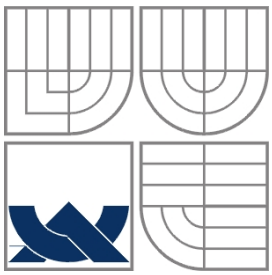


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ADOBE FLASH – INTERAKTIVNÍ PREZENTACE (OPTIMALIZACE)

ADOBE FLASH – INTERACTIVE PRESENTATION (OPTIMIZATION)

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK DOSTÁL**
AUTHOR

VEDOUCÍ PRÁCE **ING. RADOMIL MATOUŠEK, PH.D.**
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Marek Dostál

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Adobe Flash – interaktivní prezentace (optimalizace)

v anglickém jazyce:

Adobe Flash – interactive presentation (optimization)

Stručná charakteristika problematiky úkolu:

Daná BP bude pomocí technologie Adobe Flash prezentovat tři vybrané optimalizační metody z oblasti tzv. soft-computingu. Pro porovnání budou rovněž demonstrovány dva klasické numerické postupy optimalizace.

Cíle bakalářské práce:

Rešerše a základní popis zvolených optimalizačních metod.
Tvorba dvou demo aplikací pro numerickou optimalizaci.
Tvorbu tři demo aplikací pro soft-computing optimalizaci.
Vytvořené aplikace budou obsahovat nápovědu stručně popisující příslušnou metodu.
Vytvoření e-dokumentace k vytvořeným aplikacím.

Seznam odborné literatury:

Bhati, M. A. Practical Optimization Methods with Mathematica Applications. New York: Springer-Verlag, 2000.

Vedoucí bakalářské práce: Ing. Radomil Matoušek, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc. Děkan fakulty

LICENČNÍ SMLOUVA

(na místo tohoto listu vložte vyplněný a podepsaný list formuláře licenčního ujednání)

ABSTRAKT

Cílem této bakalářské práce je základní popis vybraných optimalizačních metod. Tvorba dvou demo aplikací pro numerickou optimalizaci a tří demo aplikací pro soft-computing optimalizaci. Vytvořené aplikace by měly obsahovat nápovědu, stručně popisující příslušnou metodu. Vytvoření e-dokumentace k aplikacím.

ABSTRACT

The aim of this thesis is basic description of the selected optimization methods. Creation of two demo applications for numerical optimization, and three demo applications for soft-computing optimization. Created applications should contain help, briefly describing the appropriate method. Write e-documentation for developed applications.

KLÍČOVÁ SLOVA

Flash, Actionscript, metody, Soft-computing.

KEYWORDS

Flash, Actionscript, Methods, Soft-computing.

PODĚKOVÁNÍ

Děkuji Ing. Radomilu Matouškovi, Ph.D. za odbornou pomoc a konzultace s touto prací.

OBSAH:

Zadání závěrečné práce....3
Licenční smlouva....5
Abstrakt....7
Poděkování....9
1 Úvod....13
2 Optimalizace....15
2.1 Druhy optimalizačních metod....15
2.2.1 Stochastické metody....16
2.2.2 Geometrické metody....16
2.2.3 Evoluční metody....16
2.3 Soft-computing....17
2.3.1 Druhy soft-computingu....17
2.3.2 Fuzzy sytémy....17
2.3.3 Neuronové sítě....17
2.3.4 Evoluční algoritmy....17
3 Adobe Flash CS 3....19
3.1 Historie....19
3.2 Formáty....20
3.3 Výhody Flashe....20
3.4 Nevýhody Flashe....20
3.5 Prostředí programu Flash....21
3.6 Actionscript 3....22
3.7 Ochrana kódu v actionscriptu....22
3.8 Popis vývojového prostředí ActionScript3....23
4 Matematické metody....25
4.1 Newtonova metoda....25
4.1.1 Historie....25
4.1.2 Popis algoritmu....25
4.1.3 Popis aplikace....25
4.1.4 Ukázka aplikace....26
4.2 Metoda půlení intervalu....27
4.2.1 Popis algoritmu....27
4.2.2 Popis aplikace....28
4.2.3 Ukázka aplikace....28
5 Soft-Computingové metody....29
5.1 Metoda náhodného prohledávání....29
5.1.1 Parametry metody....29
5.1.2 Princip algoritmu....29
5.1.3 Popis aplikace....29
5.1.4 Ukázka aplikace....30
5.2 Diferenciální evoluční algoritmus....31
5.2.1 Historie....31
5.2.2 Nastavení vhodných parametrů metody....31
5.2.3 Varianty algoritmu....32
5.2.4 Popis implementace....32
5.2.5 Popis aplikace....33
5.2.6 Ukázka aplikace....33
5.3 Simulované žíhání....34
5.3.1 Volba parametrů....34

5.3.2 Popis algoritmu....35
5.3.3 Popis aplikace....36
5.3.4 Ukázka aplikace....36
ZÁVĚR....37
Seznam použité literatury....39
PŘÍLOHA....41

