VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Algoritmus využívající kvadratické interpolace pro minimalizaci funkce bez omezení

An algorithm using quadratic interpolation for unconstrained optimization

2015

Štěpán Bednařík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student: **Štěpán Bednařík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Algoritmus využívající kvadratické interpolace pro minimalizaci funkce bez omezení
An algorithm using quadratic interpolation for unconstrained optimization

Zásady pro vypracování:

Pro úlohu optimalizace funkce bez omezení existuje řada iteračních metod, které vyžadují znalost první derivace (gradientu) a případně i druhé derivace (Hessiánu) funkce v každé iteraci. Jde například o tzv. gradientní či Newtonovy metody. Existují ale metody (někdy se jim říká metody nultého řádu), které dokážou minimalizovat funkci i bez použití gradientu a Hessiánu. Mezi tyto metody patří také metoda využívající kvadratickou interpolaci funkce. Úkolem práce je seznámit se s touto metodou a její následná implementace.

Práce bude mít tyto části:
Kvadratická interpolace funkce
Algoritmy pro minimalizaci spojitě diferencovatelných funkcí
Programová realizace v Matlabu
Testování algoritmů na vhodných úlohách

Seznam doporučené odborné literatury:
A.R.Conn, Ph.L.Toint: An algorithm using quadratic interpolation for unconstrained derivative free optimization, Nonlinear Optimization and Applications, 1996.

Dále dle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Beremlijski, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Abstrakt

Práce se zabývá metodami pro minimalizaci funkce bez omezení. Klasické metody vyžadují znalost první derivace (gradientu), případně i druhé derivace (Hessiánu). Pokud je náročné tyto hodnoty vypočítat, pak se používají tzv. metody nultého řádu. Mezi tyto metody patří i metoda využívající kvadratické interpolace funkce. Úkolem této práce je seznámení s touto metodou, popis jejích základních rysů a následně popis algoritmu pro výpočet a celková implementace kódu.

Klíčová slova: nelineární optimalizace, metody optimalizace nultého řádu, Trust-region, kvadratická interpolace.

Abstract

This work is devoted to methods for unconstrained minimization. Classical methods need the knowledge of first derivation (gradient) eventually second derivation (Hessian). However, if both of these numbers are hard to evaluate, we use derivative free algorithms. An algorithm using quadratic interpolation for unconstrained optimization is one of them. The goal of this work is to describe the basics features of this method, to cover the whole algorithm and then to implement the code of program.

Keywords: nonlinear optimization, derivative free algorithms, Trust-region, quadratic interpolation.

Seznam použitých zkratek a symbolů

$\arg\max$	–	argument maxima
$\arg\min$	–	argument minima
$\|\cdot\|$	–	Eukleidovská norma
$\|\cdot\|_F$	–	Frobeniova norma
$\langle\cdot,\cdot\rangle$	–	skalární součin
$\|\cdot\|$	–	velikost množiny, absolutní hodnota
v^T	–	transponování
G	–	gradient funkce
H	–	Hessián funkce

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Úkolem bakalářské práce bylo sestavení algoritmu pro metodu využívající kvadratické interpolace pro minimalizaci funkce bez omezení. Tento text se zabývá jeho tvorbou, ale i různými jinými metodami pro řešení problému nalezení minima hledané funkce.
V druhé kapitole se přesně definuje námi řešený matematický problém a popisují se jeho specifika (jaké podmínky musí splňovat funkce apod.).
Třetí kapitola obsahuje popis některých metod minimalizace. Uvádíme zde metodu gradientní a Newtonovu, které rozebíráme podrobněji a také je zde uveden algoritmus pro nalezení řešení minimalizační úlohy danou metodou. Na konci kapitoly se také objevuje krátký popis dalších tzv. genetických algoritmů.
Ve čtvrté kapitole už se poté dostáváme k metodě využívající kvadratickou interpolaci. Zde je podrobně popsáno jaké má metoda základní rysy a jak pracuje. Nejprve se zaměříme na tzv. Trust-region metody a poté na interpolaci kvadratickou funkcí. Následuje popis algoritmu úpravy geometrie množin a poté se probírají různé náležitosti daného algoritmu, jako např. hledání nového iteračního bodu, nebo správná volba ukončovacího kritéria. Na závěr kapitoly je poté uveden celkový algoritmus pro tuto metodu a ještě je přidána poznámka pro minimalizaci funkce více proměnných.
V poslední (páté) kapitole poté testujeme algoritmus na konkrétních příkladech. Numerické testy jsou provedeny na čtyřech úlohách, u kterých zkoumáme rozdíly výsledků při volbě jiných počátečních parametrů. Výsledky jsou uvedeny v tabulkách. Jsou zde uvedeny také grafy tzv. konvergenčních historií, které popisují jak rychle daná metoda konverguje k minimu.

2 Formulace problému

Metody optimalizace (viz [2], [4]) se snaží o nalezení hodnot, pro které daná funkce nabývá maximální či minimální hodnoty. Budeme se zabývat problémem minimalizace funkce, jejíž hodnoty můžou být zjištěny při nějakém reálném měření, např. při modelování fyzikálních jevů. Zaměříme se na funkce, které nekladou žádná omezení na dané proměnné. Danou úlohou optimalizace můžeme zapsat jako hledání $x^* \epsilon R^n$, které je řešením následujícího problému:

$$\min_{x \epsilon R^n} f(x). \tag{1}$$

Hledáme tedy x^*, pro které daná funkce nabývá nejmenších hodnot, tzn. pro x^* platí:

$$f(x^*) \leq f(x), \forall x \epsilon R^n. \tag{2}$$

Předpokládáme, že funkce $f(x) : R^n \rightarrow R$ je dvakrát spojitě diferencovatelná. Metoda, kterou se zabýváme, pracuje s funkcemi, u nichž je nákladné vypočítat funkční hodnotu i derivaci. Pro daný problém existuje řada klasických metod využívajících znalostí gradientu nebo Hessiánu. O těchto metodách si řekneme v příští kapitole. Protože chceme pracovat s funkcemi kde je výpočet gradientu i Hessiánu náročný, budeme se zabývat metodou, která je nepoužívá.

3 Přehled optimalizačních metod

Obvykle problém (1) řešíme pomocí gradientní nebo Newtonovy metody.

3.1 Gradientní metoda

V případě, že funkce je spojitě diferencovatelná používáme gradientní metodu (metodu největšího spádu), která v každé iteraci využívá směr poklesu (záporný gradient) funkce k nalezení menší hodnoty. Pro ilustraci dané metody uvádíme algoritmus:

```
1  Input: x_0, ε > 0
2
3  while || G(x_k) ||> ε
4      t_k = arg min_{t>0} f(x_k − t · G(x_k))
5      x_{k+1} = x_k − t_k · G(x_k)
6  end
```

Výpis 1: Gradientní metoda

G představuje gradient funkce. Gradientní metody jsou velmi jednoduché na implementaci, často ale dosahují velkého počtu iterací.

3.2 Newtonova metoda

Pokud je funkce dvakrát spojitě diferencovatelná, pak je nejvhodnější použít Newtonovu metodu. Tato metoda pracuje se znalostí gradientu a Hessiánu a celkově dosahuje daleko menšího počtu iterací než gradientní metoda. Opět uvádíme algoritmus dané metody:

```
1  Input: x_0, ε > 0
2
3  while || G(x_k) ||> ε
4      x_{k+1} = x_k − H(x_k)^{−1} · G(x_k)
5  end
```

Výpis 2: Newtonova metoda

G představuje gradient funkce, H představuje Hessián funkce.

3.3 Ostatní metody

Pokud je výpočet gradientu a Hessiánu drahý, používáme genetické algoritmy, které pracují pouze se znalostí funkčních hodnot. Hlavním rysem genetických algoritmů je výpočet značného počtu funkčních hodnot. Pokud se tomu chceme vyhnout, můžeme použít jiné metody nultého řádu, mezi které patří i metoda kvadratické interpolace. Tuto metodu popíšeme podrobněji v další kapitole. Více lze o uvedených metodách nalézt v [2], [4], [5].

4 Metoda využívající kvadratickou interpolaci

Metoda využívající kvadratickou interpolaci se uplatňuje u funkcí, pro které platí, že jejich funkční hodnoty a derivace jsou nákladné na výpočet. Jejími základními rysy jsou využití Trust-regionu a interpolace kvadratickou funkcí. Více se lze s touto metodou seznámit v [1].

4.1 Trust-region

Algoritmus, který popisujeme patří mezi tzv. Trust-region metody. Metoda je založena na zkoumání funkce v Δ okolí bodu x_c. Poloměr daného okolí (Δ) se nazývá Trust-region a udává velikost okolí, kde očekáváme dobrou shodu modelu a funkce.
Je vypočten nový iterační bod, který minimalizuje model v daném Δ okolí bodu x_c a je vypočtena skutečná hodnota funkce v daném bodě.
Pokud tato hodnota dostatečně odpovídá hodnotě vypočtené modelem, nový iterační bod je přijat a Δ je zvětšeno.
Naopak, pokud je tato hodnota nedostačující vůči hodnotě vypočtené modelem, nový iterační bod je zamítnut a Δ je zmenšeno.
Tento proces se opakuje dokud není splněna ukončující podmínka.
Poměr mezi dosaženou a předpokládanou hodnotou se vypočítá takto:

$$\rho = \frac{f(x_c + s) - f(x_c)}{m(x_c + s) - m(x_c)}. \tag{3}$$

Bod x_c představuje aktuální bod iterace, bod $x_c + s$ představuje nový iterační bod. Hodnota $f(\cdot)$ je skutečná hodnota funkce v daném bodě a hodnota $m(\cdot)$ představuje hodnotu vypočtenou modelem pro trust-region. Čitatel ve zlomku (3) se označuje jako skutečný pokles a jmenovatel jako předpokládaný pokles.

