UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

BAKALÁŘSKÁ PRÁCE

Metoda konjugovaných gradientů

Vedoucí bakalářské práce:
RNDr. Jitka Machalová, Ph.D.
Rok odevzdání: 2012

Vypracovala:
Kateřina Dudková
MAP, III. ročník

Prohlášení

Prohlašuji, že jsem vytvořila tuto bakalářskou práci samostatně za vedení RNDr. Jitky Machalové, Ph.D. a že jsem v seznamu použité literatury uvedla všechny zdroje použité při zpracování práce.

V Olomouci dne 20. dubna 2012

Poděkování

Ráda bych na tomto místě poděkovala vedoucí bakalářské práce RNDr. Jitce Machalové, Ph.D. za obětavou spolupráci i za čas, který mi věnovala při konzultacích. Dále bych chtěla poděkovat své rodině za její podporu při studiu.

Obsah

Použité symboly

$\mathbb{R}$	obor reálných čísel
$\mathbb{R}^n$	vektorový prostor dimenze n nad $\mathbb{R}$
$\mathbb{R}^{n\times m}$	množina matic typu (n, m) nad $\mathbb{R}$
$C^n(X)$	množina všech funkcí, které mají spojité derivace až do řádu n na množině X
$\mathbf{x} = (x_1, \ldots, x_n)^T$	uspořádaná n-tice reálných čísel – vektor
$\mathbf{o} = (0, 0, \ldots, 0)^T$	nulový vektor
$\|\mathbf{x}\|$	norma vektoru
$\{\mathbf{x}^k\}$	posloupnost vektorů
$\mathbf{A} = (a_{ij})_{i,j=1}^n$	reálná čtvercová matice řádu n
$\nabla f(\mathbf{x})$	gradient funkce f v bodě $\mathbf{x}$
$\mathbf{H}(\mathbf{x})$	Hessián funkce f v bodě $\mathbf{x}$

Úvodní slovo

Numerická matematika je široce obsáhlá matematická disciplína, která využívá počítačové výpočetní metody pro řešení velkého množství úloh, jež jsou pro člověka početně velice komplikované. Pro řešení takových problémů používáme tzv. numerické metody. Tyto metody nám přesně popisují řešení dané úlohy, přičemž pro konečný počet vstupních dat je přiřazen konečný počet dat výstupních. Posloupnost kroků vyjadřujících průběh dané úlohy nazýváme algoritmus.

Během svého studia na vysoké škole jsem se seznámila s optimalizačními numerickými metodami, jež mne velice zaujaly. Pro téma své bakalářské práce jsem si vybrala jednu z těchto metod a to Metodu konjugovaných gradientů.

Na začátku práce si uvedeme několik pojmů, jež budeme v textu potřebovat. Dále se krátce seznámíme s problémem optimalizace, řekneme si něco o podmínkách optimality a uvedeme názorný příklad.

Následně si představíme spádové metody, jejich základní myšlenku a obecné schéma. Uvedeme si také Metodu největšího spádu, její algoritmus a využití na příkladech.

Cílem této práce je seznámit čtenáře s Metodou konjugovaných gradientů, jejíž podstatou je nalezení minima dané funkce. Uvedeme si základní vlastnosti, její výhody a rozdíl mezi použitím pro kvadratické a obecné funkce. Poté se zaměříme na sestavení souborů v matematickém prostředí Matlab, jež budou použité k řešení daných úloh. Také si porovnáme přesné řešení s řešením, které vypočítáme pomocí dané numerické metody.

Nakonec aplikujeme Metodu konjugovaných gradientů na konkrétních úlohách a budeme zkoumat průběh výpočtu a přesnost řešení.

1 Přípravná kapitola

Na začátku této práce si uvedeme některé pojmy, které budou nezbytné v následujícím textu.

Definice 1.1. Symetrickou čtvercovou matici $\mathbf{A}$ řádu n nazýváme
(i) *pozitivně definitní*, jestliže $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{o}$ platí $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$,
(ii) *negativně definitní*, jestliže $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{o}$ platí $\mathbf{x}^T\mathbf{A}\mathbf{x} < 0$,
(iii) *pozitivně semidefinitní*, jestliže $\forall \mathbf{x} \in \mathbb{R}^n$ platí $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$,
(iv) *negativně semidefinitní*, jestliže $\forall \mathbf{x} \in \mathbb{R}^n$ platí $\mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0$,
(v) *indefinitní*, jestliže $\exists \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \neq \mathbf{o}$, $\mathbf{y} \neq \mathbf{o}$ platí

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \quad \wedge \quad \mathbf{y}^T\mathbf{A}\mathbf{y} < 0.$$

K určení pozitivní definitnosti matice lze použít např. tzv. Sylvestrovo kritérium:

Věta 1.1. *Nechť je dána symetrická čtvercová matice* $\mathbf{A}$ *řádu* n*. Jednotlivé subdeterminanty matice* $\mathbf{A}$ *označíme*

$$\mathbf{D}_k = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{12} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1k} & a_{2k} & \dots & a_{kk} \end{vmatrix}, \quad \forall k, \quad 1 \leq k \leq n.$$

Pak matice $\mathbf{A}$ *je pozitivně definitní právě tehdy, když* $\mathbf{D}_k > 0$*, pro* $1 \leq k \leq n$*.*

Důkaz: viz [1], str. 56.

Definice 1.2. Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$ je symetrická pozitivně definitní matice, $\mathbf{b}$ je vektor z $\mathbb{R}^n$ a c je reálné číslo. Pak funkce $f : \mathbb{R}^n \to \mathbb{R}$ daná předpisem

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} + c, \tag{1.1}$$

se nazývá *kvadratická funkce* proměnné $\mathbf{x} \in \mathbb{R}^n$.

Definice 1.3. Nechť je dána funkce f spojitá na množině $\mathbb{R}^n$. *Gradientem* funkce f v bodě $\mathbf{x} \in \mathbb{R}^n$ nazveme vektor tvaru

$$\nabla f(\mathbf{x}) = \mathbf{g}(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \frac{\partial f}{\partial x_2}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right)^T . \tag{1.2}$$

Poznámka 1.1. Obecně gradient $\nabla f(\mathbf{x}_0)$ určuje směr největšího růstu funkce $f(\mathbf{x})$ v okolí bodu $\mathbf{x}_0$. Záporný gradient $-\nabla f(\mathbf{x}_0)$ tedy udává směr nějvětšího spádu.

Definice 1.4. Nechť je dána funkce $f \in C^2(\mathbb{R}^n)$. *Hessiánem* funkce f v bodě $\mathbf{x} \in \mathbb{R}^n$ nazýváme symetrickou matici danou předpisem

$$\mathbf{H}(\mathbf{x}) = (h_{ij}(\mathbf{x}))_{i,j=1}^n , \qquad h_{ij}(\mathbf{x}) = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}). \tag{1.3}$$

2 Úvod do optimalizace

Optimalizace je matematická disciplína, která se zabývá hledáním extrémů dané funkce. Tedy naším úkolem je určit, ve kterém bodě má tato funkce své minimum, respektive maximum.

Nejprve se podíváme na základní pojmy související s problematikou optimalizace a na nutné a postačující podmínky optimality.

Definice 2.1. Nechť $f(\mathbf{x})$ je daná funkce n proměnných. Úloha minimalizovat funkci $f(\mathbf{x})$ pro $\mathbf{x} \in \mathbb{R}^n$ se nazývá *úlohou nepodmíněné optimalizace*. Budeme ji sručně zapisovat

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}). \tag{2.1}$$

Poznámka 2.1. Vzhledem k tomu, že platí

$$\max_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = -\min_{\mathbf{x}\in\mathbb{R}^n}(-f(\mathbf{x})),$$

budeme se zabývat pouze minimalizací funkce.

Definice 2.2. Nechť $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. Bod $\mathbf{x}^* \in \mathbb{R}^n$ se nazývá *bodem globálního minima* funkce f, jestliže platí

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

V případě, že

$$f(\mathbf{x}^*) < f(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{x}^*$$

hovoříme o *ostrém globálním minimu*.

K tomu, abychom nalezli globální minimum funkce f, je potřeba znát chování funkce f na celém jejím definičním oboru. Zpravidla známe její průběh pouze v určitých lokálních oblastech a jsme tedy schopni nalézt pouze *lokální minimum* funkce f.

Definice 2.3. Nechť $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. Bod $\mathbf{x}^* \in \mathbb{R}^n$ se nazývá *bodem lokálního minima* funkce f, jestliže existuje $\varepsilon > 0$ takové, že

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in N,$$

kde $N = \{\mathbf{y} \in \mathbb{R}^n : 0 \leq \|\mathbf{x}^* - \mathbf{y}\| < \varepsilon\}$. Pokud platí

$$f(\mathbf{x}^*) < f(\mathbf{x}), \quad \forall \mathbf{x} \in N, \mathbf{x} \neq \mathbf{x}^*,$$

hovoříme o *ostrém lokálním minimu.*

Definice 2.4. Nechť funkce $f \in C^1(\mathbb{R}^n)$. Bod $\tilde{\mathbf{x}} \in \mathbb{R}^n$ se nazývá *stacionárním bodem* funkce f, jestliže platí

$$\nabla f(\tilde{\mathbf{x}}) = \mathbf{g}(\tilde{\mathbf{x}}) = \mathbf{o}.$$

Poznámka 2.2. $\mathbf{g}(\mathbf{x})$ je gradient funkce f v bodě $\mathbf{x}$ (viz vztah (1.2)).

2.1 Podmínky optimality

Podmínky optimality se zabývají řešitelností úlohy minimalizace ve smyslu hledání lokálního minima. Tyto podmínky slouží k ověření, zda je daný bod skutečně řešením úlohy (2.1) (postačující podmínky optimality), případně ke zjištění, že daný bod není řešením úlohy (2.1) (nutné podmínky optimalizace) a k vytváření algoritmů a jejich ukončovacích kritérií.

Věta 2.1 (Nutná podmínka optimality 1. řádu)**.** *Nechť funkce $f \in C^1(\mathbb{R}^n)$ a nechť $\mathbf{x}^*$ je lokální minimum funkce f. Potom v bodě $\mathbf{x}^*$ platí*

$$\mathbf{g}(\mathbf{x}^*) = \nabla f(\mathbf{x}^*) = \mathbf{o}.$$

Důkaz: viz [3], str. 15.

Věta 2.2 (Nutná podmínka optimality 2. řádu)**.** *Nechť $f \in C^2(\mathbb{R}^n)$ a nechť $\mathbf{x}^*$ je lokální minimum funkce f. Pak v bodě $\mathbf{x}^*$ platí*

- $\mathbf{g}(\mathbf{x}^*) = \mathbf{o}$

- $\mathbf{H}(\mathbf{x}^*)$ *je pozitivně semidefinitní matice, tj.*

$$\mathbf{p}^T\mathbf{H}(\mathbf{x}^*)\mathbf{p} \geq 0, \qquad \forall \mathbf{p} \in \mathbb{R}^n.$$

Důkaz: viz [3], str. 15.

Věta 2.3 (Postačující podmínka optimality 2. řádu)**.** *Nechť funkce* $f \in C^2(\mathbb{R}^n)$ *a nechť v bodě* $\mathbf{x}^*$ *platí*

- $\mathbf{g}(\mathbf{x}^*) = \mathbf{o}$
- $\mathbf{H}(\mathbf{x}^*)$ *je pozitivně definitní matice, tj.*

$$\mathbf{p}^T\mathbf{H}(\mathbf{x}^*)\mathbf{p} > 0, \qquad \forall \mathbf{p} \in \mathbb{R}^n, \mathbf{p} \neq \mathbf{o}.$$

Pak bod $\mathbf{x}^*$ *je ostré lokální minimum funkce* f*.*

Důkaz: viz [3], str. 16.

Na následujícím příkladu si ukážeme použití výše uvedených podmínek.

Příklad 2.1. Pomocí podmínek optimality nalezněte minimum funkce

$$f(x_1, x_2) = x_1^2 + 4x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 - 5.$$

Řešení: Funkce f je definovaná na celé množině $\mathbb{R}^2$. Graf této funkce je znázorněn na Obrázku 1. K nalezení minima funkce využijeme Větu 2.3. Určíme gradient funkce f, tj.

$$\mathbf{g}(x_1, x_2) = (2x_1 + 4x_2 - 2, 4x_1 + 12x_2 + 8)^T.$$

Obě složky gradientu položíme rovny nule a dostáváme systém dvou lineárních rovnic tvaru

$$2x_1 + 4x_2 - 2 = 0$$
$$4x_1 + 12x_2 + 8 = 0$$

Po výpočtu řešení tohoto systému dostáváme bod $\mathbf{x} = (7, -3)^T$, což je tedy stacionární bod.

