UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Metody statické optimalizace funkcí jedné proměnné

Jaroslav Dašek

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jaroslav Dašek**
Osobní číslo:	**I11320**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Metody statické optimalizace funkcí jedné proměnné**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Prostudovat doporučenou literaturu.
Sestavit vývojové diagramy určených metod.
Sestavit ve výpočetním prostředí příslušný program.
Vypracovat uživatelské prostředí (GUI).
Vyzkoušet provozuschopnost programu na vybraných testovacích funkcích.
Zpracovat závěrečnou práci podle příslušných norem a pokynů vedoucího.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

BRUNOVSKÁ, A. Malá optimalizácia. Bratislava: ALFA, 1990. ISBN 80-05-00770-1.
HUDZOVIČ, P. Optimalizácia. Bratislava: STU Bratislava, 2001. ISBN 80-227-1598-0.
DRÁBEK, O.; TAUFER, I. Automatizované systémy řízení technologických procesů. Pardubice: VŠCHT Pardubice, 1985.
TAUFER, I.; KOTYK, J.; HRUBINA, K.; TAUFER, J. Algoritmy a algoritmizace a vývojové diagramy. Pardubice: Univerzita Pardubice, 2009. ISBN 978-80-7395-182-5.
TAUFER, I.; KOTYK, J.; JAVŮREK, M. Jak psát a obhajovat závěrečnou práci. Pardubice: Univerzita Pardubice, 2009. ISBN 978-80-7395-157-3.

Vedoucí bakalářské práce: **prof. Ing. Ivan Taufer, DrSc.**
Katedra řízení procesů

Datum zadání bakalářské práce: **9. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 22. 5. 2014

Jaroslav Dašek

Poděkování

Tímto bych rád poděkoval své rodině, známým a kamarádům za jejich podporu během tvorby bakalářské práce.

V Pardubicích dne 22. 5. 2014

Jaroslav Dašek

ANOTACE

Práce je věnována problematice optimalizace a hledání extrémů funkcí jedné proměnné pomocí čtyř algoritmů. Konkrétně se jedná o metodu Zlatého řezu, metodu půlení intervalů, metodu přímého porovnání funkčních hodnot a Fibonacciho metodu. Všechny metody jsou vypracovány pomocí grafického prostředí v programu MATLAB.

KLÍČOVÁ SLOVA

Hledání extrémů funkce, Matlab, optimalizace.

TITLE

Methods for static optimization of functions of one variable.

ANNOTATION

The work is dedicated to the optimization and search extremes of functions of one variable using four algorithms. Specifically, the golden section method, bisection method, the method of direct comparison function values and the Fibonacci method. All methods are drawn from the desktop in MATLAB.

KEYWORDS

Search extremes function, Matlab, optimization.

Obsah

Seznam zkratek

GUI	Graphical user interface
2-D	dvourozměrný
3-D	Trojrozměrný

Seznam symbolů

D	Diskriminant kvadratické rovnice
δ	změna hodnoty x
Δ	změna hodnoty x
ε	chyba řešení
$f(x)$	účelová funkce
$f(\boldsymbol{x})$	vektor účelových funkcí
$f'(x)$	první derivace účelové funkce
$f''(x)$	druhá derivace účelové funkce
$g(x)$	omezující funkce ve tvaru rovnice
$h(x)$	omezující funkce ve tvaru nerovnice
x_0	stacionární bod
X	množina přípustných řešení

Seznam obrázků

Seznam tabulek

Úvod

Cílem řešení bakalářské práce bylo sestavit uživatelský program pro určení (hledání) optima funkce jedné proměnné ve výpočetním prostředí Matlab pomocí Graphical User Interface (GUI).

V první kapitole jsou vysvětleny základní pojmy a klasické metody diferenciální analýzy, ve kterých je vysvětlena problematika hledání extrémů u funkcí jedné proměnné pomocí metod diferenciální analýzy.

Ve druhé kapitole je pojednáváno o numerických metodách a je v ní uveden základní popis vybraných čtyř numerických metod: metoda přímého porovnání funkčních hodnot, Fibonacciho metoda, metoda zlatého řezu a metoda půlení intervalu.

Třetí kapitola je věnována realizaci algoritmů v prostředí Matlab. V této části práce je uživatel seznámen se základními vlastnosti prostředí Matlab a obsluhou vytvořeného programu.

Ve čtvrté a poslední kapitole jsou testovány vybrané metody na zvolených třech testovacích funkcí, u kterých se hledá extrém (minima, maxima) účelové funkce *f* pro zvolený interval a chybu řešení. V závěru je uvedeno zhodnocení těchto testovaných metod.

1 Optimalizace

Obecně se pod pojmem optimalizace (Vítečková, 2003; Drábek, 1985) chápe taková činnost, která vede k nejlepším výsledkům této činnosti. Stav, při kterém se docílí nejlepších výsledků, se pak nazývá optimální stav a hodnoty parametrů, které tento stav zajišťují, optimální hodnoty parametrů.

1.1 Extrém, optimum

Z matematického hlediska je cílem optimalizace nalezení takových proměnných $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, pro které příslušná účelová funkce $f(\boldsymbol{x})$ na dané množině X nabývá extrémní (minimální, maximální) hodnoty. Platí:

$$f(\mathbf{x}_o) = \min_{\mathbf{x} \in X} \{f(\mathbf{x})\} \tag{1.1}$$

nebo

$$f(\mathbf{x}_o) = \max_{\mathbf{x} \in X} \{f(\mathbf{x})\} \tag{1.2}$$

kde $\boldsymbol{x}_0 = \{x_{1o}, x_{2o}, \ldots, x_{no}\}$ je vektor optimálních hodnot nezávislých proměnných a n je počet nezávislých proměnných. Oblast řešení může být vymezena omezujícími podmínkami. Z tohoto hlediska lze definovat dva typy extrémů: volný a vázaný.

Volný (nepodmíněný) extrém

Volný extrém je extrém, který není vázán dalšími dodatečnými podmínkami.

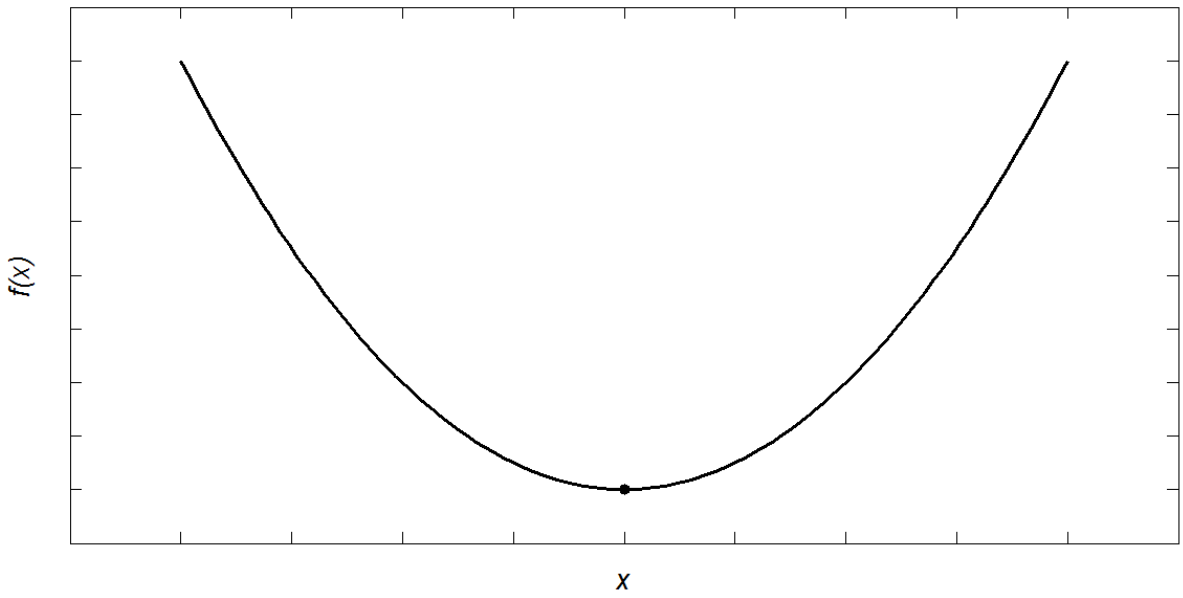
Pro volný extrém funkce jedné proměnné $f(x)$, se vychází z rovnic (1.1) resp. (1.2), pak platí:

$$f(\mathrm{x}_o) = \min_{x \in X} \{f(x)\} \tag{1.3}$$

nebo

$$f(\mathrm{x}_o) = \max_{x \in X} \{f(x)\} \tag{1.4}$$

kde x_o je optimální hodnota nezávislé proměnné.



Obrázek 1.1 – Příklad volného extrému

Vázaný (podmínění) extrém

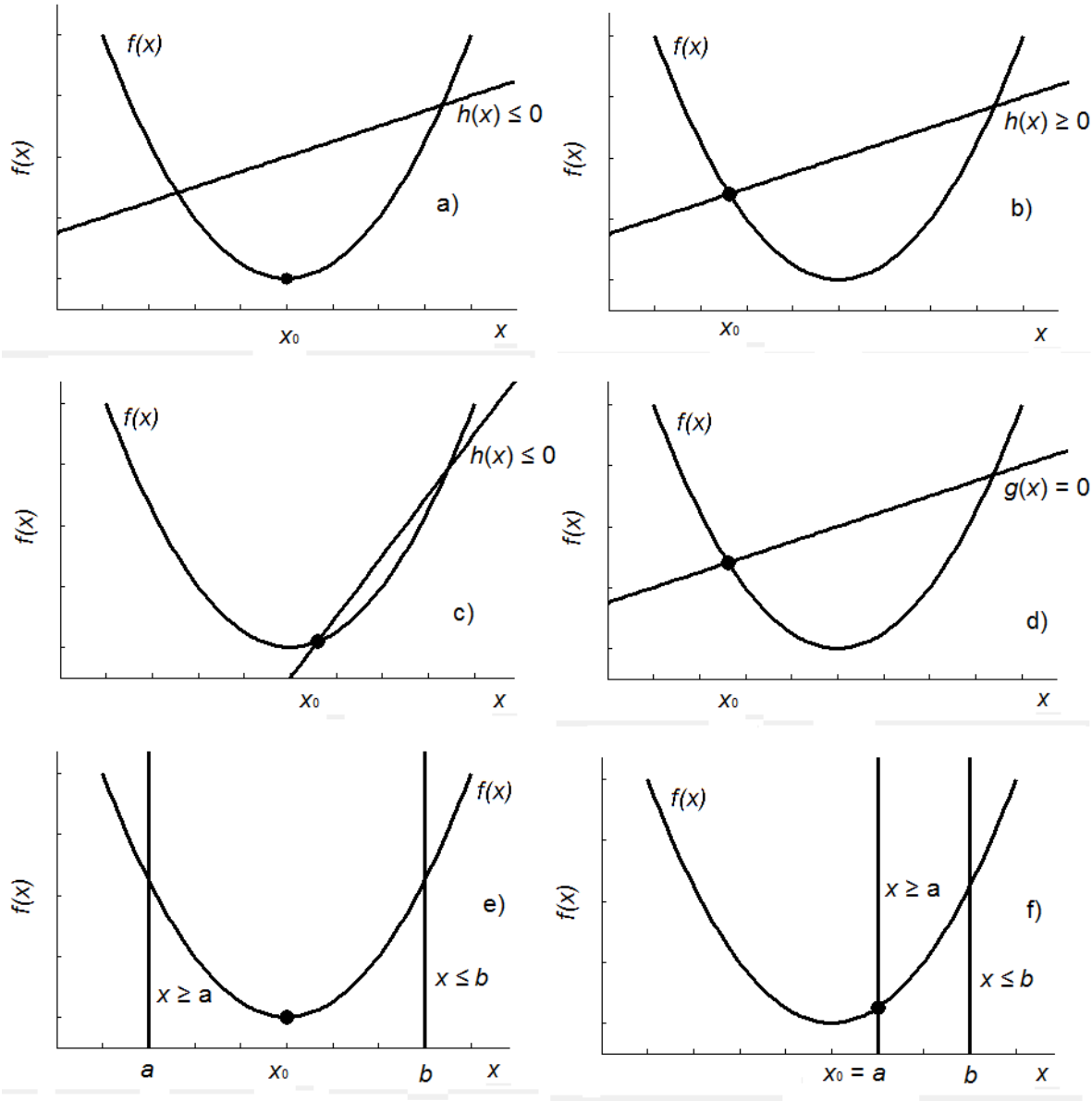
Vázaný extrém je extrém v oblasti přípustných řešení, který splňuje omezení. Omezující podmínky mohou být buď ve tvaru rovnic

$$g(x) = 0, \tag{1.5}$$

nebo ve tvaru nerovnic

$$h(x) \leq 0 \text{, resp. } h(x) \geq 0 . \tag{1.6 a, b}$$

Na obrázku 1.2 jsou uvedeny příklady možných omezení.