4.2 Interpolace kvadratickou funkcí

Základní podmínkou pro metodu trust region je volba správného modelu funkce. V případě metody kvadratické interpolace se volí kvadratický model ve formě:

$$m(x_c + s) = f(x_c) + \langle G, s\rangle + \frac{1}{2}\langle s, Hs\rangle, \tag{4}$$

kde G je gradient a H je Hessián funkce, $f(\cdot)$ značí hodnotu funkce v daném bodě a $m(\cdot)$ je hodnota modelu v daném bodě.
Jelikož počítáme s funkcemi, u kterých je složité určit gradient a Hessián pomocí první a druhé derivace, určíme model (4) interpolací pomocí funkčních hodnot předchozích bodů. Tato interpolace bude jednoznačně určena soustavou rovnic:

$$m(x) = f(x) \tag{5}$$

pro všechny body z množiny I takové, že $f(x)$ je známé pro všechny $x \epsilon I$. Velikost množiny I musí být rovna:

$$p = \frac{1}{2}(n+1)(n+2) \tag{6}$$

pro přesné určení kvadratického modelu rovnicí (5). V rovnici (6) představuje n dimenzi funkce f.
Body v množině I musí také splňovat určité geometrické vlastnosti: např. 6 bodů v přímce neurčuje kvadratický model pro funkci v R^2. Musíme tedy zjistit, zda jsou body množiny I uspořádány tak, aby bylo možné s jejich pomocí sestrojit kvadratický model funkce. Tento problém vyřešíme pomocí matice bázových vektorů:

$$\delta(I) = det \begin{pmatrix} \phi_1(x_1) & \dots & \phi_1(x_p) \\ \vdots & & \vdots \\ \phi_p(x_p) & ... & \phi_p(x_p) \end{pmatrix}, \tag{7}$$

kde $\phi_1 \dots \phi_p$ představují jednotlivé bázové vektory prostoru kvadratických funkcí v R^n. Ověřujeme, jestli je $\delta(I) \neq 0$. Pokud ano je daná množina zvolena správně a poté řekneme, že je daný model dobrý. Pokud je $\delta(I)$ nulový, body v množině I jsou uspořádány špatně a musíme tedy zvolit jinou geometrii bodů.

4.3 Algoritmus úpravy geometrie množin (A1)

Nyní popíšeme algoritmus, který se využívá k vylepšení kvadratického modelu po neúspěšné iteraci. V algoritmu se využívá množin I, J, M. I je množina, pomocí které interpolujeme kvadratický model (množina zmíněna v předchozím textu) a je to množina obsahující body, u kterých známe funkční hodnotu. M obsahuje body, u kterých byla v průběhu iteračního procesu vypočtena funkční hodnota. J je množina obsahující body, ve kterých dosud neznáme funkční hodnotu a přitom množina $I \cup J$ poskytuje dobrý kvadratický model. Daný algoritmus vypadá následovně:

1 Input: M, I, J, Δ, x_c
2 Step 1: $\forall x_i \epsilon M \setminus I : \|x_i - x_c\| \leq \Delta$
3 $x_- = \arg\max_{x_j \epsilon (I \cup J)} S(x_i, x_j)$
4 **if** $S(x_i, x_-) \geq 2$
5 $I = (I \setminus \{x_-\}) \cup \{x_i\}$
6 **return**
7 **end**;
8 Step 2: $x_- = \arg\max_{x_i \epsilon J} \|x_i - x_c\|$
9 **if** $\|x_- - x_c\| > 2\Delta$
10 $x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_-)$
11 Vypočítej $f(x_+)$
12 $M = M \cup \{x_+\}$

13 $I = I \cup \{x_+\}$
14 $J = J \setminus \{x_-\}$
15 **return**
16 **end**;
17 Step 3: $x_- = \arg\max_{x_i \epsilon I} \|x_i - x_c\|$
18 **if** $\|x_- - x_c\| > 2\Delta$
19 $x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_-)$
20 Vypočítej $f(x_+)$
21 $M = M \cup \{x_+\}$
22 $I = (I \setminus \{x_-\}) \cup \{x_+\}$
23 **return**
24 **end**;
25 Step 4: $x_- = \arg\max_{x_i \epsilon J} \left[\arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i) \right]$
26 $x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i)$
27 **if** $S(x_+, x_-) > 1$
28 Vypočítej $f(x_+)$
29 $M = M \cup \{x_+\}$
30 $I = I \cup \{x_+\}$
31 $J = J \setminus \{x_-\}$
32 **return**
33 **end**;
34 Step 5: $x_- = \arg\max_{x_i \epsilon I} \left[\arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i) \right]$
35 $x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i)$
36 **if** $S(x_+, x_-) \geq 2$
37 Vypočítej $f(x_+)$
38 $M = M \cup \{x_+\}$
39 $I = (I \setminus \{x_-\}) \cup \{x_+\}$
40 **return**
41 **end**;
42 Step 6: Algoritmus selhal v úpravě množin
43 **return**
44 **end**

Výpis 3: Algoritmus A1

Δ je poloměr daného trust-regionu a x_c je aktuální bod iterace (střed trust-regionu).
V algoritmu se objevuje funkce $S(\cdot, \cdot)$, kterou požíváme pro zjištění, jestli se dají dané množiny upravit (pokud se dá upravit geometrie daných množin).

Funkce $S(x, x_-)$ má tvar:

$$S(x, x_-) = \frac{|L(x, x_-)|}{\min\left[1, \max\limits_{t\epsilon[0,\Delta]} |L(x_c + td_-, x_-)|\right]}, \tag{8}$$

kde x_c představuje aktuální bod iterace a d_- se vypočítá jako:

$$d_- = \frac{(x_- - x_c)}{\|x_- - x_c\|}. \tag{9}$$

Ve funkci se dále objevuje výpočet pro hodnotu $|L(x, x_-)|$, která představuje hodnotu Lagrangeovy interpolační funkce (více viz [3]), jejíž hodnota je v bodě x_- rovna 1 a v ostatních bodech z množiny I je nulová.
V případě funkce jedné proměnné má funkce $L(x, x_-)$ tvar:

$$L(x, x_-) = \prod_{x_m \neq x_-} \frac{x - x_m}{x_- - x_m}, \tag{10}$$

kde body x_m jsou body z množiny $I \cup J$ a jelikož x_- patří do této množiny je potřeba dodržet podmínku $x_m \neq x_-$, aby nedošlo k dělení nulou.
Pokud chceme minimalizovat funkce více proměnných je potřeba použít Lagrangeovu interpolační funkci více proměnných, která bude popsána dále v textu.
Význam funkce L je následující. Hodnota $|L(x_+, x_-)|$ se rovná poměru:

$$\frac{|\delta(I \setminus \{x_-\} \cup \{x_+\})|}{|\delta(I)|}, \tag{11}$$

kde I obsahuje bod x_- a $\delta(\cdot)$ je dané vztahem (7). Poměr (11) udává, jak se zlepší kvadratický model funkce f sestavený z bodů v množině I. Proto hledáme takový bod x_+, který při nahrazení bodu x_- maximalizuje $|L(x_+, x_-)|$
Pokud algoritmus dospěje do kroku 6 znamená to, že geometrii modelu nelze vylepšit vzhledem k danému Δ a úpravy modelu v okolí x_c se tedy dosáhne jenom zmenšením Δ, což způsobí, že interpolační body budou blíže bodu x_c