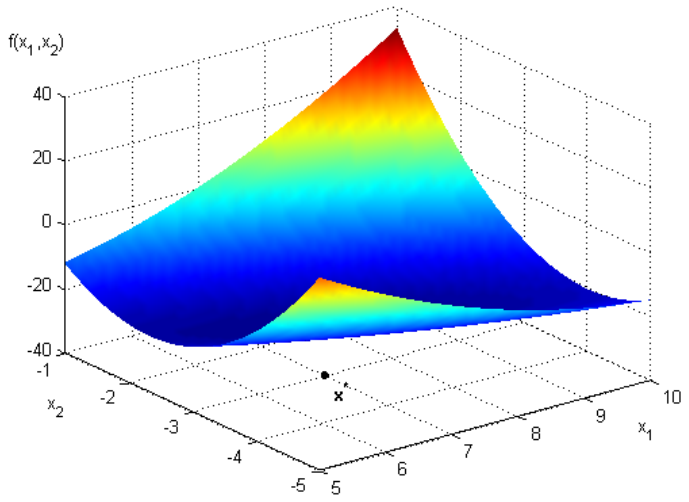
Nyní je třeba zjistit, zda je $\mathbf{x}$ minimem funkce f. Vypočítáme Hessián a dostaneme matici ve tvaru

$$\mathbf{H}(x_1, x_2) = \begin{pmatrix} 2 & 4 \\ 4 & 12 \end{pmatrix}.$$

Subdeterminanty této matice jsou rovny

$$\mathbf{D}_1 = 2 \qquad \mathbf{D}_2 = 8.$$

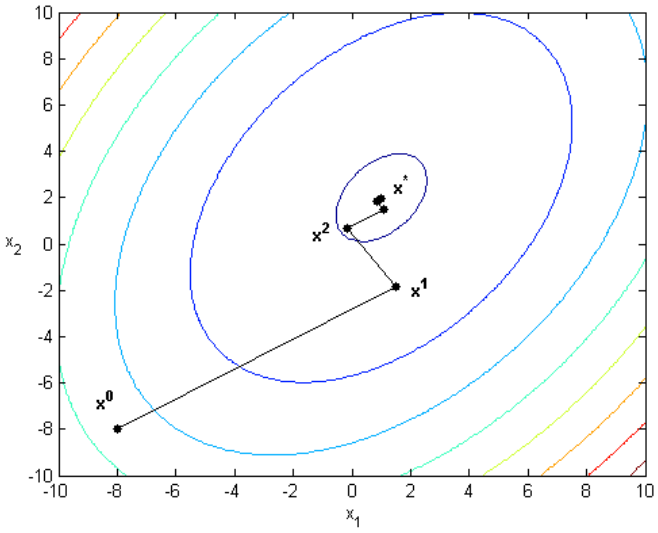
S využitím Sylvestrova kritéria vidíme, že Hessián je pozitivně definitní, a tudíž bod $\mathbf{x}^* = (7, -3)^T$ je ostré lokální minimum funkce f.



Obrázek 1: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 2.1

3 Minimalizace funkce pomocí spádových metod

Principem spádových metod je konstrukce posloupnosti bodů $\{\mathbf{x}^k\}_{k=0}^{\infty}$, která konverguje k minimu dané funkce f (viz Obrázek 2). Můžeme tedy říct, že spádové metody jsou metody iterační. V podstatě v každém kroku těchto metod hledáme minimum funkce f na polopřímce, která je určena bodem a vektorem, v jehož směru klesá funkční hodnota funkce f.



Obrázek 2: Princip spádové metody pro funkci dvou proměnných

V předchozí kapitole jsme si uvedli jednoduchý příklad výpočtu minima funkce f. V případě, že by složky gradientu $\mathbf{g}$ funkce f nebyly lineární, nalezení minima této funkce by bylo složitější. K tomu, abychom mohli najít minimum takové funkce, nám slouží právě metody spádové. Jejich základní myšlenka je následující.

Předpokládejme, že máme dánu funkci $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. Vybereme počáteční bod $\mathbf{x}^0 \in \mathbb{R}^n$ a poté generujeme posloupnost $\{\mathbf{x}^k\}$, která je dána předpisem

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{v}^k, \quad k = 0, 1, 2, \ldots,$$

přičemž vektory $\mathbf{v}^k$ se nazývají *směrové vektory* a α_k jsou kladná reálná čísla.

Naším úkolem je tedy vyřešit úlohu

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

s využitím spádové metody. Její důležitou součástí k nalezení minima je vhodně si zvolit směr, kterým se budeme pohybovat, a v tomto případě to bude směr spádový.

Definice 3.1. Nechť $f : \mathbb{R}^n \to \mathbb{R}$ je daná funkce, $\mathbf{x} \in \mathbb{R}^n$ je bod a $\mathbf{p} \in \mathbb{R}^n$ směr. Jestliže existuje $\alpha_0 > 0$ takové, že

$$f(\mathbf{x} + \alpha\mathbf{p}) < f(\mathbf{x}) \qquad \forall\alpha \in (0, \alpha_0\rangle,$$

pak $\mathbf{p}$ se nazývá *spádový směr* funkce f v bodě $\mathbf{x}$.

Nyní si uvedeme některé vlastnosti spádového směru. Následující věty lze nalézt ve většině studijních materiálů zabývajících se nepodmíněnou optimalizací, např. v [2], [3].

Věta 3.1. *Nechť je dána funkce $f \in C^1(\mathbb{R}^n)$, $\mathbf{p} \in \mathbb{R}^n$ je daný směr a nechť $\mathbf{g}(\mathbf{x})$ je gradientem funkce f v bodě $\mathbf{x}$. Jestliže*

$$\mathbf{g}^T(\mathbf{x})\mathbf{p} < 0,$$

pak $\mathbf{p}$ je spádovým směrem funkce f v bodě $\mathbf{x}$.

Věta 3.2. *Nechť je dána funkce $f \in C^2(\mathbb{R}^n)$, bod $\mathbf{x} \in \mathbb{R}^n$ a směr $\mathbf{p} \in \mathbb{R}^n$. Jestliže platí*

$$\mathbf{g}^T(\mathbf{x})\mathbf{p} = 0$$
$$\mathbf{p}^T\mathbf{H}(\mathbf{x})\mathbf{p} < 0,$$

pak $\mathbf{p}$ je spádovým směrem funkce f v bodě $\mathbf{x}$.

Věta 3.3. *Nechť je dána funkce $f \in C^1(\mathbb{R}^n)$ a bod $\mathbf{x} \in \mathbb{R}^n$. Mezi všemi spádovými směry funkce f v bodě $\mathbf{x}$ je směr*

$$\mathbf{p} = -\mathbf{g}(\mathbf{x})$$

ten, ve kterém funkce f klesá nejrychleji v okolí bodu $\mathbf{x}$.

Obecné schéma spádových metod vypadá následovně:

1. určíme počáteční bod $\mathbf{x}^0 \in \mathbb{R}^n$
2. pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria
   - určíme spádový směr $\mathbf{p}^k$ v bodě $\mathbf{x}^k$
   - stanovíme vhodnou délku kroku $\alpha_k > 0$, která splňuje podmínku
   $$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) < f(\mathbf{x}^k)$$
   - vypočítáme $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$

Snažíme se minimalizovat funkci f, tudíž postupně v každé nové iteraci postupující ve spádovém směru dostáváme menší funkční hodnotu funkce f. Vhodnou délkou kroku rozumíme $\alpha_k > 0$, které splňuje podmínku

$$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) < f(\mathbf{x}^k).$$

Takové α_k lze nalézt např. pomocí tzv. *exaktní volby* délky kroku. Položíme

$$\varphi(\alpha) = f(\mathbf{x}^k + \alpha \mathbf{p}^k), \tag{3.1}$$

což je funkce jedné proměnné α. Následně vyřešíme úlohu jednorozměrné minimalizace

$$\varphi(\alpha_k) = \min_{\alpha \geq 0} \varphi(\alpha). \tag{3.2}$$

Věta 3.4. *Nechť $f \in C^1(\mathbb{R}^n)$ je daná funkce. Pro spádovou metodu používající exaktní volbu délky kroku platí*

$$\mathbf{g}^{k+1^T} \mathbf{p}^k = 0,$$

kde $\mathbf{g}^{k+1} = \mathbf{g}(\mathbf{x}^{k+1})$ a $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$.

Poznámka 3.1. Věta 3.4 nám říká, že vektory $\mathbf{g}^{k+1}$ a $\mathbf{p}^k$ jsou ortogonální, přičemž $\mathbf{g}^{k+1}$ je gradient funkce f v bodě $\mathbf{x}^{k+1}$.

Další možností pro stanovení α_k je určení „nepřesné“ délky kroku např. pomocí Wolfeho podmínek nebo tzv. Armijovy podmínky zpětného vyhledávání (např. viz [3], str. $33 - 37$). Vzhledem k tomu, že v příkladech uvedených v této práci budeme používat právě Armijovu podmínku zpětného vyhledávání, uvedeme si zde její základní myšlenku. Ta spočívá v tom, že zkracujeme délku kroku α_k tak dlouho, dokud není splněna podmínka poklesu funkční hodnoty dané funkce f. Její algoritmus vypadá následovně.

Algoritmus 3.1 (Armijova podmínka zpětného vyhledávání)**.**

- zvolíme počáteční hodnotu $\bar{\alpha} > 0$
- zvolíme koeficienty $\gamma \in (0, 1)$ a $\delta \in (0, 1)$
- položíme $\alpha_k = \bar{\alpha}$
- dokud platí $f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) > f(\mathbf{x}^k) + \alpha_k \delta {\mathbf{p}^k}^T \mathbf{g}^k$
  - položíme $\alpha_k = \gamma \alpha_k$

Na základě Algoritmu 3.1 si naprogramujeme M-soubor pro Armijovu podmínku.

M-soubor 3.1 (Armijova podmínka zpětného vyhledávání)**.**

```
function alpha = armij(alpha0,fce,x,g,p,gamma,delta)

% armij - Armijova podmínka zpětného vyhledávání

% Vstupní hodnoty:
% alpha0 - počáteční hodnota délky kroku větší než nula
% fce - funkce zadaná pomocí 'function_handle'
% x - bod v aktuálním kroku algoritmu
% g - gradient funkce fce v bodě x
```

```
% p - spádový směr funkce fce v bodě x
% gamma - koeficient z intervalu (0,1)
% delta - koeficient z intervalu (0,1)

% Výstupní hodnota
% alpha - hledaná délka kroku

alpha = alpha0;

while fce(x+alpha*p) > fce(x) + alpha*delta*dot(p,g)
    alpha = gamma*alpha;
end;
```

Poznámka 3.2. Pro parametry $\bar{\alpha}$, γ a δ si čtenář může zvolit vlastní hodnoty. Sama jsem si vyzkoušela několik možností a rozhodla jsem se zvolit následující hodnoty

$$\bar{\alpha} = 10$$
$$\gamma = 0.5$$
$$\delta = 0.001,$$

jež také využiji v příkladech, ve kterých se bude počítat délka kroku pomocí Armijovy podmínky zpětného vyhledávání.

Poznámka 3.3. Ve druhém kroku obecného schématu spádových metod jsme zmínili ukončovací kritérium. Toto kritérium je důležité proto, abychom věděli, kdy máme ukončit algoritmus. Než začneme minimalizovat danou funkci, zadáme si nejdříve přesnost s jakou minimum hledáme. Tuto přesnost budeme značit ε.

Uvedeme si několik typů ukončovacích kritérií.

- testování gradientu (v lokálním minimu musí platit $\mathbf{g}(\mathbf{x}^*) = \mathbf{o}$)

$$\|\mathbf{g}^k\| \leq \varepsilon \tag{3.3}$$

  nebo

$$\frac{\|\mathbf{g}^k\|}{\|\mathbf{g}^0\|} \leq \varepsilon \tag{3.4}$$

- testování posunutí bodu (možno použít absolutní nebo relativní chybu)

$$\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \leq \varepsilon \tag{3.5}$$

$$\frac{\|\mathbf{x}^{k+1} - \mathbf{x}^k\|}{\|\mathbf{x}^k\|} \leq \varepsilon \tag{3.6}$$

- testování změny funkční hodnoty f

$$|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)| \leq \varepsilon \tag{3.7}$$

  nebo

$$\frac{|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)|}{|f(\mathbf{x}^{k+1})|} \leq \varepsilon \tag{3.8}$$

Za ε si volíme libovolně malé číslo větší než nula, např. $\varepsilon = 0.01$, $\varepsilon = 0.0001$.

Tímto jsme si uvedli základní kroky potřebné ke spádovým metodám. Nyní se podívejme konkrétně na Metodu největšího spádu. Tuto metodu publikoval v roce 1847 francouzský matematik Augustin Louis Cauchy. K nalezení minima dané funkce f se pohybujeme v takovém směru $\mathbf{p}^k$, který má „největší spád“. Tento směr je dán

$$\mathbf{p}^k = -\mathbf{g}^k. \tag{3.9}$$

S využitím předešlého textu můžeme vytvořit algoritmus pro Metodu největšího spádu.

Algoritmus 3.2 (MNS pro obecné funkce)**.**

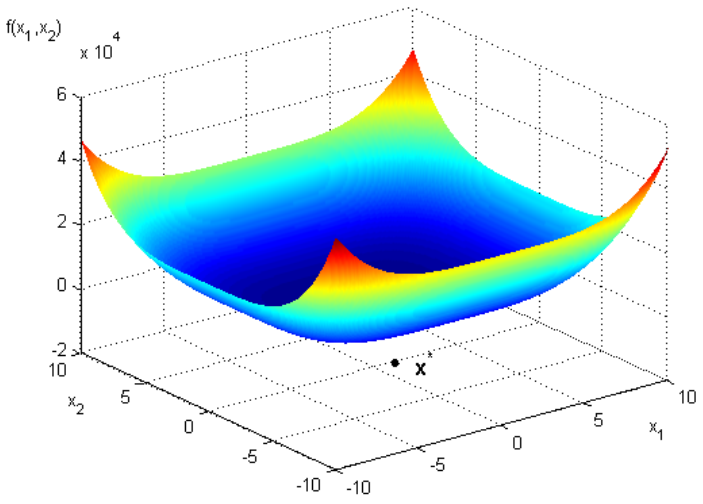
- zvolíme počáteční bod $\mathbf{x}^0$ a ε pro ukončovací kritérium
- vypočítáme gradient $\mathbf{g}^0 = \nabla f(\mathbf{x}^0)$
- pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria
  - vypočítáme α_k
  - položíme $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{g}^k$
  - vypočítáme $\mathbf{g}^{k+1} = \nabla f(\mathbf{x}^{k+1})$

Podívejme se nyní na konkrétní úlohu.

