jako studijní text pro tento předmět.

SEZNAM POUŽITÉ LITERATURY

- [1] Prokop, R.: Teória systémov a optimalizácia, CHTF STU Bratislava, 1985
- [2] Mañas, M.: Optimalizační metody, SNTL Praha 1979
- [3] Kaňka, M., Henzler, J.: Matematická analýza (Matematika B pro VŠE), VŠE v Praze, fakulta informatiky a statistiky, 1995
- [4] Brunovská, A.: Malá optimalizácia, Alfa Bratislava, 1990
- [5] Statická optimalizace, dostupný z URL:
< www.fs.vsb.cz/books/statickaoptimalizace >

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

R^n	n-rozměrný prostor
X	množina přípustných řešení
$\nabla f(x)$	gradient funkce
$H = \nabla^2 f(x_1, \dots, x_n)$	Hessova matice
λ_i	koeficient gradientní metody
ρ	délka hrany simplexu
ε	požadovaná přesnost řešení
F_i	Fibonacciho posloupnost

SEZNAM OBRÁZKŮ

Obr. 1: Extrémy funkce.....	7
Obr. 2: a) konkávní funkce b) konvexní funkce.....	8
Obr. 3: Průběh funkce-příklad 1.	11
Obr. 4: Průběh funkce-příklad 2.	12
Obr. 5: Přesnost řešení	14
Obr. 6: Dělení intervalu u Fibonacciho metody.....	15
Obr. 7: Princip metody zlatého řezu	17
Obr. 8: Princip Newtonovy metody	19
Obr. 9: Princip metody regula falsi	20
Obr. 10: Boxova-Wilsonova metoda ve dvourozměrném případě	22
Obr. 11: Hledání extrému funkce $f(x_1, x_2)$ simplexovou metodou.....	23
Obr. 12: Flexibilní simplexová metoda	24
Obr. 13: Hledání extrému funkce $f(x, y)$ gradientní metodou	28

SEZNAM PŘÍLOH

P I :		Fibonacciho metoda
P II :		Newtonova metoda
P III :		Metoda regula falsi
P IV :		Boxova-Wilsonova metoda
P V :		Metoda zlatého řezu
P VI :		Metoda pravidelného simplexu
P VII :		Metoda cyklické záměny parametrů
P VIII :		Newtonova metoda 3D
P IX :		Gradientní metoda s krátkým krokem

P I: FIBONACCIHO METODA

Zdrojový kód programu Mathematica:

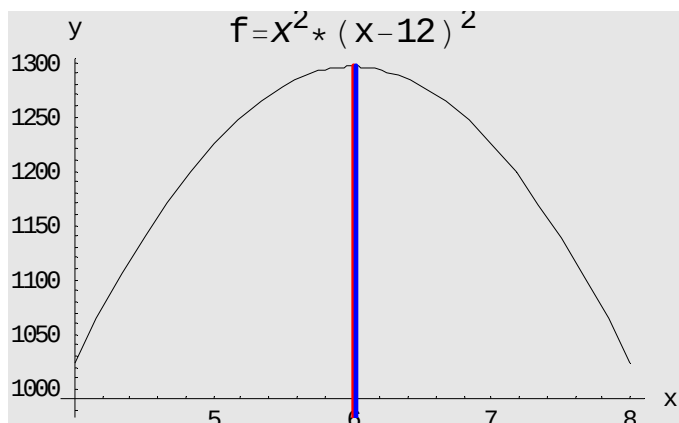
```

f[x_] := x2 (x - 12)2; aa = 4; bb = 8; n = 12; name = "f=x2*(x-12)2";
extrem = "Max";
(*f[x_] := -(x2 (x-12)2); aa=4; bb=8; n=12; name="f=-(x2*(x-12)2)";
extrem="Min";*)
Fib = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597};
a = aa; b = bb;
For[i = 1, i < n + 1, A = b -  $\frac{\text{Fib}[n - i + 1]}{\text{Fib}[n - i + 2]} * \text{Abs}[b - a]$ ;
  B = a +  $\frac{\text{Fib}[n - i + 1]}{\text{Fib}[n - i + 2]} * \text{Abs}[b - a]$ ; FA = f[A]; FB = f[B];
  If[If[extrem == "Max", FA > FB, FA < FB], a = a; b = B, a = A; b = b];
  Print[i, ". krok   Interval  <", a // N, " ; ", b // N, ">"];
  Show[Plot[f[x], {x, aa, bb}, DisplayFunction -> Identity],
    Graphics[{RGBColor[1, 0, 0], Thickness[0.01], Line[{a, 0}, {a, f[a]}]}],
    Graphics[{RGBColor[0, 0, 1], Thickness[0.01], Line[{b, 0}, {b, f[b]}]}],
    Axes -> Automatic, AxesLabel -> {"x", "y"},
    PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
    Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction]; i++];
Clear[f];