4.4 Krok trust-regionu

Po sestavení a úpravě (pokud byla potřeba) modelu je vypočten ideální krok, který minimalizuje model pomocí trust-regionu v Δ okolí bodu x_c.
Tento krok je řešením úlohy:

$$s = \arg\min_{\|s\| \leq \Delta} m(x_c + s). \tag{12}$$

Po nalezení ideálního kroku v úloze (12) je vypočítán poměr ρ mezi dosaženým a předpokládaným poklesem ze vztahu (3). Místo abychom hned přešli k další iteraci a úpravě

poloměru trust-regionu, zjišťujeme, jestli si nemůžeme dovolit tzv. dlouhý krok v případě, že model odpovídá funkci velmi dobře. Tzv. dlouhý krok využívá velmi dobré shody modelu a skutečného chování funkce a proto se pokusíme využít tento model na větším okolí bodu x_c než jen okolí o velikosti Δ
Testujeme jestli je $\rho \geq 0.9$, což značí výborný poměr mezi dosaženým a předpokládaným poklesem. V tomto případě ověřujeme úspěšnost u všech předchozích bodů $x_i \epsilon M \setminus I$, která je určena:

$$\rho_i = \frac{f(x_i) - f(x_c)}{m(x_i) - m(x_c)}. \tag{13}$$

Vypočteme maximální vzdálenost $\|x_i - x_c\|$ pro všechny x_i, pro které platí, že poměr ρ_i je pouze lehce horší než poměr ze vztahu (3) ($\rho_i \geq 0.85$). Tuto maximální vzdálenost označíme Θ. Pokud je Θ výrazně větší než Δ, je model dobrý v mnohem větším okolí, než ve kterém byl vypočten krok s. Můžeme tedy vypočítat delší krok d:

$$d = arg \min_{\|s\| \leq \Theta} m(x_c + s) \tag{14}$$

s šancí na úspěch podobnou té s původním krokem s.
Můžeme poté nahradit kratší krok s krokem d kdykoliv, kdy nastane očekávaný pokles, tzn. když:

$$\rho_d = \frac{f(x_c + d) - f(x_c)}{m(x_c + d) - m(x_c)} \geq 0.05 \qquad a \qquad f(x_c + d) < \min\{f(x_c), f(x_c + s)\}. \tag{15}$$

Tento mechanismus nám umožňuje provádět dlouhé kroky, když kvadratický model odpovídá funkci (např. pokud je zadaná funkce v Δ okolí bodu x_c přibližně nebo přesně kvadratická).

4.5 Nový iterační bod

Po nalezení kroku s pomocí minimalizační úlohy (12) (popř. dlouhého kroku d pomocí (14)) a vypočtení funkční hodnoty v bodě $x_c + s$ (nebo $x_c + d$), je potřeba rozhodnout, jestli by měl daný bod nahradit bod současné iterace (x_c).
U metod trust-regionu je nový bod přijat pokud je poměr dosaženého a očekávaného poklesu (vztah (3)) větší než nějaká malá konstanta (v našem algoritmu 0.05). Pokud tento případ nastane, bod $x_c + s$ je přidán do množiny I a současně je z množiny $I \cup J$ odebrán bod, jehož nahrazení bodem $x_c + s$ je nejvýhodnější pro geometrii daných množin. Bod $x_c + s$ může být přidán do množiny I i když nedošlo k požadovanému poklesu, ale pouze tehdy, když se geometrie daných množin výrazně zlepší, pokud odebereme bod z množiny $(I \cup J) \setminus \{x_c\}$.
Může se ovšem stát, že obě z předcházejících možností selžou. Poté je daná iterace nazvaná jako neúspěšná, x_c zůstává nezměněné a provede se algoritmus A1 pro úpravu geometrie množin. Tento krok se dá provést i v případě, že dosažený pokles je příliš malý ($\rho < 0.15$) i když je iterace úspěšná.

4.6 Úprava poloměru trust-regionu

Po provedení předchozích kroků musí být změněn poloměr trust-regionu. Můžeme uvažovat změnu poloměru trust-regionu pouze tehdy, pokud všechny předchozí metody na úpravu geometrie množin selhaly (tzn. pokud algoritmus A1 skončil neúspěšně v kroku 6) a krok trust-regionu s byl neúspěšný, ale to by znamenalo velmi pomalé zmenšování daného poloměru. Místo toho budeme uvažovat změnu poloměru při každé iteraci. Pokud je daná iterace úspěšná ($\rho \geq 0.75$), může se daný poloměr zvětšit, v opačném případě může dojít ke zmenšení poloměru.

4.7 Konvergenční test

Jako poslední část algoritmu musíme navrhnout ideální ukončující kritérium. Můžou nastat dvě možnosti:

1. Zadaná funkce už nemůže být více vylepšena, i když je poloměr trust-regionu malý (pod hranicí přípustnosti ϵ_Δ)
$$\Delta < \epsilon_\Delta. \tag{16}$$

2. Byl nalezen bod x, pro který platí, že $f(x) - f_l$ je pod hranicí přípustnosti ϵ_f. f_l označujeme jako odhad pro minimum (pokud tato informace není známa, může být nastavena na $-\infty$).
$$f(x_c) < f_l + \epsilon_f. \tag{17}$$

Pokud nastane jeden z těchto případů, algoritmus skončí v současné iteraci a vrátí x_c jako přibližné řešení problému - hledání minima zadané funkce.

4.8 Algoritmus metody

Nyní můžeme uvést celkový algoritmus metody pro minimalizaci funkce $f \epsilon R^n$ využívající kvadratickou interpolaci. V algoritmu se objevují proměnné η_r a η_a, které udávají relativní a absolutní úroveň zašumění funkce f. Na začátku algoritmu inicializujeme $\Delta = \Delta_0$ a $M = I = \{x_0, x_1\}$, kde x_0 je počáteční bod aproximace (později x_c) a x_1 je náhodný bod pro který platí $\parallel x_1 - x_0 \parallel \leq \Delta$. Množinu J inicializujeme jako $J = \{x_2, \ldots x_{p-1}\}$, kde pro body $x_2 \ldots x_{p-1}$ také platí, že jejich vzdálenost od bodu x_0 je menší než Δ (p je dané vztahem (6)). Množina $I \cup J$ musí splňovat, že $\delta(I \cup J) \neq 0$, kde δ je dané vztahem (7). Parametr r, který také zadáváme na vstupu, udává poměr, o který se poloměr trust-regionu zmenšuje při neúspěšné iteraci, nebo zvětšuje při úspěšné.
Algoritmus metody vypadá následovně:

1 Input: $x_0, \epsilon_\Delta, f_l, \epsilon_f, r > 1, \Delta_0, \eta_a, \eta_r$
2 $\Delta = \Delta_0$
3 $x_1 = x_0 + \Delta \cdot rand(1, n)$
4 Vypočítej $f(x_0)$
5 Vypočítej $f(x_1)$
6 $x_c = x_0$

$M = I = \{x_0, x_1\}$
$J = \{x_2, \ldots x_{p-1}\} = x_0 + \Delta \cdot rand(1, n)$
$p = \frac{1}{2}(n+1)(n+2)$
while $(\Delta > \epsilon_\Delta)$ and $(f(x_c) \geq f_l + \epsilon_f)$
Step 1: $x_t = \arg\min_{x \epsilon I} f(x)$
if $f(x_t) < f(x_c) - 0.1\epsilon_\Delta \Delta$
$x_c = x_t$
end;
Step 2: **if** $\mid I \mid = p$
$\forall x \epsilon I : m(x) = f(x)$
pokus se sestavit model:
$m(x_c + s) = f(x_c) + g^T s + \frac{1}{2} s^T H s$

else if $\mid I \mid < p$
pokus se vyřešit problém: $\min_{g,H} \|g\|^2 + \|H\|_F^2$
Pokud model nemůže být sestaven
Algoritmus A1
Jdi do kroku 6
end;
end;
Step 3: $s = \min_{\|s\| \leq \Delta} m(x_c + s)$
if $m(x_c) - m(x_c + s) < \max\left[\epsilon_\Delta \Delta, \frac{\eta_a}{2}, \frac{\eta_r}{2} |f(x_c + s)|\right]$
Algoritmus A1
Jdi do kroku 6
else
$M = M \cup \{x_c + s\}$
Vypočítej $f(x_c + s)$
$\rho = \frac{f(x_c+s) - f(x_c)}{m(x_c+s) - m(x_c)}$
end;
Step 4: **if** $\rho \geq 0.9$
$\forall x_i \epsilon M \setminus I : f(x_i) - f(x_c) \leq 0.85(m(x_i) - m(x_c))$
$and \parallel x_i - x_c \parallel \geq \Delta$
$\Theta = \min\{1000\Delta, \max \|x_i - x_c\|\}$
if $\Theta > (1 + r) \parallel s \parallel$
$d = \min_{\|d\| \leq \Theta} m(x_c + d)$
if $\parallel d \parallel > (1 + r) \parallel s \parallel$
$M = M \cup \{x_c + d\}$
Vypočítej $f(x_c + d)$
$\rho_d = \frac{f(x_c+d) - f(x_c)}{m(x_c+d) - m(x_c)}$
if $\rho_d \geq 0.05$ and $f(x_c + d) < \min\{f(x_c), f(x_c + s)\}$

$s = d$
$\rho = \rho_d$
end;
end;
end;
end;

Step 5: $x_I = \arg\max_{x \epsilon I \setminus \{x_c\}} S(x_c + s, x)$

$x_J = \arg\max_{x \epsilon J} S(x_c + s, x)$

$S_I = S(x_c + s, x_I)$

$S_J = S(x_c + s, x_J)$

if $\rho \geq 0.05$

  if $S_I > 2S_J$

    $I = (I \setminus \{x_I\}) \cup \{x_c + s\}$

  end;

  if $S_I \leq 2S_J$

    $I = I \cup \{x_c + s\}$

    $J = J \setminus \{x_J\}$

  end;

else if $\max\{S_I, S_J\} > 1$

  if $S_I > S_J$

    $I = (I \setminus \{x_I\}) \cup \{x_c + s\}$

  end;

  if $S_I \leq S_J$

    $I = I \cup \{x_c + s\}$

    $J = J \setminus \{x_J\}$

  end;

else if $\rho < 0.15$

  Algoritmus A1

else

  Algoritmus A1

end;