Příklad 3.1. Mějme dánu funkci f tvaru

$$f(x_1, x_2) = 2x_1^4 + 3x_2^4 - 4x_1^2 + x_1 x_2 - 3x_2^3 - 3,$$

jejíž graf je znázorněn na Obrázku 3. Pomocí Metody největšího spádu vypočítejte první tři členy posloupnosti $\{\mathbf{x}^k\}$ s počátečním bodem $\mathbf{x}^0 = (2, 0)^T$.



Obrázek 3: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 3.1

Řešení: Máme zadánu funkci f a počáteční bod $\mathbf{x}^0 \in \mathbb{R}^2$. Nejdříve si zjistíme, jak bude vypadat gradient $\mathbf{g}(x_1, x_2)$. Vypočítáme

$$\mathbf{g}(x_1, x_2) = (8x_1^3 - 8x_1 + x_2, 12x_2^3 + x_1 - 9x_2^2)^T.$$

Pro výpočet délky kroku α_k použijeme Armijovu podmínku zpětného vyhledávání. Pro prametry $\bar{\alpha}$, γ a δ si zadáme stejné hodnoty, jako jsme si uvedli v Poznámce 3.2. Nebudeme zde krok α_k počítat ručně, poněvadž bychom museli provádět velké množství výpočtů. K tomu využijeme M-soubor 3.1. Jednotlivé ruční výpočty budeme zaokrouhlovat na 5 desetinných míst.

Dále budeme postupovat podle Algoritmu 3.2. Vypočítáme

$$\mathbf{g}^0 = \nabla f(\mathbf{x}^0) = \begin{pmatrix} 48 \\ 2 \end{pmatrix}.$$

Podívejme se, jak bude vypadat 1. krok algoritmu. Pro výpočet α_0 zadáme hodnoty $\bar{\alpha} = 10$, $\gamma = 0.5$, $\delta = 0.001$ a již známé $\mathbf{x}^0$, $\mathbf{g}^0$. Za směr $\mathbf{p}^0$ dosadíme $-\mathbf{g}^0$, což nám plyne z výrazu (3.9). V matematickém prostředí Matlabu zapíšeme následující

```
x0=[2;0]; g0=[48;2];
fce=@(x)2*x(1).^4+3*x(2).^4-4*x(1).^2+x(1).*x(2)-3*x(2).^3-3;
```

Zavoláme funkci *armij*

```
alpha=armij(10,fce,x0,g0,-g0,0.5,0.001)
```

a Matlab nám vrátí výsledek

```
alpha =

    0.07812500000000
```

Dosadíme do algoritmu a pro $k = 0$ vypočítáme

$$\alpha_0 = 0.07813$$

$$\mathbf{x}^1 = \mathbf{x}^0 - \alpha_0 \mathbf{g}^0 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} - 0.07813 \begin{pmatrix} 48 \\ 2 \end{pmatrix} = \begin{pmatrix} -1.75024 \\ -0.15626 \end{pmatrix}$$

$$\mathbf{g}^1 = \nabla f(\mathbf{x}^1) = \begin{pmatrix} -29.04698 \\ -2.01578 \end{pmatrix}.$$

Pokračujeme 2. krokem algoritmu, pro výpočet α_1 zadáme do Matlabu

```
x1=[-1.75024;-0.15626]; g1=[-29.04698;-2.01578];
fce=@(x)2*x(1).^4+3*x(2).^4-4*x(1).^2+x(1).*x(2)-3*x(2).^3-3;
alpha=armij(10,fce,x1,g1,-g1,0.5,0.001)
```

a Matlab nám vrátí

```
alpha =
     0.07812500000000
```

Pro $k = 1$ vypočítáme

$$\begin{aligned}
&\alpha_1 = 0.07813\\
&\mathbf{x}^2 = \mathbf{x}^1 - \alpha_1 \mathbf{g}^1 = \begin{pmatrix} 0.51920 \\ 0.00123 \end{pmatrix}\\
&\mathbf{g}^2 = \nabla f(\mathbf{x}^2) = \begin{pmatrix} -3.03269 \\ 0.51919 \end{pmatrix}.
\end{aligned}$$

Nyní počítáme 3. krok algoritmu, přičemž pro výpočet α_2 zadáme do Matlabu

```
x2=[0.51920;0.00123]; g2=[-3.03269;0.51919];
fce=@(x)2*x(1).^4+3*x(2).^4-4*x(1).^2+x(1).*x(2)-3*x(2).^3-3;
alpha=armij(10,fce,x2,g2,-g2,0.5,0.001)
```

a Matlab nám vrátí

```
alpha =
     0.15625000000000
```

Pro $k = 2$ vypočítáme

$$\begin{aligned}
&\alpha_2 = 0.15625\\
&\mathbf{x}^3 = \mathbf{x}^2 - \alpha_2 \mathbf{g}^2 = \begin{pmatrix} 0.99306 \\ -0.07989 \end{pmatrix}\\
&\mathbf{g}^3 = \nabla f(\mathbf{x}^3) = \begin{pmatrix} -0.18978 \\ 0.92950 \end{pmatrix}.
\end{aligned}$$

Ukázali jsme si, jak se počítají jednotlivé kroky v algoritmu pro Metodu největšího spádu pro obecné funkce. V případě, že bychom chtěli pomocí Algoritmu 3.2 najít minimum funkce f, zadáme si přesnost např. $\varepsilon = 10^{-4}$ a s využitím následujícího M-souboru 3.2 dostáváme po 12 krocích hledané minimum, které po zaokrouhlení na 5 desetinných míst je tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 1.01740 \\ -0.28608 \end{pmatrix}.$$

Pro srovnání si můžeme uvést také přesné řešení, jež má souřadnice

$$\mathbf{x}^{*} = \begin{pmatrix} 1.01742 \\ -0.28607 \end{pmatrix},$$

což si můžeme ověřit analogickým postupem jako v Příkladu 2.1. Vidíme tedy, že minimum $\mathbf{x}^{\varepsilon}$ jsme skutečně našli s přesností 0.0001.

Na Příkladu 3.1 jsme si ukázali aplikaci Algoritmu 3.2 pro konkrétní funkci. Jestliže bychom chtěli pomocí ručních výpočtů nalézt minimum dané funkce, bylo by to ve většině případů velice zdlouhavé a nepraktické. Proto si nyní naprogramujeme M-soubor, který nám vypočítá minimum mnohem rychleji.

V následujícím M-souboru 3.2 si za ukončovací kritérium zvolíme test změny funkční hodnoty f, který je vyjádřen vztahem (3.8) a test posunutí bodu, jež je dán ve tvaru (3.6). V případě Metody největšího spádu je volba dvou ukončovacích kritérií výhodnější, protože s jejich použitím vypočítá algoritmus řešení s větší přesností. Navzdory tomu, že budeme používat ukončovací kritéria, se může stát, že se algoritmus nikdy neukončí, jestliže se v blízkém okolí počátečního bodu nenalézá lokální minimum (viz otázka konvergence např. v knize [3] na str. $37 - 44$). Proto si jako vstupní hodnotu v M-souboru 3.2 budeme zadávat také maximální počet iterací, jež se budou v algoritmu provádět. Pro výpočet délky kroku α_k použijeme Armijovu podmínku zpětného vyhledávání, kterou jsme si naprogramovali v M-souboru 3.1. Pro její vstupní parametry můžeme opět zvolit hodnoty, jež jsou uvedeny v Poznámce 3.2.

M-soubor 3.2 (MNS pro obecné funkce)**.**

```
function [xk,krok] = mns(fce,x,x0,epsilon,maxiter)

% mns - Metoda největšího spádu pro obecné funkce

% Vstupní hodnoty:
% fce - funkce, jejíž minimum hledáme, zadána
%       pomocí 'function_handle'
% x - symbolická proměnná
% x0 - počáteční bod
% epsilon - přesnost, s jakou mns nalezne hledané minimum
% maxiter - maximální počet iterací, které mns provede

% Výstupní hodnoty:
% xk - hledané minimum funkce
% krok - počet provedených iterací

n = length(x0);
FCE = fce(x);   % symbolická funkce

% výpočet gradientu funkce f
for (i = 1:n)
    G(i) = diff(FCE,x(i)); % symbolické derivování
end;

xk = x0;
g = (subs(G,x,xk))';
f = fce(xk);
```

```
% vstupní hodnoty pro Armijovu podmínku
alpha0 = 10;
gamma = 0.5;
delta = 0.001;

% cyklus algoritmu
for krok = 0:maxiter
    xprev = xk;
    fprev = f;

    % Armij. podm.
    alpha = armij(alpha0,fce,xk,g,-g,gamma,delta);

    xk = xk - alpha*g;
    f = fce(xk);
    g = (subs(G,x,xk))';

    % ukončovací kritérium
    if (abs((fprev-f)/f) <= epsilon)
        if (norm(xk-xprev)/norm(xprev) <= epsilon)
            break;
        end;
    end;
end;
```

Podívejme se, jak lze M-soubor 3.2 aplikovat na konkrétní úlohu.

Příklad 3.2. Pomocí Metody největšího spádu najděte minimum funkce f ve tvaru

$$f(x_1, x_2) = x_1^2 + 5x_1x_2 + x_2^4 - 2x_1 - 3x_2$$

s přesností $\varepsilon = 10^{-4}$. Za počáteční bod zvolte $\mathbf{x}^0 = (2, 1)^T$.

Řešení: Maximální počet iterací si zadáme 100 a v prostředí Matlabu napíšeme

```
x=sym('[x1,x2]');
fce=@(x)x(1).^2+5*x(1).*x(2)+x(2).^4-2*x(1)-3*x(2);
[reseni,pocet_iteraci]=mns(fce,x,[2;1],0.0001,100)
```

Matlab nám vrátí

```
reseni =
   5.60242814813319
  -1.84211725142878


pocet_iteraci =
    79
```

Zaokrouhlíme-li výsledné řešení na 5 desetinných míst, můžeme vidět, že po 79 krocích jsme s přesností $\varepsilon = 10^{-4}$ našli minimum ve tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 5.60243 \\ -1.84212 \end{pmatrix}.$$

3.1 Kvadratická funkce

Nyní si ukážeme, jak bude vypadat Metoda největšího spádu pro kvadratickou funkci tvaru

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + c. \tag{3.10}$$

V této kapitole jsme si již uvedli obecné schéma spádových metod. K nalezení minima kvadratické funkce pomocí Metody největšího spádu se budeme řídit stejnými kroky.

1. určíme počáteční bod $\mathbf{x}^0 \in \mathbb{R}^n$

2. pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria

   - určíme spádový směr $\mathbf{p}^k = -\mathbf{g}^k$ v bodě $\mathbf{x}^k$
   - stanovíme vhodnou délku kroku $\alpha_k > 0$, která splňuje podmínku

     $$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) < f(\mathbf{x}^k)$$

   - vypočítáme $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$

Pro kvadratickou funkci f dostáváme gradient a Hessián tvaru

$$\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} \tag{3.11}$$

$$\mathbf{H}(\mathbf{x}) = \mathbf{A}. \tag{3.12}$$

V dalším textu budeme pracovat s kvadratickou funkcí ve tvaru

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x},$$

protože konstanta c nám neovlivní polohu bodu minima $\mathbf{x}^*$, na c závisí pouze hodnota $f(\mathbf{x}^*)$.

V případě kvadratické funkce jsme schopni přesně určit délku kroku α_k pomocí již zmíněné *exaktní volby* délky kroku, což si následně ukážeme.

Předpokládejme, že jsme v k-tém kroku. Pohybujeme se z bodu $\mathbf{x}^k$, ve směru $\mathbf{p}^k$ a s délkou kroku α. K nalezení α využijeme výrazů (3.1) a (3.2). Do předpisu funkce f dosadíme za bod $\mathbf{x} = \mathbf{x}^k + \alpha\mathbf{p}^k$. Pak tedy

$$\begin{aligned} f(\mathbf{x}^k + \alpha\mathbf{p}^k) &= \frac{1}{2}(\mathbf{x}^k + \alpha\mathbf{p}^k)^T\mathbf{A}(\mathbf{x}^k + \alpha\mathbf{p}^k) - \mathbf{b}^T(\mathbf{x}^k + \alpha\mathbf{p}^k) = \\ &= \frac{1}{2}\mathbf{x}^{kT}\mathbf{A}\mathbf{x}^k + \frac{1}{2}\mathbf{x}^{kT}\mathbf{A}\alpha\mathbf{p}^k + \frac{1}{2}\alpha\mathbf{p}^{kT}\mathbf{A}\mathbf{x}^k + \frac{1}{2}\alpha\mathbf{p}^{kT}\mathbf{A}\alpha\mathbf{p}^k - \\ &- \mathbf{b}^T\mathbf{x}^k - \mathbf{b}^T\alpha\mathbf{p}^k. \end{aligned}$$

S využitím vztahu (3.11) dostáváme

$$f(\mathbf{x}^k + \alpha\mathbf{p}^k) = f(\mathbf{x}^k) + \alpha\mathbf{g}^{kT}\mathbf{p}^k + \frac{1}{2}\alpha^2\mathbf{p}^{kT}\mathbf{A}\mathbf{p}^k.$$

Položíme $\varphi(\alpha) = f(\mathbf{x}^k + \alpha\mathbf{p}^k)$ a hledáme minimum funkce $\varphi(\alpha)$, které označíme α_k. Tudíž funkci φ zderivujeme podle α a dostaneme

$$\varphi'(\alpha) = \mathbf{g}^{k^T}\mathbf{p}^k + \alpha\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k.$$

Tuto derivaci položíme rovnu nule

$$\mathbf{g}^{k^T}\mathbf{p}^k + \alpha\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k = 0. \tag{3.13}$$

Dále víme, že matice $\mathbf{A}$ je pozitivně definitní, tudíž platí $\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k > 0$. Můžeme tedy číslo α v rovnici (3.13) vyjádřit ve tvaru

$$\alpha = -\frac{\mathbf{g}^{k^T}\mathbf{p}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}, \tag{3.14}$$

což je stacionární bod funkce $\varphi(\alpha)$. K tomu, abychom zjistili, zda je α minimem, potřebujeme vypočítat také druhou derivaci funkce $\varphi(\alpha)$, tedy

$$\varphi''(\alpha) = \mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k.$$

Vzhledem k pozitivní definitnosti matice $\mathbf{A}$ dostáváme minimum funkce $\varphi(\alpha)$ ve tvaru

$$\alpha_k = -\frac{\mathbf{g}^{k^T}\mathbf{p}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}. \tag{3.15}$$

Využijeme-li již zmíněné obecné schéma spádových metod a výraz (3.15), jsme schopni v k-tém kroku algoritmu vypočítat

$$\begin{aligned}
\mathbf{g}^k &= \mathbf{A}\mathbf{x}^k - \mathbf{b} \\
\alpha_k &= \frac{\mathbf{g}^{k^T}\mathbf{g}^k}{\mathbf{g}^{k^T}\mathbf{A}\mathbf{g}^k} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k\mathbf{g}^k,
\end{aligned}$$

přičemž pro $\mathbf{g}^{k+1}$ platí

$$\mathbf{g}^{k+1} = \mathbf{A}(\mathbf{x}^k - \alpha_k\mathbf{g}^k) - \mathbf{b} = \mathbf{g}^k - \alpha_k\mathbf{A}\mathbf{g}^k.$$

Nyní si můžeme uvést algoritmus pro Metodu největšího spádu pro kvadratické funkce.