```

Graf funkce:

11. krok Interval < 6.00858 ; 6.02575 >



P II : NEWTONOVA METODA

Zdrojový kód programu Mathematica:

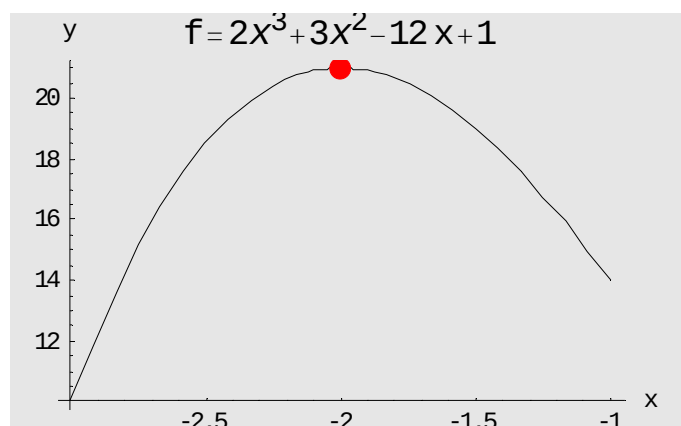
```

f[x_] := 2 x^3 + 3 x^2 - 12 x + 1; a = -3; b = -1; name = "f=2x^3+3x^2-12x+1";
ε = 0.01;
(*f[x_] := -(2 x^3 + 3 x^2 - 12 x + 1); a = -3; b = -1; name = "f=-(2x^3+3x^2-12x+1)";
ε = 0.01; *)
krok = 1; i = 0;
xk-1 = a; xk = b;
While[i < 1, yk1 = f'[xk]; yk2 = f''[xk];
  xk+1 = xk -  $\frac{y_{k1}}{y_{k2}}$ ; If[Abs[xk+1 - xk] > ε, xk = xk+1;
  Print[krok, ". krok řešení: x = ", xk+1 // N, " Hodnota: ",
    f[xk+1] // N];
  graf1 = Plot[f[x], {x, a, b}, DisplayFunction -> Identity];
  Show[graf1,
    Graphics[{RGBColor[1, 0, 0], PointSize[0.04], Point[{xk+1, f[xk+1]}]}],
    Axes -> Automatic, AxesLabel -> {"x", "y"},
    PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
    Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction];
  krok++, i = 1];
Clear[f];

```

Graf funkce:

4. krok Řešení: x= -2.00005 Hodnota: 21.



P III : METODA REGULA FALSI

Zdrojový kód programu Mathematica:

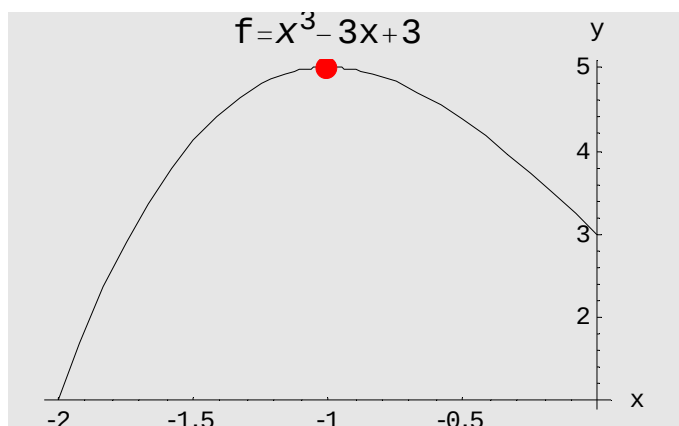
```

f[x_] := x3 - 3x + 3; a = -2; b = 0; name = "f=x3-3x+3"; ε = 0.01;
(*f[x_] := -(x3-3x+3); a=-2; b=0; name="f=-(x3-3x+3)"; ε=0.01;*)
krok = 1; i = 0;
xk-1 = a; xk = b;
While[i < 1, yk-1 = f'[xk-1]; yk = f'[xk];
  xk+1 = xk - yk *  $\frac{x_{k-1} - x_k}{y_{k-1} - y_k}$ ; If[Abs[xk+1 - xk] > ε, xk-1 = xk; xk = xk+1;
  Print[krok, ". krok  řešení: x= ", xk+1 // N, "  Hodnota: ",
    f[xk+1] // N];
  Show[Plot[f[x], {x, a, b}, DisplayFunction → Identity],
    Graphics[{RGBColor[1, 0, 0], PointSize[0.04], Point[{xk+1, f[xk+1]}]}],
    Axes → Automatic, AxesLabel → {"x", "y"},
    PlotLabel → StyleForm[TraditionalForm[name], FontSize → 15],
    Background → GrayLevel[0.9], DisplayFunction → $DisplayFunction];
  krok++, i = 1]];
Clear[f];