Step 6: **if** $\rho \geq 0.75$ and $\|s\| > \epsilon_\Delta$

  $\Delta = \min\left[r\Delta, \max(\Delta, r \parallel s \parallel)\right]$

else if $\Delta > \epsilon_\Delta$ a Krok 2 proběhl bez problému

  $\Delta = \max\{\epsilon_\Delta, \frac{\Delta}{r}\}$

  if Algoritmus A1 selhal v úpravě množin

    Algoritmus A1 s upravenou Δ

  end;

else if Algoritmus A1 selhal v úpravě množin

  $\Delta = \frac{\Delta}{r}$

end;

```
87  end;
88
89  return x_c
90
91  end
```

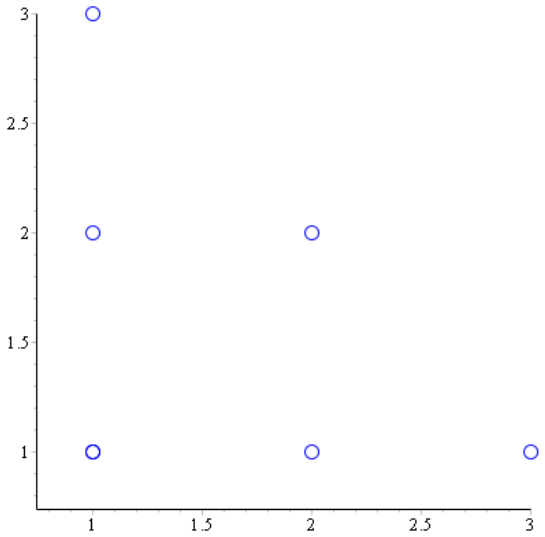
Výpis 4: Algoritmus metody využívající kvadratickou interpolaci

4.9 Funkce více proměnných

V případě, že uvažujeme úlohu (1), kde $n > 1$, je potřeba upravit funkci $L(\cdot, \cdot)$ ze vztahu (10). Tato funkce totiž obsahuje hodnotu Lagrangeovy interpolační funce, která je sestavena pro funkci jedné proměnné. Je tedy potřeba sestavit Lagrangeovu interpolační funkci pro funkci více proměnných. Interpolační funkce bude mít tvar:

$$L_d(x) = \prod_{i=1}^{d} \prod_{\substack{j=0 \\ j \neq \alpha_i}}^{2} \frac{x_i - x_{i,j}}{x_{i,\alpha_i} - x_{i,j}}. \tag{18}$$

Ve vztahu (18) udává d dimenzi funkce. Pokud ale chceme pracovat s tímto vztahem, musíme zvolit naprosto odlišnou geometrii bodů. Chceme-li totiž interpolovat danou Lagrangeovu interpolační funkci, musí být body poskládány v mřížce, ve které budou mít některé body vždy současně stejné souřadnice. Pro ilustraci této mřížky uvádíme obrázek možného rozložení bodů pro $n = 2$.



Obrázek 1: Rozložení mřížky

Pokud pracujeme s takto rozloženými body, máme dokonce zaručeno, že determinant matice $\delta(\cdot)$ ze vztahu (7) je nenulový.

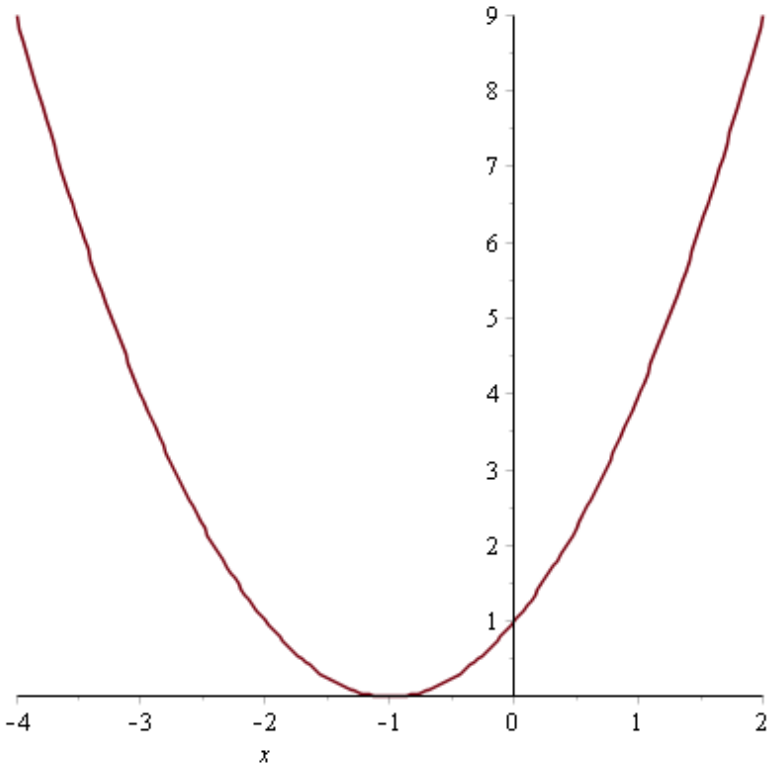
5 Numerické testy

V této kapitole otestujeme navržený algoritmus pro metodu využívající kvadratickou interpolaci na několika příkladech. Budeme demonstrovat výsledky po volbě různých počátečních parametrů funkce a budeme sledovat, jak dané volby ovlivní celkový počet iterací daného algoritmu, ale také počet vyčíslení funkčních hodnot (výpočet funkčních hodnot je nákladný) a přesnost celkového řešení.

5.1 Příklad 1

Budeme řešit úlohu:

$$\min_{x \epsilon R} x^2 + 2x + 1 \tag{19}$$



Obrázek 2: Graf funkce f_1

V následující tabulce si ukážeme, jak závisí volba jednotlivých parametrů na výsledku dané aproximace, počtu iterací a také na počtu vyčíslení funkčních hodnot dané funkce. Ukážeme také tzv. konvergenční historie dané aproximace.

Volíme počáteční parametry:

f_l - odhad funkční hodnoty minima
ϵ_Δ - hranice přípustnosti pro Δ
Δ_f - hranice přípustnosti pro funkční hodnotu minima
x_0 - počáteční bod aproximace
Δ_0 - počáteční okolí pro trust-region

Algoritmus poté vrací výsledky ve formě x^*, které představuje odhad pro minimum dané funkce, počet iterací, které proběhly při výpočtu, údaj o tom, kolikrát se musela vyčíslit daná funkce a tzv. konvergenční historii algoritmu, což je závislost mezi funkční hodnotou současného bodu iterace ($f(x_c)$) a současnou iterací. Konvergenční historie ukazuje, jak rychle se daný algoritmus blíží k minimu.

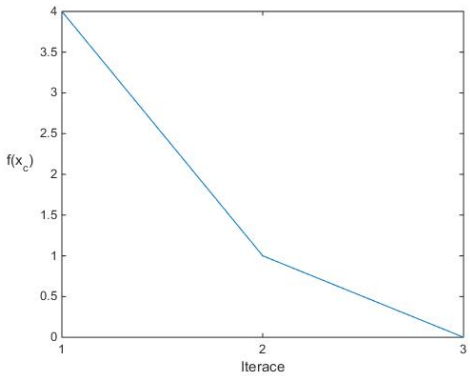
f_l	ϵ_Δ	ϵ_f	x_0	Δ_0	x^*	Počet iterací	Počet vyčíslení funkce	Konvergenční historie
0	10^{-1}	10^{-1}	1	1	-1.00	3	6	obr. 1
0	10^{-1}	10^{-1}	100	10	-1.00	6	9	obr. 2
0	10^{-3}	10^{-3}	1	1	-1.00	3	6	obr. 3
0	10^{-3}	10^{-3}	100	10	-1.00	6	9	obr. 4
$-\infty$	10^{-1}	10^{-1}	1	1	-1.00	9	15	obr. 5
$-\infty$	10^{-1}	10^{-1}	100	10	-1.00	15	26	obr. 6
$-\infty$	10^{-3}	10^{-3}	1	1	-1.00	15	26	obr. 7
$-\infty$	10^{-3}	10^{-3}	100	10	-1.00	22	38	obr. 8

Tabulka 1: Výsledky funkce f_1

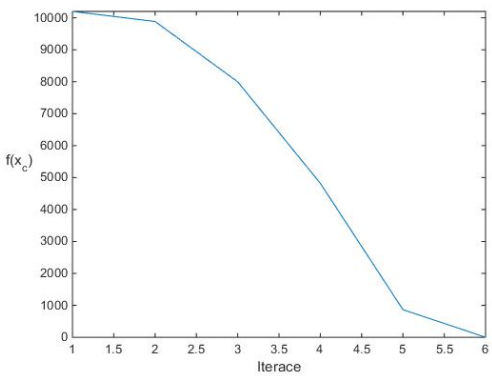
Algoritmus našel řešení úlohy (19) v $x^* = -1.00$. Hodnota funkce v minimu je $f(x^*) = 0$.