Obrázek 1.2 – Příklady vázaných extrémů (minima) funkce jedné proměnné

Optimum

Slovo optimum, můžeme obecně chápat, jako nejvhodnější řešení. Na obrázku 1.2 jsou znázorněny extrémy a současně optima dané funkce s danými omezujícími podmínkami. Avšak funkce vyšších řádů má mnoho extrémů, zatímco optimum má pouze jedno. A to v nejmenší, nebo nejvyšší funkční hodnotu.

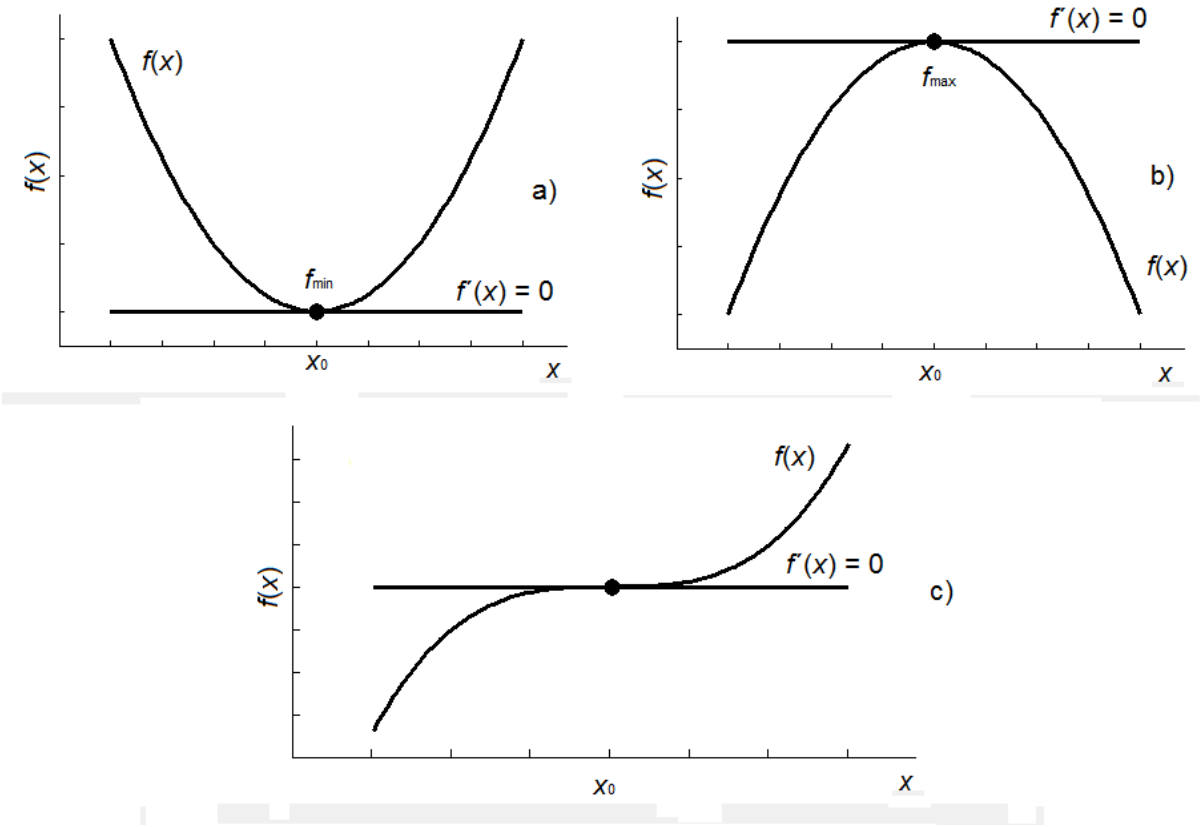
1.2 Metody diferenciální analýzy

Metody klasické diferenciální analýzy se používají v případech, kdy je znám analytický a diferencovatelný tvar účelové funkce *f*(*x*).

Potom nutnou podmínkou existence extrému je

$$f'(x) = 0, \tag{1.7}$$

jJak je graficky znázorněno na obrázku 1.3 a, 1.3 b.



Obrázek 1.3 – Funkce nutných podmínek existence extrémů

Z obrázku 1.3 je vidět, že nutné podmínky jsou splněné, ale v případě 1.3 c extrém neexistuje. Z toho plyne, že podmínka (1.7) není postačující.

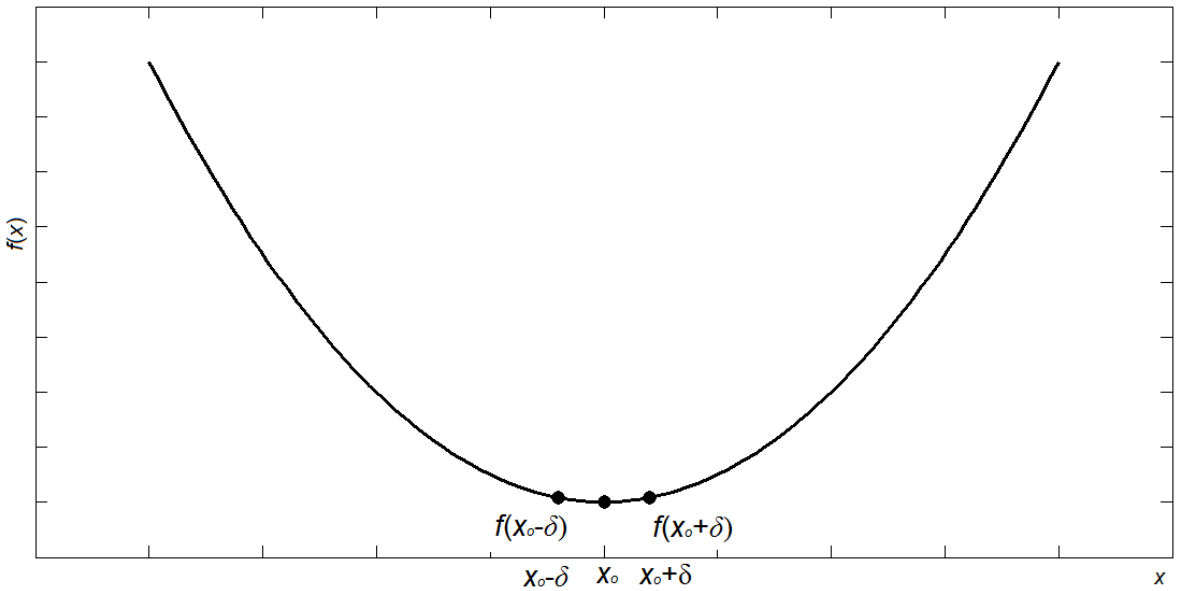
Proto pro určení typu extrémů spojité účelové funkce *f*(*x*) je třeba provézt dodatečnou analýzu pomocí jedné z následujících metod.

1.2.1 Metoda porovnání hodnot funkce

Tato metoda spočívá ve srovnání hodnot účelové funkce $f(x)$ v okolí podezřelého bodu x_o. Platí-li nerovnost

$$f(x_o) < f(x \pm \delta), \tag{1.8}$$

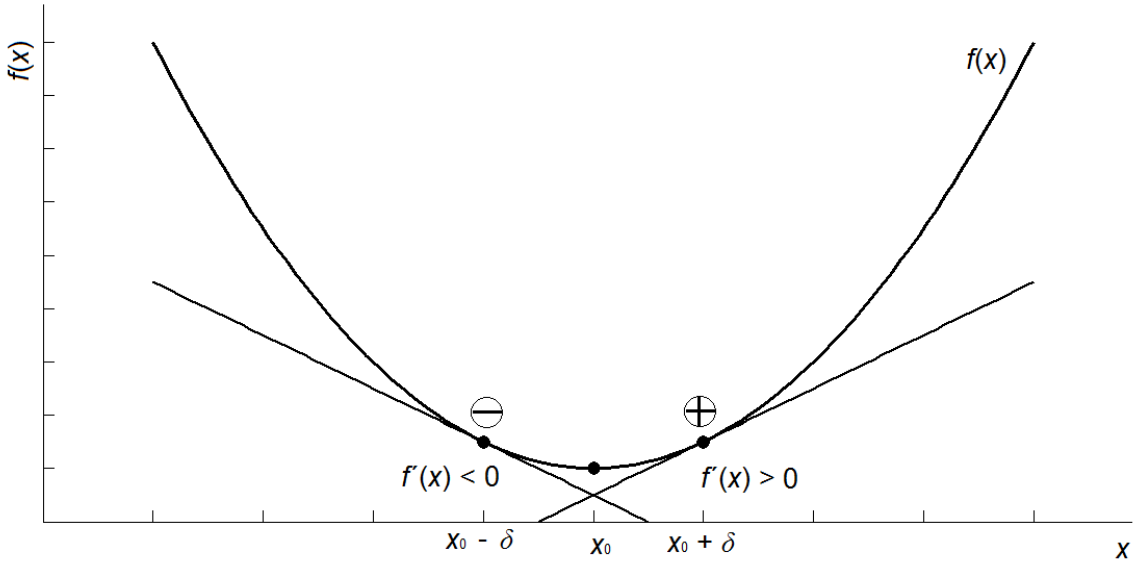
pak se v bodě x_o nachází minimum. Platí-li opačná nerovnost, pak se v bodě x_o nalézá maximum. V jiných případech extrém neexistuje.



Obrázek 1.4 – Metoda porovnávání hodnot funkce

1.2.2 Metoda porovnání znamének první derivace

Při této metodě se určí znaménka první derivace v bodech $x_o - \delta$ a $x_o + \delta$. Budou-li znaménka první derivace shodné, pak v podezřelém bodě x_0 extrém neexistuje (obrázek 1.5). Jestliže jsou znaménka různá a pro bod $x_o - \delta$ kladná, je v bodě x_0 maximum a naopak je-li znaménko pro bod $x_o - \delta$ záporné, je v bodě x_o minimum (obrázek 1.6).δ



Obrázek 1.5 – Metoda porovnání znamének první derivace

1.2.3 Metoda vyšších derivací

Nechť má funkce f v bodě x_0 první a druhou derivaci. Je-li první derivace nulová a druhá derivace nenulová, má funkce f v bodě x_0 lokální extrém. Je-li druhá derivace menší jak nula (rovnice 1.3), jedná se o lokální maximum. Je-li druhá derivace větší jak nula (rovnice 1.4), jedná se o lokální minimum.

