# ITERAČNÍ METODY OPTIMALIZACE

Bc. Martin Kočica

Diplomová práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

***nascannované zadání s. 1***

***nascannované zadání s. 2***

## ABSTRAKT

Abstrakt česky

Práce je příspěvkem k pedagogickému působení v rámci předmětu Optimalizace. Oblastí hlavní činnosti jsou iterační metody pro vyhledávání extrémů reálných funkcí jedné nebo více reálných proměnných bez omezení na definiční obor. Cílem práce bylo vytvořit algoritmy optimalizačních iteračních metod v prostředí Mathematica v obecném tvaru pro širokou třídu účelových funkcí i dimenze úlohy. Důraz je kladen na iterační metody komparativní a gradientní. Dále byly vypracovány vzorové protokoly a www stránky předmětu, které budou sloužit zejména pro kombinovanou formu studiu.



Klíčová slova: Extrém funkce, optimalizace, účelová funkce, iterace, simplex, gradient



## ABSTRACT

The contribution of the work is in the field of optimization technique methods. The main aim of the work is to create and carry out of iteration procedures for real functions of many real variables. The program realization was performed in the Mathematica environment in a general form with respect to goal function as well as to the dimension of the problem. Emphasize is laid on comparative and gradient iteration methods. Further, a set of sample default students projects for standard optimization tasks. Also www sites of special parts of the subject were created for distance study.



Keywords: Function extremes, optimization, goal function, iteration, simplex, gradient

Poděkování panu prof. Ing. Romanu Prokopovi, Csc. za cenné rady a připomínky.

V průběhu realizace své diplomové práce jsem napočítal do nekonečna. Dvakrát

Ve Zlíně ……..……..

25.května.2006 Martin Kočica

# OBSAH

# ÚVOD

Proces optimalizace znamená snahu nalezení nejvýhodnějšího vyvážení hodnot do systému vstupujících a výstupu jež je těmito hodnotami ovlivněn. Úlohy hledání extrémů jsou mnohdy velice komplikované, nicméně uspět při řešení často přináší zisk adekvátní vynaloženému úsilí. Proto se na počátku minulého století staly extremalizační úlohy hluboce studovanou matematickou disciplínou. V praktických úlohách se za čísly, jež reprezentují řešený problém pro její matematické zpracování, velice často skrývá zajímavé zadání. Snadno si lze z běžného života představit požadavek na nalezení takového způsobu vytápění bytu, jež povede k nejnižším nákladů. Obdobně lze extremalizační úlohou nazvat problém vyhledání vhodných surovin a jejich kombinací pro výrobu s nejvyšším ziskem.

Ačkoli teoreticky je samozřejmě možné definovat jakkoli obtížný problém, praxe je v tomto ohledu významným konkurentem. Z tohoto důvodu je víceméně utopií vymyslet jeden univerzální postup který by neomylně vedl k nejvýhodnějšímu řešení. Dochází tak zejména k vytváření mnoha různých postupů pro jednotlivé typy problémů a také k množství klasifikací samotných úloh. Velice důležitým rozlišovacím znakem extremalizační úlohy jsou její omezující podmínky. V případě, že na hodnoty systému nejsou kladeny žádné speciální nároky, hovoří se o úlohách bez omezení na definiční obor. Mnoho případů ovšem není možné řešit zcela bez dalších požadavků. V takovýchto případech lze rozlišit omezení typu rovnost a omezení typu nerovnost. Například právě z těchto odlišností vyplývá potřeba různého matematického aparátu. Jedním z možných postupů jsou analytické metody. Tyto metody se nejvíce blíží klasické matematické analýze, využívají prostředků diferenciálního počtu a jsou vhodné zejména pro individuální ruční výpočet. Jiným nástrojem jsou metody iterační. Jejich charakteristikou je konstrukce posloupnosti bodů definičního oboru konvergující k optimálnímu řešení. Jejich nasazení pro vyhledání extrému je velice výhodné v kombinaci s výpočetní technikou.

Práce samotná je příspěvkem k předmětu Optimalizace, který se problematikou vyhledávání extrémů zabývá. Zaměřuje se především na iterační metody optimalizace, ale nevyhýbá se ani nástinu globálnějšího pohledu na optimalizační úlohy a na metody jejich řešení. V neposlední řadě je produktem této práce čtveřice konkrétních implementací vybraných iteračních postupů extremalizace. Cílem vytvoření www prezentace je napomoci zájemcům při studiu této matematické disciplíny.

# I. TEORETICKÁ ČÁST

# 1 OPTIMALIZAČNÍ METODY

Optimalizace je proces vyhledávání nejvýhodnějšího, nejkvalitnějšího eventuálně nejvhodnějšího řešení. V praxi se s požadavkem nalezení optima setkáváme velice často. Vyhledávání extrémů je skryto v nastavování parametrů výroby, kdy se logicky snažíme skloubit minimální počet zdrojů dávající maximální zisk. S problémem optimalizace se setkáme při cestě na dovolenou, kdy se snažíme přepravit z místa na místo po co nejkratší trase, eventuálně s co nejnižšími finančními nebo časovými náklady. Podobně jsme problematikou provázeni při nákupech, kde je našim cílem co nejefektivněji vyvážit nakupované zboží a jeho cenu. V této souvislosti se velice často setkáváme s výroky typu „poměr cena – výkon“ eventuálně „cena – kvalita“ kterými obchodníci lákají na své zboží. A právě v těchto výrocích je také skryto ono vyhledávání nejvhodnějšího. Ačkoli je toto uvedení problematiky poněkud vágní, nepochybně dostačuje pro zevrubnou představu náplně disciplíny zvané optimalizace. Intuitivně si pak snadno představíme svůj konkrétní problém, který by odpovídal předchozím schématům.

Z jiného pohledu lze o optimalizaci také říci, že jde o matematickou disciplínu, ve které hledáme minimum respektive maximum dané funkce $f(x)$ a dané množině M. 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. Jejím významem je zamezit prohledávání prostoru, pro nějž nalezené řešení nebude akceptovatelné. [6]

V obecném optimalizačním procesu dochází ke změně stavů proměnných optimalizovaného objektu a ke sledování projevu těchto změn. Snahou je pak docílit takové konfigurace proměnných, že systém bude vykazovat požadované optimální vlastnosti. Je důležité si uvědomit, že v mnoha případech nelze naprosto přesně docílit minimalizace či maximalizace daných vlastností. V takovém případě je logicky snahou se k této ideální metě alespoň co nejvíce přiblížit. Odchylka aktuálních parametrů systému od parametrů žádaných je popisována právě účelovou funkcí. [5]

Optimalizační metody jsou velice silným nástrojem pro zkvalitnění či zefektivnění mnoha případů lidské činnosti. Je tedy nezbytné věnovat jim dostatečnou pozornost.

## 1.1 Význam základních pojmů a značení

Mnohé pojmy se kterými se setkáme v souvislosti z problematikou optimalizace byly již zmíněny v předchozí části. Je proto nejvyšší čas nějakým způsobem tento přehled pojmů ucelit a vysvětlit jejich význam.

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 se značením $f(x_1,..,x_n)$. V optimalizačních problémech se lze mimo funkce reálných proměnných často setkat i s jiným tipem proměnných. Jde například o proměnné diskrétní - celočíselné, omezené na interval a podobně. Je spíše filozofickou otázkou, zda reálné proměnné jsou obecnějším případem či je tomu naopak. Lze totiž namítnout, že pro proměnné definované na jiné množině než množině reálných čísel je za potřebí určitého rozšíření či doplnění podmínek a algoritmů pro hledání optimálních hodnot účelové funkce. Naopak celočíselné proměnné jsou jen podmnožinou proměnných reálných. Diskuse na toto téma není náplní textu, přesto je dobré si tuto situaci uvědomit. Podstatným závěrem by však mělo být, že vhodným výběrem a patřičnou úpravou postupů a algoritmů lze hledat a nacházet extrémy i na jiném definičním oboru než je obor reálných čísel.

Značení proměnných funkce odpovídá zvyklostem značení pro funkce většího počtu proměnných $x_1, x_2, x_3,.., x_n$. Vzhledem k praktickému využití algoritmů je velice podstatná skutečnost, že je možné řešit úlohy neomezeného počtu proměnných. Neomezenou optimalizací nazýváme problém v případě, že proměnné stavového prostoru se mohou měnit bez jakýchkoli omezení. V mnoha případech je ovšem nutné nějaká omezení připustit. Pro představu je takovým omezením například nemožnost výroby záporného počtu výrobků. Jestliže jsou proměnné nějakým způsobem limitovány jedná se o optimalizaci s omezujícími podmínkami. [5] V souvislosti s proměnnými se nabízí pojem již jednou zmíněný, rozměr funkce. Dimenzí, nebo též rozměrem, úlohy se rozumí číslo odpovídající počtu proměnných účelové funkce.

Pojem optimalizace již byl vysvětlen dříve, nicméně pro pořádek jím rozumíme postup pro hledání extrému funkce. U reálných funkcí rozlišujeme dva typy extrémů. Jde o minimum a maximum. Přičemž každý z nich může mít povahu lokálního nebo globálního. Přesná definice říká že funkce $f : X \to R$ má v bodě $x \in X$

1. Lokální minimum – na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $y \in X, \|y - x\| < \delta$ platí $f(x) \leq f(y)$
2. Ostré lokální minimum – na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $y \in X, 0 < \|y - x\| < \delta$ platí $f(x) < f(y)$
3. Globální minimum – na $X \subset R^n$, jestliže pro každé $y \in X$ platí $f(x) \leq f(y)$
4. Ostré globální minimum – na $X \subset R^n$, jestliže pro každé $y \in X, y \neq x$ platí $f(x) < f(y)$
5. Lokální maximum – na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $y \in X, \|y - x\| < \delta$ platí $f(x) \geq f(y)$
6. Ostré lokální maximum – na $X \subset R^n$, jestliže existuje $\delta > 0$ tak, že pro každé $y \in X, 0 < \|y - x\| < \delta$ platí $f(x) > f(y)$
7. Globální maximum – na $X \subset R^n$, jestliže pro každé $y \in X$ platí $f(x) \geq f(y)$
8. Ostré globální maximum – na $X \subset R^n$, jestliže pro každé $y \in X, y \neq x$ platí $f(x) > f(y)$ [7][8]

Pro většinu algoritmů hledajících extrémy funkce je nutné stanovit zásadní omezující podmínku. Tato podmínka říká, že optimalizovaná funkce má pouze jeden extrém. Poměrně striktní omezení na definičním oboru lze poněkud zmírnit, pokud omezení aplikujeme pouze na interval ve kterém bude vyhledávání prováděno. Omezení předchází chybám uváznutí v lokálním extrému, neboť v drtivé většině případů není možné rozlišit, zda nalezený extrém je pouze lokální, nebo se jedná o globální extrém funkce. Jestliže se stane, že o účelové funkci není dostatek informací a vyhledávání minima či maxima začne z nevhodného bodu, nebude možné rozhodnout jak kvalitního extrému bude dosaženo.

## 1.2 Klasifikace úloh optimalizace

Optimalizační úlohy lze klasifikovat podle druhu omezení kladeného na vstupní podmínky. Rozlišuje se trojí případ omezujících podmínek:

1. Bez omezení
2. Omezení typu rovnost: $c_i(x_1,..,x_n)=0$, kde $i \in \overline{N}^+$, $i < n$
3. Omezení typu nerovnost: $c_i(x_1,..,x_n) \geq 0$, kde $i \in \overline{N}^+$, $i < n$ a $x_j \geq 0$, $j \in \langle 1,n \rangle$

$x_1,..,x_n$ jsou proměnné systému a funkce $c$ vyjadřuje omezující podmínku. Úlohu lze za v takovém případě formulovat jako požadavek nalezení extrému funkce $f(x_1,..,x_n)$ za podmínek $c_i(x_1,..,x_n)=0$

Dle těchto kritérií vypadá klasifikace úloh následovně

1. Volný extrém – bez omezení vstupních podmínek
2. Klasický vázaný extrém – omezení typu rovnost
3. Neklasický vázaný extrém – omezení typu nerovnost

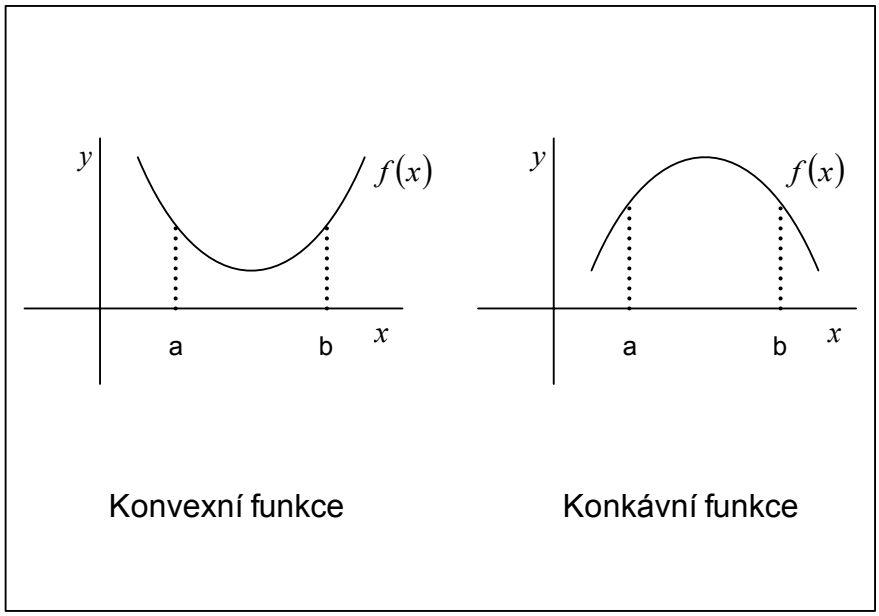
### 1.2.1 Volný extrém

#### *1.2.1.1 Jednorozměrný případ*

V případě, že účelová funkce jedné proměnné je zcela obecná lze její extrém například nalézt s využitím metod založených na Fibonacciho číslech viz. kapitola 2.1.1. V případech, že je funkce na prohledávaném intervalu hladká a spojitá lze využít známých matematických postupů. Analýza funkcí vychází z jejích základních vlastností jež je vhodné připomenout.

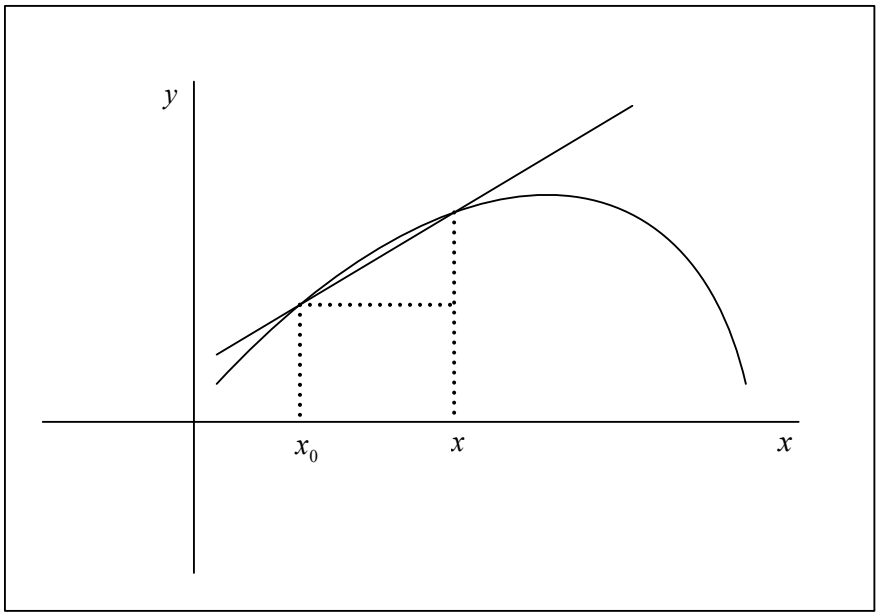
Funkce $f(x)$ je na intervalu $\langle a,b \rangle$ konvexní, Jestliže pro všechna x patřících do tohoto intervalu je druhá derivace funkce ostře větší než 0. Matematicky lze tuto vlastnost vyjádřit vzorcem: $\forall x \in \langle a,b \rangle, f''(x) > 0$. Obdobnou definici s opačným znaménkem je

vymezena funkce konkávní. Jestliže $\forall x \in \langle a,b \rangle, f''(x) < 0$ na intervalu $\langle a,b \rangle$, pak je funkce konkávní.



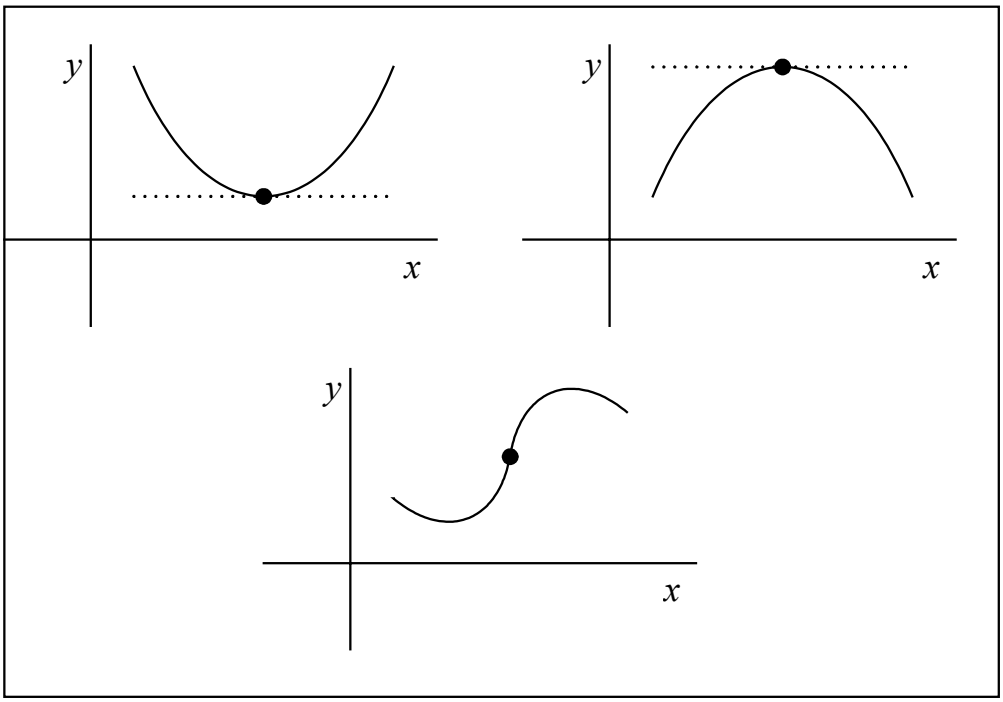

Obrázek 1 Konvexní a konkávní funkce

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.