5.1.1 Konvergenční historie

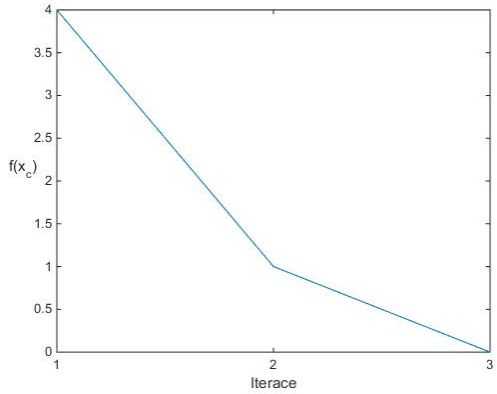
V grafech představuje osa x číslo iterace a osa y představuje funkční hodnotu současného bodu iterace.



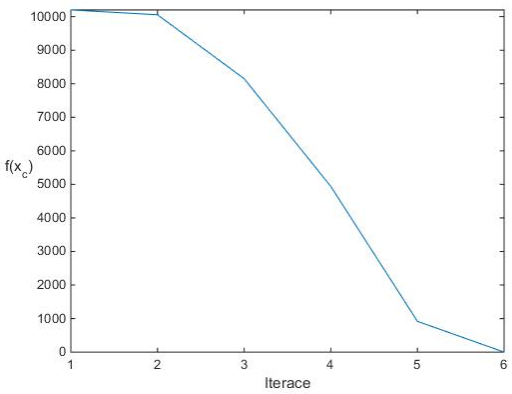
obr. 1



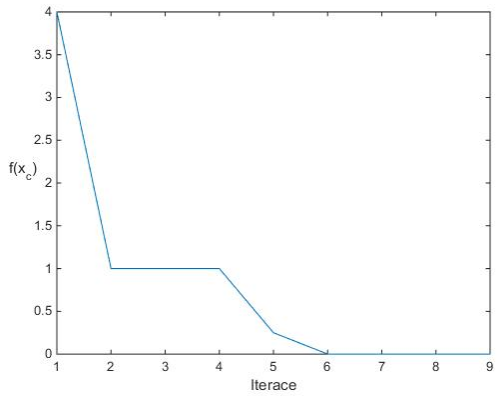
obr. 2



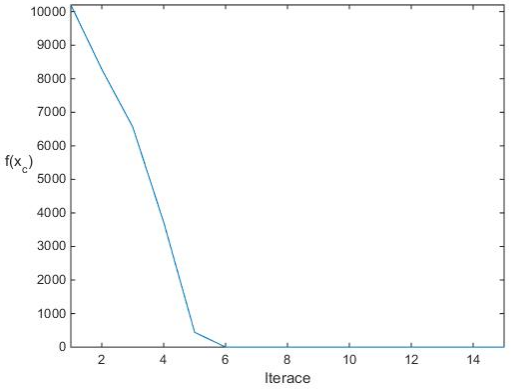
obr. 3



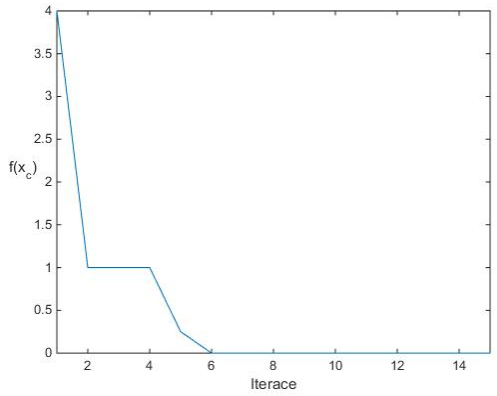
obr. 4



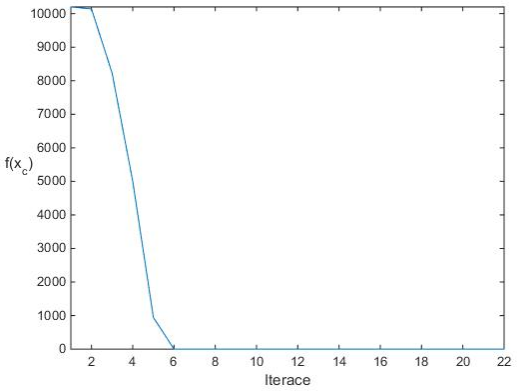
obr. 5



obr. 6



obr. 7

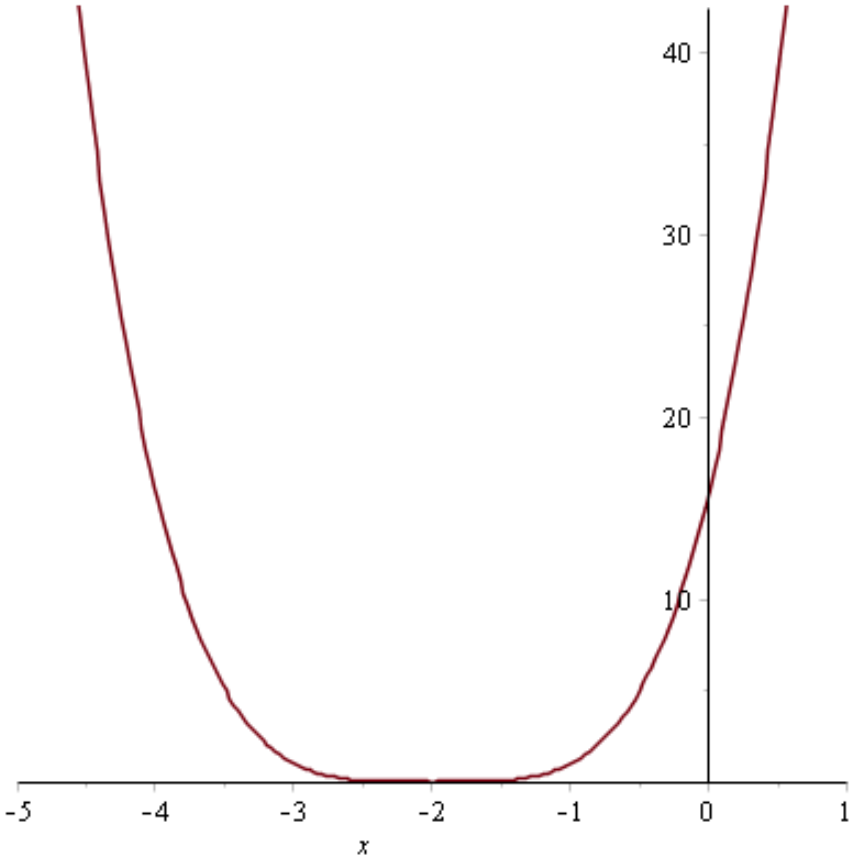


obr. 8

5.2 Příklad 2

Budeme řešit úlohu:

$$\min_{x \epsilon R} x^4 + 8x^3 + 24x^2 + 32x + 16 \tag{20}$$



Obrázek 3: Graf funkce f_2

Opět budeme stejně jako u příkladu č. 1 volit různé počáteční parametry a budeme zkoumat jak tato volba ovlivní celkové výsledky. Ukážeme si také konvergenční historie pro jednotlivé možnosti výběru počátečních parametrů. Výsledky pro jednotlivé příklady jsou uvedeny v následující tabulce, která stejně jako u příkladu 1 obsahuje 8 možností, jak nakombinovat počáteční parametry.

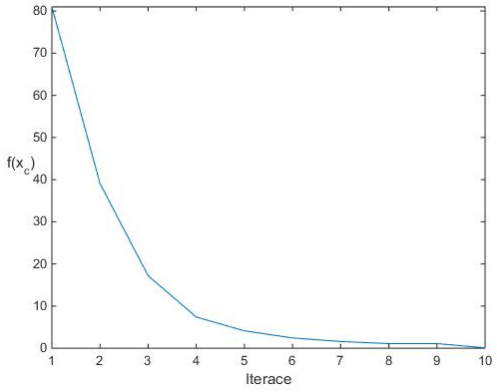
f_l	ϵ_Δ	ϵ_f	x_0	Δ_0	x^*	Počet iterací	Počet vyčíslení funkce	Konvergenční historie
0	10^{-1}	10^{-1}	1	1	-1.44	10	15	obr. 1
0	10^{-1}	10^{-1}	50	10	-1.33	53	61	obr. 2
0	10^{-3}	10^{-3}	1	1	-1.97	34	38	obr. 3
0	10^{-3}	10^{-3}	50	10	-1.84	465	472	obr. 4
$-\infty$	10^{-1}	10^{-1}	1	1	-1.65	12	18	obr. 5
$-\infty$	10^{-1}	10^{-1}	50	10	–1.49	50	58	obr. 6
$-\infty$	10^{-3}	10^{-3}	1	1	–2.00	40	51	obr. 7
$-\infty$	10^{-3}	10^{-3}	50	10	–1.98	497	509	obr. 8

Tabulka 2: Výsledky funkce f_2

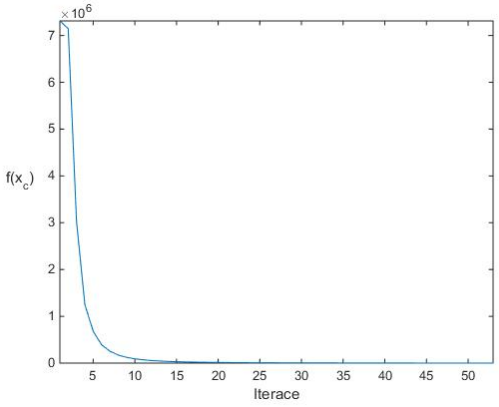
Z výsledků vyplývá, že algoritmus nalezl řešení úlohy (20) v intervalu $x^* \epsilon \langle -2.00, -1.33 \rangle$. Můžeme si všimnout, že v případě volby $f_l = 0$ ukončuje algoritmus výpočet dále od minima než v případě volby $f_l = -\infty$, protože funkce f je v okolí minima téměř konstantní a tím je splněna ukončující podmínka $f(x_c) < (f_l + \epsilon_f)$. Celkový odhad pro minimum závisí na volbě počátečních parametrů. Funkční hodnota minima je $f(x^*) = 0$.