Pokud je však současně první i druhá derivace nulová, nemůže se na základně předešlých věty určit chování funkce poblíž takového bodu. Z tohoto důvodu je potřeba použít silnější postačující podmínku.

Silnější postačující podmínka – Platí-li:

$$f'(x) = f''(x) = \ldots = f^{n}(x) = 0, \tag{1.9}$$

$$f^{n+1}(x) \neq 0, \tag{1.10}$$

kde n je liché a funkce f nabývá v bodě x lokálního extrému: pro $f^{n+1}(x) > 0$ lokálního minima a pro $f^{n+1}(x) < 0$ lokálního maxima. Je-li naopak n sudé, nemá funkce f v bodě x žádný lokální extrém.

2 Numerické (iterační) metody

Iterační metody (Vítečková, 2003; Brunovská, 1990; Taufer, 2009) pracují na principu porovnávaní hodnot účelové funkce. Z tohoto důvodu nenastává při výpočtech žádný problém s derivacemi.

V zásadě lze všechny numerické metody jednorozměrné minimalizace spojitých unimodálních funkcí charakterizovat jako metody krokového zlepšování ve směru hledaného optima. Z výchozího stavu x^k se provádí přechod do stavu x^{k+1} o hodnotu Δx, kterou nazýváme krokem. Takže platí

$$x^{k+1} = x^k + \Delta x . \tag{2.1}$$

Je zřejmé, že v případě hledání optima ve tvaru minima, tak úspěšný krok bude dán nerovností

$$f\left(x^{k+1}\right) < \left(f^{x}\right), \tag{2.2}$$

V případě opačné rovnosti je přechod do stavu x^{k+1} nežádoucí.

U numerických metod je nutné zadat požadovanou chybu řešení optimálního řešení x^k, tj. nezáporné číslo ε, pro které k-tá aproximace řešení musí vyhovovat nerovnosti

$$\left|x^{k+1} - x^k\right| \leq \varepsilon . \tag{2.3}$$

Numerické metody jednorozměrné optimalizace se dají rozdělit na dvě základní skupiny:

1. diferenciální (gradientní) metody – vyžadují výpočet hodnot účelové funkce a její první, resp. druhé derivace
2. komparativní metody – vyžadují pouze vypočítávání hodnot účelové funkce.

V následující části budou popsány čtyři komparativní metody, a to: metoda přímého porovnání funkčních hodnot, Fibonacciho metoda, metoda zlatého řezu a metoda půlení intervalu.

2.1 Metoda přímého porovnání funkčních hodnot

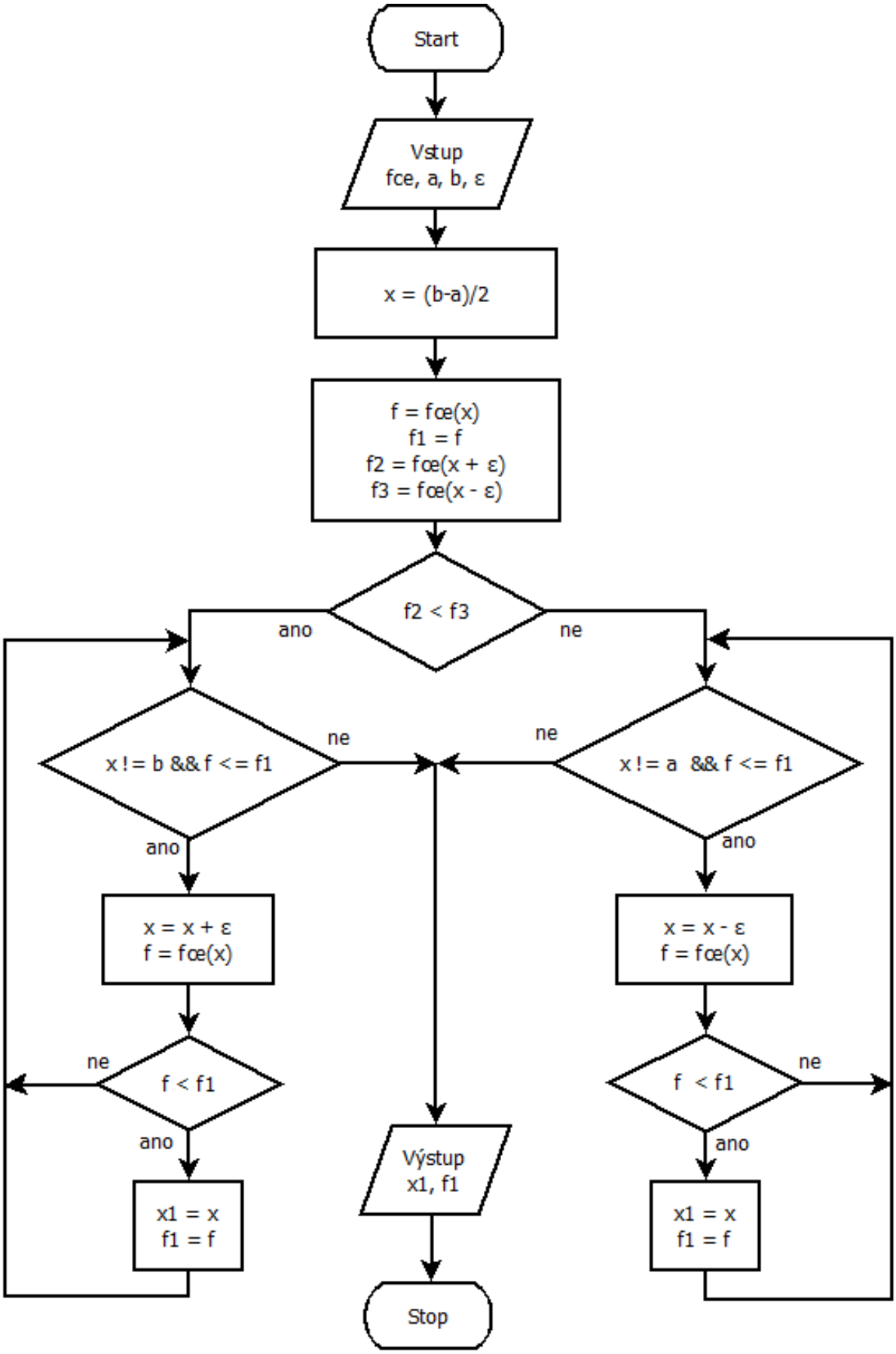
V intervalu, ve kterém se hledá extrém účelové funkce, se zvolí bod x, který je roven středu intervalu. K bodu x se přičte a odečte zvolená chyba řešení a vypočítají se dvě funkční hodnoty. Tyto dvě hodnoty se porovnají a dle rovnosti se bud snižují, nebo zvyšují s krokem zvolené chyby řešení. Při každém kroku se porovnává původní a nová funkční hodnota. Nižší hodnota se uloží do zvolené proměnné. Tímto způsobem se prohledává interval, než je nová hodnota vyšší než stávající, nebo nenarazí na začátek, či konec intervalu, pak se v tomto bodě nachází minimum funkce. Z tohoto textu je patrné, že metoda není efektivní a bude značně pomalá.

U Algoritmu pro hledání maxima funkce stačí do zvolené proměnné ukládat vyšší hodnotu, namísto té menší.

Základní popis algoritmu

1. Zadáme vstupní hodnoty proměnných *fce, a, b, ε*
2. Nastaví se proměnná x na střed intervalu
3. Vypočítají se hodnoty proměnných f, f_1, f_2 a f_3 podle vztahů uvedených v diagramu
4. Jestliže $f_2 < f_3$ vykoná se substituce:
5. Jestliže $x < b$ a zároveň $f <= f_1$ pak
   5.1 $x = x + \varepsilon$, $f = fce(x)$
   5.2 Jestliže $f < f_1$ provede se:
   $$x_1 = x,\ f_1 = f$$
   v opačném případě se pokračuje k bodu 6
6. Vypsání proměnných x_1, f_1
7. Ukončení programu

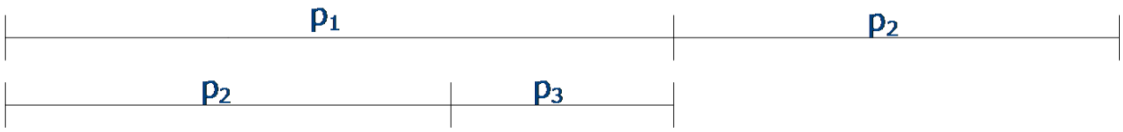
Jestliže podmínka v bodě čtyři bude neplatná, vykoná se stejná substituce s opačným znaménkem u chyby řešení ε a u cyklu while bude podmínka $f >= f_1$.



Obrázek 2.1 – Vývojový diagram metody přímého porovnání funkčních hodnot

2.2 Metoda zlatého řezu

Rozdělme úsečku p na dvě části p_1 a p_2, potom úsečku p_1 na části p_2 a p_3, jak je uvedeno na obrázku.



Obrázek 2.2 – Zlatý řez

Nechť platí:

$$\frac{p_2}{p_1} = \frac{p_3}{p_2} \tag{2.4}$$

$$p_3 = p_1 - p_2 \tag{2.5}$$

Po dosazení: $\frac{p_2}{p_1} = \frac{p_1 - p_2}{p_2}$, poté se pravá strana rovnice vynásobí členem $\frac{p_1}{p_1}$ a dostává se rovnice: $\frac{p_2}{p_1} = \frac{1 - \frac{p_2}{p_1}}{\frac{p_2}{p_1}}$, provede-li se substituce: $x = \frac{p_2}{p_1}$ a dosadí se do rovnice, vzniká rovnice: $x = \frac{1-x}{x}$, celá rovnici se vynásobí neznámou x a dostane se rovnici o jedné neznámé:

$$x^2 + x - 1 = 0 \tag{2.6}$$

Tato rovnice se poté vyřeší.

$$D = b^2 - 4 \cdot a \cdot c = 1 + 4 = 5$$

Tato kvadratická rovnice má dva kořeny, jeden záporný a druhý kladný. Výpočet kladného kořenu této kvadratické rovnice je:

$$x = \frac{-b + \sqrt{D}}{2 \cdot a} = \frac{-1 + \sqrt{5}}{2} \approx 0{,}618.$$

Tomuto poměru říkáme zlatý řez. V další části této práci je tento poměr označen jako kons.

Algoritmus metody zlatého řezu:

Na začátku algoritmu se vypočítají hodnoty bodů x_1, x_2 a jejich funkční hodnoty $f(x_1)$ a $f(x_2)$. Podle platnosti relace posoudíme dva případy:

1. $f(x_1) < f(x_2)$ – interval hledání minima je zkrácen na $\langle x_1, b \rangle$.
2. $f(x_1) > f(x_2)$ – interval hledání minima je zkrácen na $\langle a, x_2 \rangle$.

Při každém průchodu cyklem se zvolený interval zkrátí o násobek $\frac{\sqrt{5}-1}{2}$ a takto se zkracuje do doby, než se dosáhne požadované chyby řešení. Po k- takovýchto krocích se zvolený interval zkrátí o násobek $\left(\frac{\sqrt{5}-1}{2}\right)^k$ jeho původní délky.