Algoritmus 3.3 (MNS pro kvadratické funkce)**.**

- zvolíme počáteční bod $\mathbf{x}^0$ a $\varepsilon > 0$ pro ukončovací kritérium
- vypočítáme $\mathbf{g}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$
- pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria počítáme

$$\begin{aligned} \alpha_k &= \frac{\mathbf{g}^{k\,T}\mathbf{g}^k}{\mathbf{g}^{k\,T}\mathbf{A}\mathbf{g}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{g}^k \\ \mathbf{g}^{k+1} &= \mathbf{g}^k - \alpha_k \mathbf{A}\mathbf{g}^k. \end{aligned}$$

Algoritmus 3.3 použijeme při řešení následující úlohy.

Příklad 3.3. Nechť je dána funkce f tvaru

$$f(x_1, x_2) = 3{x_1}^2 - 2x_1x_2 + 2{x_2}^2 - 2x_1 - 6x_2 + 1,$$

jejíž graf je znázorněn na Obrázku 4. Pomocí Metody největšího spádu vypočítejte první tři kroky algoritmu. Za počáteční bod zvolte $\mathbf{x}^0 = (0, 0)^T$.

Řešení: Máme zadánu funkci f a počáteční bod $\mathbf{x}^0 \in \mathbb{R}^2$. Pro srovnání si zde uvedeme přesné řešení $\mathbf{x}^* = (1, 2)^T$, jež lze vypočítat analyticky. Nyní budeme postupovat podle Algoritmu 3.3. Nejprve vypočteme

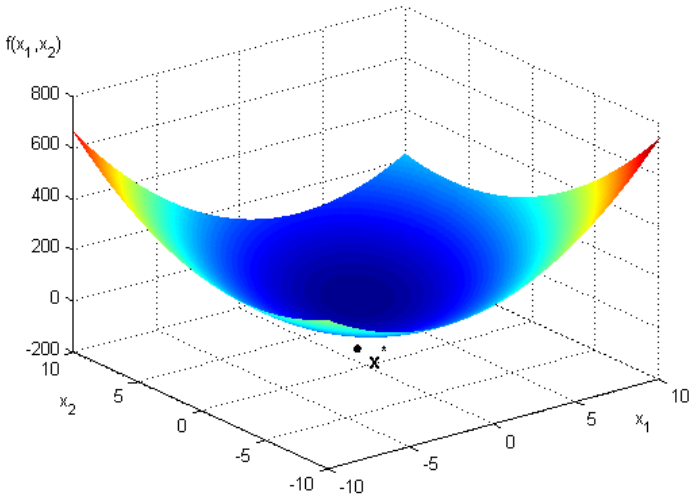
$$\mathbf{g}(x_1, x_2) = (6x_1 - 2x_2 - 2, -2x_1 + 4x_2 - 6)^T.$$

Položíme-li jednotlivé složky gradientu rovny nule, dostaneme soustavu

$$\begin{aligned} 6x_1 - 2x_2 &= 2 \\ -2x_1 + 4x_2 &= 6\ . \end{aligned} \tag{3.16}$$

Z výrazu (3.11) víme, že gradient pro kvadratickou funkci je tvaru $\mathbf{g} = \mathbf{A}\mathbf{x} - \mathbf{b}$. Ze soustavy (3.16) můžeme tedy vyjádřit matici $\mathbf{A}$ a vektor $\mathbf{b}$ ve tvaru

$$\mathbf{A} = \begin{pmatrix} 6 & -2 \\ -2 & 4 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 2 \\ 6 \end{pmatrix}.$$



Obrázek 4: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 3.3

Dále vypočítáme

$$\mathbf{g}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b} = \begin{pmatrix} 6 & -2 \\ -2 & 4 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 \\ 6 \end{pmatrix}$$

$$\mathbf{g}^0 = \begin{pmatrix} -2 \\ -6 \end{pmatrix}.$$

Dosadíme do algoritmu a pro $k = 0$ vypočítáme

$$\alpha_0 = \frac{\mathbf{g}^{0T}\mathbf{g}^0}{\mathbf{g}^{0T}\mathbf{A}\mathbf{g}^0} = \frac{(-2,-6)\begin{pmatrix} -2 \\ -6 \end{pmatrix}}{(-2,-6)\begin{pmatrix} 6 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} -2 \\ -6 \end{pmatrix}} = \frac{1}{3}$$

$$\mathbf{x}^1 = \mathbf{x}^0 - \alpha_0\mathbf{g}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \frac{1}{3}\begin{pmatrix} -2 \\ -6 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ 2 \end{pmatrix}$$

$$\mathbf{g}^1 = \mathbf{g}^0 - \alpha_0\mathbf{A}\mathbf{g}^0 = \begin{pmatrix} -2 \\ -6 \end{pmatrix} - \frac{1}{3}\begin{pmatrix} 6 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} -2 \\ -6 \end{pmatrix} = \begin{pmatrix} -2 \\ \frac{2}{3} \end{pmatrix}.$$

Pro $k = 1$ vypočítáme

$$\alpha_1 = \frac{{\mathbf{g}^1}^T \mathbf{g}^1}{{\mathbf{g}^1}^T \mathbf{A}\mathbf{g}^1} = \frac{1}{7}$$

$$\mathbf{x}^2 = \mathbf{x}^1 - \alpha_1 \mathbf{g}^1 = \begin{pmatrix} \frac{20}{21} \\ \frac{40}{21} \end{pmatrix}$$

$$\mathbf{g}^2 = \mathbf{g}^1 - \alpha_1 \mathbf{A}\mathbf{g}^1 = \begin{pmatrix} -\frac{2}{21} \\ -\frac{2}{7} \end{pmatrix}.$$

Pro $k = 2$ vypočítáme

$$\alpha_2 = \frac{{\mathbf{g}^2}^T \mathbf{g}^2}{{\mathbf{g}^2}^T \mathbf{A}\mathbf{g}^2} = \frac{1}{3}$$

$$\mathbf{x}^3 = \mathbf{x}^2 - \alpha_2 \mathbf{g}^2 = \begin{pmatrix} \frac{62}{63} \\ 2 \end{pmatrix}$$

$$\mathbf{g}^3 = \mathbf{g}^2 - \alpha_2 \mathbf{A}\mathbf{g}^2 = \begin{pmatrix} -\frac{2}{21} \\ \frac{2}{63} \end{pmatrix}.$$

Jestliže bychom chtěli pomocí Algoritmu 3.3 najít minimum funkce f, zadáme si přesnost např. $\varepsilon = 10^{-4}$ a využijeme následujícího M-souboru 3.3. Dostáváme po 7 krocích minimum ve tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 0.99999485810953 \\ 1.99998971621907 \end{pmatrix},$$

jež je po zaokrouhlení na 5 desetinných míst stejné jako analyticky nalezené $\mathbf{x}^*$.

V jednotlivých iteracích můžeme vidět, jak se postupně přibližujeme k hledanému řešení. Např. v prvním kroku jsme vypočítali, že $\mathbf{x}^1 = \left(\frac{2}{3}, 2\right)^T$, přičemž zde můžeme vidět, že druhá souřadnice bodu $\mathbf{x}^1$ je stejná jako v bodu $\mathbf{x}^{\varepsilon}$.

Podívejme se, jak bude vypadat M-soubor pro Metodu největšího spádu v případě kvadratických funkcí. Ukončovací kritéria si můžeme zvolit stejná jako jsme použili v M-souboru 3.2 u funkcí obecných a opět zadáme maximální počet iterací, které Metoda největšího spádu bude provádět.

M-soubor 3.3 (MNS pro kvadratické funkce)**.**

```
function [x,krok] = mnskvadr(A,b,c,x0,epsilon,maxiter)

% mnskvadr - Metoda největšího spádu pro kvadratické funkce
%            f(x) = 1/2*x'*A*x - x'*b + c

% Vstupní hodnoty:
% A - symetrická matice vyjádřena z předpisu pro kvadr. funkci f(x)
% b - vektor vyjádřen z předpisu pro kvadr. funkci f(x)
% c - reálné číslo vyjádřeno z předpisu pro kv. funkci f(x)
% x0 - počáteční bod
% epsilon - přesnost, jakou mnskvadr nalezne hledané minimum
% maxiter - maximální počet iterací, které mnskvadr provede

% Výstupní hodnoty:
% xk - hledané minimum funkce f(x)
% krok - počet provedených iterací

x = x0;
g = A*x - b;
f = 1/2*dot(x,A*x) - dot(x,b) + c;

% cyklus algoritmu
for krok = 0:maxiter
    xprev = x;
    fprev = f;
    alpha = dot(g,g)/dot(g,A*g);
    x = x - alpha*g;
    f = 1/2*dot(x,A*x) - dot(x,b) + c;
    g = g - alpha*A*g;
```

```
    % ukončovací kritérium
    if (abs((fprev-f)/f) <= epsilon)
        if (norm(x-xprev)/norm(xprev) <= epsilon)
            break;
        end;
    end;
end;
```

Na následující příklad aplikujeme M-soubor 3.3.

Příklad 3.4. Nechť je dána kvadratická funkce ve tvaru

$$f(x_1, x_2) = 2(x_1^2 + 2x_2^2) - 3x_1(x_2 - 4) - (2x_1 - x_2).$$

Zvolte počáteční bod $\mathbf{x}^0 = (2, 3)^T$ a přesnost $\varepsilon = 10^{-4}$. S využitím Metody největšího spádu nalezněte minimum funkce f. Výsledné řešení zaokrouhlete na 5 desetinných míst.

Řešení: Pro zadanou kvadratickou funkci f dostáváme matici $\mathbf{A}$, vektor $\mathbf{b}$ a parametr c ve tvaru

$$\mathbf{A} = \begin{pmatrix} 4 & -3 \\ -3 & 8 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} -10 \\ -1 \end{pmatrix}, \qquad c = 0.$$

Zvolíme si maximální počet iterací 100 a do Matlabu zadáme

```
A=[4,-3;-3,8]; b=[-10;-1]; c=0;
[reseni,pocet_iteraci] = mnskvadr(A,b,c,[2;3],0.0001,100)
```

Matlab nám vrátí

```
reseni =
  -3.60834823213536
  -1.47798347217009
```

```
pocet_iteraci =
    17
```

Po 17 krocích jsme s přesností $\varepsilon = 10^{-4}$ našli minimum ve tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} -3.60835 \\ -1.47798 \end{pmatrix}.$$

V této kapitole jsem představila Metodu největšího spádu pro obecné a kvadratické funkce a naprogramovala tři M-soubory, s jejichž pomocí lze nalézt minimum dané funkce s určitou přesností. Pro řešení konkrétních úloh stačí do Matlabu zadat jiné vstupní hodnoty.

4 Metoda konjugovaných gradientů

Tématem této bakalářské práce je Metoda konjugovaných gradientů. Je to jedna z nejpoužívanějších technik pro řešení velkých systémů lineárních rovnic a můžeme ji také přizpůsobit k řešení nelineárních optimalizačních problémů.

Lineární Metodu konjugovaných gradientů navrhli v padesátých letech minulého století dva matematikové Hestenes a Stiefel. Je to iterační metoda, která řeší systém lineárních rovnic s pozitivně definitní maticí.

Nelineární Metodu konjugovaných gradientů nám představili v šedesátých letech dvacátého století matematikové Fletcher a Reeves. Je to jedna z nejstarších známých technik pro řešení rozsáhlých nelineárních optimalizačních problémů. Metoda konjugovaných gradientů má dva hlavní znaky, a to: nevyžaduje ukládání všech předchozích výpočtů do paměti a je rychlejší než Metoda největšího spádu.

Než se začneme zabývat Metodou konjugovaných gradientů, je třeba si ujasnit, co je to vlastně konjugovaný směr.