Obrázek 2 Derivace funkce v bodě

Stacionární bod funkce $f(x)$ je bod $x_0$ pro který platí, že jeho první derivace je rovna nule $f'(x_0) = 0$. Ve stacionárním bodě se nachází minimum, maximu, nebo inflexní bod. Inflexním bodem je takový, ve kterém funkce $f(x)$ přechází z konvexní do konkávní nebo opačně.




Obrázek 3 Stacionární body

Pro určení zda a případně jakým je bod extrémem účelové funkce na intervalu $\langle a,b\rangle$ je za potřebí vyhodnotit tyto podmínky

1. Jestliže má funkce $f(x)$ na intervalu $\langle a,b\rangle$ první a druhou derivaci v bodě $x_0$, pak platí
   a. je-li $f'(x_0)=0$ a $f''(x_0)<0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální maximum
   b. je-li $f'(x_0)=0$ a $f''(x_0)>0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální minimum
   c. jsou-li obě derivace rovny nule platí podmínka číslo 2

2. Nechť existuje přirozené číslo i, pro které $f^{(i)} \neq 0$ a pro všechna $j < i$ je $f^{(j)}(x_0) = 0$, pak platí
   a. je-li i sudé a $f^{(i)} < 0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální maximum
   b. je-li i sudé a $f^{(i)} > 0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální minimum
   c. je-li i liché, pak má funkce $f(x)$ v bodě $x_0$ inflexní bod

3. Může se stát, že druhá derivace na intervalu $\langle a,b \rangle$ funkce $f(x)$ nebude existovat, pak platí
   a. je-li $f'(x) < 0$ v levém okolí a $f'(x) > 0$ v pravém okolí bodu $x_0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální minimum
   b. je-li $f'(x) > 0$ v levém okolí a $f'(x) < 0$ v pravém okolí bodu $x_0$, pak má funkce $f(x)$ v bodě $x_0$ ostré lokální maximum [9]

Vyhodnocení těchto podmínek zaručuje identifikaci stacionárního bodu na minimum, maximum nebo inflexní bod, nezaručuje ale nalezení všech extrémů. Nepokrývá totiž variantu, kdy funkce $f(x)$ nemá v daném bodě první derivaci.

#### *1.2.1.2 Vícerozměrný případ*

V obecném případě za předpokladu reálné funkce reálných proměnných $f(x_1,..,x_n)$, kde $n \in \overline{N}^+$ je první derivace zobecněná do podoby gradientu a zobecnění druhé derivace je Hessovou maticí.

1. První derivace: $grad(f(x_1,..,x_n)) = \nabla f(x_1,..,x_n) = \left( \frac{\partial f}{\partial x_1},..,\frac{\partial f}{\partial x_n} \right)$

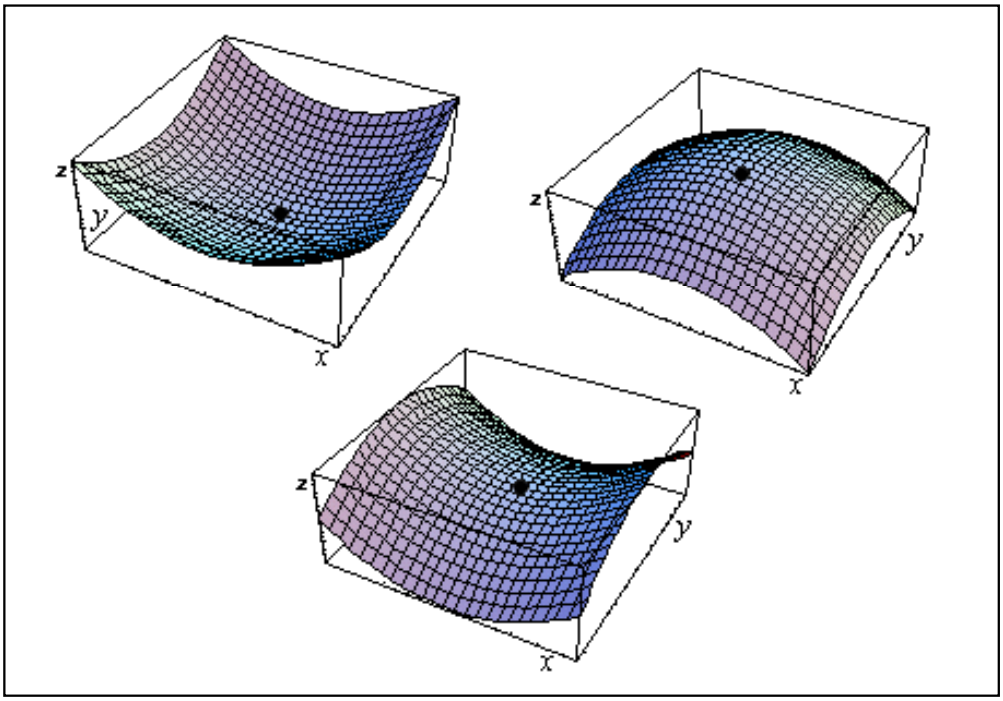
2. Druhá derivace: $H = \nabla^2 f(x_1,..,x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & .. & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ .. & & & .. \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & .. & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$

3. Zobecněním inflexního bodu je bod sedlový

Stacionárním bodem je takový, pro který platí $\nabla^2 f(x_1,..,x_n) = 0$. Lokálním maximem je bod $x_0$ tehdy, jestliže $\nabla f(x_0) = 0$ a $H(x_0)$ negativně definitní. Je-li $\nabla f(x_0) = 0$ a $H(x_0)$ je pozitivně definitní, pak je bod $x_0$ lokálním minimem.

- Pozitivně definitní je čtvercová matice tehdy, jestliže všechny její hlavní subdeterminanty jsou kladné

- Negativně definitní je čtvercová matice tehdy, jestliže její hlavní subdeterminanty střídají znaménka, ale první je záporné.




Obrázek 4 Stacionární body II

### 1.2.2 Klasický vázaný extrém

Pro jednorozměrný případ omezení typu rovnost nemá valný smysl, neboť jestliže úloze o jednom stupni volnosti definujeme bod rovností, stává se tento bod automaticky řešením celého problému.

#### *1.2.2.1 Vícerozměrný případ*

Klasické omezení rovnosti pro vícerozměrnou úlohu optimalizace je definováno soustavou rovnic, jejichž počet je menší než počet proměnných. Zajímavostí hledaného extrému za těchto podmínek je skutečnost, že úloha má obvykle řešení a to i v případě, kdy volný extrém neexistuje.

K řešení problému lze dospět v zásadě dvěma způsoby. Jedním z nich je dosazování z omezení. Druhým způsobem je metoda Lagrangeových multiplikátorů, ta vychází ze zavedení funkce $\phi(x,\lambda) = f(\underline{x}) + \sum_{j=1}^{i} \lambda_j c_j(\underline{x})$, kde $i < n$ a $c_i(x_1,..,x_n) = 0$ jsou omezující podmínky. Nechť funkce $f, c_1,..,c_i$ mají spojité první parciální derivace nechť jsou funkce $\nabla c_j(x)$ lineárně nezávislé platí: je-li $x_0$ extrém funkce $f(x_1,..,x_n)$ při omezeních $c_j(x) = 0$, pak existuje $\lambda_0(\lambda_{01},..,\lambda_{0i})$ tak, že platí $\nabla_x \phi(x_0,\lambda_0) = \nabla_\lambda \phi(x_0,\lambda_0) = 0$. Určit o jaký extrém se konkrétně jedná je nejsnazší pomocným dosazení bodu z okolí.

### 1.2.3 Neklasický vázaný extrém

Analytické řešení úlohy s omezujícími podmínkami ve tvaru nerovnosti opět vychází z Lagrangeovy funkce jako v předchozím případě s přídavkem věty o sedlovém bodě, Kun-Tuckerova věta. Jestliže existuje $x_0 \geq 0$ a $\lambda_0 \geq 0$ tak, že pro $\forall x \geq 0, \forall \lambda \geq 0$ platí $\phi(x,\lambda_0) \leq \phi(x_0,\lambda_0) \leq \phi(x_0,\lambda)$ potom $x_0$ je optimálním řešením úlohy. Je nutné poznamenat, že optimálním řešením se zde myslí maximum. Je-li potřeba nalézt minimum, nutno vynásobit účelovou funkci -1. [9]

## 1.3 Klasifikace metod optimalizace

Klasifikaci optimalizačních metod lze provést podle různých kritérií podobně, jako samotné úlohy. Zvolené rozdělení vychází ze způsobu výpočtu extrému

1. Analytické metody
   a. derivace v případě jednorozměrných úloh
   b. gradienty u problémů více proměnných
      - omezení typu rovnost
      - omezení typu nerovnost
2. Iterační metody
   a. Komparativní
      - jednorozměrné
      - mnohorozměrné
   b. Gradientní
      - bez omezení
      - s omezením
   c. Metody náhodného vyhledávání
3. Speciální metody
   a. lineární programování
   b. dynamické programování
   c. konvexní programování

## 1.4 Podmínky optimality

Podmínky optimality jsou podmínky, které platí pro optimální řešení a slouží k redukci množiny přípustných řešení. V mnoha případech jsou přímo nápomocny při řešení úlohy, například v případě analytického zkoumání. Při formulaci podmínek lze rozlišit jejich tři základní typy.

1. Nutné podmínky optimality – má-li funkce $f(x_1,..,x_n)$ v bodě $x^*$ extrém, pak v tomto bodě musí být splněny nutné podmínky existence extrému. Obráceně také

platí, pokud v bodě $x^*$ nejsou nutné podmínky optimality splněny, pak tento bod není extrémem.

2. Postačující podmínky optimality – jsou-li v bodě $x^*$ splněny postačující podmínky optimality, pak funkce $f(x_1,..,x_n)$ má v tomto bodě extrém.

3. Nutné a postačující podmínky optimality – budou-li v bodě $x^*$ splněny nutné a postačující podmínky, pak funkce $f(x_1,..,x_n)$ má v tomto bodě extrém. Lze také říci, je-li bod $x^*$ extrém funkce $f(x_1,..,x_n)$, pak splňuje nutné a postačující podmínky. [3]

# 2 ITERAČNÍ METODY

Pojem iterace znamená opakování, nebo též opětné užití daného úkonu na výsledek úkonu předchozího. [1] Iterační metoda je tedy založena na opakovaném provádění konkrétních operací či postupů. Postupně se tak vytvoří posloupnost dílčích výsledků blížících se k hledanému řešení. [2]. Matematicky lze tento postup zapsat následovně: $x_{n+1} = x_n + \Delta_n$, kde $x_n$ značí současný stav, $x_{n+1}$ stav následující , $\Delta_n$ představuje změnu ke které dojde po proběhnutí jednoho cyklu postupu, $n \in \overline{N}$ přičemž $x_0$ představuje počáteční, nebo též startovací stav. Podstatnou vlastností vznikající posloupnosti mezivýsledků $\{x_n\}$ je skutečnost, že pro rostoucí n se hodnoty $x_n$ blíží k požadovanému optimu. V limitě to tedy znamená, že iterační postup po n krocích kde n roste k nekonečnu dospěje k výslednému řešení. Tuto skutečnost lze formulovat takto: $\lim_{n\to\infty} x_n = x_{řeš}$ .

Pro optimalizační úlohy jsou obecné pojmy konkretizovány následovně. $x_{řeš}$ odpovídá řešení v podobě extrému, tedy maxima či minima. Startovacím se rozumí jakýkoli bod, pro který existuje ohodnocení optimalizační funkce. Mezivýsledky jsou opět body, pro než existuje ohodnocení optimalizační funkce, ale zároveň platí, $x_0 < x_1 < .. < x_n$ v případě, že hledáme maximum. U minima jsou znaménka nerovnosti obráceně.

Iterační metody lze klasifikovat do tří základních skupin.

1. Komparativní metody
    a. jednorozměrné
    b. mnohorozměrné
2. Gradientní metody
    a. jednorozměrné
    b. mnohorozměrné bez omezení
    c. s omezením

3. Metody náhodného prohledávání
    a. jednoduché
    b. adaptivní

## 2.1 Komparativní metody

Charakteristikou komparativních metod optimalizace je vyčíslování hodnot mezivýsledků potřebných pro výpočet jednotlivých kroků algoritmu na základě přímého dosazení do účelové funkce bez nutnosti vypočíst či odhadnout její derivaci.

Další dělení komparativních metod již bylo naznačeno. Jedná se o postupy pro jednorozměrné a mnohorozměrné problémy. Mezi postupy určené na řešení jednodimenzionálních problémů patří: Fibonacciho metoda, metoda zlatého řezu, Powelova metoda, rovnoměrná komparativní metoda a další.

### 2.1.1 Jednorozměrné komparativní metody

#### *2.1.1.1 Fibonacciho metoda*

Základem této metody je Fibonacciho posloupnost. Tato číselná posloupnost je vyjádřena rovnicí $F_{i+2} = F_{i+1} + F_i$ a počátečními hodnotami $F_1 = F_2 = 1 \text{pro}\ \ i \in \overline{N}^+$. Prvních několik členů posloupnosti tedy vypadá následovně: $F = \{1,1,2,3,5,8,13,21,34,...\}$. Samotná metoda hledání extrému předpokládá, že na daném intervalu $\langle a,b \rangle$ existuje jediný extrém. V principu potom, algoritmus určuje body ve kterých se má vypočíst hodnota účelové funkce a v definovaném počtu kroků zmenšuje interval $\langle a,b \rangle$. Nevýhodou metody je právě počáteční volba počtu kroků N. Toto omezení se projeví v okamžiku, kdy výsledek který dostaneme neodpovídá požadavkům na něj kladený. Za takové situace nezbývá než zvýšit počet kroků a rozběhnout algoritmus od začátku, neboť na proběhnutý výpočet nelze navázat. Alternativou k tomuto kroku je spuštění algoritmu s počátečními hodnotami jež dostaneme právě ukončeným výpočtem. Posledním důležitým krokem je volba znaménka nerovnosti. V algoritmu zapsaném níže, konkrétně v bodech 6 a 7 implikují znaménka hledání lokálního maxima. V případě opačného požadavku je třeba tato znaménka nerovnosti převrátit.

Algoritmus:

1. Volba počtu kroků N
2. Volba hranic intervalu a,b
3. $i = 1, \alpha_1 = a,\ \beta_i = b$
4. $\overline{\alpha}_{i+1} = \beta_i - \frac{F_{N-i+1}}{F_{N-i+2}} \cdot \left|\beta_i - \alpha_i\right|$
5. $\overline{\beta}_{i+1} = \alpha_i + \frac{F_{N-i+1}}{F_{N-i+2}} \left|\beta_i - \alpha_i\right|$
6. Jestliže $f\left(\overline{\alpha}_{i+1}\right) < f\left(\overline{\beta}_{i+1}\right)$ pak $\alpha_{i+1} = \overline{\alpha}_{i+1}$ a $\beta_{i+1} = \beta_i$
7. Jestliže $f\left(\overline{\alpha}_{i+1}\right) \geq f\left(\overline{\beta}_{i+1}\right)$ pak $\alpha_{i+1} = \alpha_i$ a $\beta_{i+1} = \overline{\beta}_{i+1}$
8. Je-li $i = N$ pak konec
9. $i = i + 1$ skok na d

### *2.1.1.2 Metoda zlatého řezu*

Zlatým řezem je nazývána hodnota limity podílu dvou sousedních členů Fibonacciho posloupnosti. $\lim_{i\to\infty} \frac{F_i}{F_{i+1}} \cong 0{,}618$. Samotný postup výpočtu je velice podobný postupu z předcházející metody viz. Fibonacciho metoda. Liší se v bodech d a e, kdy je místo poměru $\frac{F_{N-i+1}}{F_{N-i+2}}$ volena právě hodnota $\frac{\sqrt{5}-1}{2} \cong 0{,}618$ Další drobnou odchylkou je počet kroků algoritmu N. Tento počet v tomto případě definuje přesnost s jakou obdržíme výsledek a není zdrojem omezení jako v předchozím algoritmu. Nevýhodou metody zlatého řezu je její pomalejší konvergence.

### *2.1.1.3 Rovnoměrná komparativní metoda*

Principem této metody je rozdělení počátečního intervalu $\langle a,b \rangle$, v němž leží extrém, $N$ body na $N+1$ podintervalů. V těchto $N$ bodech je vyčíslena účelová funkce a hledá se minimum z těchto hodnot. Optimální bod je lokalizován s přesností dvou podintervalů přičemž společným bodem těchto podintervalů je nalezené minimum

z vypočtených hodnot. Body $x_n$ dělící původní interval $\langle a,b\rangle$ se vypočítají ze vztahu $x_n = a + n\frac{b-a}{N+1}$, kde $n \in \langle 1,N\rangle$. Pro bod $x_{\min}$ platí $f(x_{\min}) = \min(f(x_n))$, kde $n \in \langle 1,N\rangle$. O optimálním bodu $x^*$ lze potom říci $x^* = x_m \pm \frac{b-a}{N+1}$. [3] Opět je nutné poznamenat, že níže uvedený algoritmus hledá minimum. V případě požadavku na nalezení maxima je nutné v bodě 4 změnit funkci min na max.