Hledání maxima funkce je vyřešeno změnou podmínek a to:

1. $f(x_1) > f(x_2)$ – interval hledání maxima je zkrácen na $\langle x_1, b \rangle$.
2. $f(x_1) < f(x_2)$ – interval hledání maxima je zkrácen na $\langle a, x_2 \rangle$.

Základní popis metody

1. Zadají se vstupní hodnoty proměnných *fce*, *a*, *b*, ε
2. Zadá se konstanta zlatého řezu, jako *kons* = 0,618
3. Vypočítají se proměnné x_1, x_2 podle vztahu, který je uveden na vývojovém diagramu
4. Vypočítají se proměnné $f_1 = fce(x_1)$ a $f_2 = fce(x_2)$
5. Jestliže je $|b - x_1| > \varepsilon$

    5.1 Jestliže $f_1 < f_2$

    $b = x_2$

    $x_2 = x_1$

    $x_1 = a + (1 - kons)(b - a)$

    $f_2 = fce(x_2)$

    $f_1 = fce(x_1)$

    5.2 Jestliže $f_1 > f_2$

    $a = x_1$

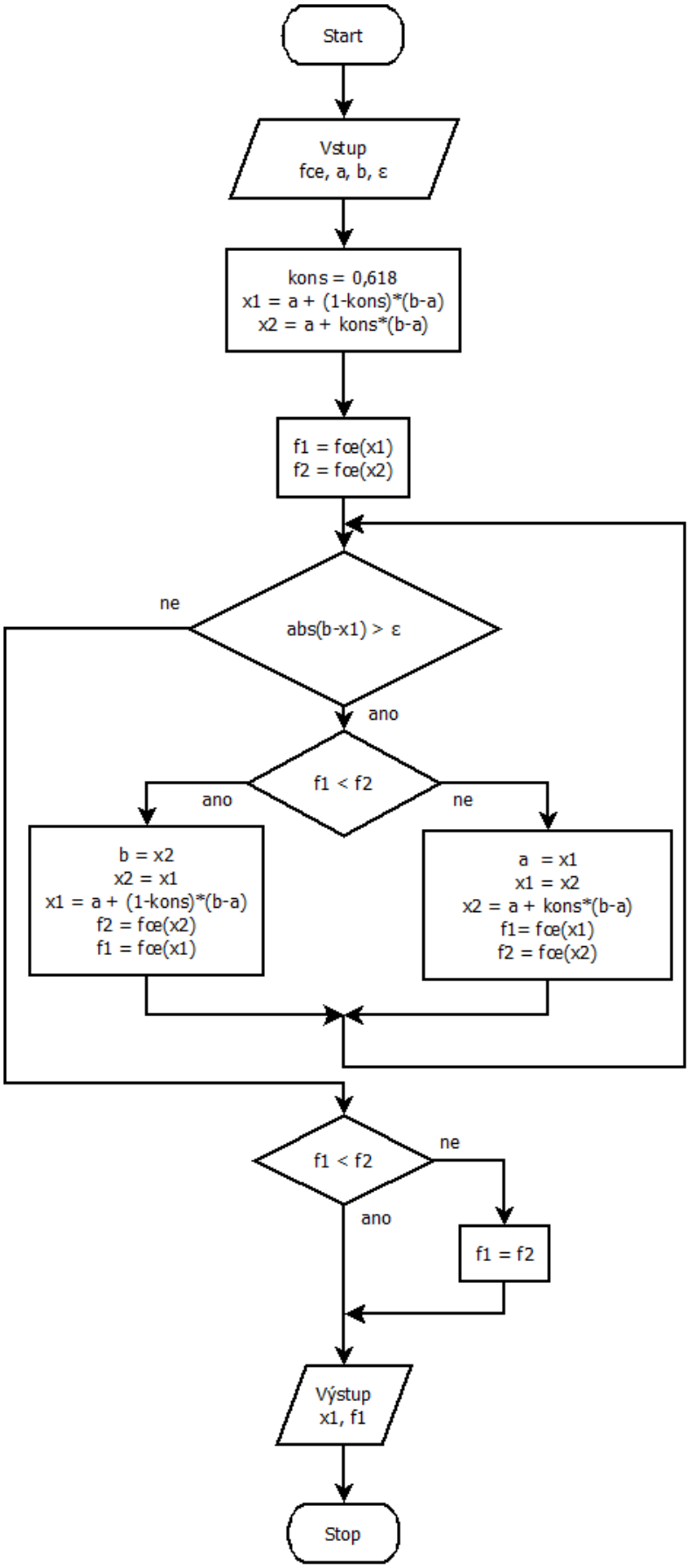
    $x_1 = x_2$

    $x_1 = a + kons*(b - a)$

    $f_2 = fce(x_2)$

    $f_1 = fce(x_1)$
6. Jestliže $f_1 > f_2$

    $f_1 = f_2$
7. Vypsání hodnot proměnných x_1 a f_1
8. Ukončení metody



Obrázek 2.3 – Vývojový diagram metody zlatého řezu

2.3 Fibonacciho metoda

Základem této metody je Fibonacciho posloupnost, která je definována jako $F_0 = F_1 = 1$ a $F_k = F_{k-1} + F_{k-2}$ pro k = 2, 3, …., n.

Pro prvních dvacet členů posloupnosti tedy máme: $\{F_n\} = \{1; 1; 2; 3; 5; 8; 13; 21; 34; 55; 89; 144; 233; 377; 610; 987; 1597; 2584; 4181; 6765; 10946\}$.

Algoritmus Fibonacciho metody:

Algoritmu je podobný jako u metody Zlatého řezu. Na začátku algoritmu se vypočítají body x_1, x_2 a jejich funkční hodnoty $f(x_1)$ a $f(x_2)$. Podle platnosti relace posoudíme dva případy:

1. $f(x_1) < f(x_2)$ – interval hledání minima je zkrácen na $\langle x_1, b \rangle$.
2. $f(x_1) > f(x_2)$ – interval hledání minima je zkrácen na $\langle a, x_2 \rangle$.

Při každém průchodu cyklem se interval zkrátí o násobek $\frac{F_{n-1}}{F_n}$ a $n = n - 1$ při každém průchodu cyklem. Z tohoto plyne, že na začátku algoritmu dochází ke stejnému zkracování jako u metody zlatého řezu. U konce algoritmu je zkracování nepravidelné a pohybuje se kolem hodnoty zlatého řezu. Jednou je zkrácení větší a po druhé menší, než hodnota zlatého řezu. Takto se interval zkracuje do doby, než dosáhne požadovaného počtu opakování *n*.

Ukázka: $\frac{F_5}{F_6} = 0{,}6154$, $\frac{F_4}{F_5} = 0625$, $\frac{F_3}{F_4} = 0{,}6$, $\frac{F_2}{F_3} = 0{,}6667$.

Z ukázky lze vidět, že na konci každého cyklu převládá větší zkracování intervalu oproti metodě zlatého řezu a z tohoto důvodu je Fibonacciho metoda efektivnější.

Hledání extrému maxima funkce je vyřešeno změnou podmínek a to:

1. $f(x_1) > f(x_2)$ – interval hledání maxima je zkrácen na $\langle x_1, b \rangle$.
2. $f(x_1) < f(x_2)$ – interval hledání maxima je zkrácen na $\langle a, x_2 \rangle$.

Základní popis metody

1. Zadají se vstupní hodnoty proměnných *fce*, *a*, *b*, ε
2. Vypočítají se hodnoty proměnných x_1, x_2 podle vztahu, který je uveden na vývojovém diagramu a nastaví $k = 1$
3. Vypočítají se hodnoty proměnné $f_1 = fce(x_1)$ a $f_2 = fce(x_2)$
4. Jestliže je $k \neq n$

   4.1 Jestliže $f_1 < f_2$

   $b = x_2$

   $x_2 = x_1$

   $f_2 = fce(x_1)$

   $x_1 = a + b - x_2$

   $f_1 = fce(x_1)$

   4.2 Jestliže $f_1 > f_2$

   $a = x_1$

   $x_1 = x_2$

   $f_1 = fce(x_2)$

   $x_1 = a + b - x_1$

   $f_2 = fce(x_2)$

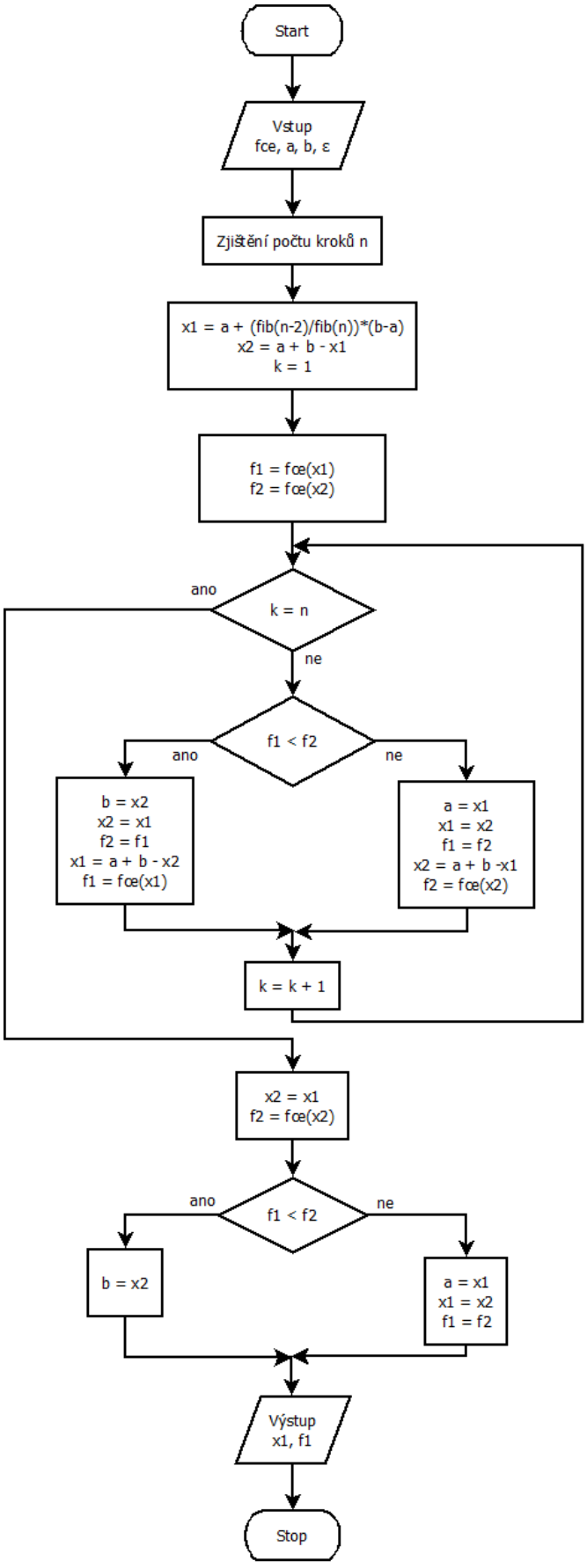
   4.3 Zvýšení hodnoty k o jedna
5. Přirazení hodnoty x_1 do proměnné x_2 a uložení funkční hodnoty $f(x_2)$ do proměnné f_2
6. Jestliže $f_1 < f_2$

   $b = x_2$
7. Jestliže $f_1 > f_2$

   $a = x_2$

   $x_1 = x_2$

   $f_1 = f_2$
8. Vypsání hodnot x_1 a f_1
9. Ukončení metody



Obrázek 2.4 – Vývojový diagram Fibonacciho metody

2.4 Metoda půlení intervalu

V intervalu, ve kterém se hledá extrém účelové funkce, se zvolí tři body, přičemž první bod x_1 je roven začátku intervalu, koncový bod x_3 je roven konci intervalu a třetí bod x_2, který je roven středu intervalu, poté se vypočítají funkční hodnoty těchto bodů. Při každém průchodu cyklem se pomocí algoritmu najde nejmenší funkční hodnota f_1 a k ní daný argument funkce x_1. K tomuto budu x_1 se přičte a odečte proměnná p, která má výchozí hodnotu jedna čtvrtina z intervalu $\langle a,b \rangle$. Takto vzniknou nové tři body, které tvoří nový interval funkce. Při každém průchodu cyklem se proměnná p zkrátí o jednu polovinu. Tímto způsobem zkracujeme interval funkce do doby, než $|x_1-x_2| < \varepsilon$.