```

Graf funkce:

5. krok řešení: x= -1.00826 Hodnota: 4.99979



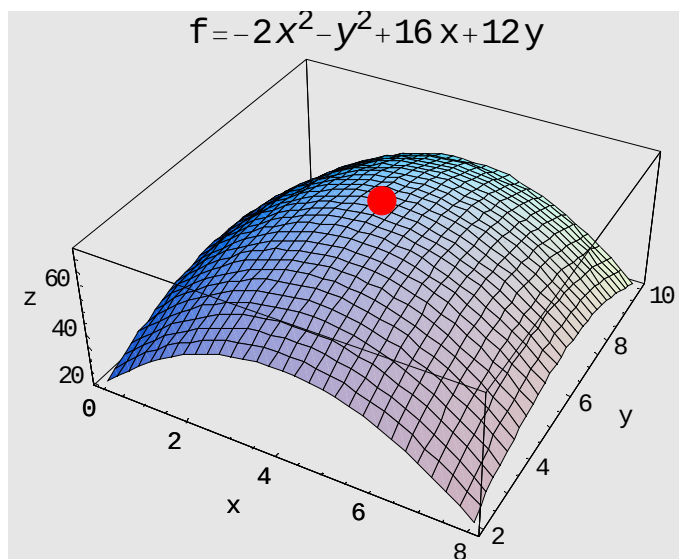
P IV : BOXOVA-WILSONOVA METODA

Zdrojový kód programu Mathematica:

```
f[x_, y_] := -2 x^2 - y^2 + 16 x + 12 y; a1 = 0; a2 = 8; b1 = 2; b2 = 10; Δ = 0.5;
name = "f=-2x^2-y^2+16x+12y"; bod = 5; extrem = "Max";
(*f[x_, y_] := -(-2 x^2 - y^2 + 16 x + 12 y); a1 = 0; a2 = 8; b1 = 2; b2 = 10; Δ = 0.5;
name = "f=-(-2x^2-y^2+16x+12y)"; bod = 5; extrem = "Min"; *)
Sx = {0}; Sy = {0};
i = 0; krok = 1;
While[i < 1,
  x = {Sx[[1]] + 0.5*Δ, Sx[[1]] + 0.5*Δ, Sx[[1]] - 0.5*Δ, Sx[[1]] - 0.5*Δ};
  y = {Sy[[1]] + 0.5*Δ, Sy[[1]] - 0.5*Δ, Sy[[1]] + 0.5*Δ, Sy[[1]] - 0.5*Δ};
  y0 = f[Sx[[1]], Sy[[1]]];
  y1 = f[x[[1]], y[[1]]];
  y2 = f[x[[2]], y[[2]]];
  y3 = f[x[[3]], y[[3]]];
  y4 = f[x[[4]], y[[4]]];
  fce = {y1, y2, y3, y4};
  M = Position[fce, If[extrem == "Max", Max[fce], Min[fce]]];
  MM = M[[1, 1]];
  If[If[extrem == "Max", Max[fce] > y0, Min[fce] < y0], Sx = {x[[MM]]};
  Sy = {y[[MM]]};, i = 1]; Print[krok, ". krok řešení: x = ", Sx[[1]],
  " y = ", Sy[[1]], " Hodnota: ", y0];
  Show[Plot3D[f[x, y], {x, a1, a2}, {y, b1, b2}, AxesLabel -> {"x", "y", "z"},
  PlotPoints -> 30, DisplayFunction -> Identity,
  Graphics3D[{RGBColor[1, 0, 0], PointSize[0.05],
  Point[{Sx[[1]], Sy[[1]], y0 + bod}]]], Axes -> Automatic,
  AxesLabel -> {"x", "y", "z"},
  PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
  Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction]; krok++;
Clear[f];
```

Graf funkce:

25. krok Řešení: $x = 4$. $y = 6$. Hodnota: 68.