1 ÚVOD

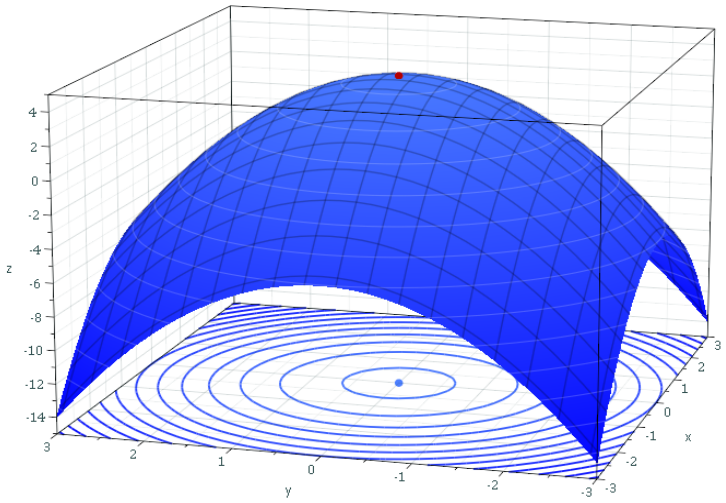
Hlavním úkolem této bakalářské práce je vytvoření interaktivních flashových aplikací, které budou sloužit jako výuková pomůcka. Aplikace budou názorně prezentovat vybrané numerické a soft-computingové metody. Tyto aplikace budou napsány v programu Adobe Flash CS3. K těmto aplikacím bude vypracována e-dokumentace.

První část práce se zabývá optimalizací a soft-computingem. Následující kapitola popisuje vyvojové prostředí programu Adobe Flash CS3 a jeho programovací jazyk Actionscript 3. Využití těchto znalostí nám dovoluje napsání výukových aplikací, které jsou popsány v následujících kapitolách.

Další kapitola popisuje dvě vybrané matematické metody a také výsledné aplikace těchto metod. Poslední kapitolou jsou soft-computingové metody. U těchto metod je taktéž uveden jejich popis a jejich následná aplikace.

2 OPTIMALIZACE

V matematice, optimalizaci, či v matematickém programování, je naším hlavním cílem výběr nejlepších řešení z poskytnuté množiny alternativ. V jednoduchých případech to znamená vyřešení problémů, které se snaží minimalizovat nebo naopak maximalizovat reálnou funkci systematickým výběrem reálných nebo celočíselných proměnných ze předem daného výčtu řešení.



Obr. 1 Maximum paraboloidu[O1].

2.1 Druhy optimalizačních metod

Existuje velké množství optimalizačních metod. A existuje také nepřeberné množství modifikací jednotlivých metod[1].

- Stochastické
- Geometrické – simplexová metoda
- Evoluční – diferenciální evoluce

2.2.1 Stochastické metody

Nejjednodušší stochastické metody využívají pro nalezení extrému, či minima náhody. Princip je takový, že se generuje náhodně dvojice čísel z dané oblasti, tato čísla se pak kontrolují na shodu s funkčními hodnotami optimalizované funkce.

Velká role náhody v této metodě nám znemožňuje cokoliv předpovídat o celkovém průběhu metody. Nejznámější stochastickou metodou je metoda Monte Carlo.[1]

2.2.2 Geometrické metody

Tyto metody aplikují geometrii pro nalezení místa extrému funkce. [1]

2.2.3 Evoluční metody

Patří k nejmladším optimalizačním metodám. Rozvoj těchto metod je spojen hlavně s nástupem počítačů. Využití počítačů je při těchto metodách nevyhnutelné. První metodou těchto algoritmů je metoda simulovaného žíhání. Další algoritmy evolučních metod jsou diferenciální evoluce, tabu search a další[1].

Evoluční metody se hodně inspirují ze zákonitostí přírody. Příkladem je třeba simulované žíhání, kdy se jedná o napodobení metody žíhání z metalurgie.

2.3 Soft-computing

Je poměrně mladý vědní obor. Softcomputingový přístup k výpočtům je podobný, jako je lidské myšlení. Soft-computing je na rozdíl od konvečního computingu tolerantní na nepřesnosti, částečné správnosti a aproximaci. Tyto metody se využívají křešení optimalizačních, biologických nebo matematických problémů. Velkou výhodou těchto soft-computingových metod je velká jednoduchost v matematickém zápise metody. Všeobecně lze říci, že ne každá metoda je vhodná na daný problém, proto tyto metody můžeme vzájemně kombinovat tak, abychom dosáhli co nejlepšího výsledku. Pro úspěšné aplikování těchto metod je nutné znát charakteristické vlastnosti metody, vstupní data a přesnost výsledku metody[2].

2.3.1 Druhy soft-computingu

- Fuzzy sytémy
- Neuronové sítě
- Evoluční algoritmy

2.3.2 Fuzzy sytémy

Ve Fuzzy systémech neplatí boolovská logika. Fuzzy množina využívá slovní popis hodnoty, k určení míry příslušnosti k množině. Fuzzy systémy jsou v dnešní době hojně využívané[2].

2.3.3 Neuronové sítě

Tyto algoritmy se inspirují ve funkci jednotlivých neuronů v lidském mozku. Jedná se tu o obdobu funkcí lidských neuronů, které spolu nezávisle vytvářejí spojení, vzájemně si předávají informační impulzy, jsou rozloženy do vrstev. Aplikováním těchto pravidel do metod dostáváme poměrně silný nástroj na rozpoznávání ručně psaného textu, hlasu[2].

2.3.4 Evoluční algoritmy

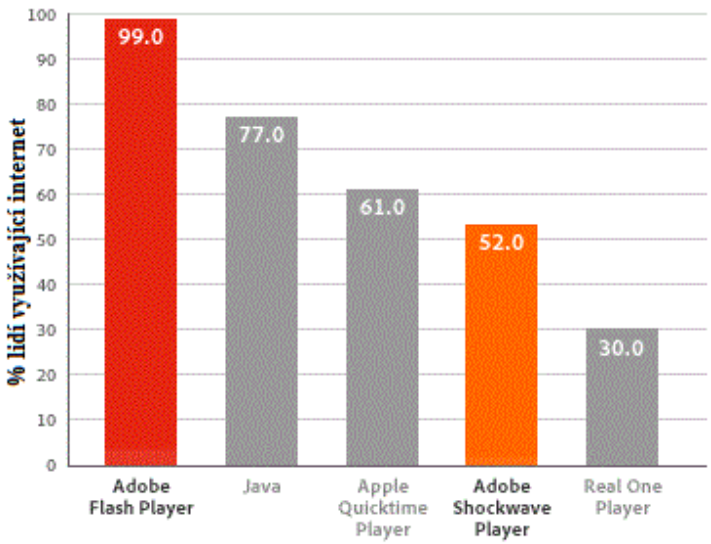
Evoluční algoritmy jsou inspirovány přírodní evolucí. Využívají mutace a křížení mezi první populací a mutantní populací ke generování populace s lepšími výsledky než je ta první.