Definice 4.1. Nenulové vektory $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ se nazývají *konjugované směry* (neboli *sdružené*) vzhledem k symetrické pozitivně definitní matici $\mathbf{A} \in \mathbb{R}^{n \times n}$, jestliže platí

$$\mathbf{p}^T \mathbf{A} \mathbf{q} = 0.$$

Množina $\{\mathbf{p}^0, \mathbf{p}^1, \ldots, \mathbf{p}^m\}$ nenulových vektorů se nazývá *konjugovaná* vzhledem k symetrické pozitivně definitní matici $\mathbf{A}$, jestliže splňuje

$$\mathbf{p}^{iT} \mathbf{A} \mathbf{p}^j = 0, \qquad \forall i \neq j.$$

Poznámka 4.1. Lze také dokázat, že konjugované směry jsou lineárně nezávislé.

Metoda konjugovaných gradientů je obecně spádová metoda, přičemž místo spádových směrů volíme směry konjugované. V předchozí kapitole jsme si již uvedli obecné schéma spádových metod, které použijeme také nyní a proto si ho znovu připomeneme.

1. určíme počáteční bod $\mathbf{x}^0 \in \mathbb{R}^n$
2. pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria
   - určíme spádový směr $\mathbf{p}^k$ v bodě $\mathbf{x}^k$
   - stanovíme vhodnou délku kroku $\alpha_k > 0$, která splňuje podmínku

     $$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) < f(\mathbf{x}^k)$$

   - vypočítáme $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$

Pro jednodušší vysvětlení, jak Metoda konjugovaných gradientů pracuje, se nejdříve podíváme na lineární Metodu konjugovaných gradientů, což je metoda pro nalezení minima kvadratických funkcí. Poté se zaměříme na Metodu konjugovaných gradientů pro funkce obecné, nebo-li metodu nelineární, jejíž algoritmus vychází právě z metody lineární.

4.1 Lineární Metoda konjugovaných gradientů

Předpokládejme, že máme dánu kvadratickou funkci tvaru

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{x}^T \mathbf{b} + c. \tag{4.1}$$

Z předchozího textu víme, že

$$\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}. \tag{4.2}$$

Dosadíme-li za $\mathbf{x} = \mathbf{x}^k$, dostaneme gradient pro k-tou iteraci ve tvaru

$$\mathbf{g}^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}. \tag{4.3}$$

Mějme počáteční bod $\mathbf{x}^0 \in \mathbb{R}^n$ a již zmíněnou kvadratickou funkci f. Minimum této funkce hledáme generováním posloupnosti $\{\mathbf{x}^k\}$, která je dána předpisem

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k, \tag{4.4}$$

kde $\mathbf{p}^k$ jsou konjugované směry vzhledem k matici $\mathbf{A}$.

V každém kroku algoritmu je třeba určit vhodnou délku kroku α_k. Stejně jako u Metody největšího spádu pro kvadratické funkce můžeme α_k určit pomocí tzv. *exaktní volby* délky kroku. Tudíž α_k bude ve tvaru

$$\alpha_k = -\frac{\mathbf{g}^{k^T}\mathbf{p}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}. \tag{4.5}$$

V následující větě si uvedeme důležitou vlastnost pro Metodu konjugovaných gradientů.

Věta 4.1. *Nechť je dán bod* $\mathbf{x}^0 \in \mathbb{R}^n$ *a předpokládejme, že posloupnost* $\{\mathbf{x}^k\}$ *splňuje podmínky (4.4) a (4.5). Pak*

$$\mathbf{g}^{k+1^T}\mathbf{p}^j = 0, \qquad \forall j = 0, \ldots, k, \tag{4.6}$$

přičemž $\mathbf{p}^j$ *jsou konjugované směry vzhledem k matici* $\mathbf{A}$.

Důkaz: viz [3], str. 106.

Poznámka 4.2. Věta 4.1 nám říká, že gradient $\mathbf{g}^{k+1}$ je kolmý na každý předchozí směr $\mathbf{p}^j$, pro $j = 0, \ldots, k$.

Metoda konjugovaných gradientů pracuje obdobně jako Metoda největšího spádu. Jejich rozdíl je ve volbě směrů, kterými se pohybujeme tak, abychom nalezli minimum dané funkce. V Metodě největšího spádu počítáme v každém kroku gradient funkce, přičemž právě gradient záporný nám udává směr největšího spádu. V případě Metody konjugovaných gradientů se nám výběr vhodného směru liší, a to tak, že volíme směry konjugované.

Předpokládejme, že máme dán bod $\mathbf{x}^0$. První směr $\mathbf{p}^0$ volíme jako záporný gradient $\mathbf{g}^0$, což podle Věty 3.3 je směr největšího spádu. Následující směry $\mathbf{p}^k$, pro $k = 1, 2, \ldots$, určíme jako lineární kombinace záporného gradientu $\mathbf{g}^k$ a předchozího směru $\mathbf{p}^{k-1}$, konkrétně

$$\mathbf{p}^k = -\mathbf{g}^k + \beta_k \mathbf{p}^{k-1}, \tag{4.7}$$

přičemž koeficienty $\beta_k \in \mathbb{R}$ musí splňovat takový požadavek, aby směry $\mathbf{p}^{k-1}$ a $\mathbf{p}^k$ byly směry konjugované vzhledem k matici $\mathbf{A}$. Vynásobíme-li vztah (4.7) zleva vektorem ${\mathbf{p}^{k-1}}^T\mathbf{A}$, pak za předpokladu konjugovanosti směrů $\mathbf{p}^{k-1}$, $\mathbf{p}^k$ (tj. ${\mathbf{p}^{k-1}}^T\mathbf{A}\mathbf{p}^k = 0$ (viz Definice 4.1)), dostáváme β_k ve tvaru

$$\beta_k = \frac{{\mathbf{g}^k}^T\mathbf{A}\mathbf{p}^{k-1}}{{\mathbf{p}^{k-1}}^T\mathbf{A}\mathbf{p}^{k-1}}. \tag{4.8}$$

Nyní si můžeme uvést základní verzi algoritmu Metody konjugovaných gradientů pro kvadratické funkce.

Algoritmus 4.1 (Základní algoritmus MKG pro kvadratické funkce)**.**

- zvolíme počáteční bod $\mathbf{x}^0$ a $\varepsilon > 0$ pro ukončovací kritérium
- vypočítáme $\mathbf{g}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$
- položíme $\mathbf{p}^0 = -\mathbf{g}^0$
- pro $k = 0, 1, 2, \ldots$ až po splnění ukončovacího kritéria vypočítáme

$$\begin{aligned}
\alpha_k &= -\frac{{\mathbf{g}^k}^T\mathbf{p}^k}{{\mathbf{p}^k}^T\mathbf{A}\mathbf{p}^k} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k\mathbf{p}^k \\
\mathbf{g}^{k+1} &= \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b} \\
\beta_{k+1} &= \frac{{\mathbf{g}^{k+1}}^T\mathbf{A}\mathbf{p}^k}{{\mathbf{p}^k}^T\mathbf{A}\mathbf{p}^k} \\
\mathbf{p}^{k+1} &= -\mathbf{g}^{k+1} + \beta_{k+1}\mathbf{p}^k.
\end{aligned}$$

V přesné aritmetice (bez zaokrouhlování) je algoritmus pro Metodu konjugovaných gradientů takzvaně finitní, což znamená, že nalezneme řešení $\mathbf{x}^*$ nejvýše v n krocích, viz následující věta.

Věta 4.2. *Předpokládejme, že k-tá iterace $\mathbf{x}^k$ generovaná Metodou konjugovaných gradientů není řešení $\mathbf{x}^*$. Platí-li podmínky*

$$\mathbf{p}^{i^T}\mathbf{A}\mathbf{p}^j = 0, \qquad \forall i \neq j \tag{4.9}$$

$$\mathbf{g}^{i^T}\mathbf{g}^j = 0, \qquad \forall i \neq j, \tag{4.10}$$

potom posloupnost $\{\mathbf{x}^k\}$ konverguje k $\mathbf{x}^*$ nejvýše v n krocích.

Důkaz: viz [3], str. 109.

Již jsme si uvedli základní algoritmus pro Metodu konjugovaných gradientů. Ovšem s využitím Vět 4.1 a 4.2 jsme schopni formulovat mnohem praktičtější verzi algoritmu. Tzn. že některé výpočty v Algoritmu 4.1 nahradíme jinými, lépe konstruovanými vzorečky. Tato vyjádření jsou výhodná z několika důvodů. Zajišťují nám rychlejší výpočty, šetří nám místo v paměti počítače a redukují počet násobení matice s vektorem.

Za prvé, s využitím výrazů $\mathbf{p}^{k+1}$ v Algoritmu 4.1 a (4.6) můžeme odvodit nový vzoreček pro α_k, a to následovně

$$\alpha_k = -\frac{\mathbf{g}^{k^T}\mathbf{p}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k} = -\frac{\mathbf{g}^{k^T}(-\mathbf{g}^k + \beta_k\mathbf{p}^{k-1})}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k},$$

přičemž po roznásobení čitatele dostáváme vztah

$$\alpha_k = \frac{\mathbf{g}^{k^T}\mathbf{g}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}. \tag{4.11}$$

Za druhé, pomocí výrazů (4.3) a (4.4) dostaneme

$$\mathbf{g}^{k+1} = \mathbf{g}^k + \alpha_k\mathbf{A}\mathbf{p}^k. \tag{4.12}$$

Odtud a opět s využitím výrazů $\mathbf{p}^{k+1}$ v Algoritmu 4.1 a (4.6) a také vztahu (4.10) odvodíme β_{k+1} následujícím způsobem

$$\beta_{k+1} = \frac{\mathbf{g}^{k+1^T}\mathbf{A}\mathbf{p}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k} = \frac{\mathbf{g}^{k+1^T}(\mathbf{g}^{k+1} - \mathbf{g}^k)}{\alpha_k\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k} =$$

$$= \frac{\mathbf{g}^{k+1^T}(\mathbf{g}^{k+1} - \mathbf{g}^k)}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}\frac{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k}{\mathbf{g}^{k^T}\mathbf{g}^k}.$$

Poté dostaneme

$$\beta_{k+1} = \frac{\mathbf{g}^{k+1^T}\mathbf{g}^{k+1}}{\mathbf{g}^{k^T}\mathbf{g}^k}. \tag{4.13}$$

Jsme tedy schopni vyjádřit modifikovaný tvar algoritmu pro Metodu konjugovaných gradientů.

Algoritmus 4.2 (Modifikovaný algoritmus MKG pro kvadratické funkce)**.**

- zvolíme počáteční bod $\mathbf{x}^0$ a $\varepsilon > 0$ pro ukončovací kritérium
- vypočítáme $\mathbf{g}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$
- položíme $\mathbf{p}^0 = -\mathbf{g}^0$
- pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria vypočítáme

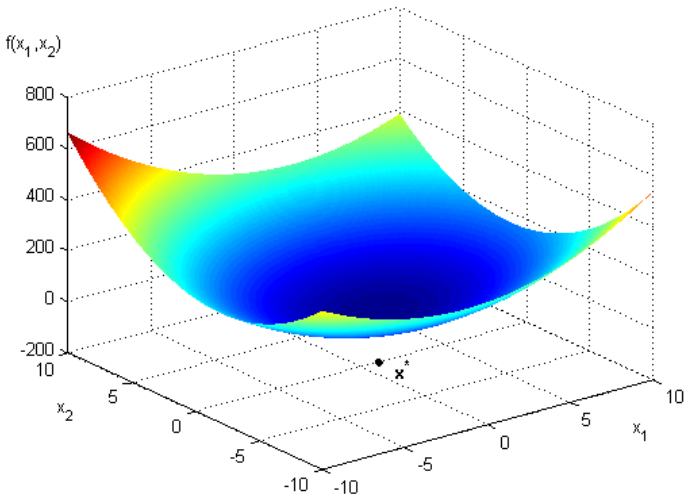
$$\begin{aligned}
\alpha_k &= \frac{\mathbf{g}^{k^T}\mathbf{g}^k}{\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k\mathbf{p}^k \\
\mathbf{g}^{k+1} &= \mathbf{g}^k + \alpha_k\mathbf{A}\mathbf{p}^k \\
\beta_{k+1} &= \frac{\mathbf{g}^{k+1^T}\mathbf{g}^{k+1}}{\mathbf{g}^{k^T}\mathbf{g}^k} \\
\mathbf{p}^{k+1} &= -\mathbf{g}^{k+1} + \beta_{k+1}\mathbf{p}^k.
\end{aligned}$$

Podívejme se nyní na konkrétní úlohu.

Příklad 4.1. Je dán bod $\mathbf{x}^0 = (0, 0)^T$. S použitím Metody konjugovaných gradientů nalezněte minimum funkce f ve tvaru

$$f(x_1, x_2) = 2x_1^2 - x_1x_2 + 3x_2^2 - 4x_1 + 2x_2 - 1.$$

Graf funkce f je znázorněn na Obrázku 5.