Jestliže chceme, aby tato metoda fungovalo pro hledání maxima funkce, stačí jen pozměnit algoritmus na místo hledání nejmenší funkční hodnoty, nalézt nejvyšší funkční hodnotu.

Základní popis metody

1. Zadají se vstupní hodnoty proměnných *fce, a, b, ε*
2. Nastaví se proměnné x_1, x_2 a x_3 po celém rozsahu intervalu
3. Nastaví se proměnná p na jednu čtvrtinu délky intervalu
4. Jestliže $|x_1 - x_2| > \varepsilon$ provede se:

   4.1 Výpočet funkčních hodnot argumentů funkcí

   4.2 Nalezení nejmenší funkční hodnoty a přiřazení do proměnné f_1

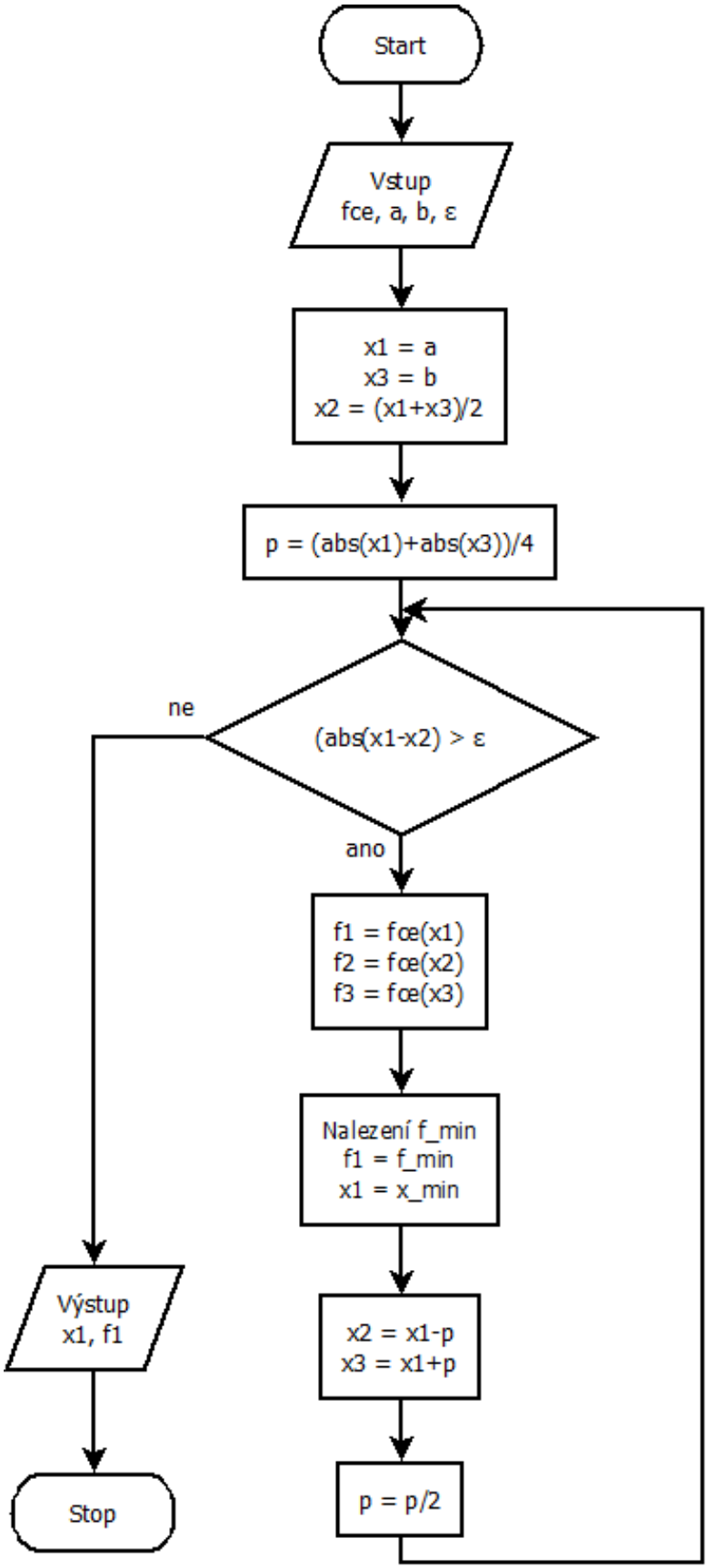
   4.3 Argument funkce s nejmenší funkční hodnotou přiřadit do proměnné x_1

   4.4 Výpočet proměnné x_2 zmenšené o p a proměnné x_3 zvětšené o p

   4.5 Zmenšení proměnné p o jednu polovinu

   4.6 Návrat k bodu 4
5. Jestliže $|x_1 - x_2| < \varepsilon$ provede se:

   Vypsání hodnoty proměnných x_1 a f_1
6. Ukončení metody.



Obrázek 2.5 – Vývojový diagram metody půlení intervalu

3 Realizace algoritmů v prostředí Matlab

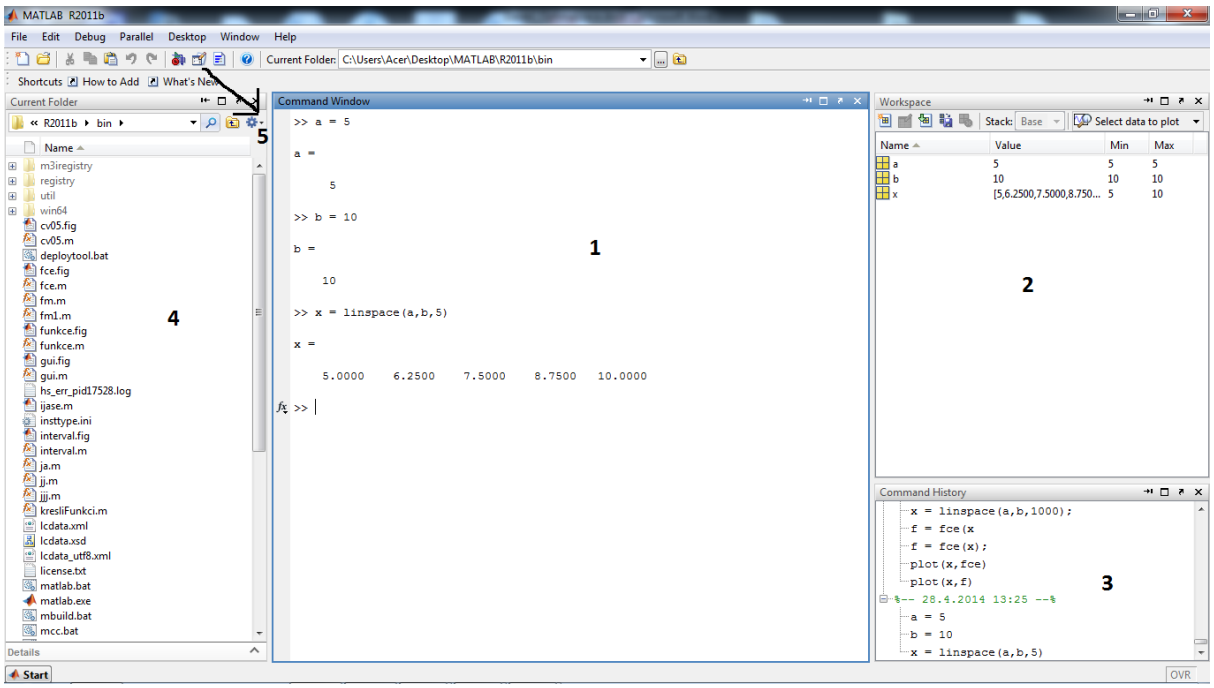
Jako programovací jazyk pro tuto práci se zvolilo vývojové prostředí Matlab, toto prostředí je popsáno v kapitole níže.

3.1 Popis vývojového prostředí Matlab

Matlab, neboli matrix laboratory je interaktivní programové prostředí. Tento program je vyvíjen společností MathWorks a je k dispozici jak pro Windows, Linux, tak i Mac OS. Program Matlab umožnuje například práci s vektory a maticemi, vykreslování 2-D a 3-D grafů, nebo také vytváření aplikací pomocí uživatelského rozhraní GUI. Vlastní programovací jazyk vychází z jazyka Fortran.

3.1.1 Úvodní okno Matlabu

Při spuštění programu se zobrazí úvodní menu, které je vidět na obrázku 3.1. Toto okno lze rozdělit na čtyři části. První část označena jako *Command Window*, neboli příkazový řádek je uživatelem nejčastěji využívaný a slouží k zápisu příkazů. Druhá část pojmenována *Workspace* slouží k uložení všech dat, které uživatel zapsal do příkazové řádky. Třetí část značena jako *Command history* ukládá všechny použité příkazy, které uživatel použil. Ve čtvrté části pojmenované *Curennt Folder* se zobrazují všechny soubory, které jsou uloženy v daném adresáři. Jedná se o různá skripta, funkce, ale také o další adresáře, atd. Pod číslem pět se skrývá ikona pro spuštění GUI (Graphical user interface), které se dá spustit levým kliknutím myši, nebo napsáním do příkazového řádku *guide*. Toto prostředí bude popsáno v kapitole 3.1.2.