3 ADOBE FLASH CS 3

Je multimediální platforma, která je nejčastěji využívaná k vytvoření nejrůznějších animací, videí, ale především k vytváření multimediálních interaktivních webových stránek. Další využití se nabízí, jako vhodný nástroj pro tvorbu interaktivních reklamních sdělení, a taktéž jako nástroj pro tvorbu webových her.

Flash pracuje s vektorovou a rastrovou grafikou, se kterou vytváří animace textu, grafiky, a také jednotlivých obrázků. Taktéž poskytuje podporu bidirectionalního streamovaného audia a videa. Dokáže ovládat vstupy z myši, klávesnice, mikrofonu a webové kamery. Programovacím jazyk pro Flashové aplikace je objektově orientovaný ActionScript[3].

Flashové aplikace se dají spustit na rozličných operačních systémech a zařízení, které využívají Adobe Flash Player. Tento přehrávač je běžný doplněk webových prohlížečů, některých mobilních telefonů a dalších elektronických zařízení. Tyto zařízení využívají Flash Lite přehrávač.



Obr. 2 Procentuální rozložení softwarových technologií na internetu[O2].

3.1 Historie

Flash byl představen v roce 1996 firmou Macromedia. Nyní je vyvíjen a distribuován firmou Adobe Systems.

Předchůdce Flashových aplikací byl SmartSketch, grafická aplikace pro kapesní počítače, na kterých běžel PenPoint OS jehož autor byl Jonathan Gay. Když PenPoint propadl na trhu, SmartSketch byl přepsán pro Microsoft Windows, Mac OS. V tom jak se internet stával více a více populární SmartSketch byl znovu uveden jako FutureSplash Animator pro rozličné platformy. V roce 1996 získala FutureSplash firma Macromedia a na základě tohoto produktu vyvinula Flash, který byl pojmenován po zkratkách jeho předchůdce „Future“ a „Splash“[4].

3.2 Formáty

Nejčastějším formátem Flash souborů je SWF formát, který se tradičně nazývá „ShockWave Flash“. Soubory mají obvykle příponu .swf a jsou používány jako forma web stránkového pluginu, který je po té „přehráván“ ve Flash Playeru. Nebo také může být začleněný do samo spouštějícího Projector movie (s .exe příponou na platformě Microsoft Windows) .

FLA formát je zdrojovým souborem Flashe, který obsahuje veškeré informace o výsledné aplikaci. Tento soubor můžeme otevřít jen ve vývojovém prostředí Flash.

3.3 Výhody Flashe

- Výhodou Flashe je, že není nutná instalace programů pro přehrávání u každé webové aplikace. Pouze stačí si do svého prohlížeče nainstalovat Flashový přehrávač a ten po té již přehřeje veškeré webové prezentace.

- Výsledné soubory Flashové prezentace mají malou velikost a to díky převládajícímu využívání vektorové grafiky nad bitmapovou.

- Interaktivita Flash může reagovat na pohyby myši, stlačení kláves a atd.

- Podporuje streaming videa a zvuků.

- Možnost použití jakéhokoliv písma.

- Provázanost s externími soubory.

- Další výhoda je taktéž v jeho velké rozšířenosti.

3.4 Nevýhody Flashe

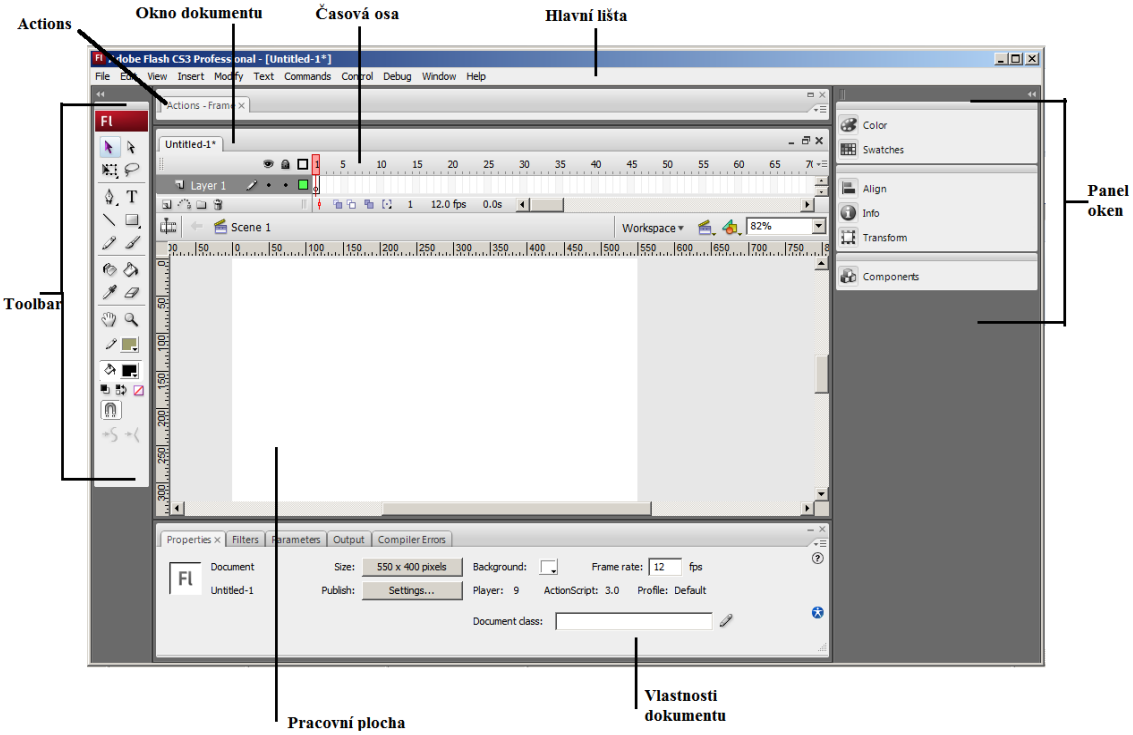
- Hlavními nevýhodami jsou velká náročnost na Hardware.