Obrázek 5: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 4.1

Řešení: Máme zadánu funkci f a počáteční bod $\mathbf{x}^0 \in \mathbb{R}^2$. Lokální minimum funkce f nalezené analyticky je bod $\mathbf{x}^* = \left(\frac{22}{23}, -\frac{4}{23}\right)^T$. Jednotlivé výsledky nebudeme zaokrouhlovat, ale ponecháme je ve zlomcích. Nemusíme tedy použít ukončovací kritérium, protože podle Věty 4.2 dospějeme k přesnému řešení nejvýše ve dvou krocích. Nejdříve vypočítáme gradient

$$\mathbf{g}(x_1, x_2) = (4x_1 - x_2 - 4, -x_1 + 6x_2 + 2)^T.$$

Jeho složky položíme rovny nule a dostaneme soustavu

$$\begin{aligned} 4x_1 - x_2 &= 4 \\ -x_1 + 6x_2 &= -2\ . \end{aligned} \tag{4.14}$$

Odtud můžeme vyjádřit matici $\mathbf{A}$ a vektor $\mathbf{b}$ ve tvaru

$$\mathbf{A} = \begin{pmatrix} 4 & -1 \\ -1 & 6 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 4 \\ -2 \end{pmatrix}.$$

Dále budeme postupovat podle Algoritmu 4.2. Vypočítáme

$$\mathbf{g}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b} = \begin{pmatrix} 4 & -1 \\ -1 & 6 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 4 \\ -2 \end{pmatrix}$$

$$\mathbf{g}^0 = \begin{pmatrix} -4 \\ 2 \end{pmatrix}.$$

Položíme

$$\mathbf{p}^0 = -\mathbf{g}^0 = \begin{pmatrix} 4 \\ -2 \end{pmatrix}.$$

Dosadíme do algoritmu a pro $k = 0$ vypočítáme

$$\alpha_0 = \frac{\mathbf{g}^{0T}\mathbf{g}^0}{\mathbf{p}^{0T}\mathbf{A}\mathbf{p}^0} = \frac{(-4,2)\begin{pmatrix} -4 \\ 2 \end{pmatrix}}{(4,-2)\begin{pmatrix} 4 & -1 \\ -1 & 6 \end{pmatrix}\begin{pmatrix} 4 \\ -2 \end{pmatrix}} = \frac{5}{26}$$

$$\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0\mathbf{p}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{5}{26}\begin{pmatrix} 4 \\ -2 \end{pmatrix} = \begin{pmatrix} \frac{10}{13} \\ -\frac{5}{13} \end{pmatrix}$$

$$\mathbf{g}^1 = \mathbf{g}^0 + \alpha_0\mathbf{A}\mathbf{p}^0 = \begin{pmatrix} 4 \\ -2 \end{pmatrix} + \frac{5}{26}\begin{pmatrix} 18 \\ -16 \end{pmatrix} = \begin{pmatrix} -\frac{7}{13} \\ -\frac{14}{13} \end{pmatrix}$$

$$\beta_1 = \frac{\mathbf{g}^{1T}\mathbf{g}^1}{\mathbf{g}^{0T}\mathbf{g}^0} = \frac{(-\frac{7}{13},-\frac{14}{13})\begin{pmatrix} -\frac{7}{13} \\ -\frac{14}{13} \end{pmatrix}}{(-4,2)\begin{pmatrix} -4 \\ 2 \end{pmatrix}} = \frac{49}{676}$$

$$\mathbf{p}^1 = -\mathbf{g}^1 + \beta_1\mathbf{p}^0 = \begin{pmatrix} \frac{7}{13} \\ \frac{14}{13} \end{pmatrix} + \frac{49}{676}\begin{pmatrix} 4 \\ -2 \end{pmatrix} = \begin{pmatrix} \frac{140}{169} \\ \frac{315}{338} \end{pmatrix}.$$

Pro $k = 1$ vypočítáme

$$\alpha_1 = \frac{\mathbf{g}^{1T}\mathbf{g}^1}{\mathbf{p}^{1T}\mathbf{A}\mathbf{p}^1} = \frac{26}{115}$$

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha_1\mathbf{p}^1 = \begin{pmatrix} \frac{22}{23} \\ -\frac{4}{23} \end{pmatrix}$$

$$\mathbf{g}^2 = \mathbf{g}^1 + \alpha_1\mathbf{A}\mathbf{p}^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\beta_2 = \frac{\mathbf{g}^{2T}\mathbf{g}^2}{\mathbf{g}^{1T}\mathbf{g}^1} = 0$$

$$\mathbf{p}^2 = -\mathbf{g}^2 + \beta_2\mathbf{p}^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Ve druhém kroku algoritmu vidíme, že gradient $\mathbf{g}^2$ a směr $\mathbf{p}^2$ jsou nulové a tudíž jsme skutečně dospěli k minimu funkce f ve tvaru

$$\mathbf{x}^* = \begin{pmatrix} \frac{22}{23} \\ -\frac{4}{23} \end{pmatrix}.$$

Na základě Algoritmu 4.2 jsem vytvořila následující M-soubor 4.1. Pro ukončovací kritérium si zvolíme test gradientu, jež je vyjádřen vztahem (3.4) a opět si zadáme maximální počet iterací, které Metoda konjugovaných gradientů bude provádět.

M-soubor 4.1 (MKG pro kvadratické funkce)**.**

```
function [x,krok] = mkgkvadr(A,b,c,x0,epsilon,maxiter)

% mkgkvadr - Metoda konjugovaných gradientů pro kvadratické funkce
%            f(x) = 1/2*x'*A*x - x'*b + c

% Vstupní hodnoty:
% A - symetrická matice vyjádřena z předpisu pro kvadr. funkci f(x)
% b - vektor vyjádřen z předpisu pro kvadr. funkci f(x)
% c - reálné číslo vyjádřeno z předpisu pro kv. funkci f(x)
% x0 - počáteční bod
% epsilon - přesnost, s jakou mkgkvadr nalezne hledané minimum
% maxiter - maximální počet iterací, které mkgkvadr provede

% Výstupní hodnoty:
% x - hledané minimum funkce f(x)
% krok - počet provedených iterací

x = x0;
g = A*x - b;
```

```
f = 1/2*dot(x,A*x) - dot(x,b) + c;
p = -g;
g0 = g;

% cyklus algoritmu
for krok = 1:maxiter
    gprev = g;
    alpha = dot(g,g)/dot(p,A*p);
    x = x + alpha*p;
    g = g + alpha*A*p;
    beta = dot(g,g)/dot(gprev,gprev);
    p = -g + beta*p;

    % ukončovací kritérium
    if (norm(g)/norm(g0) <= epsilon)
        break;
    end;
end;
```

S pomocí M-souboru 4.1 vyřešíme následující úlohu.

Příklad 4.2. Nechť je dána funkce tvaru

$$f(x_1, x_2) = 2x_1^2 + x_2^2 - x_1(2 + x_2) + 3(x_1 - x_2) - 2.$$

Zvolte si počáteční bod $\mathbf{x}^0 = (5, -3)^T$ a přesnost $\varepsilon = 10^{-4}$. S využitím Metody konjugovaných gradientů nalezněte minimum funkce f. Výsledné řešení zaokrouhlete na 5 desetinných míst.

Řešení: Nejdříve si zjistíme, jak vypadají matice $\mathbf{A}$, vektor $\mathbf{b}$ a parametr c. Z výrazu (4.1) dostáváme

$$\mathbf{A} = \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}, \qquad c = -2.$$

Maximální počet iterací si zvolíme 100 a do Matlabu zadáme

```
A=[4,-1;-1,2]; b=[-1;3]; c=-2;
[reseni,pocet_iteraci] = mkgkvadr(A,b,c,[5;-3],0.0001,100)
```

Matlab nám vrátí

```
reseni =
   0.14285714285714
   1.57142857142857


pocet_iteraci =
     2
```

Vidíme tedy, že již po 2 iteracích jsme s přesností $\varepsilon = 10^{-4}$ dospěli k hledanému minimu funkce f ve tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 0.14286 \\ 1.57143 \end{pmatrix}.$$

4.2 Nelineární Metoda konjugovaných gradientů

V předchozí kapitole jsme si představili Metodu konjugovaných gradientů pro kvadratické funkce a nyní se podíváme, jak to bude vypadat v případě funkcí obecných. Fletcher a Reeves rozšířili Metodu konjugovaných gradientů pro nelineární funkce několika jednoduchými změnami v Algoritmu 4.2.

Předpokládejme, že jsme v k-tém kroku. K určení parametru α_k zde nebudeme používat tzv. *exaktní volbu* délky kroku, poněvadž pro obecné funkce je to ve většině případů velice časově náročné. Proto využijeme např. Wolfeho podmínky či Armijovu podmínku zpětného vyhledávání. V případě gradientu $\mathbf{g}^k$ budeme obecně počítat gradient nelineární funkce. Prostřednictvím těchto změn si můžeme představit algoritmus pro Metodu konjugovaných gradientů obecné funkce.

Algoritmus 4.3 (Algoritmus MKG pro obecné funkce (F-R))**.**

- zvolíme počáteční bod $\mathbf{x}^0$ a $\varepsilon > 0$ pro ukončovací kritérium
- vypočítáme $\mathbf{g}^0 = \nabla f(\mathbf{x}^0)$
- položíme $\mathbf{p}^0 = -\mathbf{g}^0$
- pro $k = 0, 1, 2, \ldots$ až do splnění ukončovacího kritéria
  - vypočítáme α_k
  - položíme $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$
  - vypočítáme

$$\begin{aligned} \mathbf{g}^{k+1} &= \nabla f(\mathbf{x}^{k+1}) \\ \beta_{k+1} &= \frac{\mathbf{g}^{k+1^T}\mathbf{g}^{k+1}}{\mathbf{g}^{k^T}\mathbf{g}^k} \\ \mathbf{p}^{k+1} &= -\mathbf{g}^{k+1} + \beta_{k+1}\mathbf{p}^k. \end{aligned}$$

Algoritmus 4.3, jež si nazveme Algoritmus F-R, je vhodný pro velké nelineární optimalizační problémy, protože v každém jeho kroku je potřeba vypočítat pouze funkční hodnotu dané funkce f a její gradient. Nemusíme počítat s maticemi a v paměti máme uloženo pouze několik vektorů. Metoda konjugovaných gradientů s variantou Fletcher-Reeves je globálně konvergentní, což je detailněji popsáno v knize [3] na str. 121 - 122.

Algoritmus pro Metodu konjugovaných gradientů má několik variant, které se liší volbou výpočtu β_{k+1}. Další možností je např. varianta Polak-Ribière, pro níž je parametr β_{k+1} dán ve tvaru

$$\beta_{k+1} = \frac{(\mathbf{g}^{k+1} - \mathbf{g}^k)^T \mathbf{g}^{k+1}}{\mathbf{g}^{k^T}\mathbf{g}^k}. \tag{4.15}$$

Algoritmus se vzorečkem (4.15) budeme nazývat Algoritmus P-R. V případě, že funkce f je kvadratická a parametr α_k je určen exaktní volbou délky kroku,

jsou Algoritmus F-R a Algoritmus P-R identické. Vyplývá to z výrazu (4.10), který nám říká, že gradienty jsou vzájemně ortogonální. Jestliže však hledáme minimum funkce obecné, chování těchto dvou algoritmů se liší. Při studiu této problematiky jsem se v knize [3] dočetla, že Algoritmus P-R provádí sice větší množství výpočtů v jednotlivých krocích, ale zato je dle autorů účinnější.

Další variantou algoritmu Metody konjugovaných gradientů je varianta Hestenes-Stiefel, v tomto případě je parametr β_{k+1} vyjádřen následovně

$$\beta_{k+1} = \frac{(\mathbf{g}^{k+1} - \mathbf{g}^k)^T \mathbf{g}^{k+1}}{(\mathbf{g}^{k+1} - \mathbf{g}^k)^T \mathbf{p}^k}. \tag{4.16}$$

Algoritmus se vzorečkem pro β_{k+1} ve tvaru (4.16) si označíme jako Algoritmus H-S. Algoritmus P-R a Algoritmus H-S jsou si velice podobné, a to vzhledem k vlastnostem globální konvergence a také praktického využití.

Varianty Fletcher-Reeves, Polak-Ribière a Hestenes-Stiefel pro výpočet parametru β_{k+1} jsou tři často používané vztahy v algoritmu pro Metodu konjugovaných gradientů při hledání minima obecné funkce.

Nyní se můžeme podívat na konkrétní příklad. Pro názornost použijeme pro výběr parametru β_{k+1} variantu Fletcher-Reeves. Pokud bychom chtěli počítat i s ostatními variantami, stačí v Algoritmu 4.3 pouze nahradit výpočet pro β_{k+1} příslušným vztahem (4.15), resp. (4.16).

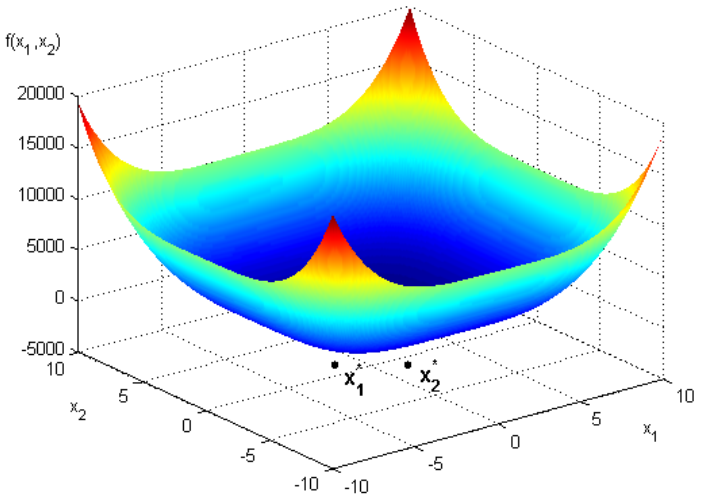
Příklad 4.3. Nechť je dána funkce

$$f(x_1, x_2) = x_1^4 + x_2^4 - 3x_1^2 + 2x_1x_2 - x_2^2 + 1$$

a počáteční bod $\mathbf{x}^0 = \left(-1, \frac{1}{2}\right)^T$. Pomocí Metody konjugovaných gradientů vypočítejte první dvě iterace posloupnosti $\{\mathbf{x}^k\}$. Graf dané funkce je znázorněn na Obrázku 6.