5.2.1 Konvergenční historie

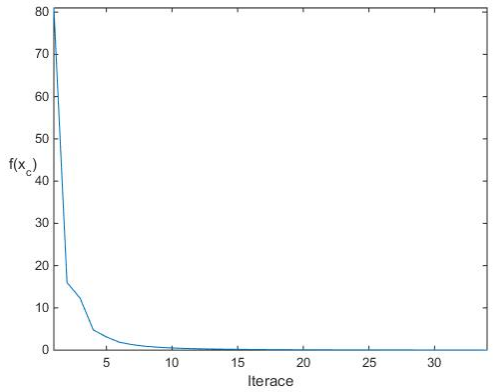
V grafech představuje osa x číslo iterace a osa y představuje funkční hodnotu současného bodu iterace.



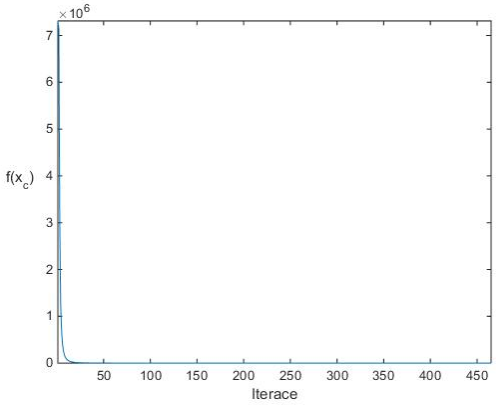
obr. 1



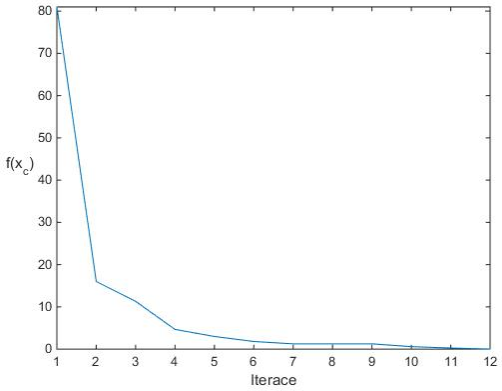
obr. 2



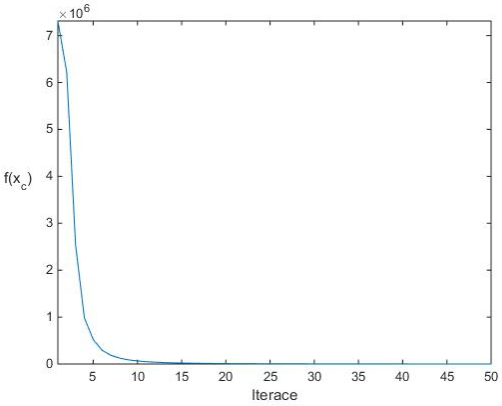
obr. 3



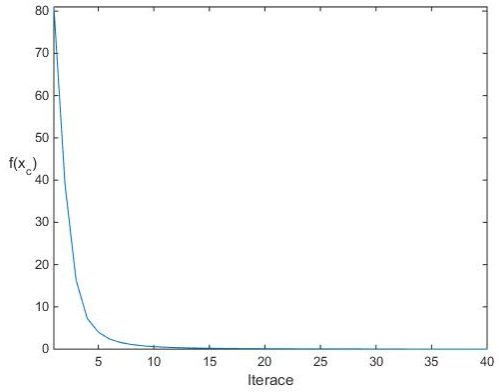
obr. 4



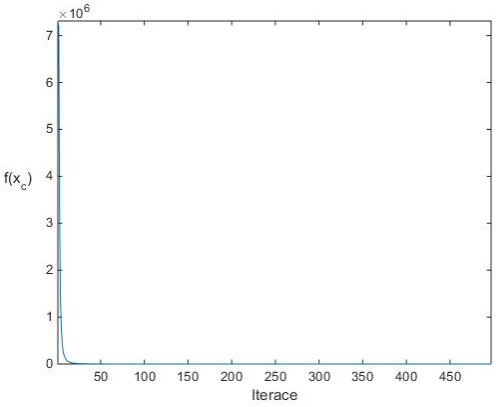
obr. 5



obr. 6



obr. 7

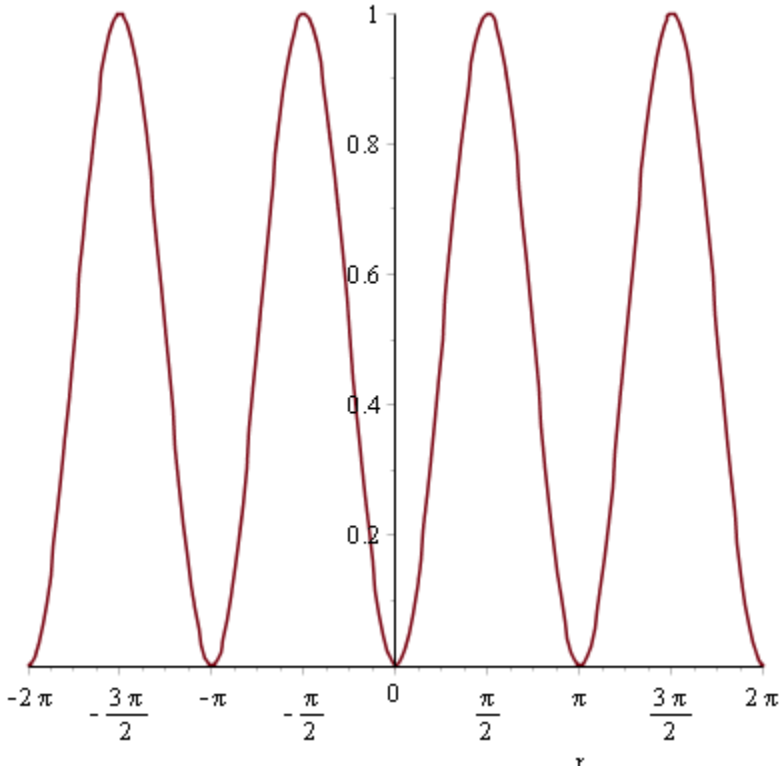


obr. 8

5.3 Příklad 3

Budeme řešit úlohu:

$$\min_{x \epsilon R} sin^2(x) \tag{21}$$



Obrázek 4: Graf funkce f_3

Z grafu vyplývá, že funkce má mnoho lokálních minim. Nalezené minimum tedy bude pravděpodobně záviset na volbě počátečního bodu iterace. Pojďme se tedy podívat jak volby daných parametrů ovlivní algoritmus. Výsledky jsou uvedeny v následující tabulce.

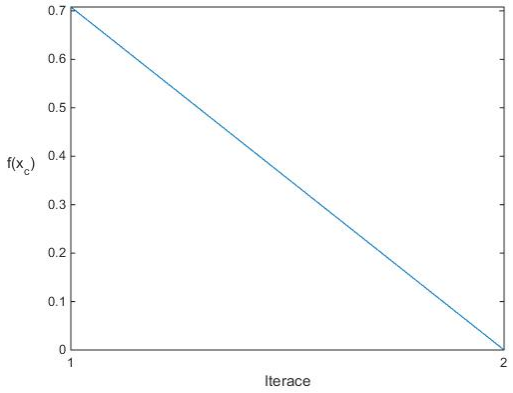
f_l	ϵ_Δ	ϵ_f	x_0	Δ_0	x^*	Počet iterací	Počet vyčíslení funkce	Konvergenční historie
0	10^{-1}	10^{-1}	1	1	0	2	4	obr. 1
0	10^{-1}	10^{-1}	100	10	97.50	3	5	obr. 2
0	10^{-3}	10^{-3}	1	1	0	2	4	obr. 3
0	10^{-3}	10^{-3}	100	10	97.40	7	14	obr. 4
$-\infty$	10^{-1}	10^{-1}	1	1	0	6	10	obr. 5
$-\infty$	10^{-1}	10^{-1}	100	10	97.34	7	11	obr. 6
$-\infty$	10^{-3}	10^{-3}	1	1	0	12	16	obr. 7
$-\infty$	10^{-3}	10^{-3}	100	10	97.39	16	26	obr. 8