- Možnost dekompilace SWF souboru na následný zdrojový FLA souboru.

- Nutnost stáhnout celou aplikaci před spuštěním a tedy její možná modifikace.

3.5 Prostředí programu Flash

Bylo navrženo pro snadnou tvorbu aplikací a flashových prezentací s přehledným uživatelsky přívětivým interfacem a podobá se ostatním programům od společnosti Adobe.



Obr. 3 Vývojové prostředí Adobe Flash CS3.

- **Okno dokumentu –** Zobrazuje otevřené Flash dokumenty.
- **Časová osa –** Zobrazuje visuální reprezentaci jednotlivých snímků, vrstev, scény.
- **Hlavní lišta –** Obsahuje obvykle používané příkazy. Např. File, Open, Save a apod.
- **Panel oken –** Tyto okna umožňují editaci jednotlivých prvků, jako jsou například Components – zde je výběr jednotlivých hotových komponent, nebo Library – poskytuje přehled využitých prvků v dokumentu.
- **Vlastnosti dokumentu –** Dává možnost změnit vlastnosti dokumentu, nastavit filtry a parametry, v neposlední řadě možnost prohlédnout si zprávu po kompilaci.
- **Pracovní plocha –** Poskytuje prostor pro vytváření scény.
- **Toolbar** – Obsahuje nástroje pro tvorbu grafiky a jejich vlastností.

- **Actions –** Panel pro psaní ActionScriptu 3.

3.6 Actionscript 3

Je skriptovací jazyk založený na ECMAScript. Tento jazyk se primárně využívá k vývoji webových stránek a programů využívající Adobe Flash a v některých databázových aplikacích (Aplha Five). Jazyk původně vyvinula společnost Macromedia, ale nyní je vlastněn společností Adobe. ActionScript byl původně vyvinut pro kontrolu jednoduchých 2D vektorových animací ve Flashi. V nynější formě jazyk dovoluje vytváření online her a streamování videa a audia v prostředí internetového prohlížeče[4].

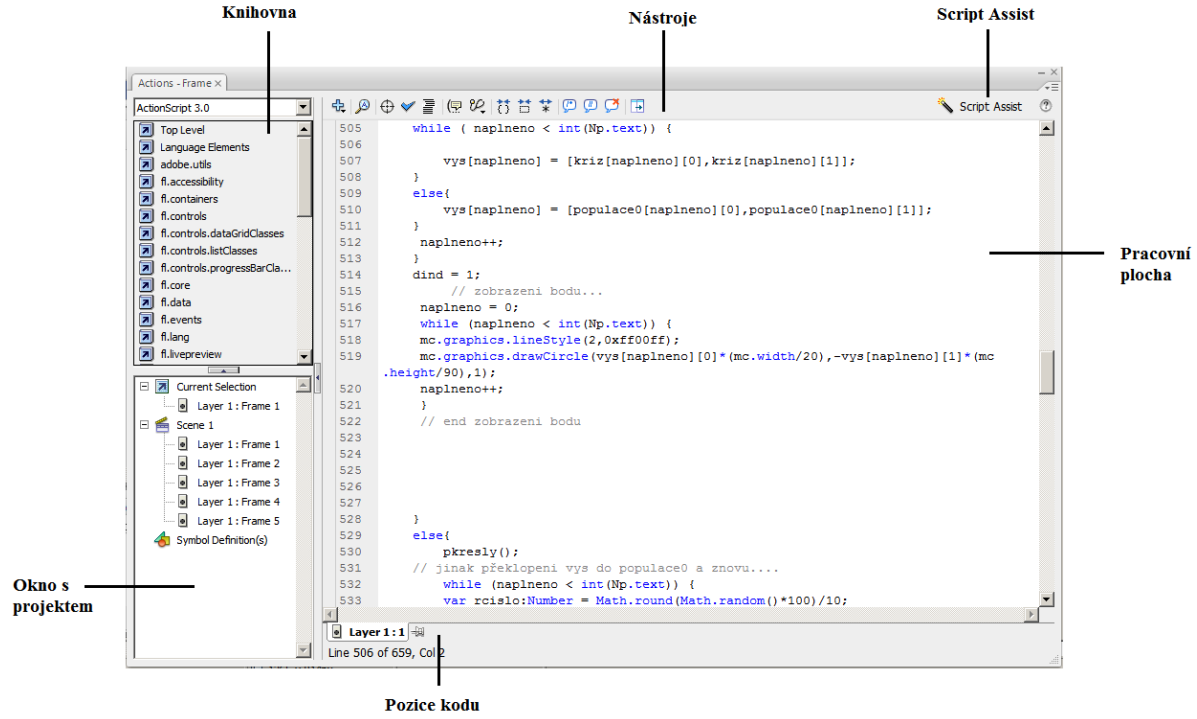
```
package
{
    public class Greeter
    {
    public static function sayHello():String

          {

          var greet:String=”Hello, world!”;

          return greet;

          }

     }

}
```