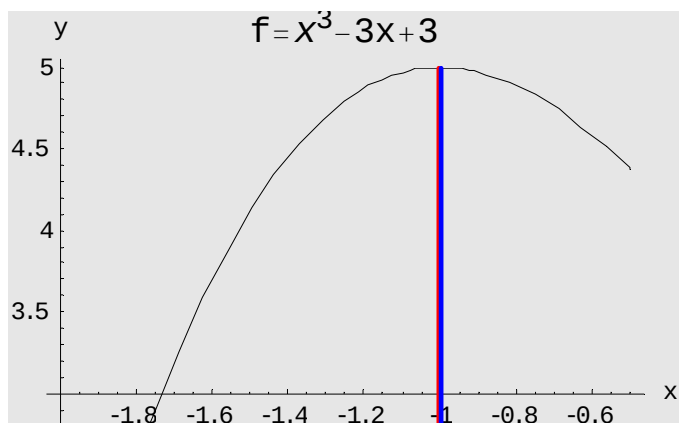
P V : METODA ZLATÉHO ŘEZU

Zdrojový kód programu Mathematica:

```
f[x_] := x3 - 3x + 3; aa = -2; bb = -0.5; n = 11; name = "f=x3-3x+3";
extrem = "Max";
(*f[x_] := -(x3 - 3x + 3); aa = -2; bb = -0.5; n = 11; name = "f=-(x3-3x+3)";
extrem = "Min";*)
zl = 0.618; a = aa; b = bb;
For[i = 1, i < n + 1, A = b - zl * Abs[b - a];
  B = a + zl * Abs[b - a]; FA = f[A]; FB = f[B];
  If[If[extrem == "Max", FA > FB, FA < FB], a = a; b = B, a = A; b = b];
  Print[i, ". krok   Interval  <", a // N, " ; ", b // N, ">"];
  Show[Plot[f[x], {x, aa, bb}, DisplayFunction -> Identity],
    Graphics[{RGBColor[1, 0, 0], Thickness[0.01], Line[{a, 0}, {a, f[a]}]}],
    Graphics[{RGBColor[0, 0, 1], Thickness[0.01], Line[{b, 0}, {b, f[b]}]}],
    Axes -> Automatic, AxesLabel -> {"x", "y"},
    PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
    Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction]; i++];
Clear[f];
```

Graf funkce:

11. krok Interval < -1.00441 ; -0.996879 >



P VI: METODA PRAVIDELNÉHO SIMPLEXU

Zdrojový kód programu Mathematica:

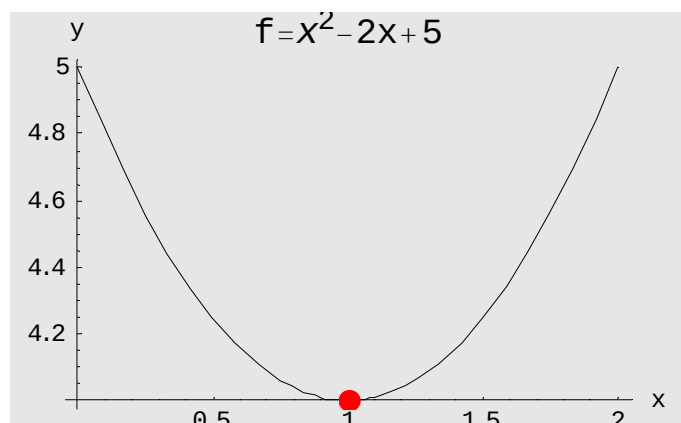
```
f[x_] := x2 - 2 x + 5; a1 = 0; b1 = 2; ε = 0.01; ρ = 0.005; name = "f=x2-2x+5";
extrem = "Max";
(*f[x_] := -(x2-2 x+5); a1=0; b1=2; ε=0.01; ρ=0.005; name="f=-(x2-2x+5)";
extrem="Min";*)

n = 2; d = ρ *  $\frac{\sqrt{n+1}-1}{\sqrt[n]{2}}$ ; e = d +  $\frac{\rho}{2}$ ;

a = 0; krok = 1; i = 0; aa = a;
While[i < 1, x1 = a; x2 = a + d + e; x3 = a + d; f1 = f[x1]; f2 = f[x2]; f3 = f[x3];
fce = {f1, f2, f3};
mm = Position[fce, If[extrem == "Max", Max[fce], Min[fce]]];
m = mm[[1, 1]];
c = (x1 + x2 + x3 - xm) / n;
a = 2 c - xm;
If[Abs[aa - a] > ε, aa = a; Print[krok, ". krok řešení: x= ", a // N,
" Hodnota: ", f[a] // N];
Show[Plot[f[x], {x, a1, b1}, DisplayFunction → Identity],
Graphics[{RGBColor[1, 0, 0], PointSize[0.04], Point[{a, f[a]}]}],
Axes → Automatic, AxesLabel → {"x", "y"},
PlotLabel → StyleForm[TraditionalForm[name], FontSize → 15],
Background → GrayLevel[0.9], DisplayFunction → $DisplayFunction];
krok++, i = 1]];
Clear[f, fce];
```