Tabulka 3: Výsledky funkce f_3

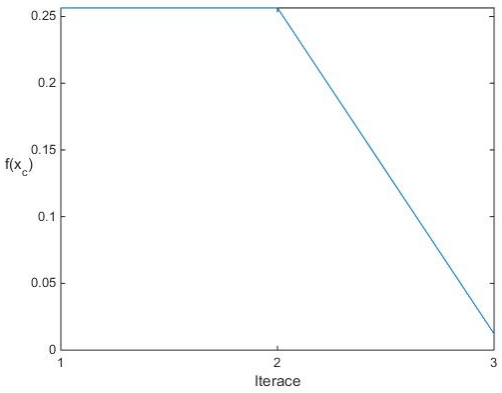
Výsledky z tabulky potvrdily, že funkce z úlohy (21) má několik lokálních minim. Minima zadané funkce jsou $x^* = k * \pi, k\epsilon N$. Záleží tedy na volbě počátečního bodu algoritmu. Pro volbu $x_0 = 1$ nám algoritmus našel minimum $x^* = 0$ - v tomto případě $k = 0$, a pro volbu $x_0 = 100$ našel algoritmus nejpřesněji bod $x^* = 97.39$, což odpovídá $k = 31$. Hodnota funkce v minimu je $f(x^*) = 0$.

5.3.1 Konvergenční historie

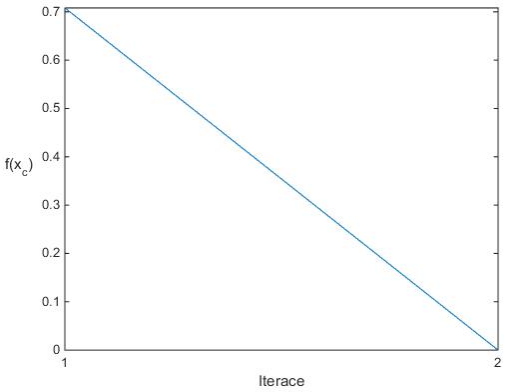
V grafech představuje osa x číslo iterace a osa y představuje funkční hodnotu současného bodu iterace.



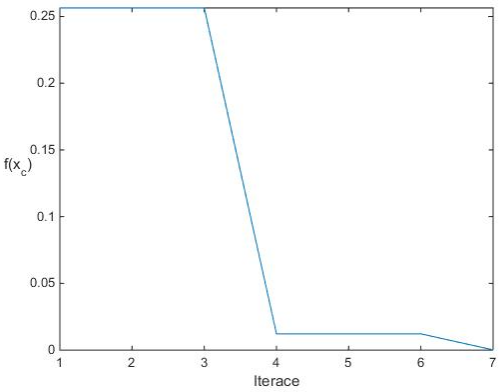
obr. 1



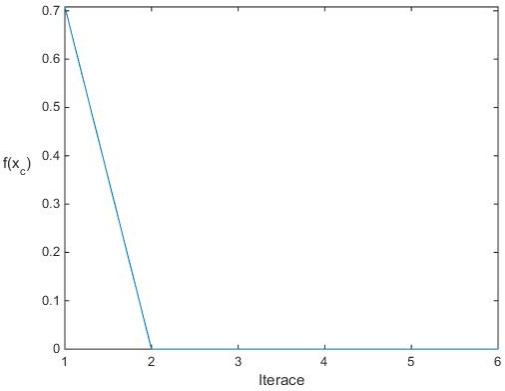
obr. 2



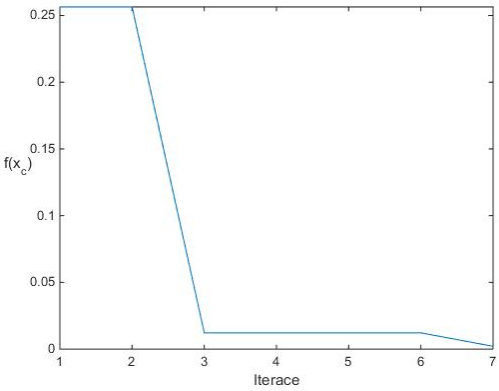
obr. 3



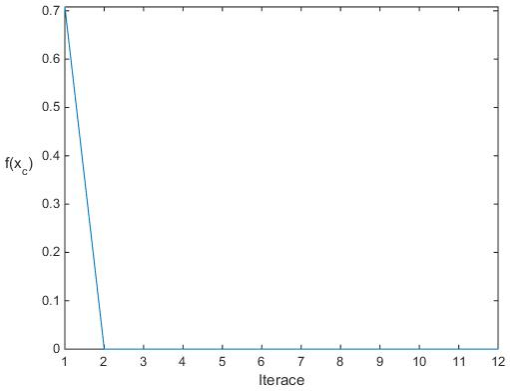
obr. 4



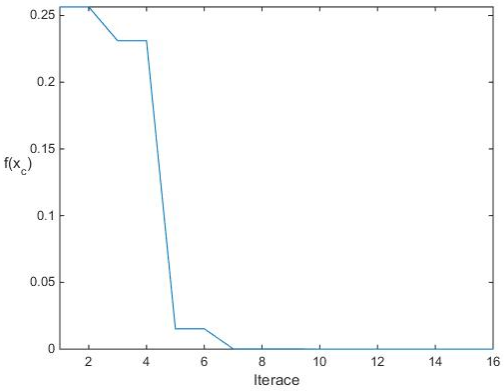
obr. 5



obr. 6



obr. 7

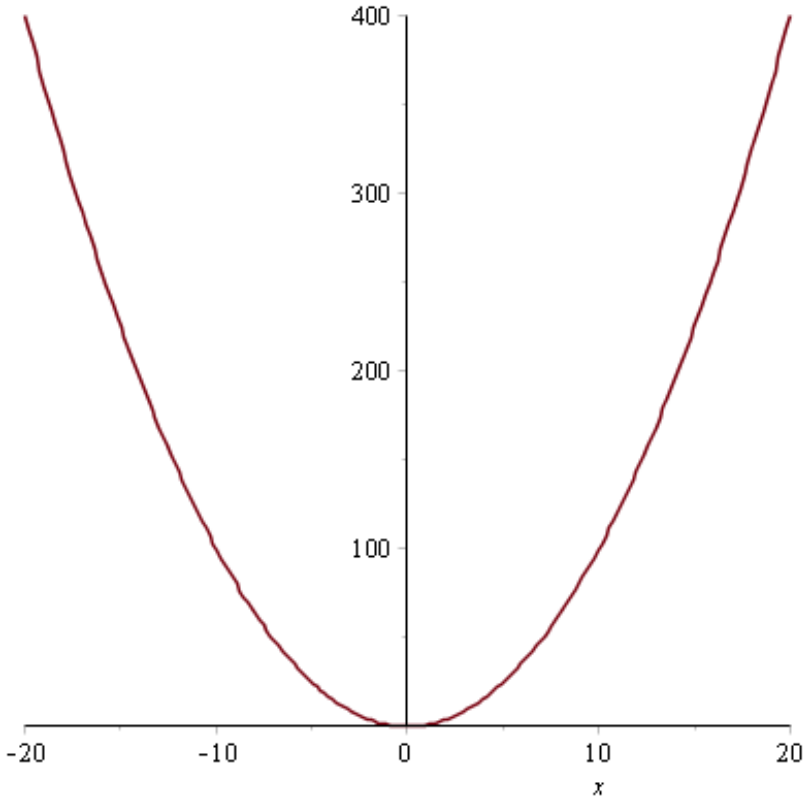


obr. 8

5.4 Příklad 4

U posledního příkladu si ukážeme, jak ovlivní volba počátečních šumů (η_a - absolutní šum a η_r - relativní šum) výsledky algoritmu. Budeme řešit úlohu:

$$\min_{x \epsilon R} x^2 \left(\frac{sin(x)}{100} + 1 \right) \tag{22}$$



Obrázek 5: Graf funkce f_4

Volíme hodnoty $\eta_a = \{0, \frac{1}{2}\}$ a $\eta_r = \{0, \frac{1}{100}\}$ a zkoumáme, jak se dané výsledky liší. Budeme také porovnávat grafy dosažených konvergenčních historií, a budeme zkoumat, při kterých počátečních parametrech metoda konverguje dříve. Výsledky jsou zaznamenány v následující tabulce.