Obr. 4 Ukázkový kód Action scriptu

3.7 Ochrana kódu v actionscriptu

Protože je zdrojový soubor SWF uložen lokálně na disku. Tak tento soubor může být rekompilován a následně můžeme získat kompletní zdrojový soubor. Tomuto případu Actionscript se snaží zabránit různými způsoby. Avšak výsledky těchto akcí přináší je částečné úspěchy dekompilátory jsou stále schopny získat zdrojové soubory, avšak již takto získaný zdrojový kód je pro člověka hůře čitelný[4].

3.8 Popis vývojového prostředí ActionScript3




```
505     while ( naplneno < int(Np.text)) {
506
507         vys[naplneno] = [kriz[naplneno][0],kriz[naplneno][1]];
508     }
509     else{
510         vys[naplneno] = [populace0[naplneno][0],populace0[naplneno][1]];
511     }
512      naplneno++;
513     }
514     dind = 1;
515          // zobrazeni bodu...
516      naplneno = 0;
517      while (naplneno < int(Np.text)) {
518      mc.graphics.lineStyle(2,0xff00ff);
519      mc.graphics.drawCircle(vys[naplneno][0]*(mc.width/20),-vys[naplneno][1]*(mc
.height/90),1);
520      naplneno++;
521      }
522      // end zobrazeni bodu
523
524
525
526
527
528     }
529     else{
530         pkresly();
531     // jinak překlopení vys do populace0 a znovu....
532         while (naplneno < int(Np.text)) {
533         var rcislo:Number = Math.round(Math.random()*100)/10;
```

Obr. 5 Panel editace Actionscriptu.

- **Knihovna –** Obsahuje ukázky a popis funkcí, syntaxi jazyka apod.
- **Nástroje –** Obsahuje nejčastější příkazy při psaní kódu, jako je například kontrola syntaxe, vyhledávání apod.
- **Script Assist** – Pomůcka pro psaní kódu.
- **Pracovní plocha** – Oblast pro psaní kódu.
- **Pozice kódu** - Aktuální snímek s kódem Actionscriptu.
- **Okno s Projektem** – Slouží k rychlému pohybu mezi snímky v Actionscriptu.

4 MATEMATICKÉ METODY

4.1 Newtonova metoda

Newtonova metoda je v matematice velmi dobře známý algoritmus pro hledání kořenů rovnic v jedné nebo ve více dimenzích. Taktéž se používá k nalezení lokálních minim a maxim funkcí.

4.1.1 Historie

Metoda byla poprvé popsána Isacem Newtonem v *De analysi per aequationes numero terminorum infinitas* napsána v roce 1671. Newton tuto metodu aplikoval pouze na polynomy. Nepočítal aproximace Xn, ale počítal se sekvencemi polynomů a to jenom na konci. Výsledkem byla aproximace pro kořen x. Newton pokládal tuto metodu jako čistě algebraickou. Tuto metodu si Newton pravděpodobně vyvodil z podobné, ale méně praktické metody, vymyšlené Franciscus Vietou. Podstatu Vietovy metody zase můžeme nalézt v práci perského matematika Sharaf al-Din al-Tusi[5].

Newtonova metoda byla poprvé uveřejněna v roce 1685 v díle Johna Wallise *Treatise of Algebra both Historical and Practical.* Roku 1690 Joseph Raphson vydal zjednodušený popis metody v *Analysis aequationum universalis.* Raphson se opět díval na Newtonu metodu čistě jako algebraickou a s výhradním využitím na polynomech, ale oproti Newtovi popsal metodu v podmínkách po sobě jdoucích aproximací xn. Konečně roku 1740 Tomas Simpson popsal Newtonu metodu jako iterační metodu pro řešení hlavních nelineárních rovnic[5].