Graf funkce:

98. krok Řešení: x= 1.00593 Hodnota: 4.00004



P VII : METODA CYKlickÉ ZÁMĚNY PARAMETRŮ

Zdrojový kód programu Mathematica:

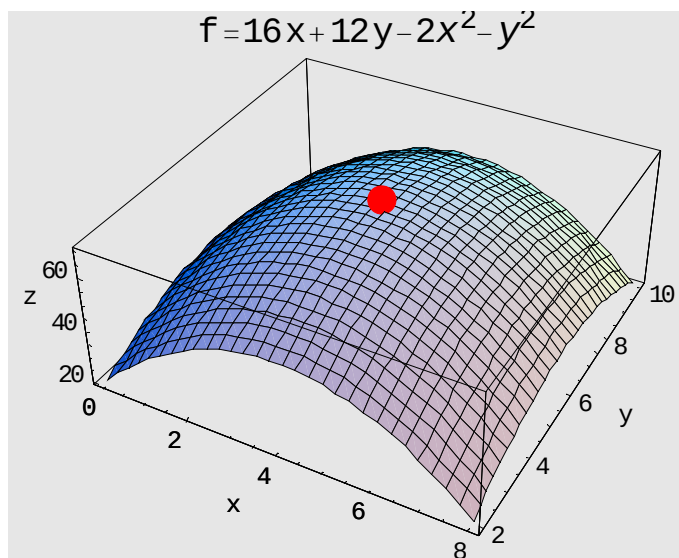
```
f[x_] := x2 - 2 x + 5; a1 = 0; b1 = 2; ε = 0.01; ρ = 0.005; name = "f=x2-2x+5";
extrem = "Max";
(*f[x_] := -(x2-2 x+5); a1=0; b1=2; ε=0.01; ρ=0.005;
name="f=-(x2-2x+5)"; extrem="Min"; *)

n = 2; d = ρ *  $\frac{\sqrt{n+1} - 1}{\sqrt[n]{2}}$ ; e = d +  $\frac{\rho}{2}$ ;

a = 0; krok = 1; i = 0; aa = a;
While[i < 1, x1 = a; x2 = a + d + e; x3 = a + d; f1 = f[x1]; f2 = f[x2];
f3 = f[x3];
fce = {f1, f2, f3};
mm = Position[fce, If[extrem == "Max", Max[fce], Min[fce]]];
m = mm[[1, 1]];
c = (x1 + x2 + x3 - xm) / n;
a = 2 c - xm;
If[Abs[aa - a] > ε, aa = a;
Print[krok, ". krok řešení: x= ", a // N, " Hodnota: ", f[a] // N];
Show[Plot[f[x], {x, a1, b1}, DisplayFunction -> Identity],
Graphics[{RGBColor[1, 0, 0], PointSize[0.04], Point[{a, f[a]}]}],
Axes -> Automatic, AxesLabel -> {"x", "y"},
PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction];
krok++, i = 1]];
Clear[f, fce];
```

Graf funkce:

1 . krok Řešení: $x= 4$ $y= 6$ Hodnota: 68



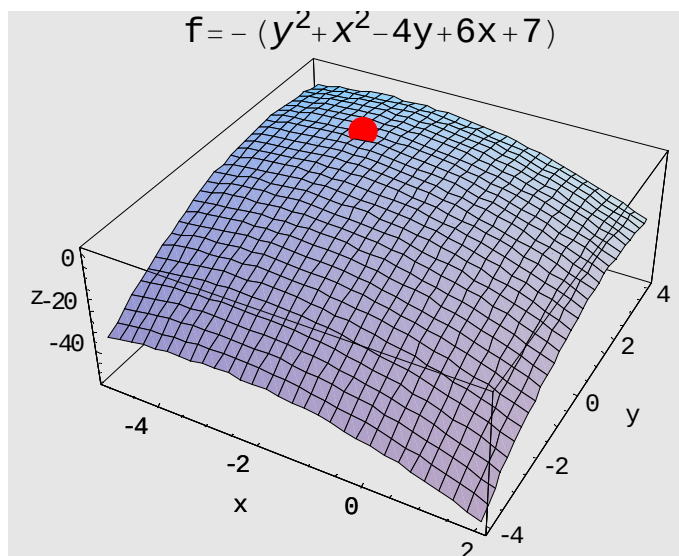
P VIII : NEWTONOVA METODA 3D

Zdrojový kód programu Mathematica:

```
f = -(y2 + x2 - 4 y + 6 x + 7); a1 = -5; a2 = 2; b1 = -4; b2 = 4; ε = 0.01;
name = "f=-(y2+x2-4y+6x+7)"; bod = 1;
(*f=y2+x2-4y+6x+7;a1=-5;a2=2;b1=-4;b2=4;ε=0.01;
name="f=y2+x2-4y+6x+7";bod=1;*)
xx = ∂xf; yy = ∂yf;
xyx = ∂x,y f; yyy = ∂y,x f;
q = ∂x,x f; r = ∂y,y f;
x = 0; y = 0; krok = 1; c = f; i = 0;
H = {{q, xyx}, {yyy, r}};
H1 = Inverse[H];
H2 = {{xx}, {yy}};
p = {{x}, {y}};
While[i < 1, xk+1 = p - H1.H2;
x = xk+1[[1, 1]];
y = xk+1[[2, 1]];
x2 = p - H1.H2;
xk+1 = x2;
If[Abs[c - f] > ε, c = f;
Print[krok, ". krok řešení: x= ", x // N, " y= ", y // N,
" Hodnota: ", f // N];
Show[Plot3D[f, {x, a1, a2}, {y, b1, b2}, AxesLabel -> {"x", "y", "z"},
PlotPoints -> 30, DisplayFunction -> Identity],
Graphics3D[{RGBColor[1, 0, 0], PointSize[0.05], Point[{x, y, f + bod}]}],
Axes -> Automatic, AxesLabel -> {"x", "y", "z"},
PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction];
krok++, i = 1]]; Clear[f, x, x2, y, xx, yy, xyx, yyy, H, H1, H2,
p, q, r];
```

Graf funkce:

1 . krok Řešení: $x = -3$. $y = 2$. Hodnota: 6.



P IX: GRADIENTNÍ METODA S KRÁTKÝM KROKEM

Zdrojový kód programu Mathematica:

```
f = 16 x + 12 y - 2 x2 - y2 + 1; a1 = 0; a2 = 8; b1 = 2; b2 = 10; λ = 0.3; ε = 0.0001;
name = "f=16x+12y-2x2-y2+1"; bod = 1;
(*f=- (16 x+12 y-2 x2-y2+1) ;a1=0;a2=8;b1=2;b2=10; λ=-0.3;
ε=0.0001;name="f=- (16x+12y-2x2-y2+1) ";bod=1;*)
xx = ∂xf; yy = ∂yf;
x = 0; y = 0; krok = 1; c = f; i = 0;
While[i < 1, bb = {{x}, {y}} + λ * {{xx}, {yy}};
  x = bb[[1, 1]]; y = bb[[2, 1]];
  x2 = {{x}, {y}} + λ * {{xx}, {yy}}; bb = x2;
  If[Abs[c - f] > ε, c = f; Print[krok, ". krok   řešení: x= ", x,
    "   y= ", y, "   Hodnota: ", f];
  Show[Plot3D[f, {x, a1, a2}, {y, b1, b2}, AxesLabel -> {"x", "y", "z"},
    PlotPoints -> 30, DisplayFunction -> Identity],
    Graphics3D[{RGBColor[1, 0, 0], PointSize[0.05], Point[{x, y, f + bod}]}],
    Axes -> Automatic, AxesLabel -> {"x", "y", "z"},
    PlotLabel -> StyleForm[TraditionalForm[name], FontSize -> 15],
    Background -> GrayLevel[0.9], DisplayFunction -> $DisplayFunction];
  krok++, i = 1]];
Clear[f, x, y, xx, yy];
```

Graf funkce:

7. krok Řešení: x= 4.00005 y= 5.99017 Hodnota: 68.9999