Algoritmus:

1. Volba počtu bodů N
2. Volba hranic intervalu a,b
3. Výpočet dělících bodů For [ $n = 1,\ N, x_n = a + n\frac{b-a}{N+1}$ ]
4. Nalezení minimální hodnoty funkce $f(x_m) = \min_{1\le n\le N}(f(x_n))$
5. $x^* = x_m \pm \frac{b-a}{N+1}$

Výčet není vyčerpávající a není to ani jeho cílem. Možnost rozšíření těchto metod do více dimenzí brání především enormní nárůst výpočetní složitosti. Především z tohoto důvodu nevychází algoritmy pro více rozměrné úlohy z těchto jedno rozměrných metod jak by bylo možné předpokládat.

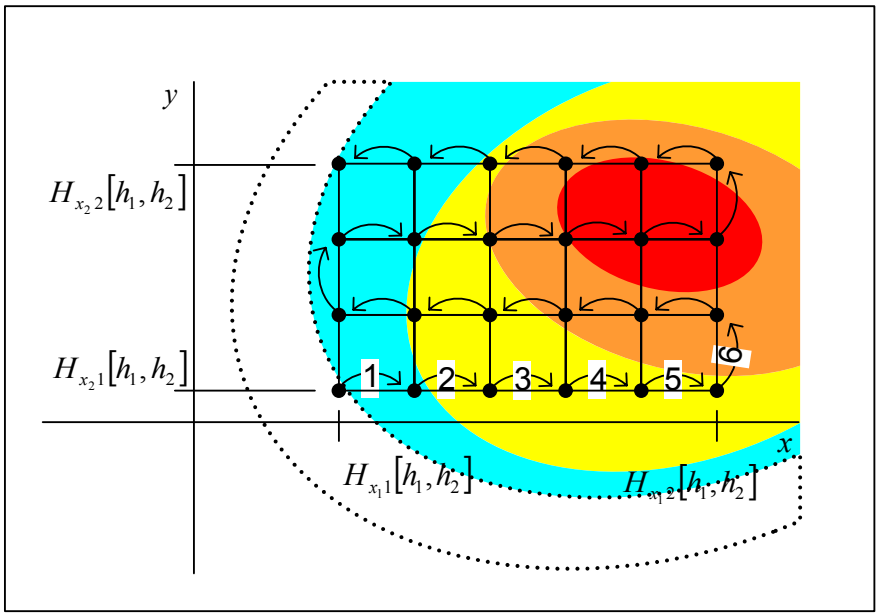
### 2.1.2 Mnohorozměrné komparativní metody

#### *2.1.2.1 Metoda mapování kriteriální plochy*

Prvotním pokusem vytvořit algoritmus pro hledání extrému funkce využitelný na počítacích strojích bylo mapování kriteriální plochy. Motivací pro práci byla snaha, využít potenciálu rozvíjející se počítačové techniky a eliminovat, nebo co nejvíce usnadnit ruční počítání. Z dnešního pohledu je hodnocení úspěšnosti tohoto počínání poměrně kontroverzní, neboť proti současnosti byly v té době počítače naprosto v počátcích a přesto ani dnes vzhledem k vysoké výpočetní náročnosti metody ji nelze rozumně využít.

Tvůrcům ovšem nelze upřít snahu a co je snad důležitější, předvedení nových možností využití techniky.

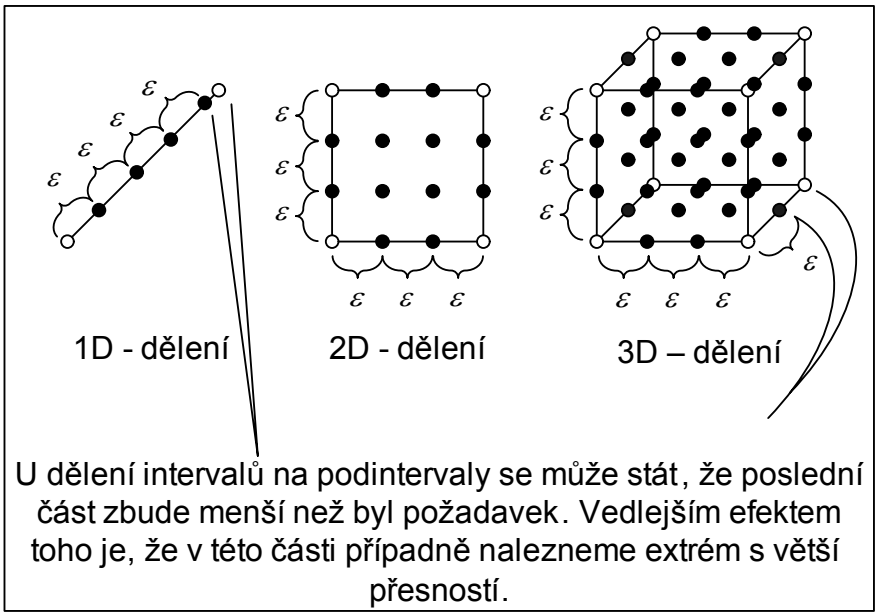
Metoda předpokládá, že je možné předem určit interval ve kterém se extrém nachází. Fungování algoritmu pro funkci dvou proměnných lze popsat následovně. Nejdříve se definují mezní přímky, které jsou hranicemi plochy v níž hledání proběhne. O takto vymezeném prostoru víme, že uvnitř něj extrém leží. Typicky se plocha vymezí přímkami rovnoběžnými s osami a průsečíky na osách tvoří body hraničního intervalu. V druhé fázi se vymezená plocha opět rozdělí rovnoběžkami s hranicemi této plochy, přičemž počet těchto rovnoběžek odpovídá požadované přesnosti nalezení extrému. Účelová funkce se vyčíslí ve všech průsečících rovnoběžek uvnitř vymezené oblasti včetně hraničních přímek a pro všechna tato vyčíslení se hledá v závislosti na požadavku maximum nebo minimum. Ačkoli je možné oblast pro prohledávání teoreticky definovat jakkoli komplikovanou, je jistě nejjednodušší definovat vždy dvojici hraničních bodů pro každou z proměnných účelové funkce. V takovémto případě se i zjednoduší hledání bodů ve kterých je nutné funkci vyčíslit.




Obrázek 5 Mapování kriteriální plochy, vyčíslované vrcholy

Zobecnění postupu pro funkci více proměnných pouze znamená zavedení více hraničních intervalů. V N-rozměrném prostoru tedy bude za potřebí $2N$ hraničních bodů.

Případy, kdy by bodů bylo potřeba více zbytečně komplikují samotné hledání a proto není vhodné o těchto případech uvažovat, byť vyloučit je zcela nelze.




Obrázek 6 Dělení intervalu v jedno až tří-rozměrném prostoru

Nevýhody algoritmu již byly zmíněny dříve. Obrovským problémem je extrémní nárůst výpočetní náročnosti jednak se zvyšujícím se počtem dimenzí úlohy, ale také se stoupající přesností. Praktická použití tohoto algoritmu jsou silně omezena.

Provedením níže popsaných kroků algoritmu řešení dostaneme extrém na vymezené ploše v podobě maxima. Bude-li požadavek na hledaní hodnoty minimální, je nutné v bodě 6 zaměnit funkci max za min.

Algoritmus:

1. Definice účelové funkce $f(x_1,..,x_n)$
2. Volba hranic prohledávané plochy
   $H_{x_1 1}(h_1,..,h_n), H_{x_1 2}(h_1,..,h_n),.., H_{x_n 1}(h_1,..,h_n), H_{x_n 2}(h_1,..,h_n)$
3. Volba přesnosti $\varepsilon$
4. Rozdělení vymezujících intervalů na podintervaly o velikosti $\varepsilon$
5. Nalezení všech průsečíků $P_1,.., P_m$
6. $f(M) = \max(f(P_1),.., f(P_m))$

### *2.1.2.2 Box – Wilson*

Pod názvem Box – Wilsonova metoda se skrývá jeden z nejzákladnějších komparativních optimalizačních algoritmů. Ačkoli byl publikován již v roce 1951 i v současnosti zůstává vhodným nástrojem pro vyhledávání extrémů účelové funkce. Svou pozici si vydobyl zejména dvojicí vlastností jež mají pozitivní dopad pro praktické použití. Základním rysem metody je její snadná algoritmizovatelnost a principiální jednoduchost, což přímo předurčuje k využití zejména tam, kde je za potřebí jakýmkoli způsobem dospět k uspokojivému řešení optimalizačního problému bez velkých nároků na podrobnější specifikace. Druhou cennou vlastností algoritmu je jeho nezávislost na počtu proměnných prohledávané funkce a případná modifikovatelnost.

Samotné fungování algoritmu lze zřejmě nejnázorněji vysvětlit na případu optimalizace úlohy dvou proměnných.

Nejdříve je zvolen startovací bod, označme jej $S[x_{S1}, x_{S2}]$ se souřadnicemi $x_{S1}, x_{S2}$, který slouží k rozběhnutí celého iterativního procesu výpočtu. Do polohy tohoto bodu se v nejlepším případě promítne počáteční znalost problému a za startovací se vybere takový, jež nejrychleji povede k řešení. V případě velké složitosti úlohy či jiných problémů lze jako startovací zvolit bod zcela libovolný, s oblibou se v takovém případě využije počátku souřadného systému. Druhým a také posledním vstupním parametrem, pomineme-li samotnou optimalizovanou funkci, je hodnota definující délku strany čtverce. Pro tuto hodnotu zavedeme označení $\Delta$. Konstrukce čtverce bude popsána níže, pro tuto chvíli

stačí konstatování, že v jeho geometrickém středu se v prvním kroku algoritmu bude nacházet zvolený startovací bod. Velikost čtverce, který volbou parametru vymezíme má klíčový vliv na následující běh algoritmu. Přímo se tímto parametrem ovlivní jednak samotná doba výpočtu a jednak přesnost řešení. Pro dobu výpočtu lze říci, že je nepřímo úměrná velikosti hrany čtverce, což tedy znamená, že čím větší délku zvolíme, tím kratší čas zabere docílení extrému. V případě přesnosti je úměra opačná. Menší hrana čtverce implikuje větší přesnost dosaženého výpočtu.

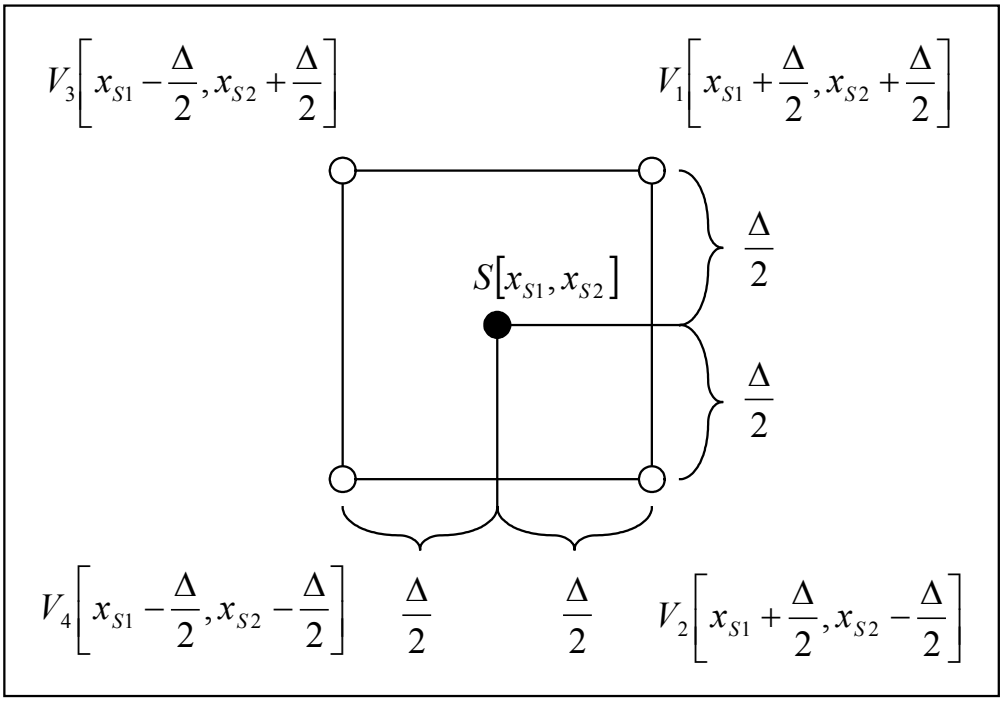
Funkce pro niž hledáme extrém, startovací bod a délka strany čtverce jsou veškeré parametry potřebné pro odstartování prvního kroku algoritmu. Jak již bylo zmíněno, nejprve je za potřebí zkonstruovat, v našem dvou-dimenzionálním případě, čtverec, přesněji tedy postačí jeho vrcholy. Jestliže máme střed čtverce, kterým pro nás je zvolený startovací bod $S[x_{S1}, x_{S2}]$, pak jednoho vrcholu získáme, pokud 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 u $\frac{\Delta}{2}$. Označíme-li jednotlivé vrcholy a jejich příslušné souřadnice $V_1\lfloor x_{v_1 1}, x_{v_1 2}\rfloor$ až $V_4\lfloor x_{v_4 1}, x_{v_4 2}\rfloor$, pak jejich jednotlivé souřadnice budou rovny:

$$V_1 : x_{v_1 1} = x_{S1} + \frac{\Delta}{2}, x_{v_1 2} = x_{S2} + \frac{\Delta}{2}$$

$$V_2 : x_{v_2 1} = x_{S1} + \frac{\Delta}{2}, x_{v_2 2} = x_{S2} - \frac{\Delta}{2}$$

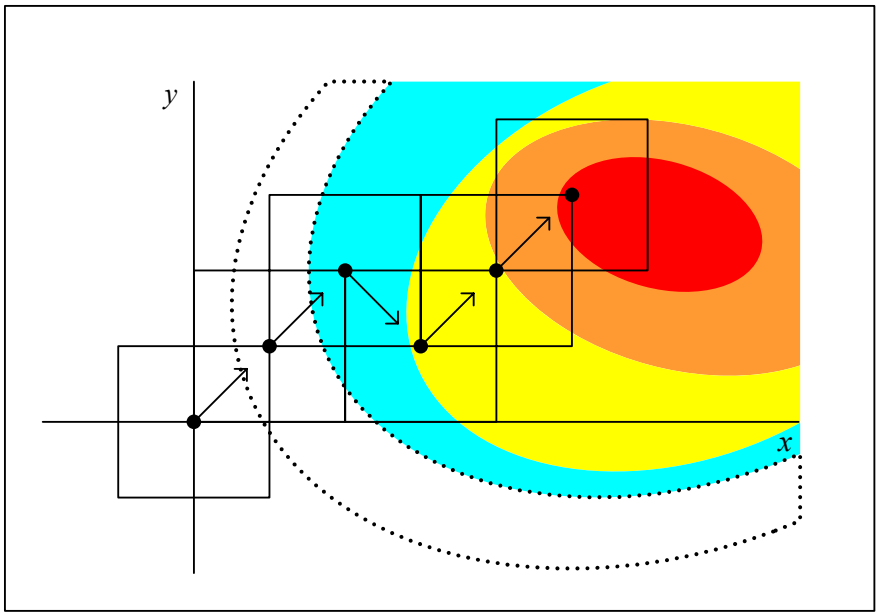
$$V_3 : x_{v_3 1} = x_{S1} - \frac{\Delta}{2}, x_{v_3 2} = x_{S2} + \frac{\Delta}{2}$$

$$V_4 : x_{v_4 1} = x_{S1} - \frac{\Delta}{2}, x_{v_4 2} = x_{S2} - \frac{\Delta}{2}.$$




Obrázek 7 Konstrukce vrcholů

V dalším kroku je nutné vypočítat hodnotu účelové funkce středu $S[x_{S1},x_{S2}]$ a vrcholů $V_1\lfloor x_{v_1 1},x_{v_1 2}\rfloor$ až $V_4\lfloor x_{v_4 1},x_{v_4 2}\rfloor$. Z této pětice vypočtených údajů vybereme ten, jehož hodnota je nejvyšší. V případě, že tuto nejvyšší hodnotu má střed, je výpočet u konce a za maximum naší funkce prohlásíme tento střed. Je-li největší hodnota v jednom z vrcholů čtverce, pak za střed pro další iteraci prohlásíme tento vrchol a algoritmus pokračuje novým výpočtem čtverce a opětovným ohodnocením jeho vrcholů.




Obrázek 8 Box - Wilson, pohyb vrcholů směrem k extrému

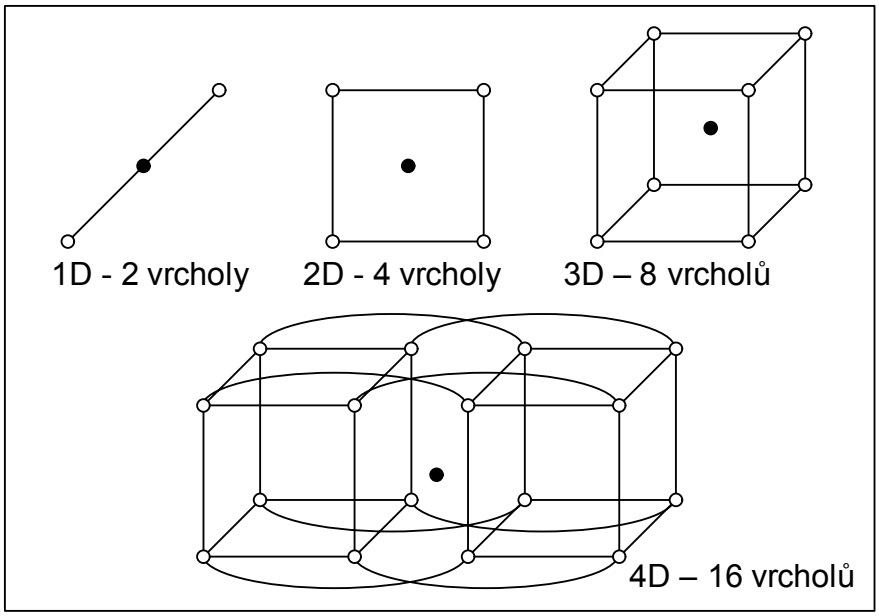
Z výše popsaného je již jistě zřejmé jak počáteční nastavení středu a délky strany čtverce přesně v algoritmu působí, zároveň ale vyvstává několik dalších otázek, jež je ještě nutno zodpovědět. První z těchto otázek se týká hledaného extrému. Box – Wilsonova metoda je ve svém základu implementována pro hledání maxima dané funkce. Z toho také pramení v algoritmu vyhledávání nejvyšší hodnoty účelové funkce ve vrcholech čtverce. V případě, že je v našem zájmu najít minimum dané funkce $f(x,y)$ je možno jednak přepsat původní algoritmus tak, že bude vyhledávat v každém kroku vrchol s nejmenší hodnotou, což ale nemusí být vždy možné, nebo můžeme hledat maximum funkce $-f(x,y)$.