4.1.2 Popis algoritmu

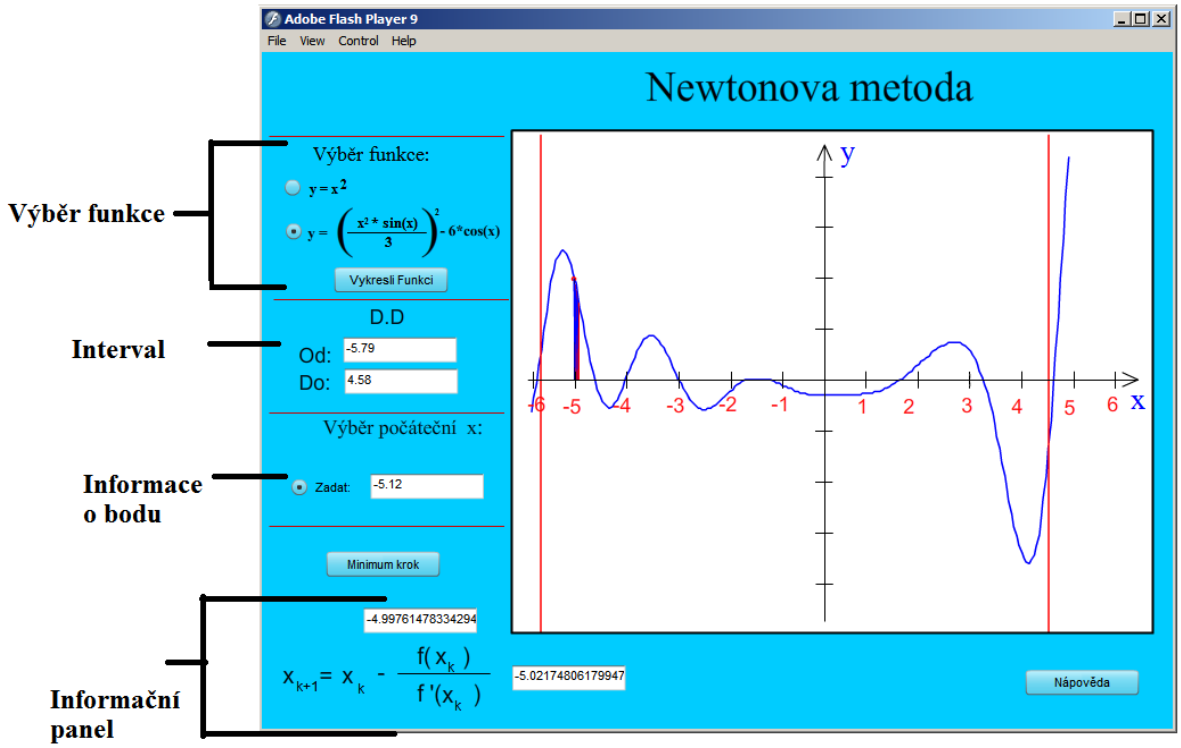
Newtonova metoda vzorec (1), se využívá k nalezení minima funkcí. Nechť je dán počáteční bod x_n. Tento bod se poté dosadí do následujícího vzorce a následně vypočítá druhý bod, který má oproti původnímu bodu menší funkční hodnotu. Takto se dále pokračuje, dokud není nalezeno minimum.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (1)$$

4.1.3 Popis aplikace

Tato aplikace demonstruje grafické znázornění Newtonovy metody na dvou funkcích. K vytvoření této aplikace byl použit Adobe Flash CS3 se skriptovacím jazykem Acionscript 3. Účelem aplikace je demonstrovat studentům tuto metodu v přehledné grafické úpravě. Studenti mohou sledovat nalezení minima funkce po okamžitém proběhnutí metody, i po jednotlivých krocích. Celá metoda je rovněž popsána v nápovědě.

4.1.4 Ukázka aplikace



Obr. 6 Ukázka aplikace Newtonovy metody

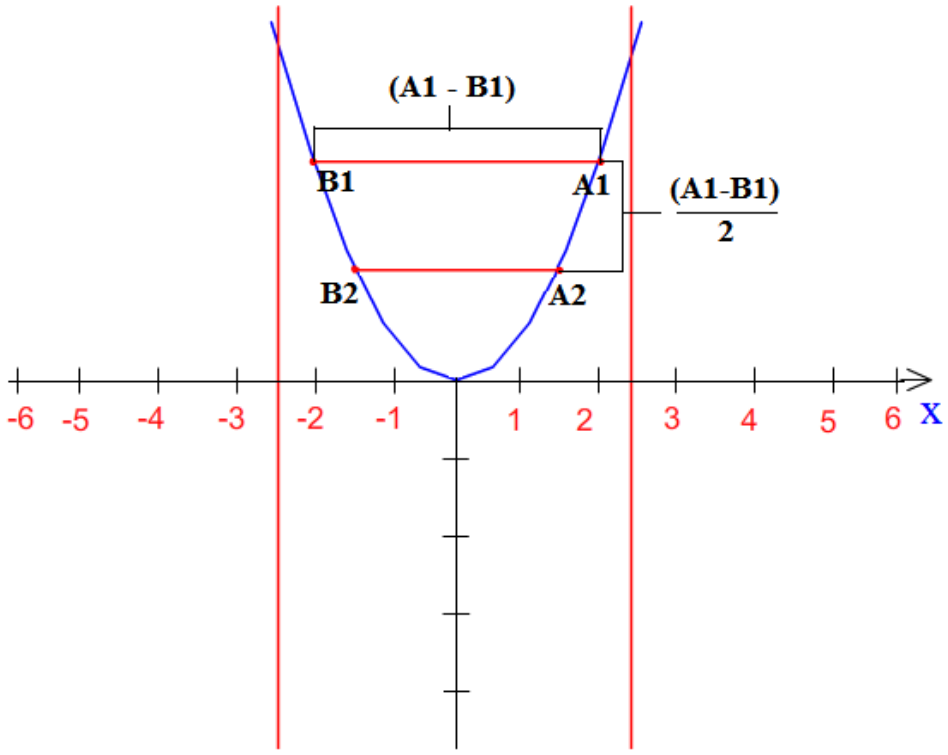
- **Výběr funkce –** umožňuje zvolit funkci
- **Interval** – interval, ve kterém bude metoda aplikována
- **Informace o bodu –** umožňuje zvolení bodu a ukazuje jeho přesnou hodnotu
- **Informační panel** – zobrazuje aktuální hodnoty
- **Nápověda** – nápověda k aplikaci

4.2 Metoda půlení intervalu

Je matematická metoda, která slouží k hledání průsečíku s x osou dané funkce. Její modifikací jsme získali metodu, která je schopná hledat minimum funkce.

4.2.1 Popis algoritmu

Mějme základní funkci x2 a na ní daný bod A1. Pro tento bod se vyhledá opoziční bod na dané funkci a spočítá se jejich vzdálenost. Tato vzdálenost se poté rozpůlí a od bodu A1 se na dané funkci nalezne druhý bod A2, který leží v poloviční vzdálenosti (A1 – B1). Tento postup se pak opakuje do té míry než algoritmus uvázne v minimu funkce. Minimum funkce může být lokální nebo globální, tato metoda je nerozlišuje.