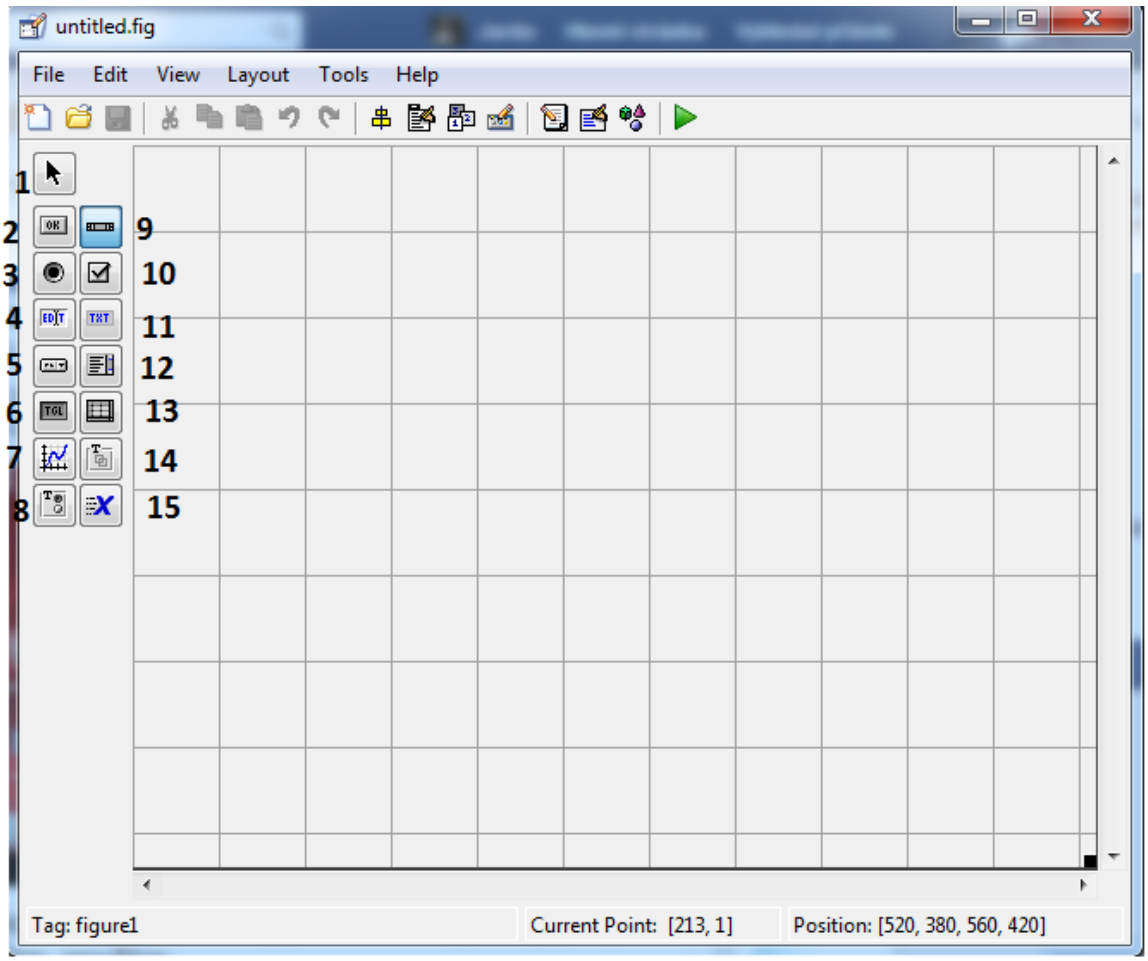


Obrázek 3.1 – Úvodní okno Matlabu

3.1.2 Graphical user interface

Graphical user interface, neboli uživatelské rozhraní slouží k vytváření různých aplikací. V následujících patnácti bodech budou popsány položky GUI.

1. Select – slouží k výběru objektů
2. Push button – jednostavové tlačítko, které slouží k zavolání příkazu
3. Radio button – přepínač, například hodnoty max a min
4. Edit text – editovatelný text, umožnuje načíst řetězec
5. Pop-up menu – rozvinovací menu, slouží k výběru položky
6. Toggle button – tlačítko, které umožňuje měnit nastavení mezi dvěma stavy
7. Axes – Prostor pro vykreslení grafu
8. Button group – seskupení tlačítek, stisknutí jednoho tlačítka vypne jiné
9. Slider – jezdec, který se pohybuje po skocích
10. Check box – zaškrtávací tlačítko, při změně stavu nastává callback
11. Static text – needitovatelný text, zpravidla se používá jako popiska
12. ListBox – seznam údajů, lze vybrat jedno i více položek
13. Table – tabulka
14. Panel – slouží k seskupení objektů
15. ActiveX control – má mnoho funkcí, slouží například k načtení videa

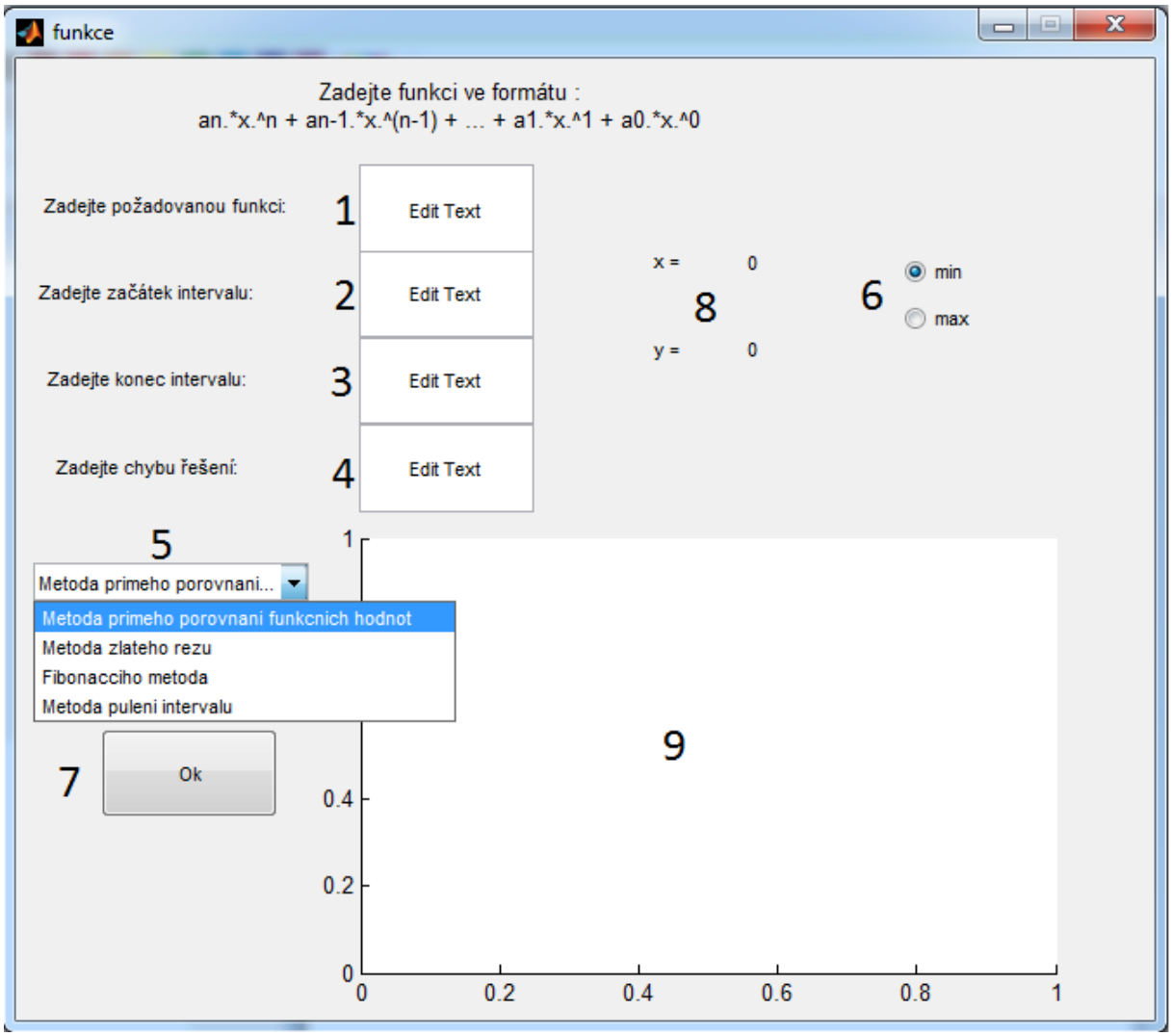


Obrázek 3.2 – Uživatelské rozhraní GUI

3.2 Popis programového řešení

V prvním kroku se spustí prostředí Matlab a do příkazové řádky se napíše příkaz *funkce*, nebo se v okně *Curennt Folder* spustí soubor *funkce.m* pomocí dvojkliku levého tlačítka myši, a poté se spustí program pomocí tlačítka run. *Funkce* je jméno souboru, který se zvolil pro grafické rozhraní. Po jedné z těchto možností se spustí program a naběhne úvodní dialogové okno, které je vidět na obrázku 3.3.

Vybrané metody jsou napsány jako funkce a při spuštění programu je jedna z těchto funkcí zavolána. Tyto čtyři funkce mají vstupní proměnné *fce*, *a*, *b*, *h*, *ε*. Přičemž proměnná *fce* je zvolená funkce, proměnné *a*, *b* umožňují zadávat začátek a konec intervalu, proměnná *h* rozhoduje, zda uživatel chce vyhledat minimum, nebo maximum funkce a proměnnou *ε* uživatel zadává zvolenou chybu řešení.



Obrázek 3.3 – Úvodní dialogové okno

První kolonka slouží pro zadání požadované funkce. Program je napsaný v prostředí Matlab a z tohoto důvodu se musí vstupní funkce zadat ve formátu:

$a_n.*x.\hat{}n + a_{n-1}.*x.\hat{}(n-1) + ... + a_1.*x + a_0$. Neboli, před každým operátorem se musí uvést tečka.

Například: $5.*x.\hat{}3 + 8.*x.\hat{}2 - x + 5$

Do druhé a třetí kolonky uživatel zadává reálné číslo a slouží jako začátek a konec intervalu funkce.

Do čtvrté kolonky uživatel zadává reálné číslo, které reprezentuje chybu řešení.

Pátá kolonka, slouží k vybrání metody, která najde extrém účelové funkce. Uživatel má na výběr ze čtyř metod.

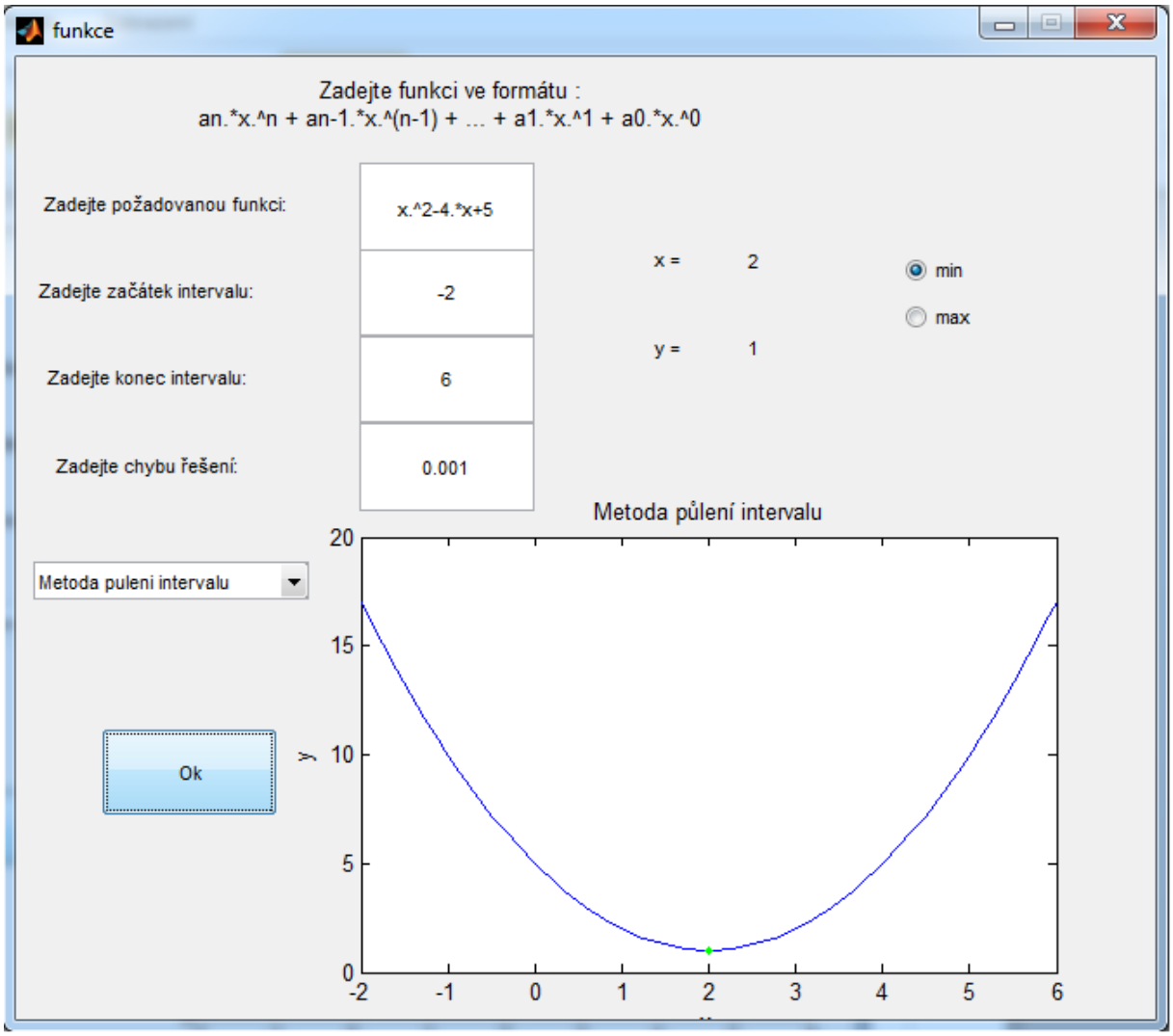
Jako šestý bod, se může uvést výběr, zda uživatel chce nalézt minimum, nebo maximum funkce.