Druhý problém se týká zastavení algoritmu. Za předpokladu, že jde vše přesně podle očekávaných předpokladů, je ukončení algoritmu triviální záležitostí a přesně koresponduje s ukončením výše popsaným. V případě, že o účelové funkci nemáme dostatek informací, může se ale velice snadno stát, že startovací bod zvolíme natolik nešikovně, že se nepodaří extrém nalézt. K takovému problému se například dostaneme v případě, že funkce nemá žádný extrém, nebo má pouze minimum a podobně. Taková situace v základním algoritmu řešena není a vedla by k nekonečnému výpočtu. Skutečnost, že při praktický úlohách velice často nemáme potřebné znalosti o úloze, je nějaký mechanizmus nouzového zastavení spíše nutností. Nejsnazší a také nejúčinnější se jeví

prosté omezení množství iterací algoritmu. Je-li běh výpočtu násilně ukončen po zadaném počtu iterací ale po přehodnocení nadále trvá předpoklad postupování správným směrem, je velice snadné celý algoritmus znovu spustit z bodu, kde byl přerušen.

Třetí otázka je zaměřena na přesnost spočteného výsledku. Ukončovací pravidlo říká, že za výsledek se bere střed čtverce v případě, že neexistuje vrchol s vyšším ohodnocením. To ovšem nevylučuje skutečnost, že uvnitř čtverce neexistuje bod, jehož ohodnocení by bylo ještě vyšší. Přesnost je tedy dána hranou čtverce. Tato skutečnost přímo vybízí k úpravě algoritmu tak, aby se v každém iteračním kroku upravovala i velikost hrany čtverce. Např.: $\Delta_{n+1} = \Delta_n \cdot \frac{1}{2}$. Takto navržený postup je skutečně možné realizovat, ale také je nutné si uvědomit rizika. Opět zde totiž vyvstává otázka ukončení algoritmu. Při zmenšování strany čtverce se opět snadno dostaneme do situace, kdy ke zmenšování muže docházet neustále a k ukončení bude opět za potřebí buď čítač nebo bude přerušení výpočtu dánu minimální velikostí strany čtverce $\Delta$. Navržené vylepšení ale není zcela jednoznačné, neboť je nutno počítat jednak s notným zpomalením a také zmohutněním základního algoritmu. Proto je na zvážení, zda není výhodnější po nalezení extrému spustit tento algoritmus ještě jednou z nalezeného bodu a menším $\Delta$.

Poslední otázkou je rozšíření algoritmu na více, než dvě dimenze. V případě tří-dimenzionálního problému je místo čtverce použito k výpočtu krychle, přičemž její vrcholy se spočítají analogicky jako v případě čtverce. Pro další rozměry je již vícedimenzionální krychle těžko představitelná, nicméně narůst počtu vrcholů odpovídá mocnině dvou a lze tedy zapsat $2^{\dim enze}$ a výpočet jejich pozic se nikterak nemění ani pro tyto vyšší dimenze. Pro úplnost nelze opomenout jedno-dimenzionální úlohu, kde se základem výpočtu stávají body konců úsečky.




Obrázek 9 Jedno až čtyř - rozměrná krychle

Samotný závěr popisu Box – Wilsonovy metody bude patřit rozšířenému algoritmu pro obecný počet dimenzí, ale ještě před tím je nutno zmínit nevýhody, kterými je tato metoda provázena. Jednoznačnou nevýhodou této metody je počet ohodnocení účelové funkce jež je v každém kroku roven $2^{\dim enze}+1$. Tento počet velice rychle roste s počtem dimenzí a v případě, kdy se v průběhu výpočtu zmenšuje strana $\Delta$ narůstá množství výpočtů ještě více. Popsaná vlastnost má samozřejmě přímý dopad na rychlost algoritmu. Tyto nevýhody jsou ovšem vyváženy jednoduchostí a průhledností prováděných operací.

Algoritmus:

$n \in \overline{N}^{+}$, $k \in \langle 1,2^n \rangle$, $\delta \in \{0,1\}$-tak aby byly zajištěny všechny kombinace

1. Definice účelové funkce $f(x_1,..,x_n)$
2. Volba středu $S[x_{S1},..,x_{Sn}]$
3. Volba velikosti strany $\Delta$
4. Vypočet vrcholů For[k=1, $2^n$, $V_k[x_{v_i1},..,x_{v_in}] = S[x_{S1},..,x_{Sn}] + (-1)^{\delta}\frac{\Delta}{2}$]
5. Výpočet $f(V_1),..,f(V_k)$ a $f(S)$

6. $f(M) = \max(f(V_1),..,f(V_k),f(S))$
7. Je-li $M = S$ pak konec
8. $S = M$ skok na 3

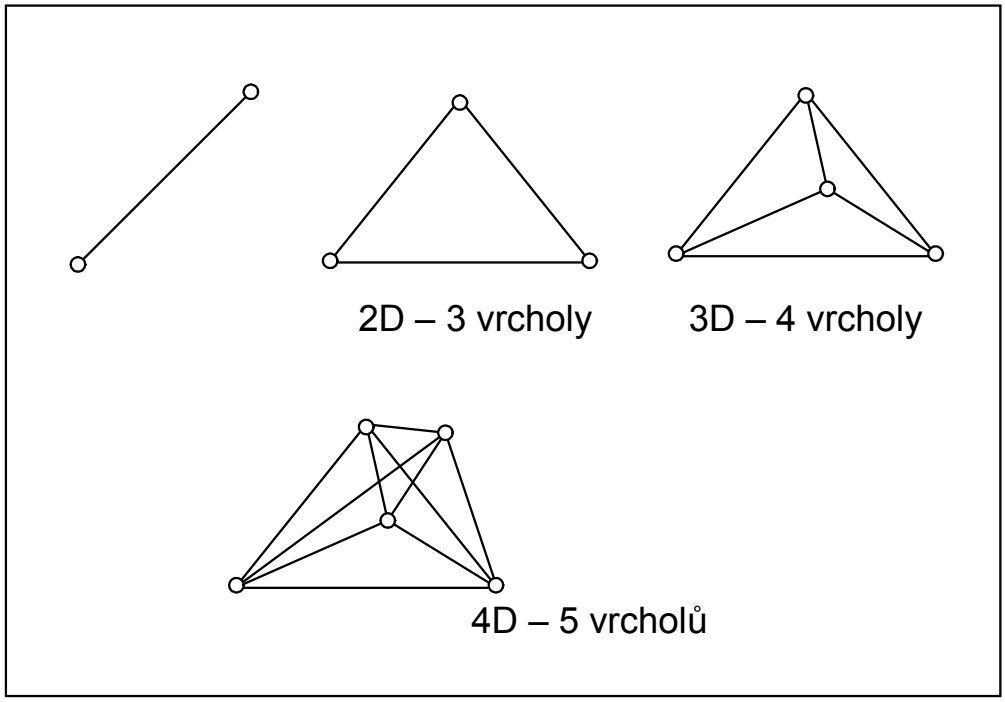
#### *2.1.2.3 Simplexová metoda*

Motivací pro práci na simplexové metodě byly nevýhody plynoucí z vysoké výpočetní náročnosti Box – Wilsonovy metody. Ačkoli daná skutečnost byla dobře známa, trvalo přes deset let, přesněji do roku 1962, než se podařilo snížit počet ohodnocení účelové funkce a vytvořit algoritmus nazvaný simplexová metoda.

Díky skutečnost, že nový postup výpočtu se snaží vylepšit ten stávající a také z něj do jisté míry vychází, lze vysledovat některé podobné vlastnosti. Například stejně jako u Box – Wilsonova algoritmu jsou v jednotlivých cyklech ohodnocovány vrcholy a střed n-rozměrné krychle tak v tomto novém algoritmu jsou počítány hodnoty ve vrcholech simplexu. Odtud také plyne název metody. Podobnost lze také vysledovat ve volitelných parametrech, jako je délka hrany či startovací bod. Pravděpodobně nejzásadnějším rozdílem je zacházení s perspektivním, tedy nejlepším a neperspektivním, nejhorším stavem. V Box – Wilsonově metodě je vždy v popředí zájmu vrchol s nejvyšším ohodnocením, v případě maxima, a po jednom kroku algoritmu se tento stává nosným bodem dalšího výpočtu. Všechny ostatní hodnoty se po ukončení iterace stávají nezajímavými a jsou zapomenuty. U simplexové metody je naopak vybírán vrchol s nejhorším ohodnocením. Tento bod je poté daným postupem nahrazen novým.

Než bude prezentováno samotné fungování simplexové metody, je za potřebí řádně objasnit s jakým útvarem se v postupu pracuje.

Pojmem simplex je označován nejmenší konvexní polyedr v daném prostoru. Jinými slovy, v N dimenzionálním prostoru definujeme N+1 bodů jež tvoří simplex. Z dané definice tedy plyne, že tento útvar v dvou-rozměrném prostoru odpovídá trojúhelníku. Pro tří-rozměrný prostor jde o čtyřstěn. Limitním případem je prostor jedno-dimenzionální, kde se útvar redukuje na úsečku, nicméně prakticky není aplikace simplexové metody v jedno-rozměrném prostoru používána, byť tomu žádná omezení nebrání.



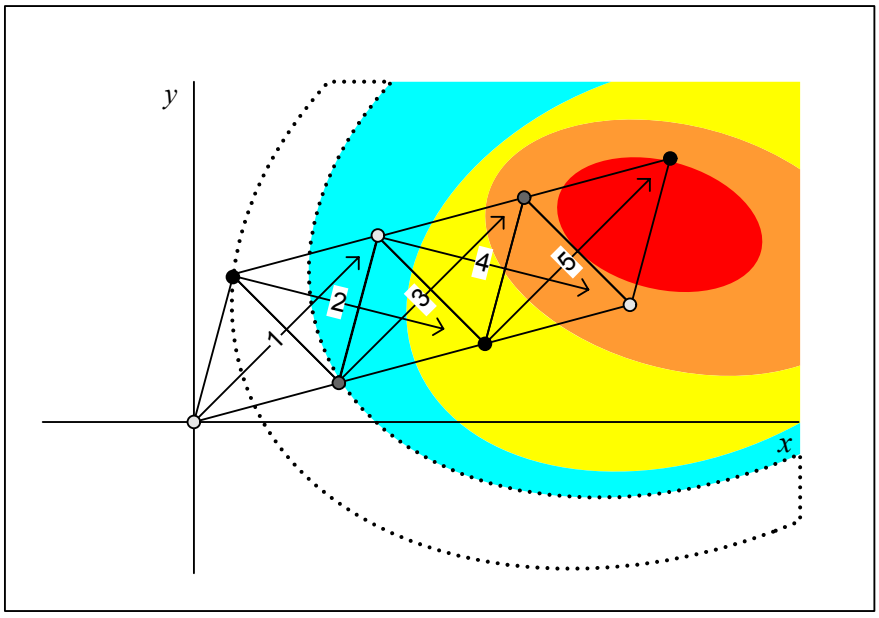

Obrázek 10 Simplex v jedno až čtyřrozměrném prostoru

Stejně jako v případě předchozí metody lze pro popis fungování algoritmu s výhodou využít dvou-dimenzionálního problému. Vzhledem k již zmíněné podobnosti algoritmu s algoritmem Box – Wilsnovým i v této metodě nejprve zvolíme startovací bod $S[x_{S1}, x_{S2}]$, kde $x_{S1}, x_{S2}$ jsou souřadnice tohoto bodu. Symbolem $\Delta$ označme délku strany simplexu, které v konečném důsledku má opět vliv na rychlost konvergence algoritmu a také na přesnost nalezeného výsledku. Obdobně jako u Box – Wilsonovy metody jsou požadavky na přesnost a rychlost postaveny proti sobě. Vyšší přesnost znamená pomalejší algoritmus, zatímco rychlost znamená zvýšení chyby.

Jsou-li k dispozici dva základní parametry, tedy startovací bod a délka hrany simplexu, nic nebrání spuštění algoritmu. V prvém kroku postupu je zkonstruován simplex. V metodě se užívá pravidelného simplexu a konstrukce jeho vrcholů $V_1[x_{v_1 1}, x_{v_1 2}]$ až $V_3[x_{v_3 1}, x_{v_3 2}]$ je dána výrazem: $V_1 = S$, $V_2 = V_1 + \begin{pmatrix} 2d+e \\ d \end{pmatrix}$, $V_3 = V_1 + \begin{pmatrix} d \\ 2d+e \end{pmatrix}$, kde $d = \Delta \frac{\sqrt{2+1}-1}{\sqrt{2}}$ a $e = \frac{\Delta}{2}$. Součty je samozřejmě nutné aplikovat na odpovídající složky vektoru, výsledný body simplexu tedy v prvním kroku jsou:

$V_1[x_{S1}, x_{S2}], V_2\left[x_{S1} + 2\Delta\frac{\sqrt{3}-1}{\sqrt{2}} + \frac{\Delta}{2}, x_{S2} + \Delta\frac{\sqrt{3}-1}{\sqrt{2}}\right],$

$V_3\left[x_{S1} + \Delta\frac{\sqrt{3}-1}{\sqrt{2}}, x_{S2} + 2\Delta\frac{\sqrt{3}-1}{\sqrt{2}} + \frac{\Delta}{2}\right]$. V okamžiku, kdy jsou již k dispozici všechny body simplexu, provede se v těchto vrcholech ohodnocení účelové funkce. Z trojice bodů se vybere takový, pro který platí, že hodnota funkce v tomto bodě je nejvyšší. Pro tento bod jakožto nejhorší v celém simplexu se hledá jeho osově souměrný obraz s osou procházející zbylými dvěma body. V případě, že funkční hodnota v obrazu nejhoršího bodu je menší než v bodu samotném provede se výměna těchto bodů a vzniká nový, posunutý simplex pro nějž součet funkčních hodnot ve všech bodech je menší než v simplexu předchozím. V případě, že nově nalezený bod nevyhovuje podmínce s nižší hodnoty účelové funkce algoritmus končí a za výsledek se považuje bod s nejnižší hodnotou účelové funkce v posledním simplexu. V opačném případě, tedy po nalezení nového vrcholu pokračuje algoritmus s novým simplexem.




Obrázek 11 Simplex, pohyb vrcholů k extrému

Simplexová metoda v podobě jak je popsána výše pro dvou-dimenzionální problém hledá minimum optimalizační funkce. Obdobně jako v Box –Wilsonově metodě lze upravit samotný algoritmus, kdy nahrazován bude vrchol s nejnižší hodnotou, nebo hledat

minimum funkce s opačným znaménkem. Ukončování algoritmu je také shodné s Box – Wilsonovou metodou. V případě, že není zaručeno, že funkce má ze startovacího bodu dosažitelní minimum, je vhodné definovat maximální počet iterací, které budou provedeny před násilným ukončením algoritmu.

Rozšíření algoritmu pro vícerozměrné úlohy je více méně závislé pouze na schopnosti výpočtu počátečního simplexu. Pro výpočet vrcholů pravidelného simplexu v N rozměrném prostoru je definován následující vztah: $V_1 = \begin{pmatrix} x_{S1} \\ x_{S2} \\ .. \\ x_{SN} \end{pmatrix}$, $V_2 = V_1 + \begin{pmatrix} 2d+e \\ d \\ .. \\ d \end{pmatrix}$, .. , $V_{N+1} = V_1 + \begin{pmatrix} d \\ d \\ .. \\ 2d+e \end{pmatrix}$, kde $d = \Delta \frac{\sqrt{N+1}-1}{\sqrt[N]{2}}$ , $e = \frac{\Delta}{2}$ a $\Delta$ je délka strany simplexu. Ani v dalších krocích nepředstavuje pro algoritmus více rozměrný prostor žádný problém. Snad jen osa souměrnosti se mění na N-dimenzionální rovinu.

Výhodou simplexové metody je bezesporu menší počet ohodnocení účelové funkce proti Box –Wilsonově metodě. Tento počet je v jednom kroku roven $N+1$ proti původní $2^N+1$. Nevýhodou je konstantní rychlost postupování k extrému i v případě velkých hodnotových rozdílů.