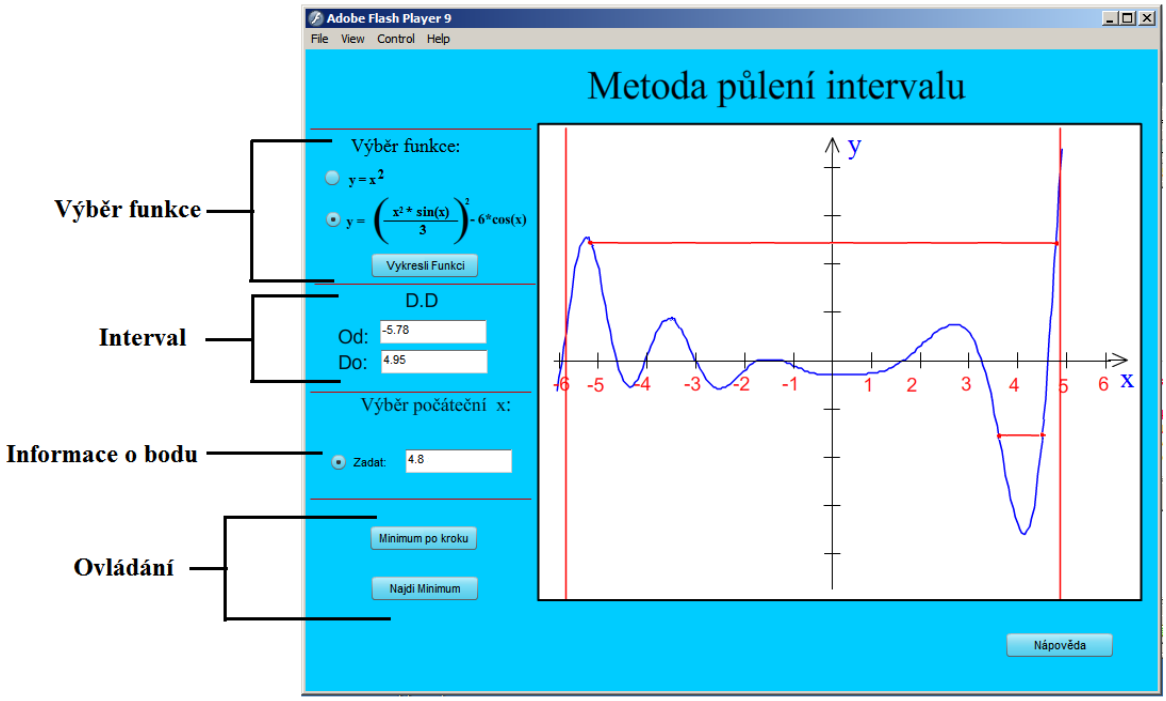


Obr. 7 Grafické znázornění metody půlení intervalu.

4.2.2 Popis aplikace

Tato aplikace demonstruje grafické znázornění metody půlení intervalu na dvou funkcích. K vytvoření této aplikace byl použit Adobe Flash CS3 se skriptovacím jazykem Acionscript 3. Účelem programu je demonstrovat studentům tuto metodu v přehledné grafické úpravě. Studenti mohou sledovat nalezení minima funkce po okamžitém proběhnutí metody, i po jednotlivých krocích.

4.2.3 Ukázka aplikace



Obr. 8 Ukázka aplikacemetody Půlení intervalu.

- **Výběr funkce** – umožňuje výběr funkce
- **Interval** – interval, ve kterém bude funkce pracovat
- **Informace o bodu** – hodnota bodu a možnost výběru bodu
- **Ovládání** – spouští metodu, možnost výběru vykreslování (najednou, po krocích)
- **Nápověda** – nápověda k aplikace

5 SOFT-COMPUTINGOVÉ METODY

5.1 Metoda náhodného prohledávání

Je pravděpodobně nejjednodušší metoda stochastické optimalizace, a ve vhodném nastavení může být velmi efektivní. Metoda náhodného prohledávání využívá náhodných prvků a pravděpodobnosti. Předností této metody je jednoduché naprogramování. Nevýhoda metody je, že snadno uvázne v lokálním minimu. Závisí také na velké míře pravděpodobnosti, kdy se náhodně generovaný bod musí střetnout s bodem funkce.