Řešení: Máme zadánu funkci f a počáteční bod $\mathbf{x}^0 \in \mathbb{R}^2$. Výsledky jednotlivých výpočtů budeme zaokrouhlovat na 5 desetinných míst. Nejdříve si zjistíme, jak bude vypadat gradient $\mathbf{g}(x_1, x_2)$. Vypočítáme

$$\mathbf{g}(x_1, x_2) = (4x_1^3 - 6x_1 + 2x_2, 4x_2^3 + 2x_1 - 2x_2)^T.$$



Obrázek 6: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 4.3

Opět jako v Příkladu 3.1 použijeme pro výpočet délky kroku α_k Armijovu podmínku zpětného vyhledávání. Nebudeme zde α_k počítat ručně, ale s pomocí M-souboru 3.1. Za parametry $\bar{\alpha}$, γ a δ si zvolíme hodnoty

$$\bar{\alpha} = 10$$
$$\gamma = 0.5$$
$$\delta = 0.001.$$

Dále budeme postupovat podle Algoritmu 4.3. Vypočítáme

$$\mathbf{g}^0 = \nabla f(\mathbf{x}^0) = \begin{pmatrix} 3 \\ -2.5 \end{pmatrix}$$

a položíme

$$\mathbf{p}^0 = -\mathbf{g}^0 = \begin{pmatrix} -3 \\ 2.5 \end{pmatrix}.$$

Do prostředí Matlabu si pro výpočet α_0 zadáme

```
x0=[-1;0.5];
g0=[3;-2.5];
```

```
p0=[-3;2.5];
fce=@(x)x(1).^4+x(2).^4-3*x(1).^2+2*x(1).*x(2)-x(2).^2+1;
alpha=armij(10,fce,x0,g0,p0,0.5,0.001)
```

a vrátí se nám

```
alpha =
   0.15625000000000
```

Dosadíme do algoritmu a pro $k = 0$ vypočítáme

$$\alpha_0 = 0.15625$$

$$\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0 \mathbf{p}^0 = \begin{pmatrix} -1 \\ 0.5 \end{pmatrix} + 0.15625 \begin{pmatrix} -3 \\ 2.5 \end{pmatrix} = \begin{pmatrix} -1.46875 \\ 0.89063 \end{pmatrix}$$

$$\mathbf{g}^1 = \nabla f(\mathbf{x}^1) = \begin{pmatrix} -2.07995 \\ -1.89289 \end{pmatrix}$$

$$\beta_1 = \frac{\mathbf{g}^{1T}\mathbf{g}^1}{\mathbf{g}^{0T}\mathbf{g}^0} = \frac{(-2.07995, -1.89289)\begin{pmatrix} -2.07995 \\ -1.89289 \end{pmatrix}}{(3, -2.5)\begin{pmatrix} 3 \\ -2.5 \end{pmatrix}} = 0.51864$$

$$\mathbf{p}^1 = -\mathbf{g}^1 + \beta_1 \mathbf{p}^0 = \begin{pmatrix} 2.07995 \\ 1.89289 \end{pmatrix} + 0.51864 \begin{pmatrix} -3 \\ 2.5 \end{pmatrix} = \begin{pmatrix} 0.52403 \\ 3.18949 \end{pmatrix}.$$

Pro výpočet α_1 napíšeme do Matlabu

```
x1=[-1.46875;0.89063];
g1=[-2.07995;-1.89289];
p1=[0.52403;3.18949];
fce=@(x)x(1).^4+x(2).^4-3*x(1).^2+2*x(1).*x(2)-x(2).^2+1;
alpha=armij(10,fce,x1,g1,p1,0.5,0.001)
```

a Matlab nám vrátí výsledek

```
alpha =
   0.07812500000000
```

Pro $k = 1$ vypočítáme

$$\begin{aligned}
\alpha_1 &= 0.07813 \\
\mathbf{x}^2 &= \mathbf{x}^1 + \alpha_1 \mathbf{p}^1 = \begin{pmatrix} -1.42781 \\ 1.13982 \end{pmatrix} \\
\mathbf{g}^2 &= \nabla f(\mathbf{x}^2) = \begin{pmatrix} -0.79667 \\ 0.78811 \end{pmatrix} \\
\beta_2 &= \frac{\mathbf{g}^{2T}\mathbf{g}^2}{\mathbf{g}^{1T}\mathbf{g}^1} = 0.15878 \\
\mathbf{p}^2 &= -\mathbf{g}^2 + \beta_2 \mathbf{p}^1 = \begin{pmatrix} 0.87988 \\ -0.28168 \end{pmatrix}.
\end{aligned}$$

V případě, že bychom chtěli pomocí Algoritmu 4.3 najít minimum funkce f, zvolíme si přesnost např. $\varepsilon = 10^{-4}$ a maximální počet iterací 100, pak s využitím následujícího M-souboru 4.2 dostáváme po 14 krocích minimum, které po zaokrouhlení na 5 desetinných míst je tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} -1.37437 \\ 1.06901 \end{pmatrix}.$$

Analytickým postupem bychom našli dvě lokální minima o souřadnicích

$$\mathbf{x}_1^* = \begin{pmatrix} -1.37438 \\ 1.06903 \end{pmatrix}, \qquad \mathbf{x}_2^* = \begin{pmatrix} 1.37438 \\ -1.06903 \end{pmatrix}.$$

Můžeme tedy vidět, že pomocí Metody konjugovaných gradientů jsme našli právě jedno minimum ze dvou možných, a to s přesností $\varepsilon = 10^{-4}$, což je způsobeno volbou počáteční aproximace. Pro nalezení druhého lokálního minima stačí zvolit jiný počáteční bod, např. $\mathbf{x}^0 = \left(1, -\frac{1}{2}\right)^T$.

Nyní si uvedeme M-soubor, který jsem vytvořila pro Algoritmus 4.3. Obdobně jako v ostatních M-souborech si mezi vstupní argumenty zadáme přesnost, s jakou budeme hledat minimum dané funkce a také maximální počet iterací, jež bude algoritmus provádět. Pro výpočet délky kroku α_k použijeme Armijovu

podmínku zpětného vyhledávání (viz M-soubor 3.1) a pro parametry $\bar{\alpha}$, γ a δ použijeme hodnoty z Poznámky 3.2. Ukončovací kritérium si můžeme zvolit stejné jako v případě Metody konjugovaných gradientů pro kvadratické funkce, což je testování gradientu vyjádřeno vztahem (3.4).

M-soubor 4.2 (MKG pro obecné funkce)**.**

```
function [xk,krok] = mkg(fce,x,x0,epsilon,maxiter)

% mkg - Metoda konjugovaných gradientů pro obecné funkce

% Vstupní hodnoty:
% fce - funkce, jejíž minimum hledáme, zadána
%       pomocí 'function_handle'
% x - symbolická proměnná
% x0 - počáteční bod
% epsilon - přesnost, s jakou mkg nalezne hledané minimum
% maxiter - maximální počet iterací, které mkg provede

% Výstupní hodnoty:
% xk - hledané minimum funkce
% krok - počet provedených iterací

n = length(x0);
FCE = fce(x);   % symbolická funkce

% výpočet gradientu funkce f
for (i = 1:n)
    G(i) = diff(FCE,x(i)); % symbolické derivování
end;

xk = x0;
```

```
g0 = (subs(G,x,x0))';
p = -g0;
g = g0;
f = fce(xk);

% vstupní hodnoty pro Armijovu podmínku
alpha0 = 10;
gamma = 0.5;
delta = 0.001;

% cyklus algoritmu
for krok = 0:maxiter
    xprev = xk;
    gprev = g;
    fprev = f;

    % Armij. podm.
    alpha = armij(alpha0,fce,xk,g,p,gamma,delta);

    xk = xk + alpha*p;
    g = (subs(G,x,xk))';
    beta = dot(g,g)/dot(gprev,gprev);
    p = -g + beta*p;
    f = fce(xk);

    % ukončovací kritérium
    if (norm(g)/norm(g0) <= epsilon)
            break;
    end;
end;
```

Podívejme se, jak M-soubor 4.2 pracuje, aplikujeme-li jej v následující úloze.

Příklad 4.4. Pomocí Metody konjugovaných gradientů nalezněte minimum funkce f s přesností $\varepsilon = 10^{-4}$. Funkce f bude mít stejné zadání i počáteční bod $\mathbf{x}^0$ jako v Příkladu 3.2.

Řešení: Zvolíme si maximální počet iterací 100 a v prostředí Matlabu napíšeme

```
x=sym('[x1,x2]');
fce=@(x)x(1).^2+5*x(1).*x(2)+x(2).^4-2*x(1)-3*x(2);
[reseni,pocet_iteraci]=mkg(fce,x,[2;1],0.0001,100)
```

Matlab nám vrátí

```
reseni =
   5.60636510899077
  -1.84278973425401


pocet_iteraci =
    22
```

Jestliže zaokrouhlíme na 5 desetinných míst, dostáváme s přesností $\varepsilon = 10^{-4}$ minimum ve tvaru

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 5.60637 \\ -1.84279 \end{pmatrix}$$

již po 22 krocích. Porovnáme-li $\mathbf{x}^{\varepsilon}$ s výsledkem v Příkladu 3.2, vidíme, že se nám jednotlivá řešení liší, což je způsobeno volbou ukončovacího kritéria. V případě Metody největšího spádu jsme zvolili testování funkční hodnoty a posunutí bodu a u Metody konjugovaných gradientů jsme použili testování gradientu.

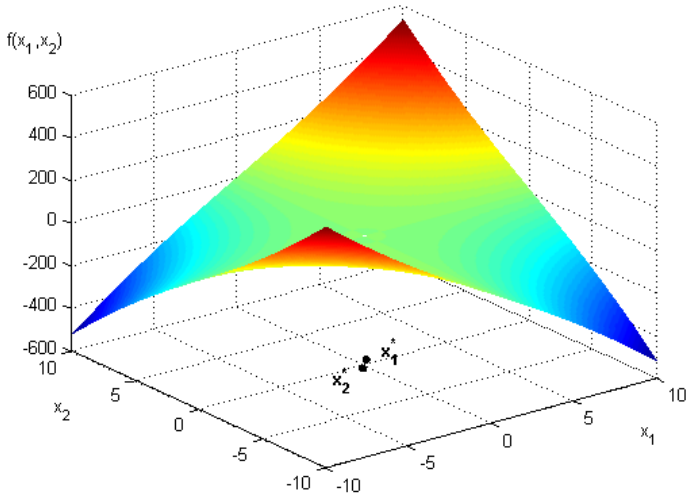
4.3 Praktické příklady

V předchozích podkapitolách 4.1 a 4.2 jsme si ukázali podstatu Metody konjugovaných gradientů. Dokázali jsme pomocí uvedených M-souborů nalézt s určitou přesností minimum dané funkce. Nyní se podívejme, jak se metoda bude chovat v případě, změníme-li některé vstupní hodnoty. Jednotlivé výsledky v následujících příkladech budeme zaokrouhlovat na 8 desetinných míst.

Příklad 4.5. Nechť funkce f je dána předpisem

$$f(x_1, x_2) = x_1 x_2 \ln(x_1^2 + x_2^2).$$

Její graf je znázorněn na Obrázku 7. Pomocí Metody konjugovaných gradientů nalezněte minimum funkce f s přesností $\varepsilon = 10^{-4}$. Pro parametr β_{k+1} použijte variantu Fletcher-Reeves. Zvolte si 5 různých počátečních bodů $\mathbf{x}_1^0 = \left(1, \frac{1}{2}\right)^T$, $\mathbf{x}_2^0 = \left(\frac{1}{2}, \frac{1}{2}\right)^T$, $\mathbf{x}_3^0 = \left(-1, -\frac{1}{2}\right)^T$, $\mathbf{x}_4^0 = (1, 1)^T$, $\mathbf{x}_5^0 = \left(\frac{3}{4}, 1\right)^T$ a porovnejte výsledná řešení.



Obrázek 7: Zobrazení funkce $f(x_1, x_2)$ z Příkladu 4.5

Řešení: Máme zadanou funkci f. Analyticky lze najít dvě různá lokální minima ve tvaru

$$\mathbf{x}_1^* = \begin{pmatrix} \frac{1}{\sqrt{2e}} \\ \frac{1}{\sqrt{2e}} \end{pmatrix}, \qquad \mathbf{x}_2^* = \begin{pmatrix} -\frac{1}{\sqrt{2e}} \\ -\frac{1}{\sqrt{2e}} \end{pmatrix}.$$

Vzhledem k tomu, že zadaná funkce f je obecně nekvadratická, k nalezení minima použijeme Metodu konjugovaných gradientů pro obecné funkce. Budeme tedy počítat pomocí M-souboru 4.2. Maximální počet iterací si zvolíme 100. Vezmeme si počáteční bod $\mathbf{x}_1^0$ a do prostředí Matlabu napíšeme

```
x=sym('[x1,x2]');
fce=@(x)x(1).*x(2).*log(x(1).^2+x(2).^2);
[reseni,pocet_iteraci]=mkg(fce,x,[1;1/2],0.0001,100)
```

a vrátí se nám

```
reseni =
  -0.42884449108892
  -0.42888561280323


pocet_iteraci =
    10
```

Analogickým postupem získáme řešení i pro zbývající čtyři počáteční body. Výsledná data zaokrouhlíme na 8 desetinných míst a zapíšeme do Tabulky 1.