Algoritmus:

$n \in \overline{N}^+$, $k \in \langle 1, N+1 \rangle$, $\delta$ odpovídající vektor $\begin{pmatrix} 2\Delta \frac{\sqrt{N+1}-1}{\sqrt[N]{2}} + \frac{\Delta}{2} \\ \Delta \frac{\sqrt{N+1}-1}{\sqrt[N]{2}} \\ .. \\ \Delta \frac{\sqrt{N+1}-1}{\sqrt[N]{2}} \end{pmatrix}$

1. Definice účelové funkce $f(x_1,..,x_n)$
2. Volba startovacího bodu $S[x_{S1},..,x_{Sn}]$
3. Volba velikosti strany simplexu $\Delta$
4. Výpočet vrcholů simplexu $V_1 = S$ ; For[k=1,n+1, $V_k = V_1 + \delta$ ]

5. $f(V_m) = \max(f(V_1),..f(V_{N+1}))$

6. Výpočet překlopeného bodu $V_{preklopeny} = V_m + 2\frac{1}{N}\sum_{\substack{j=1 \\ j \neq m}}^{N+1} V_j$

7. Je-li $f(V_m) < f(V_{preklopeny})$ pak konec

8. $V_m = V_{preklopeny}$ skok na 5

#### *2.1.2.4 Flexibilní simplexová metoda*

Ačkoli je simplexová metoda výpočetně méně náročná než metoda Box – Wilsonova, jeví se její konstantní krok jako brzdící. Metoda se nedokáže přizpůsobit funkci pro níž se snaží nalézt extrém a je tak odkázána na kvalitu počátečních parametrů.

Roku 1965 byla publikována nově přepracovaná simplexová metoda jež možnosti té dosavadní významně rozšiřuje, Metoda je nazývána dle svých autorů Nelder – Mead method, nebo též flexibilní simplexová metoda. Základem metody je opět útvar zvaný simplex, nicméně zásadním rozdílem v tomto algoritmu jsou dynamické změny jeho rozměrů. Dodatečnými výpočty jsou ohodnocovány další body v prostoru a záměna nejhoršího bodu je provedena za ten nejvýhodnější. Tímto postupem je dosaženo patrně nejdokonalejšího iteračního algoritmu současnosti.

Další testované body jsou opět odvozeny od nejhoršího bodu v dané iteraci a nejsou tedy dílem náhody. V závislosti na tom, jakého ohodnocení dosahuje zrcadlený bod vzhledem k jeho obrazu vypočítává se také ohodnocení bodu s dvojnásobnou nebo poloviční vzdáleností od roviny souměrnosti. Případně se ohodnocuje bod ve středu vzdálenosti mezi bodem s nejhorším ohodnocením a rovinou zbylých bodů. Simplex se tak přizpůsobuje průběhu účelové funkce.

Do samotného algoritmu nově přibývají další vstupní parametry. Označují se $\alpha$, $\beta_1$, $\beta_2$ a $\gamma$. Umožňují ovlivňovat velikosti změn simplexu, jejich konkrétní význam vyplyne z rovnic. Konstrukce nového simplexu u Nelder – Mead simplexové metody je složena ze čtveřice operací, přičemž vždy k následující se přistupuje jen v případě nesplnění podmínek. První operací je reflexe. Ta je známa již z klasické simplexové metody, tedy zrcadlení bodu. Druhou operací je expanze, ta v případě, že zrcadlený bod má nejnižší hodnotu účelové funkce v nově vzniklém simplexu testuje, zda by nebylo ještě

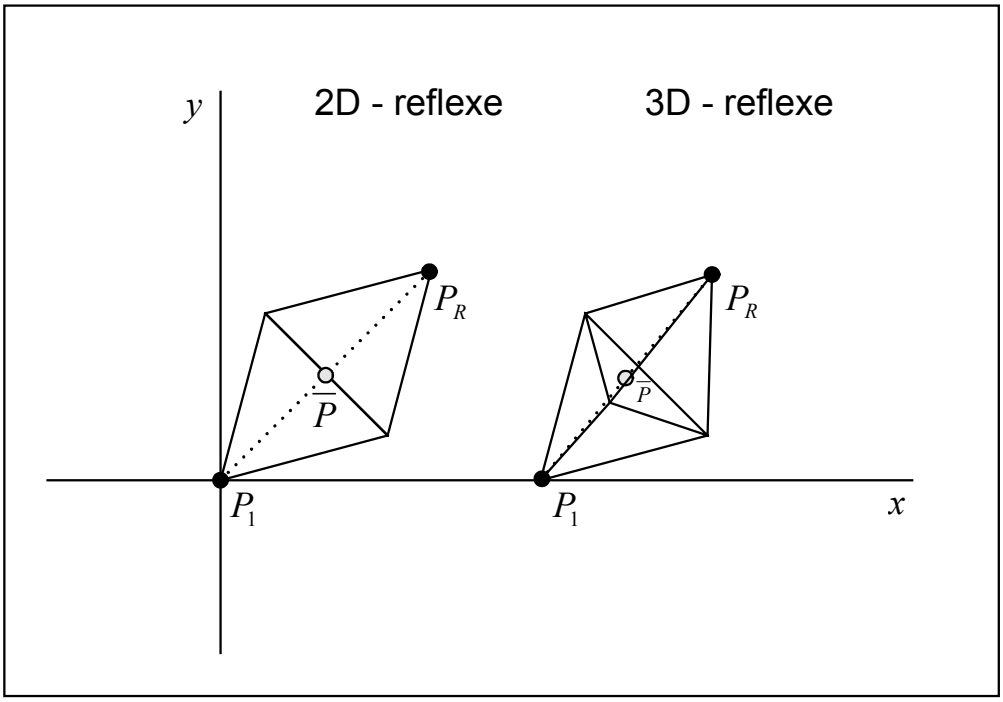
výhodnější překlopit obraz bodu do dvojnásobné vzdálenosti. Třetí operací je kontrakce, zde se naopak v případě ne příliš výhodné hodnoty zrcadleného bodu testuje bod překlopený pouze do poloviční vzdálenosti. Poslední operací je redukce, kdy se provede celkové zmenšení simplexu.

Rovnicemi lze jednotlivé kroky a podmínky pro přijetí konkrétního nového vrcholu simplexu napsat několika způsoby. Ten následující povede opět na hledání minima funkce. Pro snazší orientaci je výhodné speciálně dodefinovat následující body. Symbolem $P_1$ je označen ten vrchol, který má v současném simplexu největší ohodnocení účelové funkce. $P_2$ značí vrchol s druhým nejhorším ohodnocením účelové funkce. $P_L$ představuje vrchol v daném simplexu nejlepší, tedy s nejnižším ohodnocením účelové funkce. A posledním je bod $\overline{P}$, který představuje střed N-rozměrné roviny tvořené všemi body simplexu kromě bodu $P_1$.

Nejprve se v algoritmu provede **reflexe**. Zrcadlený bod $P_R$ se nalezne zrcadlením bodu $P_1$ přes vrchol $\overline{P}$.

$$P_R = (1+\alpha)\cdot \overline{P} - \alpha \cdot P_1,$$

kde $\alpha$ je zrcadlící faktor. V původní metodě Nelder – Mead je $\alpha = 1$. Jestliže $f(P_L) < f(P_R) < f(P_1)$, pak je bod $P_1$ nahrazen bodem $P_R$.

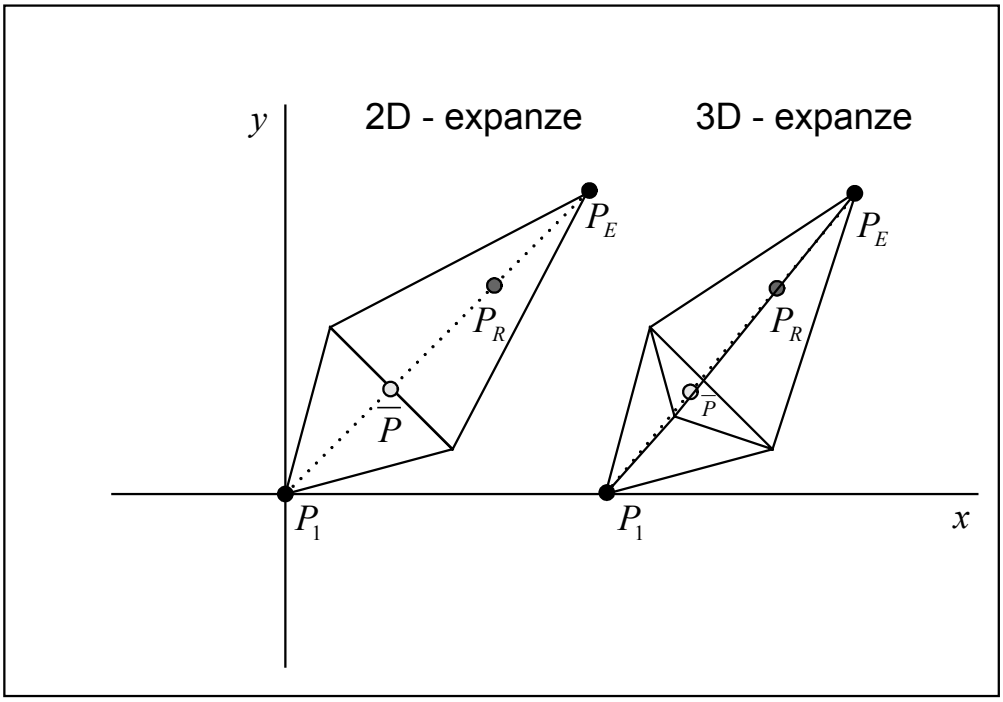



Obrázek 12 Reflexe

Jestliže $f(P_R) \leq f(P_L)$ je výhodné provést **expanzi** v naději, že expandovaný bod $P_E$ bude ještě výhodnější než bod reflektovaný $P_R$. Expanze je definována vztahem

$$P_E = (1-\gamma)\cdot \overline{P} + \gamma \cdot P_R ,$$

kde $\gamma$ je expanzní faktor. Nelder – Mead dávají $\gamma = 2$. Jestliže $f(P_E) < f(P_L)$, pak je bod $P_1$ nehrazen bodem $P_E$. V případě, že funkční hodnota není menší, je bod $P_1$ nahrazen bodem $P_R$.

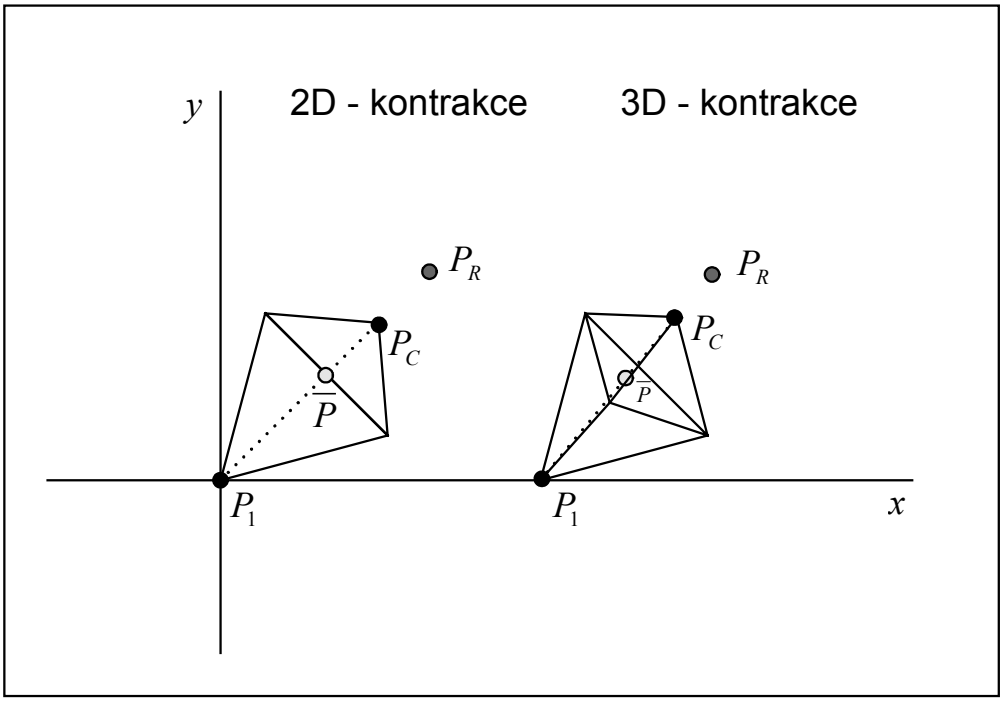



Obrázek 13 Expanze

Je-li $f(P_R) \geq f(P_2)$ provádí se **kontrakce** do bodu $P_C$ vztahem

$$P_C = (1 - \beta_1) \cdot \overline{P} + \beta_1 \cdot P_0 ,$$

kde $\beta_1$ je faktor kontrakce. Nelder – Mead definují $\beta_1 = 0.5$. Jako bod $P_0$ se zvolí ten z dvojice bodů $P_1$ nebo $P_R$ který má menší funkční hodnotu. Jestliže $f(P_C) < f(P_0)$, pak je bod $P_1$ nahrazen bodem $P_C$.

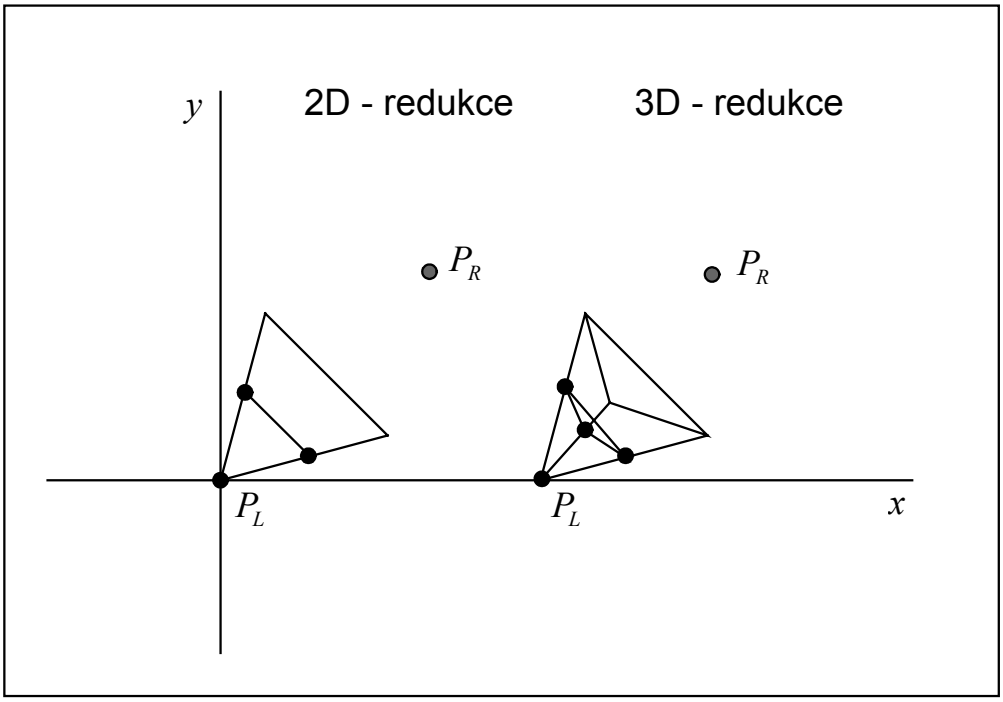



Obrázek 14 Kontrakce varianta $P_0 = P_R$

V případě, že $f(P_C) \geq f(P_0)$ je provedena **redukce**. Při této operaci se zmenší celý simplex. Jediným bodem, který zůstane je bod s nejnižším ohodnocením účelová funkce $P_L$. Všechny ostatní se změní dle předpisu

$$P_i = (1 - \beta_2) \cdot P_L + \beta_2 \cdot P_i,$$

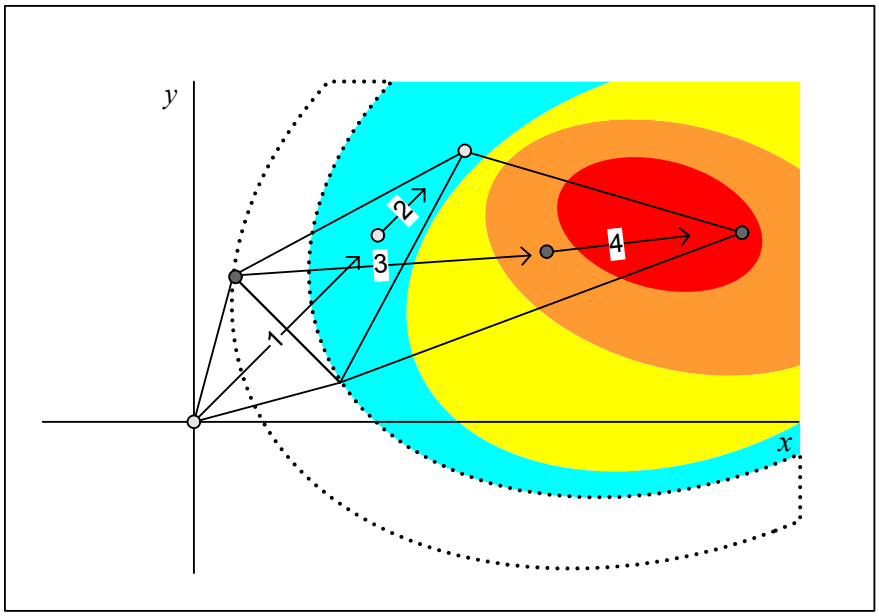
kde $\beta_2$ je faktorem redukce. Nelder – Mead opět definují $\beta_2 = 0.5$. Body $P_i$ jsou všechny body simplexu mimo $P_L$.




Obrázek 15 Redukce

Iterace je ukončena v okamžiku, kdy je nahrazen bod $P_1$. [4]

Jednotlivé kroky algoritmu jsou shodné s klasickou simplexovou metodou, rozšíření spočívá v podmínkách výběru bodu pro nahrazení, které jsou ovšem popsány výše. Tudíž není důvod algoritmus ještě jednou přepisovat.




Obrázek 16 Flexibilní simplex, pohyb vrcholů k extrému

Daleko zajímavější je možnost ukončení algoritmu. Jestliže je možné aby se simplex neustále zmenšoval, bude tato operace v okamžiku, kdy už se blíží extrém funkce probíhat neustále, ale bez možnosti ukončení. Počet iterací je opět řešením a není dobré jej z algoritmu zcela vypustí, zejména pro případy, kdy se extrém nalézt nedaří, ale v případě úspěšného hledání je mnohem výhodnější ukončovat algoritmus definováním minimálního možného rozdílu funkčních hodnot nejlepšího a nejhoršího bodu simplexu.