Sedmý bod je poslední funkční tlačítko OK a slouží ke spuštění samotného programu.

V osmém bodě se vypisují hodnoty proměnných *x* a *y*, které udávají souřadnice nalezeného řešení.

Posledním devátým objektem je plocha pro vykreslení grafu a zobrazení nalezeného extrému.

Jak lze vidět z obrázku 3.4, po zvolení funkce, vyplnění začátku a konce intervalu a zvolené metodě půlení intervalu se vykreslil graf s vyznačeným bodem a vypsanými souřadnicemi nalezeného řešení.



Obrázek 3.4 – Ukázka dialogového okna

4 Ověření funkčnosti zpracovaných programů

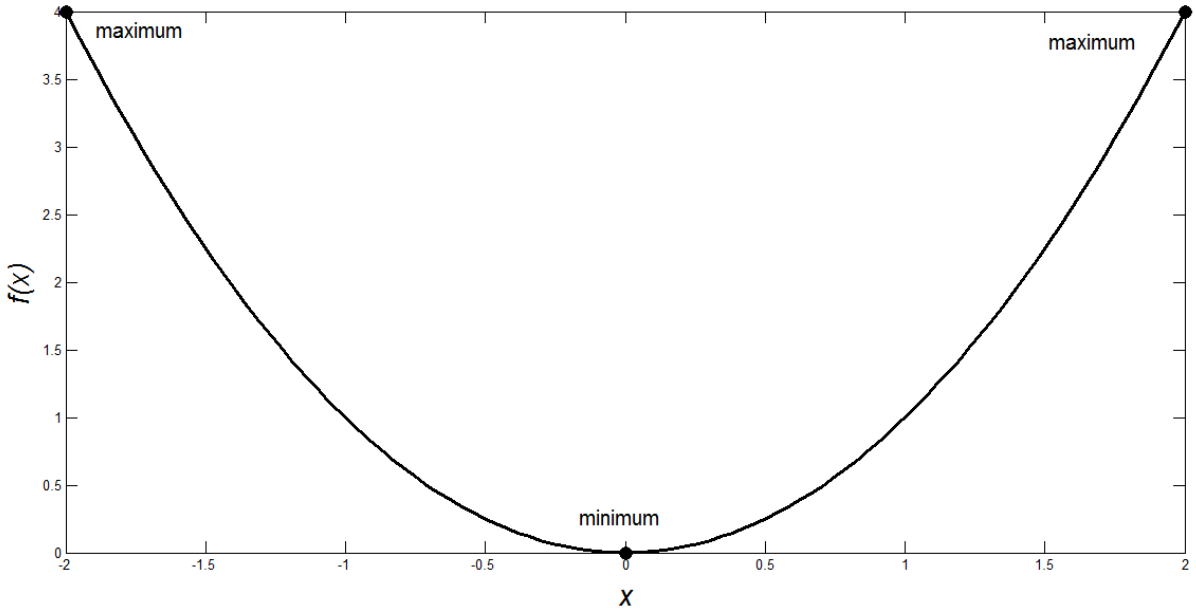
Pro ověření funkčnosti programů byly vybrány tři testovací funkce, jejichž popis je uveden níže.

4.1 Testovací funkce

První testovací funkce je:

$$f(x) = x^2, \text{ pro každé } x \in \langle -2; 2 \rangle \tag{4.1}$$

Tato testovací funkce má na intervalu $\langle -2; 2 \rangle$ tři lokální extrémy. Jeden pro extrém minima funkce [0; 0] a dva pro extrém maxima funkce [-2; 4], [2; 4]. Zároveň jsou tyto lokální extrémy i optimem funkce.

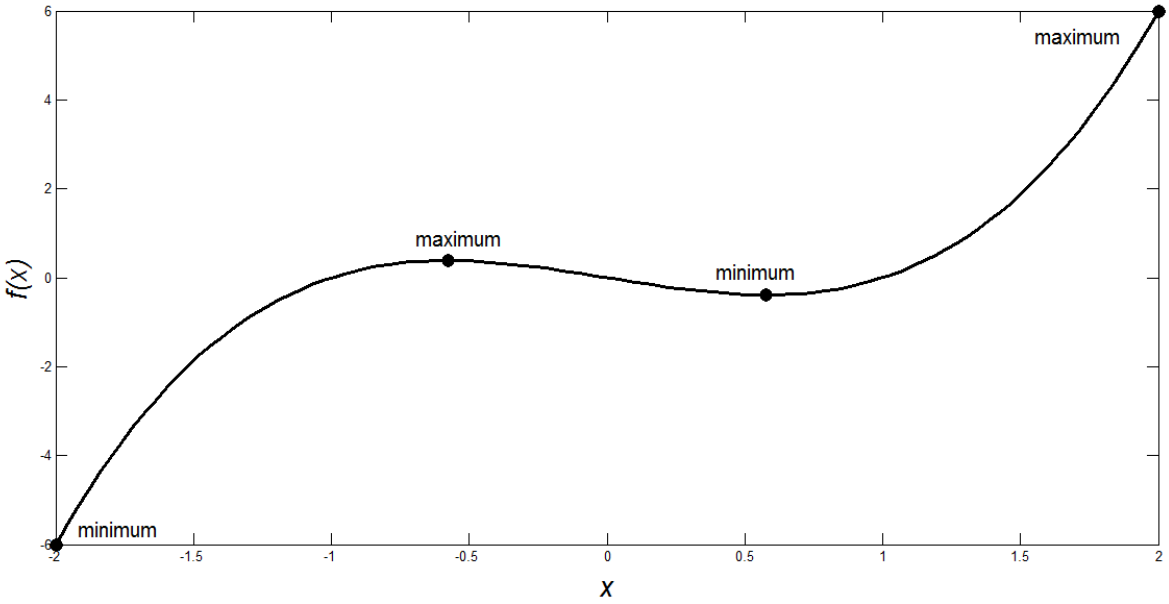


Obrázek 4.1 – Graf testovací funkce $f(x) = x^2$

Jako druhá testovací funkce se zvolila:

$$f(x) = x^3 - x, \text{ pro každé } x \in \langle -2; 2 \rangle \tag{4.2}$$

Tato testovací funkce má na intervalu $\langle -2; 2 \rangle$ čtyři lokální extrémy, dva pro minimum [+0,57735; -0,3849], [-2; -6] a dva pro maximum funkce [-0,57735; +0,3849], [2; 6].

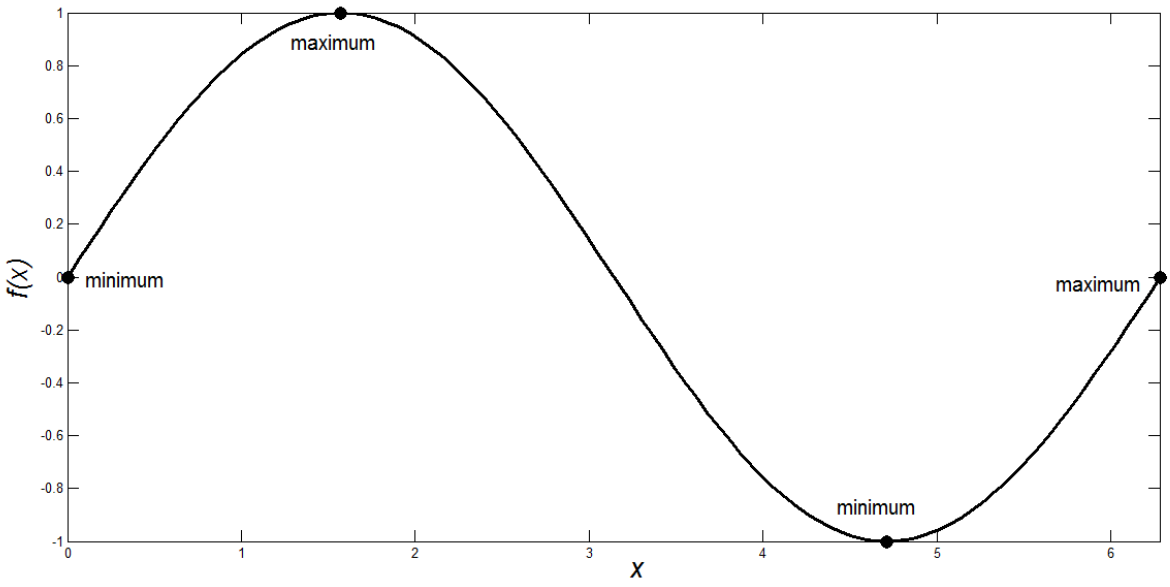


Obrázek 4.2 – Graf testovací funkce $f(x) = x^3 - x$

Jako třetí testovací funkce se zvolila:

$$f(x) = \sin(x), \text{ pro každé } x \in \langle 0; 2 \cdot \pi \rangle \tag{4.3}$$

Tato testovací funkce má na intervalu $\langle 0; 2 \cdot \pi \rangle$ čtyři lokální extrémy, dva pro minimum [0, 0], [4,71239; -1] a dva pro maximum funkce [1,5708; +1], [6,283; 0].



Obrázek 4.3 – Graf testovací funkce $f(x) = \sin(x)$

4.2 Výsledky testování

Testování provozuschopnosti jednotlivých programů bylo provedeno na výše definovaných funkcí. Testování proběhlo pro různé intervaly a přípustnou chybu řešení (nalezení optima).

Níže jsou uvedeny v tab. 4.1 – 4.6 vybrané výsledky testování.