Víme, že funkce f má dvě lokální minima, jež jsme si uvedli výše. Pro přehlednost si je vyjádříme v desetinných číslech. Tedy

$$\mathbf{x}_1^* = \begin{pmatrix} 0.42888194 \\ 0.42888194 \end{pmatrix}, \qquad \mathbf{x}_2^* = \begin{pmatrix} -0.42888194 \\ -0.42888194 \end{pmatrix}.$$

Z Tabulky 1 vidíme, že nalezená řešení se pohybují velice blízko obou lokálních minim. Můžeme tedy vidět, že výběr počátečního bodu má vliv na rychlost nalezení řešení a v případě, že má daná funkce více než jedno lokální minimum,

Tabulka 1: Výsledná řešení z Příkladu 4.5 pro různé počáteční body

počáteční bod $\mathbf{x}^0$	počet iterací	minimum $\mathbf{x}^\varepsilon$
$\mathbf{x}_1^0 = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}$	$k = 10$	$\mathbf{x}_1^\varepsilon = \begin{pmatrix} -0.42884449 \\ -0.42888561 \end{pmatrix}$
$\mathbf{x}_2^0 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$	$k = 6$	$\mathbf{x}_2^\varepsilon = \begin{pmatrix} 0.42887942 \\ 0.42887942 \end{pmatrix}$
$\mathbf{x}_3^0 = \begin{pmatrix} -1 \\ -\frac{1}{2} \end{pmatrix}$	$k = 10$	$\mathbf{x}_3^\varepsilon = \begin{pmatrix} 0.42884449 \\ 0.42888561 \end{pmatrix}$
$\mathbf{x}_4^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$k = 7$	$\mathbf{x}_4^\varepsilon = \begin{pmatrix} -0.42883068 \\ -0.42883068 \end{pmatrix}$
$\mathbf{x}_5^0 = \begin{pmatrix} \frac{3}{4} \\ 1 \end{pmatrix}$	$k = 6$	$\mathbf{x}_5^\varepsilon = \begin{pmatrix} -0.42881381 \\ -0.42888836 \end{pmatrix}$

má také vliv na to, ke kterému z nich bude posloupnost generovaná danou metodou konvergovat. Čím blíže si zvolíme počáteční bod od hledaného minima, tím méně iterací se provádí.

Nyní se podívjeme, jaký vliv na hledání minima bude mít volba parametru β_{k+1}.

Příklad 4.6. Nechť je dána funkce f ve tvaru

$$f(x_1, x_2) = 3(x_1 - 2x_2)^2 + 7x_1x_2 - 2(1 - 4x_2) - 7x_1$$

a počáteční bod $\mathbf{x}^0 = \left(-\frac{1}{2}, \frac{3}{4}\right)^T$. Pomocí Metody konjugovaných gradientů najděte s přesností $\varepsilon = 10^{-4}$ minimum funkce f. Pro výpočet β_{k+1} si zvolte varianty

Fletcher-Reeves, Polak-Ribière a Hestenes-Stiefel a porovnejte výsledná řešení.

Řešení: Víme, že zadaná funkce f je kvadratická, proto použijeme Metodu konjugovaných gradientů pro kvadratické funkce. K nalezení minima využijeme příslušný M-soubor 4.1. Maximální počet iterací si zvolíme 100. Vzhledem k tomu, že v M-souboru 4.1 je varianta Fletcher-Reeves pro β_{k+1} již uvedena, stačí do Matlabu napsat

```
A=[6,-5;-5,24]; b=[7;-8]; c=-2;
[reseni,pocet_iteraci]=mkgkvadr(A,b,c,[-1/2;3/4],0.0001,100)
```

a Matlab nám vrátí

```
reseni =
   1.07563025210084
  -0.10924369747899


pocet_iteraci =
     2
```

Varianta Polak-Ribière parametru β_{k+1} je vyjádřena vztahem (4.15). V M-souboru 4.1 nahradíme stávající β_{k+1} výrazem

```
beta = dot((g-gprev),g)/dot(gprev,gprev);
```

a zavoláme funkci

```
[reseni,pocet_iteraci]=mkgkvadr(A,b,c,[-1/2;3/4],0.0001,100)
```

V tomto případě nám Matlab vrátí

```
reseni =
   1.07563025210084
  -0.10924369747899


pocet_iteraci =
     2
```

Pro variantu Hestenes-Stiefel, jež je vyjádřena ve tvaru (4.16), přepíšeme v M-souboru 4.1 vzorec pro β_{k+1} následovně

```
beta = dot((g-gprev),g)/dot((g-gprev),p);
```

a opět zavoláme funkci

```
[reseni,pocet_iteraci]=mkgkvadr(A,b,c,[-1/2;3/4],0.0001,100)
```

a vrátí se nám

```
reseni =
   1.07563025210084
  -0.10924369747899


pocet_iteraci =
     2
```

Můžeme tedy vidět, že volba parametru β_{k+1} nám nijak neovlivnila řešení a ve všech třech případech jsme již po 2 krocích s přesností $\varepsilon = 10^{-4}$ dospěli k hledanému minimu

$$\mathbf{x}^{\varepsilon} = \begin{pmatrix} 1.07563025 \\ -0.10924370 \end{pmatrix}.$$

Jestliže bychom chtěli $\mathbf{x}^{\varepsilon}$ srovnat s přesným řešením $\mathbf{x}^*$, jež bychom nalezli analyticky a zaokrouhlili na 8 desetinných míst, dospěli bychom k závěru, že $\mathbf{x}^* = \mathbf{x}^{\varepsilon}$.

Nyní se podívejme, jaký vliv bude mít výběr parametru β_{k+1}, jestliže budeme hledat minimum obecné nekvadratické funkce.

Příklad 4.7. Je dána funkce f ve tvaru

$$f(x_1, x_2) = x_1^4 + x_2^4 + 6(2x_1^2 + x_2^2) - 3x_1x_2 - (x_1 - x_2) - 1.$$

Za počáteční bod zvolte $\mathbf{x}^0 = (-2, 3)^T$. S využitím Metody konjugovaných gradientů najdetě minimum funkce f s přesností $\varepsilon = 10^{-4}$. Pro parametr β_{k+1} zadejte postupně varianty Fletcher-Reeves, Hestenes-Stiefel a Polak-Ribière.

Řešení: Vzhledem k tomu, že zadaná funkce f je obecná, k výpočtu řešení využijeme M-soubor 4.2, jenž je naprogramován pro Algoritmus F-R. Zvolíme si maximálně 100 iterací a zapíšeme do Matlabu

```
x=sym('[x1,x2]');
fce=@(x)
x(1).^4+x(2).^4+6*(2*x(1).^2+x(2).^2)-3*x(1).*x(2)-(x(1)-x(2))-1;
[reseni,pocet_iteraci]=mkg(fce,x,[-2;3],0.0001,100)
```

a vrátí se nám

```
reseni =
   0.03262691405957
  -0.07589329068817


pocet_iteraci =
    35
```

Analogicky jako v předchozím Příkladu 4.6 si pro variantu Hestenes-Stiefel přepíšeme v M-souboru 4.2 vzorec pro β_{k+1} a opět zavoláme funkci

```
[reseni,pocet_iteraci]=mkg(fce,x,[-2;3],0.0001,100)
```

Matlab nám vrátí

```
reseni =
   0.03269031501108
  -0.07561817408165


pocet_iteraci =
    52
```

V případě varianty Polak-Ribière dostáváme

```
reseni =
   0.03295961278966
  -0.07528772530672

pocet_iteraci =
    88
```

Výsledná data zaokrouhlíme na 8 desetinných míst a zapíšeme do Tabulky 2.

Tabulka 2: Řešení Příkladu 4.7 při různé volbě parametru β_{k+1}

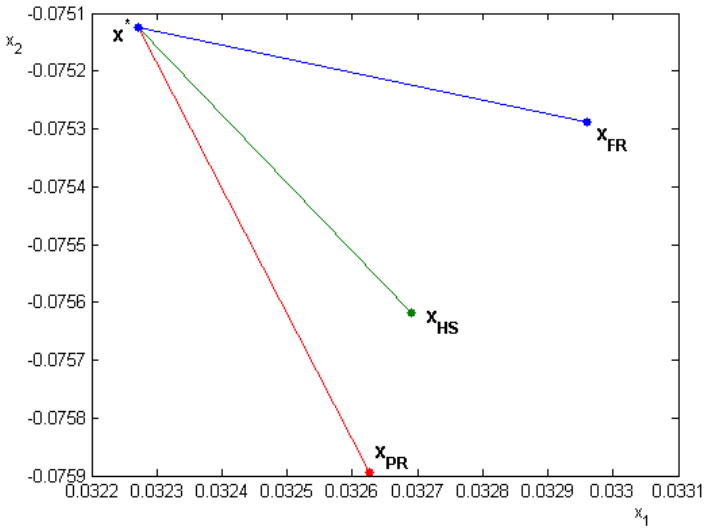
varianta parametru β_{k+1}	minimum $\mathbf{x}^{\varepsilon}$	počet iterací
Fletcher-Reeves	$\mathbf{x}^{\varepsilon}_{\mathrm{FR}} = \begin{pmatrix} 0.03262691 \\ -0.07589329 \end{pmatrix}$	$k = 35$
Hestenes-Stiefel	$\mathbf{x}^{\varepsilon}_{\mathrm{HS}} = \begin{pmatrix} 0.03269032 \\ -0.07561817 \end{pmatrix}$	$k = 52$
Polak-Ribière	$\mathbf{x}^{\varepsilon}_{\mathrm{PR}} = \begin{pmatrix} 0.03295961 \\ -0.07528773 \end{pmatrix}$	$k = 88$

V Tabulce 2 můžeme vidět, že jednotlivá řešení a rychlost jejich nalezení se výběrem parametru β_{k+1} liší. V tomto případě je varianta Fletcher-Reeves nejrychlejší, našli jsme minimum s přesností $\varepsilon = 10^{-4}$ již po 35 krocích. Naopak v případě varianty Polak-Ribiere bylo potřeba vypočítat až 88 iterací.

Výsledná data můžeme také porovnat s přesným řešením

$$\mathbf{x}^{*} = \begin{pmatrix} 0.03227052 \\ -0.07512438 \end{pmatrix},$$

které lze nalézt analyticky. Na Obrázku 8 vidíme, že bod $\mathbf{x}^{\varepsilon}_{\mathrm{HS}}$ vypočítán pomocí Algoritmu H-S se nachází nejblíže přesnému řešení $\mathbf{x}^{*}$.



Obrázek 8: Grafické znázornění bodů z Příkladu 4.7

V průběhu této práce jsme v jednotlivých příkladech pracovali pouze s funkcemi dvou proměnných, a to především z důvodu přehlednosti. Metodu konjugovaných gradientů však lze aplikovat také na funkce tří a více proměnných. Rozdíl při aplikaci M-souborů bude pouze v zadávání vstupních hodnot.

Závěr

V prvních dvou kapitolách této práce jsme si definovali potřebné pojmy a seznámili jsme se problematikou optimalizace, konkrétně s úlohou nepodmíněné optimalizace. Poté jsme si uvedli nutné a postačující podmínky optimality, jež jsme následně použili na konkrétním příkladu.

Ve třetí kapitole jsme si ukázali podstatu spádových metod a jak je lze použít pří minimalizaci funkcí. Řekli jsme si něco o Armijově podmínce zpětného vyhledávání a potřebných ukončovacích kritériích. Poté jsme si představili Metodu největšího spádu, jež hledá minimum funkce ve směru záporného gradientu, nebo-li „největšího spádu“.

V poslední, klíčové, kapitole jsme se seznámili s Metodou konjugovaných gradientů. Zjistili jsme, že její podstata je obdobná jako u Metody největšího spádu, ovšem s tím rozdílem, že při hledání minima funkce se pohybujeme v tzv. konjugovaných směrech. Tuto kapitolu jsme si rozdělili na tři části. Nejdříve jsme si uvedli metodu lineární, jež se zabývá kvadratickými funkcemi. Ukázali jsme si, jak vypadá základní algoritmus a poté jsme jednoduchými úpravami vytvořili početně ekonomičtější, tzv. modifikovaný algoritmus. Na základě těchto informací jsme naprogramovali M-soubor, jež jsme následně aplikovali na příkladech. Ve druhé části kapitoly si představili nelineární Metodu konjugovaných gradientů, nebo-li metodu pro obecné funkce. Zavedli jsme si tři nejpoužívanější varianty algoritmu - Fletcher-Reeves, Polak-Ribière a Hestenes-Stiefel, jež se v praktickém využití trochu liší. V poslední části jsme si uvedli několik příkladů, na kterých jsme aplikovali Metodu konjugovaných gradientů pro kvadratické a obecné funkce. Pozměnili jsme některé vstupní hodnoty a zjišťovali jsme, jak nám to ovlivní rychlost nalezení řešení a jejich přesnost.

Při psaní této práce jsem čerpala zejména z anglicky psané literatury a vycházela jsem také ze znalostí získaných z předmětu Numerické metody optimalizace, který vyučuje RNDr. Horymír Netuka, Ph.D., jehož studijní podklady jsou dostupné v elektronické podobě na http://elearning.math.upol.cz. Bakalářská práce pro mne byla velkým přínosem, neboť jsem získala mnoho zajímavých informací a

rozšířila jsem si své znalosti týkající se použití matematických programů Matlab a Tex.

Jsem si plně vědoma obsáhlosti problematiky optimalizace, jejíž důležitou součástí je také konvergence. Vzhledem k rozsahu práce jsem se rozhodla zaměřit prezentaci nastudovaných metod na konkrétní úlohy a konstrukci M-souborů. Ovšem otázce konvergence bych se ráda věnovala ve své diplomové práci. V případě zájmu si čtenář může nastudovat konvergenci Metody konjugovaných gradientů např. v uvedené literatuře.

Literatura

[1] Fiedler M., *Speciální matice a jejich použití v numerické matematice*, SNTL, Praha 1981

[2] Fletcher R., *Practical Methods Of Optimization*, Second Edition, Wiley, 2000

[3] Nocedal J., Wright S. J., *Numerical Optimization*, Second Edition, Springer, 2006