### *2.1.2.5 Metoda cyklické záměny parametrů*

Metodu cyklické záměny parametrů, jiným názvem též Gauss – Seidlovu metodu, je možné zařadit na pomezí komparativních a gradientních metod. V průběhu iterace je zapotřebí nalézt extrém funkce jedné proměnné, což může odpovídat hledání kořene pomocí derivace funkce. V takovém případě; tedy zcela neplatí, že není nutný výpočet derivace, nicméně tato se nepočítá z původní účelové funkce. Algoritmem je opět možné optimalizovat funkce o jakékoli dimenzi, ale pro snazší pochopení znovu začněme u dvou-rozměrné funkce.

Ani tato metoda nezačíná ničím jiným než definicí startovacího bodu $S[x_{S1}, x_{S2}]$. Na tento bod nejsou kladeny rozdílné požadavky než u metod předchozích. Nejlépe je znovu vycházet ze znalosti problému a bod zvolit co nejblíže optimu. Další parametrem, který je

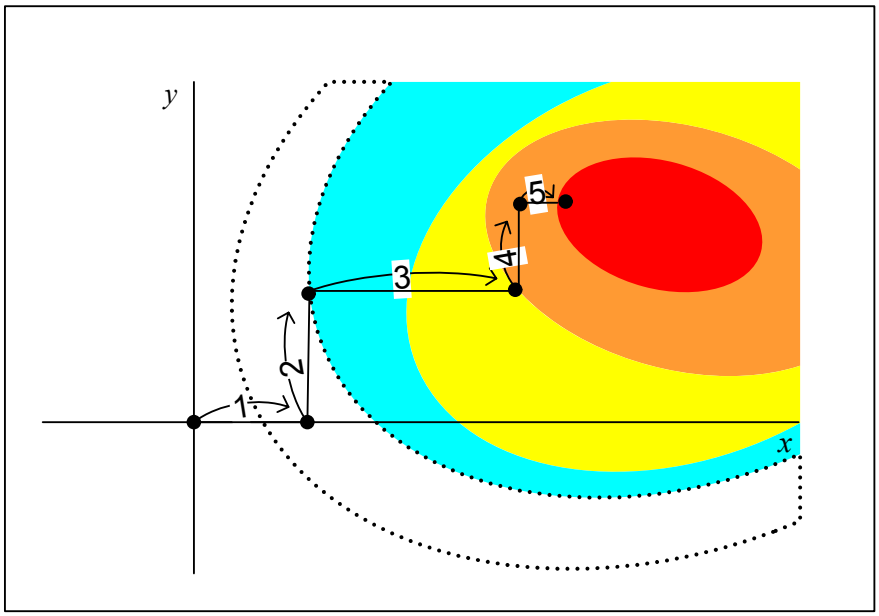
za potřebí před spuštěním algoritmu zvolit je $\varepsilon$. Tento parametr lze nazvat přesností a bude potřebný pro ukončení algoritmu. Na rozdíl od předchozích metod se nejedná o žádnou hranu, proto je značení odlišné. Posledním parametrem, který algoritmus potřebuje je počet proměnných vyhodnocované účelové funkce. Tato hodnota se zadává jako parametr i když to v obecném případě není podmínka neboť identifikace počtu proměnných ze samotného zápisu funkce je možná. V obecném případě je ovšem zjištění počtu proměnných z funkce dosti komplikované a proto je lépe využít možnosti počet proměnných zadat zvlášť. Parametr by nebylo třeba značit, neboť dle zvyklostí z předchozích algoritmů je počet proměnných zřejmý, ale z formálních důvodů a také pro přehlednost zaveďme označení $d$ jako označení dimenze úlohy.

Fungování samotného algoritmu je odlišné proti metodě Box – Wilson nebo simplexové metodě nejen ve významu parametru, ale také ve skutečnosti, že účelová funkce není v průběhu algoritmu vyčíslována pro konkrétní body. V prvém kroku postupu je do funkce dosazen startovní bod přičemž k hodnotě dosazované za první proměnnou je přičten parametr $\xi_1$. Tímto krokem vznikne parametrizovaná funkce. Díky předchozímu kroku se v tom následujícím hledá extrém funkce jedné proměnné a to za pomoci derivace. Položením derivace rovno nule nalezneme hodnotu parametru $\xi_1$. Nový bod $V_1$ jež se přiblíží k hledanému extrému dostaneme přičtením parametru $\xi_1$ k hodnotě první proměnné startovacího bodu $V_1\left[x_{S1}+\xi_1, x_{S2}\right]$. Dále je už podstatná pouze hodnota součtu x-ové souřadnice, pro zdůraznění tohoto faktu proto $x_{S1}+\xi_1 = x_{v_1 1}$. Nyní následuje analogický postup pro druhou proměnnou s tím rozdílem, že dále bude využito již nového bodu $V_1$. Do funkce tedy dosadíme za první parametr hodnotu prvního parametru tohoto bodu a za druhý parametr hodnotu $x_{S2}$ zvětšenou o $\xi_2$.Opět nalezneme extrém a hodnotu přičteme. Vznikne bod $V_2\left[x_{v_1 1}, x_{S2}+\xi_2\right]$. Bod $V_2$ již je konečným stavem prvního iteračního kroku. Nová iterace je spuštěna v případě, že $\xi_1^2+\xi_2^2 \geq \varepsilon$ a novým startovacím bodem je bod $V_2$. Je-li nerovnost opačná je dosaženo výsledku s požadovanou přesností a výpočet je ukončen. Za hodnotu extrému je považována hodnota v bodě $V_2$.

Zajímavostí tohoto algoritmu je samotný výsledek. Je-li požadavkem nalézt algoritmem cyklické záměny parametrů extrém funkce, jež je například těžko představitelná, za předpokladu, že tento extrém existuje a je ze startovacího bodu jediný

dosažitelný proběhne úspěšně, nicméně o výsledku který metoda vrátí nebude možné bez dalšího výpočtu rozhodnout, zda se jedná o minimum, nebo maximum. Vyplývá to ze způsobu výpočtu jednotlivých zlepšujících bodů. Není-li možné získat funkční hodnoty mezivýsledků, z kterých by bylo možno snadno určit zda se hodnota účelové funkce v další iteraci snížila či zvýšila a z toho odvodit zda je výsledný extrém minimem či maximem, je za potřebí vyčíslit funkci ještě jednou v nějakém bodě a prostým porovnáním učinit rozhodnutí. I zde je ovšem potřeba opatrnosti. Bude-li pro porovnání volený bod zcela náhodně může to vést v fatálnímu omylu. Důvodem je skutečnost, že vypočtený extrém přeci jen nemusí být skutečným extrémem funkce. V tomto případě náhodná volba bodu, který bude použit pro rozhodnutí o maximu či minimu, v nejhorším případě vybere bod ještě blíže skutečnému extrému. Učiněný závěr z porovnání funkčních hodnot v takovémto případě povede na přesně opačné tvrzení, než je skutečnost.

Nepovinné, nicméně velice výhodné je obdobně jako v případech jiných metod přidat do algoritmu možnost násilného ukončení výpočtu po určitém počtu provedených iterací. Je-li hledání extrému úspěšné, je ukončovací podmínka součástí algoritmu. Pokud ovšem například extrém nebude existovat, pak nebude podmínka ukončení s vysokou pravděpodobností splněna v žádném kroku a běh se nikdy nezastaví.




Obrázek 17 Cyklická záměna parametrů

Expanze algoritmu pro větší počet proměnných není příliš obtížná. V každé iteraci budou vystřídány všechny pozice proměnných pro které se provede přičtení parametru $\xi$ a vyhledání extrému. Je tedy nutné vyhradit parametry $\xi_1,..,\xi_d$ a body $V_1,..,V_d$, přičemž až poslední bod $V_d$ bude výstupem jednoho iteračního cyklu. Ukončovací podmínku lze v obecném případě zapsat: je-li $\sum_{i=1}^{d}\xi_i^2 < \varepsilon$ pak konec. Bez zajímavosti není, že jednotlivé proměnné není nutno střídat podle jejich pořadí, ale toto střídání je možné určit jiným klíčem. Změnou lze dosáhnout různé kvality výsledku či rychlosti konvergence. Je však velice pravděpodobné, že ze zadané úlohy nejvýhodnější pořadí nebude patrné a proto je vhodnou volbou právě postupné procházení proměnných účelové funkce. Limitní případ, kdy se počet vnitřních proměnných sníží na pouho jednu lze algoritmem řešit také, ale jedná se spíše o kuriozitu.

Při bližším pohledu na algoritmus je možné dojít k závěru, že je složen ze dvou základních částí. V jedné části jsou nastavovány potřebné proměnné a vyhodnocovány patřičné podmínky a v druhé části, která je jako by vnořena do té první se hledá extrém funkce jedné proměnné. Právě tato druhá část je pro tuto chvíli zajímavá. Pro hledání extrému jedné proměnné neslouží jen derivace zmiňovaná výše v popisu, ale velká spousta dalších metod, například i iteračních. Z tohoto pohledu se Gauss – Seidlova metoda stává vysoce modifikovatelnou či modulární. Úvodní slova o nutnosti v metodě derivovat lze tedy s úspěchem napadnou v případě, že pro vyhledávání extrému funkce jedné proměnné uvnitř algoritmu použijeme jinou techniku.

Hovořit o výhodách a nevýhodách metody cyklické záměny parametrů znamená zhodnotit použitý algoritmus pro nalezení extrému jedné proměnné. Jde tedy spíše o problematiku výhod a nevýhod jednotlivých vyhledávacích algoritmů. V případě postupu s derivací je nevýhodou čas strávený k výpočtu derivace ale zejména počet dosazení do účelové funkce jež je roven v každé iteraci počtu proměnných této funkce.

Algoritmus.

$n \in \overline{N}^+$, $d = n$ - počet proměnných účelové funkce. V bodě 4 $V_0 = S$

1. Definice účelové funkce $f(x_1,..,x_n)$
2. Volba startovacího bodu $S[x_{S1}, x_{Sn}]$

3. Volba $\varepsilon$
4. $i = i + 1;\ p_i(\xi_i) = f\left(x_{v_{i-1}1},..,x_{v_{i-1}i} + \xi_i,..,x_{v_{i-1}n}\right)$
5. $p_i'(\xi_i) = 0 \Rightarrow \xi_i = číslo$
6. $V_i = \left\lfloor x_{v_i1},..,x_{v_ii} + \xi_i,..,x_{v_in} \right\rfloor$
7. Je-li $i < d$ skok na 4
8. Je-li $\sum_{i=1}^{d} \xi_i^2 < \varepsilon$ pak konec
9. $i = 0$ ; skok na 4

## 2.2 Gradientní metody

Na rozdíl od komparativních metod, využívají gradientní metody pro svůj běh výpočet nebo odhad gradientu funkce. Iterační gradientní metoda je tedy taková, ve které jsou mezivýsledky jednotlivých iteračních kroků zvyšovány úměrně gradientu účelové funkce. Obecně lze hodnotu mezivýsledku v dalším kroku napsat jako $V_{i+1} = V_i + \lambda_i \cdot grad\left(f\left(V_i\right)\right)$. Parametr $\lambda$ ovlivňuje svou hodnotou hledaný extrém. Je-li $\lambda_i < 0$ jde o postup k minimu funkce, jestliže je $\lambda_i > 0$ jde o postup k maximu funkce. Bude-li $\lambda$ v průběhu výpočtu konstantní, jedná se o gradientní metodu s krátkým krokem, v opačném případě jde o metodu s dlouhým krokem. Další možné dělení gradientních metod je na jednorozměrné, vícerozměrné s omezením a vícerozměrné bez omezení.

### 2.2.1 Jednorozměrné gradientní metody

Mezi nejznámější jednorozměrné gradientní metody patří Newtonova metoda a metoda regula falsi

#### *2.2.1.1 Newtonova metoda*

Principem metody je provedení aproximace účelové funkce funkcí kvadratickou v aktuálním odhadu optima. [5] Ke svému startu tedy metoda opět vyžaduje startovací bod $S\left[x_{S1}, x_{S2}\right]$. Odhad je proveden pomocí první a druhé derivace účelové funkce a nový bod

přibližující se k extrému vyjde z rovnice $V_{i+1} = V_i - \frac{f'(V_i)}{f''(V_i)}$. Podmínkou metody tedy je existence druhé derivace účelové funkce. Zajímavostí metody jistě je, že vzhledem k podstatě fungování metody je rychlost její konvergence závisí na „podobnosti“ účelové funkce funkci kvadratické. Ukončovací podmínkou v algoritmu je rozdíl vzdáleností po sobě jdoucích nalezených bodů. $|V_{i+1} + V_i| \leq \varepsilon$, kde $\varepsilon$ lze nazvat přesností a je nutné jí zadat před spuštěním algoritmu.

#### *2.2.1.2 Metoda regula falsi*

Tato metoda je metodě Newtonově velice podobná. První rozdíl je ve skutečnosti, že druhá derivace v podílu je nahrazena jejím odhadem. $f''(V_i) = \frac{f'(V_i) - f'(V_{i-1})}{V_i - V_{i-1}}$. Po dosazení tohoto odhadu lze iterační krok metody regula falsi definovat vztahem $V_{i+1} = V_i - f'(V_i)\frac{V_i - V_{i-1}}{f'(V_i) - f'(V_{i-1})}$. Z uvedeného zápisu vyplývá druhý rozdíl proti Newtonově metodě a to skutečnost, že iterační krok vyžaduje pro svoji funkci dva body. Na počátku je tedy nutno zadat dva startovací body a v dalších iteračních krocích se využívá bodů spočtených ve dvou předchozích krocích. Zbylé vlastnosti odpovídají metodě Newtonově.

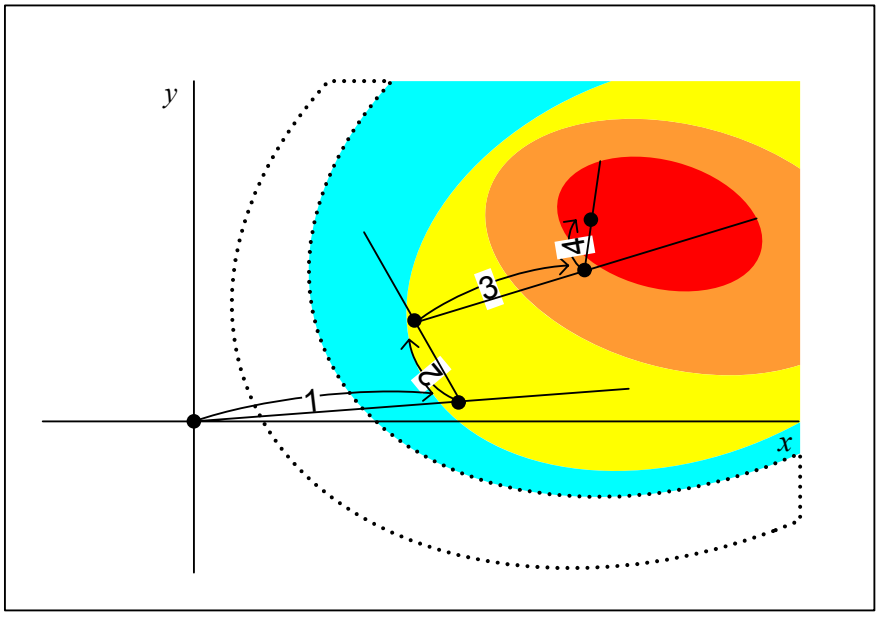
### 2.2.2 Mnohorozměrné gradientní metody

#### *2.2.2.1 Gradientní metoda s dlouhým krokem*

Principiálně je gradientní metoda s dlouhým krokem podobná metodě Gauss – Seidlově. Připusťme, že Gauss – Seidlova metoda při vyhledávání extrému provádí pomyslné řezy účelovou funkcí vždy rovnoběžně s osami souřadného systému, tedy v každém kroku zafixuje všechny proměnné až na jednu, a zde poté vyhledá podextrém. Metoda gradientní s dlouhým krokem provádí obdobně řezy plochou, ale tentokráte bez fixace proměnných.

Provádění řezů bude patrnější z postupu fungování metody pro dvou-dimenzionální úlohu. Metoda, obdobně jako ostatní, začíná ze startovacího bodu $S[x_{S1}, x_{S2}]$. Pro ukončení výpočtu je za potřebí parametr $\varepsilon$, jež obdobně jako v jiných metodách ukončí

algoritmus v okamžiku, kdy vzdálenost bodů v současné a předchozí iteraci klesne pod tuto zadanou hodnotu. Po započetí výpočtu algoritmu je nutné vytvořit novou funkci $g(\lambda)$, na což je zapotřebí provést derivaci zadané účelové funkce $f(x_1, x_2)$, teprve poté začíná samotný iterační cyklus. Nová funkce je definována takto $g(\lambda) = f((x_1, x_2) + \lambda(grad(f(x_1, x_2))))$. Od tohoto okamžiku začíná iterativní výpočet dosazením bodu $S[x_{S1}, x_{S2}]$ do nové funkce. V dalším kroku se nalezne hodnota $\lambda$, tedy extrém jednorozměrné funkce, z rovnice $\frac{dg}{d\lambda} = 0$. Výstupním bodem z iterace je bod $V_1[x_{v_1 1}, x_{v_1 2}] = S[x_{S1}, x_{S2}] + \lambda(grad(f(x_1, x_2)))$ kdy hodnotu gradientu získáme dosazením $S[x_{S1}, x_{S2}]$.




Obrázek 18 Gradientní metoda s dlouhým krokem

Rozšíření na více dimenzí nepřináší žádné komplikace a je zcela triviální. Důležitější je skutečnost, že metoda opět nerozlišuje mezi maximem a minimem funkce a o podobě nalezeného extrému je nutné až dodatečně rozhodnout. Ukončení algoritmu je opět vhodné řešit jak podmínkou rozdílu vzdáleností při úspěšném hledání, tak množstvím provedených iterací.