Tabulka 4.1 – funkce $f(x) = x^2$ pro hledání minima [0; 0]

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle -2;+2 \rangle$	0,01	1	[0; 0]
Fibonacciho metoda	$\langle -2;+2 \rangle$	0,01	12	[0,002; 0]
metoda zlatého řezu	$\langle -2;+2 \rangle$	0,01	12	[-0,0015; 0]
metoda půlení intervalu	$\langle -2;+2 \rangle$	0,01	8	[0; 0]

Tabulka 4.2 – funkce $f(x) = x^2$ pro hledání maxima [-2; 4], [2; 4]

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle -2;+2 \rangle$	0,01	200	[-2; 4]
Fibonacciho metoda	$\langle -2;+2 \rangle$	0,01	12	[1,996; 3,9477]
metoda zlatého řezu	$\langle -2;+2 \rangle$	0,01	12	[1,9923; 3,981]
metoda půlení intervalu	$\langle -2;+2 \rangle$	0,01	8	[-2; 4]

Tabulka 4.3 – funkce $f(x) = x^3 - x$ pro hledání minima [-2; -6], [0,57735; -0,3849]

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle -2;+2 \rangle$	0,01	59	[0,58; -0,3849]
Fibonacciho metoda	$\langle -2;+2 \rangle$	0,01	12	[0,5775; -0,3849]
metoda zlatého řezu	$\langle -2;+2 \rangle$	0,01	12	[0,5745; -0,3849]
metoda půlení intervalu	$\langle -2;+2 \rangle$	0,01	8	[-2; -6]

Tabulka 4.4 – funkce $f(x) = x^3 - x$ pro hledání maxima [-0,57735; 0,3849], [2; 6].

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle -2;+2 \rangle$	0,01	59	[-0,58; 0,3849]
Fibonacciho metoda	$\langle -2;+2 \rangle$	0,01	12	[-0,5775; 0,3849]
metoda zlatého řezu	$\langle -2;+2 \rangle$	0,01	12	[-0,5774; 0,3849]
metoda půlení intervalu	$\langle -2;+2 \rangle$	0,01	8	[2; 6]

Tabulka 4.5 – funkce $f(x) = \sin(x)$ pro hledání minima [0, 0], [4,71239; -1]

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle 0,+2 \cdot \pi \rangle$	0,01	158	[4,71; -1]
Fibonacciho metoda	$\langle 0,+2 \cdot \pi \rangle$	0,01	13	[4,711; -1]
metoda zlatého řezu	$\langle 0,+2 \cdot \pi \rangle$	0,01	13	[4,712; -1]
metoda půlení intervalu	$\langle 0,+2 \cdot \pi \rangle$	0,01	10	[4,71; -1]

Tabulka 4.6 – funkce $f(x) = \sin(x)$ pro hledání maxima [1,5708; +1], [6,283; 0]

Metoda	interval $\langle a,b \rangle$	Chyby řešení	Počet iterací	Nalezený bod
metoda přímého porovnání funkčních hodnot	$\langle 0,+2\cdot\pi \rangle$	0,01	158	[1,57; 1]
Fibonacciho metoda	$\langle 0,+2\cdot\pi \rangle$	0,01	13	[1,569; 1]
metoda zlatého řezu	$\langle 0,+2\cdot\pi \rangle$	0,01	13	[1,5697; 1]
metoda půlení intervalu	$\langle 0,+2\cdot\pi \rangle$	0,01	10	[1,57; 1]

U všech nalezených bodů se souřadnice zaokrouhlily na čtyři desetinné místa.

4.3 Zhodnocení výsledků

Z tabulek 4.1 – 4.6 je vidět, že vybrané čtyři programy jsou provozuschopné pro zvolené testovací funkce, avšak testování programů se vyzkoušelo na mnoha funkcích. Z rozsáhlejšího testování plyne, že naprogramované programy jsou provozuschopné pro jakoukoli funkci o jedné proměnné.

V programech je poměrně velký rozdíl v počtu iterací a nepatrný rozdíl chyby řešení nalezených souřadnic.

Všechny programy určily souřadnice extrému účelové funkce s menší chybou řešení, než byla zvolená, jak lze vidět z tabulek 4.1 – 4.6.

Z těchto dvou důvodu tvrdím, že cíl bakalářské práce byl splněn.

Závěr

Cíl bakalářské práce byl splněn. Koncepce řešení umožňuje další rozšíření programu (rozšíření o další metody řešení).

Zpracované programy čtyř metod hledání extrému funkce jedné proměnné prokázaly na základě požadovaných výsledků požadovanou provozuschopnost.

Ve vytvořeném programu si uživatel může vybrat jednu ze čtyř metod, kterou chce nalézt extrém účelové funkce. Velkou předností tohoto programu je, že si uživatel může zadat jakoukoli rovnici o jedné neznámé. V programu si také uživatel můžu vybrat, zda chce nalézt extrém maxima nebo minima funkce a zvolit si libovolný interval funkce. Poslední výhodou tohoto programu je vykreslení zadané funkce společně se znázorněným bodem a vypsání souřadnic nalezeného bodu.

Literatura

BRUNOVSKÁ, A. 1990. *Malá optimalizácia*. Bratislava: ALFA, ISBN 80-05-00770-1.

DRÁBEK, O.; TAUFER, I. 1985. *Automatizované systémy řízení technologických procesů*. Pardubice: Vysoká škola chemicko-technologická Pardubice.

TAUFER, I.; KOTYK, J.; HRUBINA, K.; TAUFER, J. 2009. *Algoritmy a algoritmizace - vývojové diagramy*. Pardubice: Univerzita Pardubice. ISBN 978-80-7395-182-5.

VÍTEČKOVÁ M.; JEDLIČKA, D. 2003. *Statická optimalizace*. [on-line]. Ostrava: VŠB-TUO, KATŘ. [cit. 11. 5. 2014] Dostupné na http://books.fs.vsb.cz/StatickaOptimalizac

Přílohy

A – Metoda přímého porovnání funkčních hodnot

B – Metoda zlatého řezu

C – Metoda půlení intervalu

D – Fibonacciho metoda

Příloha A – Metoda přímého porovnání funkčních hodnot

```
function [x1,f1] = mppfh(fce,a,b,h,presnost)
x=linspace(a,b,100);
f=fce(x);
plot(x,f)
hold on
x=(a+b)/2;
f = fce(x);
f1 = f;
f2 = fce(x+presnost);
f3 = fce(x-presnost);
if(h == 1)
    if(f2<=f3)
        while((x<b)&&(f<=f1))
             x = x+presnost;
             f=fce(x);
             if(f<f1)
                 x1 = x;
                 f1 = f;
             end
         end
    else
        while((x>a)&&(f<=f1))
            x = x-presnost;
            f = fce(x);
            if(f<f1)
                x1 = x;
                f1 = f;
            end
        end
    end
else
    if(f2>f3)
        while((x<b)&&(f>=f1))
            x = x+presnost;
            f = fce(x);

            if(f>f1)
                x1 = x;
                f1 = f;
            end
        end
    else
        while((x>a)&&(f>=f1))
            x = x-presnost;
            f = fce(x);

            if(f>f1)
                x1 = x;
                f1 = f;
            end
        end
    end
end
plot(x1,f1,'g.');
hold off
```

Příloha B – Metoda zlatého řezu

```
function [x1,y1] = mzr(fce,a,b,h,presnost)
x=linspace(a,b,100);
f=fce(x);
plot(x,f)
hold on
kons=(sqrt(5)-1)/2;
x1=a+(1-kons)*(b-a);
x2=a+kons*(b-a);
f1=fce(x1);
f2=fce(x2);
if(h == 1)
    while(abs(b-x1)>presnost)
        if(f1<f2);
            b=x2;
            x2=x1;
            x1=a+(1-kons)*(b-a);
            f1=fce(x1);
            f2=fce(x2);
        else
            a=x1;
            x1=x2;
            x2=a+kons*(b-a);
            f1=fce(x1);
            f2=fce(x2);
        end
    end
    if(f1<f2)
        y1 = f1;
        plot(x1,f1,'g.')
    else
        y1 = f2;
        plot(x2,f2,'g.')
    end
else
    while(abs(b-x1)>presnost)
        if(f1>f2);
            b=x2;
            x2=x1;
            x1=a+(1-kons)*(b-a);
            f1=fce(x1);
            f2=fce(x2);
        else
            a=x1;
            x1=x2;
            x2=a+kons*(b-a);
            f1=fce(x1);
            f2=fce(x2);
        end
    end
    if(f1>f2)
        y1 = f1;
        plot(x1,f1,'g.')
    else
        y1 = f2;
        plot(x2,f2,'g.')
    end
end
hold off
```

Příloha C – Metoda půlení intervalu

```
function [x1,y1] = mpi(fce,a,b,h,presnost)
x=linspace(a,b,100);
f = fce(x);
plot(x,f)
hold on
x1 = a;
x3 = b;
x2 = (x1+x3)/2;
p = (abs(x1)+abs(x3))/4;
if(h == 1 )
    while(abs(x1-x2)>presnost)
        f1 = fce(x1);
        f2 = fce(x2);
        f3 = fce(x3);
        if(f1<=f2 && f1<=f3)
        elseif(f2<=f1 && f2<=f3)
            f1 = f2; x1 = x2;
        elseif(f3<=f1 && f3<=f2)
            f1 = f3; x1 = x3;
        end
        x2=x1-p;
        if(x2<a && fce(x2)<f1)
            x2 = a;
        end
        x3=x1+p;
        if(x3>b && fce(x3)<f1)
            x3 = b;
        end
        p = p/2;
    end
else
    while(abs(x1-x2)>presnost)
        f1 = fce(x1);
        f2 = fce(x2);
        f3 = fce(x3);
        if(f1>=f2 && f1>=f3)
        elseif(f2>=f1 && f2>=f3)
            f1 = f2; x1 = x2;
        elseif(f3>=f1 && f3>=f2)
            f1 = f3; x1 = x3;
        end
        x2=x1-p;
        if(x2<a && fce(x2)>f1)
            x2 = a;
        end
        x3=x1+p;
        if(x3>b && fce(x3)>f1)
            x3 = b;
        end
        p = p/2;
    end
end
y1 = f1;
plot(x1,y1,'g.');
hold off
```

Příloha D – Fibonacciho metoda

```
function [x1,y1] = fm(fce,a,b,h,presnost)
x=linspace(a,b,100);
f=fce(x);
plot(x,f)
hold on
k = 22;
fib(1) = 1;
fib(2) = 1;
for n = 3:1:k
    fib(n) = fib(n-2) + fib(n-1);
end;
r = (b-a)/presnost;
n = 1;
while(fib(n)<r)
    n = n + 1
end
x1 = a+(b-a)*fib(n-1)/fib(n+1);
x2 = a+b-x1;
f1 = fce(x1);
f2 = fce(x2);
k = 1;
if(h == 1)
    while(k ~= n-1)
        k = k+1;
        if(f1<f2);
            b = x2;
            x2 = x1;
            f2 = f1;
            x1 = a+b-x2;
            f1 = fce(x1);
        else
            a = x1;
            x1 = x2;
            f1 = f2;
            x2 = a+b-x1;
            f2 = fce(x2);
        end
    end

    x2 = x1 ;
    f2 = fce(x2);

    if(f1<f2)
        b = x2;
        y1 = f1;
    else
        a = x1;
        x1 = x2;
        y1 = f2;
    end
else
    while(k ~= n-1)
        k = k+1;
        if(f1>f2);
```

```
            b = x2;
            x2 = x1;
            f2 = f1;
            x1 = a+b-x2;
            f1 = fce(x1);
        else
            a = x1;
            x1 = x2;
            f1 = f2;
            x2 = a+b-x1;
            f2 = fce(x2);
        end
    end

    x2 = x1 ;
    f2 = fce(x2);

    if(f1>f2)
        b = x2;
        y1 = f1;
    else
        a = x1;
        x1 = x2;
        y1 = f2;
    end
end
plot(x1,y1,'g.');
hold off
```