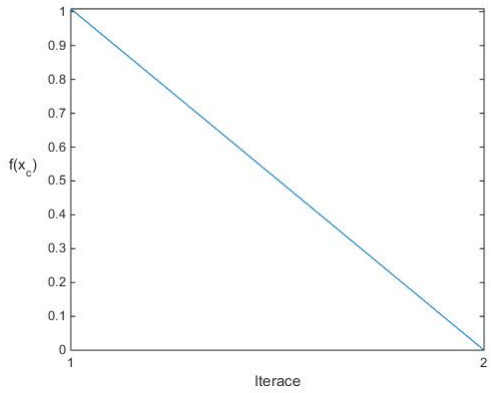
f_l	ϵ_Δ	ϵ_f	x_0	Δ_0	η_a	η_r	x^*	Počet iterací	Počet vyčíslení funkce	Konvergenční historie
0	10^-1	10^-1	1	1	0	0	0	2	5	obr. 1
0	10^-1	10^-1	100	10	0	0	0.14	8	9	obr. 2
0	10^-3	10^-3	1	1	0	0	$8.75 \cdot 10^{-9}$	2	4	obr. 3
0	10^-3	10^-3	100	10	0	0	-0.02	12	14	obr. 4
0	10^-1	10^-1	1	1	$\frac{1}{2}$	$\frac{1}{100}$	$8.7 \cdot 10^{-9}$	2	5	obr. 5
0	10^-1	10^-1	100	10	$\frac{1}{2}$	$\frac{1}{100}$	-0.02	8	10	obr. 6
0	10^-3	10^-3	1	1	$\frac{1}{2}$	$\frac{1}{100}$	$7.94 \cdot 10^{-9}$	2	5	obr. 7
0	10^-3	10^-3	100	10	$\frac{1}{2}$	$\frac{1}{100}$	-10^{-3}	8	9	obr. 8

Tabulka 4: Výsledky funkce f_4

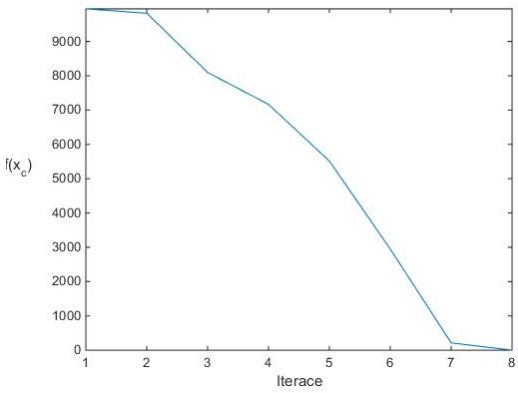
Z výsledků v tabulce vyplývá, že algoritmus našel minimum funkce z úlohy (22) v bodě $x^* \doteq -0.005$. Přesnost aproximace opět záleží na volbě počátečních parametrů. Po volbě různých úrovní šumů jsme dospěli k závěru, že algoritmus našel aproximaci řešení po přibližně stejném množství iterací, nicméně výsledné minimum je jiné. Funkce je totiž v okolí minima zašuměná, a proto algoritmus nachází jiné výsledky. Hodnota funkce v minimu je $f(x^*) \doteq 0$.

5.4.1 Konvergenční historie

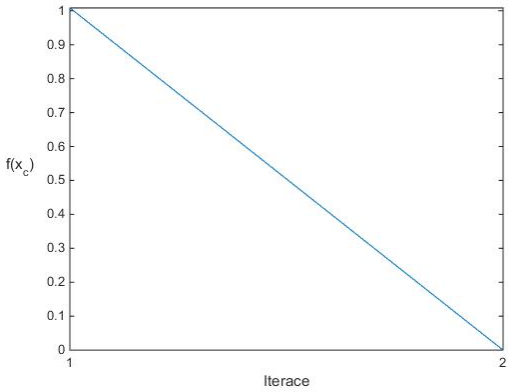
Pro porovnání uvádíme jednotlivé konvergenční historie pro daný algoritmus. Z grafů lze vyčíst, jak rychle algoritmus konverguje pro různou volbu počátečních šumů ($\eta_a = 0$, $\eta_r = 0$ a $\eta_a = \frac{1}{2}$, $\eta_r = \frac{1}{100}$). V grafech představuje osa x číslo iterace a osa y představuje funkční hodnotu současného bodu iterace.



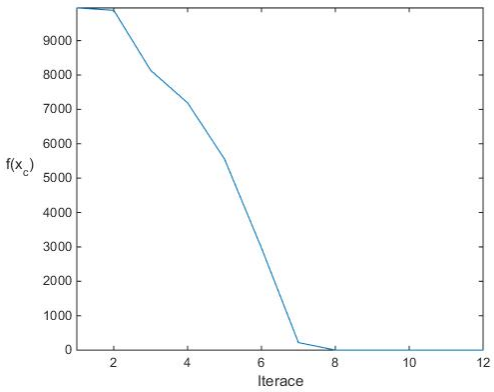
obr. 1



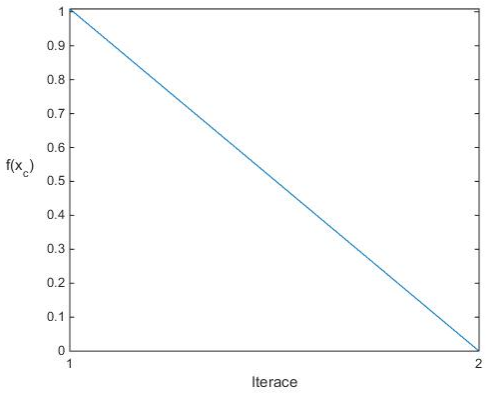
obr. 2



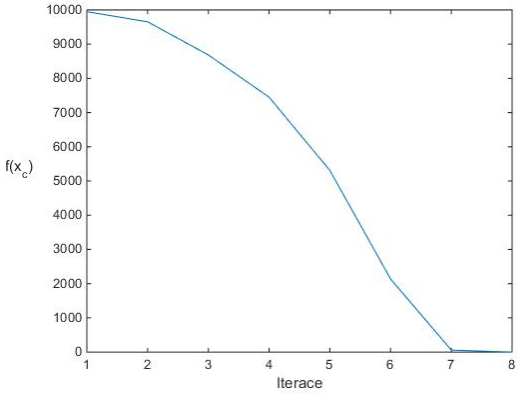
obr. 3



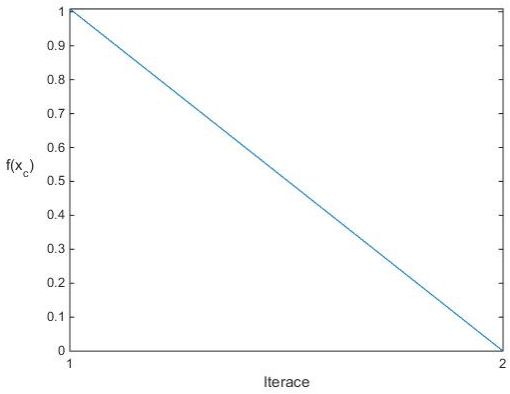
obr. 4



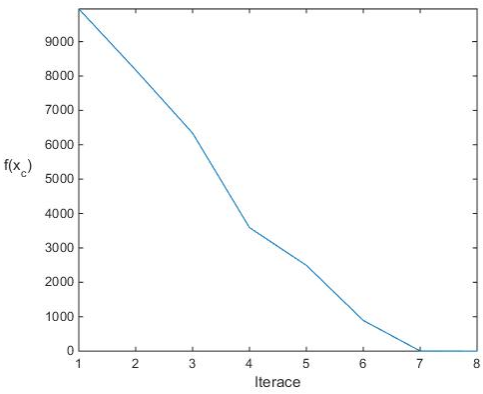
obr. 5



obr. 6



obr. 7



obr. 8

6 Závěr

Hlavním cílem práce bylo seznámení se s metodou využívající kvadratické interpolace funkce a její následná implementace. V teoretické části byly nejprve popsány jiné metody řešící zadaný matematický problém a poté byly vysvětleny základní rysy a náležitosti metody kvadratické interpolace.
Po úspěšné implementaci a otestování kódu jsme se potom zaměřili na numerické testy k ověření správnosti výsledků metody. Zjistili jsme, že efektivita algoritmu je závislá na volbě počátečních parametrů a také na velikosti absolutního a relativního šumu. Pokud jsme zvolili dostatečně malé hodnoty pro ověření konvergenčního testu, dostali jsme dobrou aproximaci pro výsledné minimum funkce, čímž jsme ověřili správnost této metody. Pokud byla funkce v Δ okolí bodu x_c kvadratická, dostali jsme dokonce přesný výsledek.
Veškeré testy byly provedeny na funkci jedné proměnné. V případě využití uvedeného algoritmu pro funkce více proměnných je potřeba použít Lagrangeovu interpolační funkci pro více proměnných. Tento postup je v práci navržen. V práci je také navrženo rozložení bodů, které je potřeba dodržet.

7 Reference

[1] CONN, A.R. a Ph.L. TOINT. *An algorithm using quadratic interpolation for unconstrained derivative free optimization.* Nonlinear Optimization and Applications. 1995.

[2] DOSTÁL, Zdeněk a Petr BEREMLIJSKI. *Metody optimalizace.* [online]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/metody_optimalizace.pdf. 2012.

[3] KAPUSTA, Joanna a Ryszard SMARZEWSKI. *Fast algorithms for multivariate interpolation and evaluation at special points.* Journal of Complexity. [online]. vol. 25, issue 4, s. 332-333. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0885064X09000041. 2009.

[4] NOCEDAL, Jorge a Stephen J WRIGHT. *Numerical optimization.* 2nd ed. New York: Springer, 2006.

[5] POSPÍŠIL, Lukáš a Vít VONDRÁK. *Numerické metody 1.* [online]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/numericke_metody.pdf. 2012.