Algoritmus

$n \in \overline{N}^{+}$, $V_0 = S$

1. Definice účelové funkce $f(x_1,..,x_n)$
2. Volba startovacího bodu $S[x_{S1},..,x_{Sn}]$
3. Volba $\varepsilon$
4. $g(\lambda) = f((x_1,..,x_n) + \lambda(grad(f(x_1,..,x_n))))$; i=1
5. Dosazení do funkce $g(\lambda)$ bod $V_{i-1}$
6. $\frac{dg}{d\lambda} = 0 \Rightarrow \lambda$
7. $V_i[x_{v_i1},..,x_{v_in}] = V_{i-1}[x_{v_{i-1}1},..,x_{v_{i-1}n}] + \lambda(grad(f(x_1,..,x_n)))$
8. Je-li $|V_i - V_{i-1}| < \varepsilon$ pak konec
9. $i++$; skok na 5

Gradientní metoda s krátkým krokem se od této liší pouze v tom, že hodnota $\lambda$ je fixní, na počátku algoritmu nastavitelný parametr.

#### *2.2.2.2 Další mnohorozměrné gradientní metody*

Optimalizačních postupů je samozřejmě mnohem více než tato práce zachycuje. K výčtu dalších metod například patří metody kvazinewtonské jež vychází z aproximace účelové funkce $f(x_1,..,x_n)$ kvadratickou formou. Dalším příkladem je metoda konjugovaných gradientů nebo Metoda paralelních tečen. V případě omezujících podmínek lze jmenovat metodu projekce gradientu, která řeší úlohy s lineárním omezením typu nerovnost.

## 2.3 Metody náhodného vyhledávání

Mezi metody náhodného prohledávání patří jednak metody jejichž strategie jsou velice jednoduché a předem definované, nicméně odlišné od gradientních či komparačních postupů, tak metody adaptační či s prvky umělé inteligence. V jednotlivých krocích iterace jsou výsledky získávány alternativními způsoby. Příkladem snadné metody může být sice

nevhodné, leč k užití možné zcela náhodné prohledávání přípustného prostoru. Další možností jsou algoritmy jako simulované ochlazování, či genetické algoritmy. O těchto algoritmech, stejně jako o všech ostatních lze říci, že mají mnohé výhody ale také nevýhody. Je tedy spíše otázkou zkušeností nebo pokusů jaký postup na jakou úlohu je vhodné použít případně jaké modifikace v algoritmu provést. Spektrum optimalizačních úloh je natolik široké, že každý z postupů nepochybně najde své uplatnění.

# II. PRAKTICKÁ ČÁST

# 3 PROGRAMOVÁ REALIZACE

Cílem programové realizace bylo navrhnout a vytvořit vlastní realizaci vybraných algoritmů iteračních metod. Prostředím k realizaci byl vybrán software s názvem Mathematica. Důvodů proč nebylo využito klasických programovacích jazyků jako například C , C++ nebo JAVA ač by právě Javovské aplety mohl být přímo prezentován na webových stránkách je několik. Záměrem bylo vytvořit jednotlivé metody co možná nejuniverzálnější, proto byla například snaha vyhnout se omezenému počtu proměnných funkce. Definice funkce s neomezeným počtem proměnných v „klasických“ programovacích jazycích by znamenala vynaložit nemalé úsilí, tedy rozsáhlý zdrojový kód, na analýzu každé jednotlivé funkce. Dalším závažným důvodem je definice samotných funkcí. Ačkoli existují mnohé knihovny s matematickými funkcemi přeci jen v případě zcela obecných funkcí nepokryjí celé spektrum možností. Programový produkt Mathematica je svou podstatou vytvořen pro matematické výpočty a veškerou podporu má již v sobě implementovánu. Jeví se tedy zcela zbytečné vynakládat úsilí na realizaci něčeho již hotového. Nevýhodou volby je dosti pomalý běh. Mathematica je interpret jazyka a jako takový se i chová.

Při výběru metod k interpretaci padla volba na Box – Wilsonovu metodu, metodu flexibilního simplexu, Gauss – Seidlovu metodu a metodu gradientu s dlouhým krokem. Box – Wilsonova metoda je typický zástupce komparativních metod. Je snadno pochopitelná a velice názorná. Metoda flexibilního simplexu je pravděpodobně nejlepší komparativní metodou vůbec. Realizace klasické simplexové metody není sice specielně naprogramována, ale správnou volbou parametrů lze docílit shodného chování i u metody flexibilního simlexu. Gaus – Seidlova metoda je zajímavá svou vnitřní modifikovatelností a metoda gradientní s dlouhým krokem je zástupce iteračních gradientních metod.

Pro odzkoušení metod je vhodné seznámit se s nastavováním potřebných parametrů jež je popsáno v příloze P I.

## 3.1 Box – Wilsonova metoda

Tato metoda byla naprogramována tak, aby vždy hledala maximum funkce. Tato vlastnost vychází z původního návrhu metody. Pro vyhledávání minima účelové funkce je nutno tuto přizpůsobit vynásobením -1. Metoda funguje pro neomezený počet proměnných.

Zdrojový kód viz. Příloha PII.

## 3.2 Metoda flexibilního simplexu

Podobně jako metoda Box –Wilson i metoda flexibilního simplexu vyhledává fixně jeden extrém, tentokráte však jde o minimum. Přizpůsobení optimalizované funkce lze opět provést vynásobením -1.

Při vhodné volbě parametrů lze metodu z flexibilní přeměnit téměř na obyčejnou simplexovou metodu. Důvod proč toho nelze dosáhnout přesně je popsán o odstavec níže. Aby bylo této zajímavé změny docíleno je za potřebí změnit následující parametry na tyto konkrétní hodnoty: $\gamma = 1, \beta_1 = 1, \beta_2 = 1$. Parametr $\alpha$ ovlivňuje zrcadlení bodu a pro klasický případ by měl mít hodnotu $\alpha = 1$, nicméně jeho změnou lze docílit odlišného chování metody. V případě, že tento parametr nastavíme na hodnotu $\alpha > 1$ bude reflektovaný bod ve větší vzdálenosti a v konečném důsledku se bude velikost simplexu neustále zvětšovat, což nepřináší valný užitek. Bude-li hodnota $\alpha < 1$ bude efekt naprosto opačný a velikost simplexu se v každém kroku zmenší. Zmenšování simplexu lze potom chápat jako zpřesňování výpočtu a je tedy svým způsobem zajímavé. Rizikem je příliš vzdálený startovací bod od extrému, v tom případě se simplex může zmenšit natolik, že algoritmus bude ukončen dříve, vysokým počtem iterací, než se některý z vrcholů simplexu dostane k extrému.

Zvláštností naimplementovaného algoritmu je prvotní vytvoření simplexu. Ve většině případů se simplex konstruuje jako pravidelný útvar. Konstrukce pravidelného simplexu je popsána v kapitole 2.1.2.4. Proti tomu v naprogramovaném algoritmu je využito náhodné konstrukce kolem startovacího bodu. Fakticky je konstrukce provedena jako součet startovacího bodu zmenšeného o $\gamma$ a náhodného čísla, jež je generováno v rozsahu hodnot 0 až $2 \cdot \gamma$ s přesností na čtyři desetinná místa.Samozřejmě se provede

kontrola, zda jde o platný simplex. Námitkou k takto pojaté konstrukci může být závěr, že vygenerovaný simplex může být velice nepravidelný a preferovat či potlačovat tak některý ze směrů, v nejhorším případě potlačí ten správný. Ano tato námitka je opodstatněná a lze ji považovat i za správnou. Nicméně připusťme, že o zadané účelové funkci nemáme žádné bližší údaje. Pak lze jen velice těžko odhadnout jak startovací bot tak také délku hrany simplexu. Za tohoto předpokladu je i z uživatelovy strany spíše dílem náhody volba těchto parametrů. Náhodná volba vrcholů simplexu ušetří starost s dálkou počáteční hrany a umožní úplné vypuštění nastavení tohoto parametru. Další důvod, proč není náhodné vygenerování simplexu zcela zavrhnutelné je skutečnost právě flexibilního přizpůsobení tvaru simplexu průběhu účelové funkce. Jestliže je vygenerována velice nevhodná podoba prvotního simplexu, je tento již v několika málo iteracích přetvořen do mnohem lepší podoby a ztrátu tak brzy vyrovná.

Náhodnost generování simplexu neumožňuje přesně vytvořit z flexibilní metody metodu obyčejného simplexu, neboť ta již symetrickou podobu útvaru vyžaduje.

Metoda funguje pro neomezený počet proměnných.

Zdrojový kód viz. Příloha P III.

## 3.3 Gauss – Seidlova metoda

Metoda je naprogramována k hledání univerzálního extrému funkce více proměnných. O nalezeném extrému není rozhodnuto zda se jedná o maximum nebo minimum. Tohoto zpřesnění lze dosáhnout dosazením hodnoty z blízkého okolí a porovnáním jednotlivých výsledků.

Zdrojový kód viz. Příloha P IV.

## 3.4 Gradientní metoda s dlouhým krokem

Jediná metoda pro kterou není naprogramována možnost zcela univerzálního množství dimenzí účelové funkce bez zásahu do vnitřního kódu. Důvodem je nutnost vygenerování nové funkce z funkce původní, tedy pro obecný případ ne zcela triviální operace. Limitním počtem proměnných je v původní verzi programu deset. V případě nutnosti vyššího počtu dimenzí účelové funkce je za potřebí rozšířit abecedu pro nově generovanou funkci jež je použita pro označení proměnných ve vygenerované funkci. Tato změna se v programu týká

dvou konkrétních proměnných a to proměnné s názvem abeceda a abecedafce. Po rozšíření těchto proměnných je opět možno program využít na více-dimenzionální problémy.

Zdroj Zdrojový kód viz. Příloha P V.

## 3.5 Testování naprogramovaných metod

S cílem ověřit správnost implementace jednotlivých metod bylo provedeno testování na několika příkladech kriteriálních funkcí.

Funkce: $f(x) = (x-a)^2 + b$

Extrém funkce v bodě $K[a]$ a hodnotou $E = b$

| $f(x) = (x-a)^2 + b$ | | | | | |
|---|---|---|---|---|---|
| | Start | Box – Wilson $\Delta = 0{,}1$ | | Simplex $\varepsilon = 0{,}1$ | |
| | | $f(x)$ | $-f(x)$ | $f(x)$ | $-f(x)$ |
| $a = 3, b = 5$ $K[3], E = 5$ | $S[3]$ | Nenalezeno | $V[3], E = -5$ | $V[3{,}16], E = 5{,}03$ | Nenalezeno |
| | $S[10]$ | Nenalezeno | $V[3], E = -5$ | $V[2{,}91], E = 5{,}01$ | Nenalezeno |
| | $S[-7]$ | Nenalezeno | $V[3], E = -5$ | $V[2{,}95], E = 5{,}01$ | Nenalezeno |

Tabulka 1 Funkce $f(x) = (x-a)^2 + b$, Box – Wilson a Simplex

| $f(x)=(x-a)^2+b$ | | | | | |
|---|---|---|---|---|---|
| | Start | Gauss – Siedl $\varepsilon = 0{,}1$ | | Gradient s dlouhým krokem $\varepsilon = 0{,}1$ | |
| | | $f(x)$ | $-f(x)$ | $f(x)$ | $-f(x)$ |
| $a=3, b=5$ $K[3], E=5$ | $S[3]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |
| | $S[10]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |
| | $S[-7]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |

Tabulka 2 Funkce $f(x)=(x-a)^2+b$, Gauss – Seidl a Gradient

Funkce: $f(x)=(x-a)^3+b$

Extrém funkce neexistuje, v bodě $K[a]$ je pouze inflexní bod

| $f(x)=(x-a)^3+b$ | | | | | |
|---|---|---|---|---|---|
| | Start | Box – Wilson $\Delta = 0{,}1$ | | Simplex $\varepsilon = 0{,}1$ | |
| | | $f(x)$ | $-f(x)$ | $f(x)$ | $-f(x)$ |
| $a=3, b=5$ Extrém neexistuje | $S[3]$ | Nenalezeno | Nenalezeno | Nenalezeno | Nenalezeno |
| | $S[10]$ | Nenalezeno | Nenalezeno | Nenalezeno | Nenalezeno |
| | $S[-7]$ | Nenalezeno | Nenalezeno | Nenalezeno | Nenalezeno |

Tabulka 3 Funkce $f(x)=(x-a)^3+b$, Box – Wilson a Simplex

| $f(x)=(x-a)^3+b$ | | | | | |
|---|---|---|---|---|---|
| | Start | Gauss – Siedl $\varepsilon=0{,}1$ | | Gradient s dlouhým krokem $\varepsilon=0{,}1$ | |
| | | $f(x)$ | $-f(x)$ | $f(x)$ | $-f(x)$ |
| $a=3,b=5$ Extrém neexistuje | $S[3]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |
| | $S[10]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |
| | $S[-7]$ | $V[3], E=5$ | $V[3], E=-5$ | $V[3], E=5$ | $V[3], E=-5$ |

Tabulka 4 Funkce $f(x)=(x-a)^3+b$, Gauss – Seidl a Gradient

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

Extrém funkce v bodě $K[a,b]$ a hodnotou $E=1$

| $f(x,y)=e^{-(x-a)^2-(y-b)^2}$ | | | | | |
|---|---|---|---|---|---|
| | Start | Box – Wilson $\Delta=0{,}1$ | | Simplex $\varepsilon=0{,}1$ | |
| | | $f(x,y)$ | $-f(x,y)$ | $f(x,y)$ | $-f(x,y)$ |
| $a=1,b=2$ $K[1,2], E=1$ | $S[1,2]$ | $V[1,2], E=1$ | Nenalezeno | $V[0.07,-3.96]$ $E=1{,}53758\cdot10^{-16}$ | V[1.2,2.3] E=-0,81 |
| | $S[4,4]$ | $V[1,2], E=1$ | Nenalezeno | $V[-9.59,-3.12]$ $E=8{,}91\cdot10^{-68}$ | V[2.5,5.23] E=-2,84.$10^{-6}$ |
| | $S[-5,-5]$ | $V[1,2], E=1$ | Nenalezeno | $V[7.04,6.09]$ $E=7{,}34\cdot10^{-24}$ | V[-4.4,-2.0] E=-6.4.$10^{-20}$ |

Tabulka 5 Funkce $f(x,y)=e^{-(x-a)^2-(y-b)^2}$, Box – Wilson a Simplex

| $f(x,y)=e^{-(x-a)^2-(y-b)^2}$ | | | | | |
|---|---|---|---|---|---|
| | Start | Gauss – Siedl $\varepsilon = 0{,}1$ | | Gradient s dlouhým krokem $\varepsilon = 0{,}1$ | |
| | | $f(x,y)$ | $-f(x,y)$ | $f(x,y)$ | $-f(x,y)$ |
| $a=3, b=5$ $K[1,2], E=1$ | $S[1,2]$ | $V[1,2], E=1$ | $V[1,2], E=-1$ | $V[1,2], E=1$ | $V[1,2], E=-1$ |
| | $S[4,4]$ | Nesmyslný výsledek | Nesmyslný výsledek | Nesmyslný výsledek | Nesmyslný výsledek |
| | $S[-5,-5]$ | Nesmyslný výsledek | Nesmyslný výsledek | Nesmyslný výsledek | Nesmyslný výsledek |

Tabulka 6 Funkce $f(x,y)=e^{-(x-a)^2-(y-b)^2}$ , Gauss – Seidl a Gradient

### 3.5.1 Zhodnocení testů

Výsledky testů mají prověřit naimplementované algoritmy a upozornit na záludnosti jednotlivých metod. Jako ukázky testování byly vybrány jen některé, u nichž lze snadno demonstrovat již zmíněné specifika jednotlivých metod.

Test funkce $f(x)=(x-a)^2+b$ demonstruje jednak schopnost metod pracovat i v případě jednodimenzionálních úloh, ale zejména upozorňuje na typ nalezeného extrému. U metody Box – Wilson vyhledává metoda maximum funkce a proto nenalezne řešení kvadratické funkce která má minimum. Metoda flexibilního simplexu naopak vyhledává minimum a proto nenalezne extrém u funkce jež má pouze maximu.

Na funkci $f(x)=(x-a)^3+b$ se ukazuje záludnost gradientních metod, jež flexibilní bod vyhodnotí nesprávně jako extrém.

Funkce $f(x,y)=e^{-(x-a)^2-(y-b)^2}$ je z pohledu dosažených výsledků velice kritická. Bez problémů byl nalezen extrém pouze metodou Box – Wilson. U simplexové metody se projevila ukončovací podmínka, jež výpočet přeruší v okamžiku, kdy rozdíl nejmenší

největší funkční hodnoty klesne pod zadanou mez. Vzhledem k vlastnostem funkce při zvoleném $\varepsilon$ je tato podmínka dosažena velice brzy a výsledky tak jsou velice zavádějící. V případě obou gradientních metod jsou nesmyslné výsledky dány skutečností, že se nepodařilo nalézt kořeny derivované funkce.

Cílem provedených testů bylo také ukázat, že ačkoli jsou samotné nástroje na výpočet extrému velice silné a užitečné, je za potřebí velmi dobře zvážit a kvalifikovaně interpretovat výsledek kterého se podařilo s jejich pomocí dosáhnout.

# 4 WWW STRÁNKY

# ZÁVĚR

S optimalizačními problémy se v každodenním životě setkáváme neustále. V průmyslových aplikacích se disciplína optimalizace uplatňuje jak v projektové tak provozní oblasti. K řešení extremalizačních úloh existuje velké množství postupů, které jsou dle základních charakteristik členěny do několika skupin. Tato práce se zabývá jednou z těchto skupin algoritmů a to iteračními optimalizačními metodami.

Neustálý rozvoj výpočetní techniky znamená také nárůst výpočetní výkonnosti. Tento zmiňovaný nárůst umožňuje širší uplatnění právě iteračních metod, které jsou svojí podstatou naprosto nevhodné pro ruční výpočet ale naopak téměř výlučně připravené pro zpracování na počítači. Práce se snaží zájemcům představit iterační metody optimalizace, demonstrovat jejich sílu při řešení praktických problémů, ale také upozornit na důležitost správné interpretace dosažených výsledků. Naprogramované algoritmy, jež jsou také součástí práce mohou posloužit jako prostředek při řešení konkrétního optimalizačního problému.

Snahou při zpracování tohoto tématu bylo také připravit pro zájemce podkladový materiál ke studiu optimalizace, jež je vyučován na fakultě aplikované informatiky univerzity Tomáše Bati ve Zlíně. Do jaké míry se to ve skutečnosti podařilo se zjistí až z případných ohlasů studentů.

# SEZNAM POUŽITÉ LITERATURY

[1] Ottova encyklopedie nové doby, heslo na URL: http://coto.je/

[2] Encyklopedie Universum, heslo na URL: http://coto.je/

[3] Vítečková M.; Jedlička D., Statická optimalizace systémů, 2003, URL: http://www.fs.vsb.cz/books/StatickaOptimalizace/

[4] Koshel J. R., Enhancement of the downhill simplex method of optimization, Optical Society of America, 2002

[5] Raida Z., Optimalizace v elektrotechnice, URL: http://www.urel.feec.vutbr.cz/~raida/optimalizace/index.htm

[6] Wikipedie – otevřená encyklopedie, URL: http://cs.wikipedia.org/wiki/Optimalizace

[7] Dupačová J.; Lachout P., Úvod do optimalizace, 2006, URL: http://www.karlin.mff.cuni.cz/~lachout/Vyuka/U-Optima/U-opt-text.pdf

[8] Dupačová J.; Lachout P.; Úvod do optimalizace, 2005, URL: http://www.karlin.mff.cuni.cz/~lachout/Vyuka/Optima1/Opt-text-051021.pdf

[9] Prokop R., přednáškové slajdy k předmětu optimalizace

[10] Box G. E. P.; Wilson K. B., On the Experimental Attainment of Optimum Conditions, Journal of the Royal Statistical Society, Series B, Vol. 13, pp. 1-45. 1991

[11] Chelouah R.; Siarry P., Genetic and Nelder – Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operation Research, 2002

# SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

| | |
|---|---|
| $\overline{N}^{+}$ | Množina přirozených čísel bez nuly. Tedy čísla 1,2,3,.. |
| $\overline{N}$ | Množina přirozených čísel |
| $f : X \to R$ | Funkce je zobrazení z X do R, kde R je množina reálných hodnot |
| $f^{(i)}(x_0)$ | i-tá derivace funkce v bodě $x_0$ |
| $x^*$ | Bod prohlášený za extrém |

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# SEZNAM PŘÍLOH

# PŘÍLOHA P I: NASTAVENÍ PROMĚNNÝCH PROGRAMU

Správná funkce programů v prostředí Mathematica vyžaduje dodržení správné syntaxe tohoto programu. Nejvíce se tato vlastnost při zadání samotné účelové funkce, neboť právě její zadání se zvyklostem vzdaluje nejvíce.

Zadání účelové funkce:

Příklad: $f[\{x\_, y\_\}] := x^2 - y^2$;

$f$ je jméno funkce. V realizacích všech programů je toto jméno shodné a také je s tímto jménem funkce počítáno v dalším výpočtu. Jméno se tedy nemůže měnit

$[\{\ \}]$ proměnné funkcí se zapisují do dvojice závorek, přesněji se do hranatých závorek zapisuje seznam proměnných jež indikují závorky složené

$x, y$ jsou proměnné definované funkce, pro zápis na levé straně výrazu (uvnitř dvojice závorek) je nutné tyto proměnné opatřit na konci _ ( podtržítkem ). Uvnitř závorek tedy MUSÍ být napsáno proměnná_ . Tento zápis dává matematice na vědomí, že za tyto hodnoty bude později dosazovat hodnoty,

:= je přiřazení a je povinné za tímto přiřazením již dále lze zapsat funkci dle zvyklostí ukončenou ; ( středníkem ).

Pro zápis funkcí lze využít znaků celé abecedy, nicméně některé znaky jako např. $\pi, e$ a podobně mohou a ve většině případů i mají v zápisu funkcí specielní význam na což je potřeba si dát pozor. Další znaky například $\alpha, \beta, \gamma, \Delta, \varepsilon$ jsou využity v programu jako proměnné a proto se také jejich využití při zápisu funkce silně nedoporučuje. Je vhodné využít zvyklostí pro zápis funkcí, tedy užívat proměnné x, y, z apod. V případě, že je funkce velkého počtu proměnných, pak je možné proměnné číslovat například x1, x2, x3, .. , xn. V takovém případě je pro přehlednost velice vhodné využívat znaménko krát ve všech případech násobení aby nedošlo ke kolizi jako například x2x3. Nelze určit zda jde o násobení či označení proměnné ( $x \cdot 2 \cdot x3$ nebo $x2 \cdot x3$ což je podstatný rozdíl).

Další příklady zápisu funkcí :

$f[\{x1\_, x2\_, x3\_, x4\}] := x1 + x2 + x3 \times x4$;

$f[\{x\_, y\_, z\_\}] := 2\pi \times x + yz$;

Zadání dimenze úlohy:

Příklad: dimenze = 2;

Parametr dimenze udává rozměr úlohy a musí odpovídat počtu proměnných funkce.

Další příklady významu proměnné dimenze:

$f[\{x\_, y\_\}] := x^2 - y^2$; dimenze = 2;

$f[\{x1\_, x2\_, x3\_, x4\}] := x1 + x2 + x3 \times x4$; dimenze = 4;

$f[\{x\_, y\_, z\_\}] := 2\pi \times x + yz$; dimenze = 3;

Zadání bodu startu:

Příklad: stred=Table[0,{dimenze+1}];

Takto zadaný bod představuje startovací bod v počátku souřadnic. Za předpokladu, že se využije tohoto zápisu a místo nuly se zadá jiná hodnota, bude mít startovací bod vždy souřadnice [hodnota, .. ,hodnota] to znamená, že všechny budou stejné a rovny zadanému číslu. Tento zápis je velice výhodný právě pro zadání středu souřadnicového systému což bývá velice typické pro většinu úloh u nichž nemáme dostatek informací o průběhu účelové funkce. Požadujeme-li zadat hodnotu zcela obecnou, je nutné změnit zápis na takto:

stred={1,5,7,0} ale pozor! takto zadaný střed bude v pozdějším výpočtu interpretován jako vrchol o TŘECH dimenzích tedy [1,5,7]. Poslední hodnota 0 bude v nejbližším okamžiku nahrazena funkční hodnotou v tomto bodě. V podstatě lze na toto místo dát jakoukoli číselnou hodnotu, ale MUSÍ tam být. Z tohoto důvodu je i v původním zápise generováno o jednu dimenzi více.

Další příklady zápisu startovacího bodu:

$f[\{x\_, y\_\}] := x^2 - y^2$; dimenze = 2; stred=Table[0,{dimenze+1}];

$f[\{x\_, y\_\}] := x^2 - y^2$; dimenze = 2; stred={-1,5,2,6};

$f[\{x1\_, x2\_, x3\_, x4\}] := x1 + x2 + x3 \times x4$; dimenze = 4; stred=Table[4,{dimenze+1}];

$f[\{x1\_, x2\_, x3\_, x4\}] := x1 + x2 + x3 \times x4$; dimenze = 4; stred={5,7,9,21,0};

Parametr $\Delta$ pouze Box – Wilson

Příklad: $\Delta = 0.1$;

Parametr definuje délku hrany čtverce ( či krychle apod. v dalších dimenzích ) od středu je vertikální a horizontální vzdálenost vrcholů tedy $\frac{\Delta}{2}$.

Parametr $\varepsilon$ flexibilní simplex, Gauss –Seidl, gradientní metoda s dlouhým krokem

Příklad: $\varepsilon = 0.5$;

Parametr sloužící k ukončení iteračních cyklů. Vyhodnocuje se, zda je rozdíl vzdáleností bodů menší než tento parametr.

Parametry $\alpha, \beta_1, \beta_2, \gamma$ pouze flexibilní simplex

Příklad: $\alpha = 1; \beta 1 = \beta 2 = 0.5; \gamma = 2.0;$

Parametry k nastavení parametrů reflexe, kontrakce a expanze. Parametr $\beta$ bez indexu v základním algoritmu je mírně matoucí, má pouze zdůraznit skutečnost, že v podání Nelder – Meadeho jsou parametry $\beta 1$ a $\beta 2$ stejné. Ve výpočtu se s neindexovaným parametrem nikde nepočítá ☺.

Parametr nouzového zastavení:

Příklad: kritickystop=100;

Parametr omezuje maximální počet iterací jež budou v programu provedeny.

# PŘÍLOHA P II: BOX – WILSONOVA METODA

```
f[{x_}] := -((x - 3)^2 + 5);
dimenze = 1;
Δ = 0.1;
stred = Table[-7, {dimenze + 1}];
kritickystop = 1000;

delta = Δ/2;
stred[[dimenze + 1]] = f[Take[stred, dimenze]];
box = Table[0, {2^dimenze}, {dimenze + 1}];

preteceni = 0;
While[preteceni < kritickystop,
  createbox = Table[1, {dimenze}];
  For[i = 1, i ≤ 2^dimenze, i++,
   For[j = 1, j ≤ dimenze, j++,
    box[[i, j]] = delta × createbox[[j]] + stred[[j]];
   ];
   box[[i, dimenze + 1]] = f[Take[box[[i]], dimenze]];
   If[i == 2^dimenze, fx = 0, fx = 1];
   While[fx > 0,
    If[ createbox[[fx]] == 1, createbox[[fx]] = -1; fx = 0,
      createbox[[fx]] = 1; fx = fx + 1];
   ];
  ];

  i = Ordering[-Drop[box, None, dimenze]][[1]];

  If[stred[[dimenze + 1]] < box[[i, dimenze + 1]],
   stred = box[[i]], preteceni = kritickystop + 10];
  preteceni++;
 ];

If[preteceni < kritickystop + 5, Print["Nepodařilo se najít řešení"],
  Print["Řešením je bod:", Take[stred, dimenze],
   " s hodnotou extrému: ", Last[stred], " a s délkou hrany: ", Δ]];
Clear[f];
```

# PŘÍLOHA P III: FLEXIBILNÍ SIMPLEXOVÁ METODA

```
f[{x_}] := (x - 3)^2 + 5;
dimenze = 1;
ϵ = 0.1;
kritickystop = 100;
stred = Table[-7, {dimenze + 1}];
α = 1;
β1 = β2 = β = 0.5;
γ = 2.0;

SeedRandom[];
simplex = Table[0, {dimenze + 1}, {dimenze + 1}];

kontrolasimplexu = "False";
While[kontrolasimplexu == "False",
  For[i = 1, i ≤ dimenze + 1, i++,
   For[j = 1, j ≤ dimenze, j++,
    simplex[[i, j]] = stred[[j]] + Random[Real, {0, 2×γ}, 4] - γ
   ];
   simplex[[i, dimenze + 1]] = f[Take[simplex[[i]], dimenze]];
  ];
  sim = Flatten[Drop[simplex, None, {dimenze + 1}]];
  If[Length[sim] == Length[Union[sim]],
   kontrolasimplexu = "True", kontrolasimplexu = "False"];
 ];

simplex = Sort[simplex, OrderedQ[{Take[#1, -1], Take[#2, -1]}] &];
preteceni = 0;
While[preteceni < kritickystop,
 For[j = 1, j ≤ dimenze, j++, stred[[j]] = ∑_{i=1}^{dimenze} simplex[[i, j]] / dimenze];
 stred[[dimenze + 1]] = f[Take[stred, dimenze]];
 obraz = (1 + α) × stred - α × simplex[[dimenze + 1]];
 obraz[[dimenze + 1]] = f[Take[obraz, dimenze]];
```

```
If[simplex[[1, dimenze + 1]] ≤ obraz[[dimenze + 1]] <
     simplex[[dimenze + 1, dimenze + 1]], simplex[[dimenze + 1]] = obraz,
    If[obraz[[dimenze + 1]] < simplex[[1, dimenze + 1]],
     expanze = (1 - γ) × stred - γ × obraz;
     expanze[[dimenze + 1]] = f[Take[expanze, dimenze]];
     If[expanze[[dimenze + 1]] < simplex[[1, dimenze + 1]],
      simplex[[dimenze + 1]] = expanze, simplex[[dimenze + 1]] = obraz],
     kontrakce1d = (1 - β1) × stred + β1 × simplex[[dimenze + 1]];
     kontrakce1d[[dimenze + 1]] = f[Take[kontrakce1d, dimenze]];
     If[kontrakce1d[[dimenze + 1]] < simplex[[dimenze + 1, dimenze + 1]],
      simplex[[dimenze + 1]] = kontrakce1d, For[i = 2, i ≤ dimenze + 1,
       i++, simplex[[i]] = (1 - β2) simplex[[1]] + β2simplex[[i]];
       simplex[[i, dimenze + 1]] = f[Take[simplex[[i]], dimenze]]]]]];
  simplex = Sort[simplex, OrderedQ[{Take[#1, -1], Take[#2, -1]}] &];

  preteceni++;
  If[Abs[simplex[[1, dimenze + 1]] - simplex[[dimenze + 1, dimenze + 1]]] < ϵ,
    preteceni = kritickystop + 10,];
  ];
For[j = 1, j ≤ dimenze, j++, stred[[j]] = ∑_{i=1}^{dimenze} simplex[[i, j]] / dimenze];
stred[[dimenze + 1]] = f[Take[stred, dimenze]];
If[preteceni < kritickystop + 5, Print["Nepodařilo se najít řešení"],
  Print["Řešením je bod:", Take[stred, dimenze],
   " s hodnotou extrému: ", Last[stred], " a s krokem: ", ϵ]];
Clear[f];
```

# PŘÍLOHA P IV: GAUSS – SEIDLOVA METODA

```
f[{x_}] := (x - 3)^2 + 5;
dimenze = 1;
ϵ = 0.1;
stred = Table[10, {dimenze + 1}];
kritickystop = 100;

stred[[dimenze + 1]] = f[Take[stred, dimenze]];
preteceni = 0;
ukonceni = 0;

While[preteceni < kritickystop,
  For[i = 1, i ≤ dimenze, i++, stred[[i]] = stred[[i]] + ξ;
    g[{ξ_}] = f[Take[stred, dimenze]]; h[{ξ_}] := D[g[{ξ}], ξ];
    koren = Replace[koren = x, FindRoot[h[{x}] == 0, {x, 0}]];
    stred[[i]] = stred[[i]] + koren - ξ;
    ukonceni = ukonceni + koren^2;
    If[ukonceni < ϵ, preteceni = kritickystop + 10, preteceni++];

    stred[[dimenze + 1]] = f[Take[stred, dimenze]];
    ukonceni = 0;
   ];
 ];
If[preteceni < kritickystop + 5, Print["Nepodařilo se najít řešení"],
  Print["Řešením je bod:", Take[stred, dimenze], " s hodnotou extrému: ",
   Last[stred], " a rozdílovou hodnotou: ", ϵ]];
Clear[f]; Clear[g];
Clear[h];
```

# PŘÍLOHA P V: GRADIENTNÍ METODA S DLOUHÝM KROKEM

```
f[{x_}] := -((x - 3)^2 + 5);
dimenze = 1;
ϵ = 0.1;
stred = Table[-7, {dimenze + 1}];
kritickystop = 100;

stred[[dimenze+1]]=f[Take[stred,dimenze]];
abeceda={ττ1,ττ2,ττ3,ττ4,ττ5,ττ6,ττ7,ττ8,ττ9,ττ10};
abecedafce={ττ1_,ττ2_,ττ3_,ττ4_,ττ5_,ττ6_,
      ττ7_,ττ8_,ττ9_,ττ10_};
promenne=Take[abeceda,dimenze];
promennefce=Take[abecedafce,dimenze];
prirustek=Table[0,{dimenze}];
prirustekvbode=Table[0,{dimenze}];
stredfce=Table[0,{dimenze}];
stredfce=Take[stred,dimenze];
For[i=1,i≤dimenze,i++,
    prirustek[[i]]=D[f[promenne],promenne[[i]]];
    ];
preteceni=0;
While[preteceni<kritickystop,
    For[i=1,i≤dimenze,i++,
      g[promennefce]=prirustek[[i]];
      prirustekvbode[[i]]=g[Take[stred,dimenze]];
      ];
    stredfce=stredfce+λ×prirustekvbode;
    g[{λ_}]=f[stredfce];
    koren=Replace[koren=x,FindRoot[D[g[{x}],x]==0,{x,0}]];
    For[i=1,i≤dimenze,
      i++,stred[[i]]=stred[[i]]+koren×prirustekvbode[[i]]];
    ukonceni=stred[[dimenze+1]];
    stred[[dimenze+1]]=f[Take[stred,dimenze]];
    stredfce=Take[stred,dimenze];
    If[Abs[ukonceni-stred[[dimenze+1]]]<ϵ,
      preteceni=kritickystop+10,preteceni++];
    ];
If[preteceni<kritickystop+5,Print["Nepodařilo se najít
řešení"],Print["Řešením je bod:",Take[stred,dimenze],
      " s hodnotou extrému: ",Last[stred]," a rozdílovou
hodnotou: ",ϵ]];
```

# PŘÍLOHA P VI: VZOROVÉ ZADÁNÍ PROTOKOLU

Stanovte extrém funkce $f(x, y) = -x^2 - y^2$

1. Analytickým způsobem
2. Box – Wilsonovou metodou ze startovacího bodu [4,3] a délkou hrany 2
3. V kolika bodech mimo středu je potřeba vyčíslit účelovou funkci v jedné iteraci?
4. Za jakých podmínek umožní metoda Box – Wilson nalezení minima funkce?
5. Lze něco říci o přesnosti dosaženého výsledku?