5.1.1 Parametry metody

- Počet bodů
- Poloměr okolí
- Způsob generování okolí

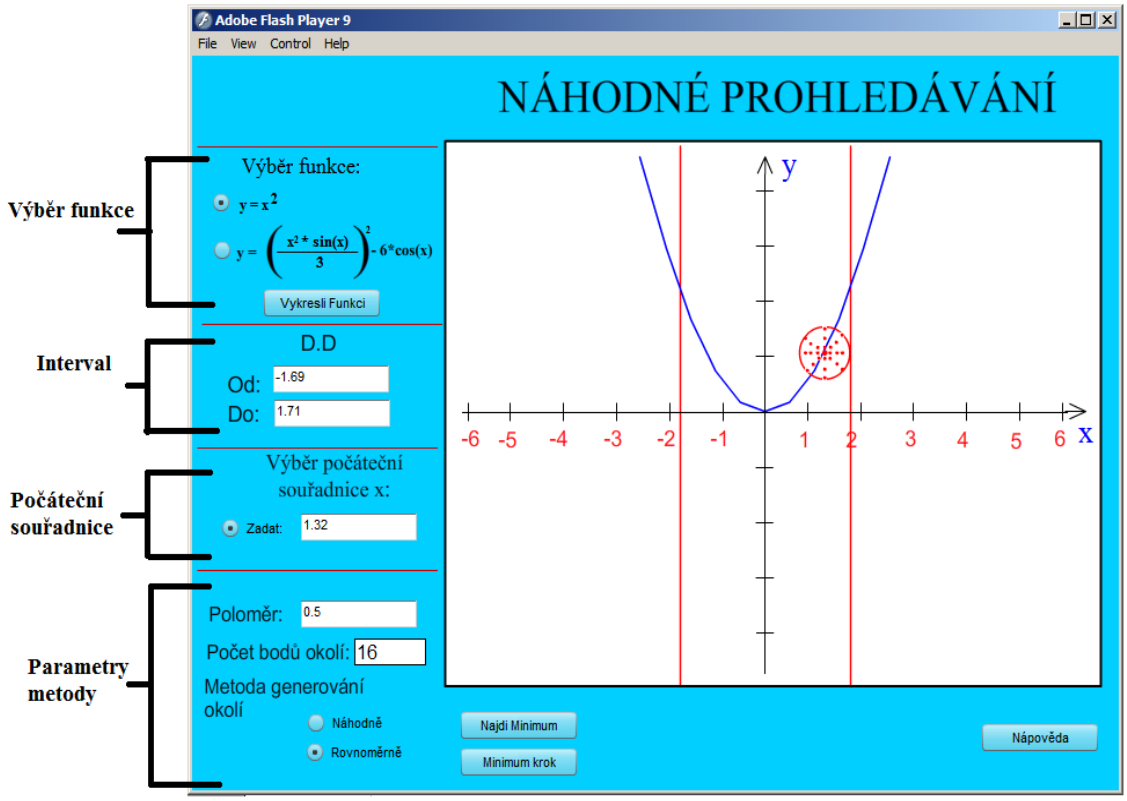
5.1.2 Princip algoritmu

Algoritmus pracuje tak, že v okolí daného bodu, které je dané proměnnou (Poloměr okolí) se vygeneruje určitý počet bodů. Způsob generování bodů v okolí a počet bodů jsou opět dány proměnnými. Tyto náhodné body se následně testují, zda se shodují s nějakou funkční hodnotou optimalizované funkce. V případě shody se tento bod označí, jako výhodnější. Algoritmus se následně přesune do tohoto bodu a následně spustí celou proceduru znovu

5.1.3 Popis aplikace

Tato aplikace demonstruje grafické znázornění metody půlení intervalu na dvou funkcích. K vytvoření této aplikace byl použit Adobe Flash CS3 se skriptovacím jazykem Acionscript 3. Účelem programu je demonstrovat studentům tuto metodu v přehledné grafické úpravě. Studenti mohou sledovat nalezení minima funkce po okamžitém proběhnutí metody, i po jednotlivých krocích. Celé metoda je objasněna v nápovědě aplikace.

5.1.4 Ukázka aplikace



Obr. 9 Ukázka aplikace metody Náhodného prohledávání.

- **Výběr funkce** – umožňuje výběr funkce
- **Interval –** určuje interval, ve kterém bude metoda pracovat
- **Počáteční souřadnice** – určuje počáteční bod metody
- **Parametry metody** – nastavují se zde parametry metody
- **Nápověda** - nápověda k metodě náhodného prohledávání

5.2 Diferenciální evoluční algoritmus

Diferenciální evoluční algoritmus je metoda numerické optimalizace bez nutné znalosti gradientu řešeného problému, který má být optimalizován. Algoritmus optimalizuje problém obsluhou populace kandidátních řešení a vytvořením nových kandidátních řešení, které jsou kombinací existujících řešení podle jednoduché formulace vektorových křížení a mutací. A následné uchovávání kandidátních řešení s nejlepší hodnotou nebo výkonem v optimalizačním problému. Tento druh optimalizačního problému si můžeme představit jako černou skříňku, která pouze poskytuje měřítko dané kvality a kandidátního řešení a to bez předchozí nutnosti znalosti gradientu[6].

5.2.1 Historie

Diferenciální algoritmus vyvinul Kenneth Price, když se pokoušel vyřešit „Chebyshev polynomial fitting problem“, který mu byl zadán Rainerem Stornem. Tento problém vyřešil modifikováním genetického žíhání, které bylo původně vymyšleno Pricem k užití plovoucí čárky místo bitově-řetězcového kódování a aritmeticko-vektorových operací místo logických operací[7].

Komunita vyvíjející tento algoritmus se vytvořila mezi lety 1994 – 1996 a počet vědců pracující na zdokonalení tohoto algoritmu se stále zvětšuje. Velkým přáním Price a Storna bylo, aby jejich algoritmus byl dále rozvíjen vědci na celém světě a aby byl diferenciální evoluční algoritmus užitečný více lidem v jejich každodenní práci. Diferenciální algoritmus nebyl dosud patentován[7].

Diferenciální evoluční algoritmus byl prvně demonstrován na První mezinárodní soutěži evoluční optimalizace v dubnu 1996, který byl pořádán Mezinárodní konferenci evolučních výpočtů (IEEE). Algoritmus se umístil na třetím místě v navržených testech[7].

5.2.2 Nastavení vhodných parametrů metody

K nastavení vhodných parametrů této metody (F, CR, NP,…) bylo vynaloženo mnoho úsilí v oblasti výzkumu. Jednoduchý a efektivní způsob, jak nastavit parametry byl představen Pedersnem, který experimentoval s diferenciálními optimalizačními problémy a jejich nastaveními. Tato technika se jmenuje meta-optimalizace[1].

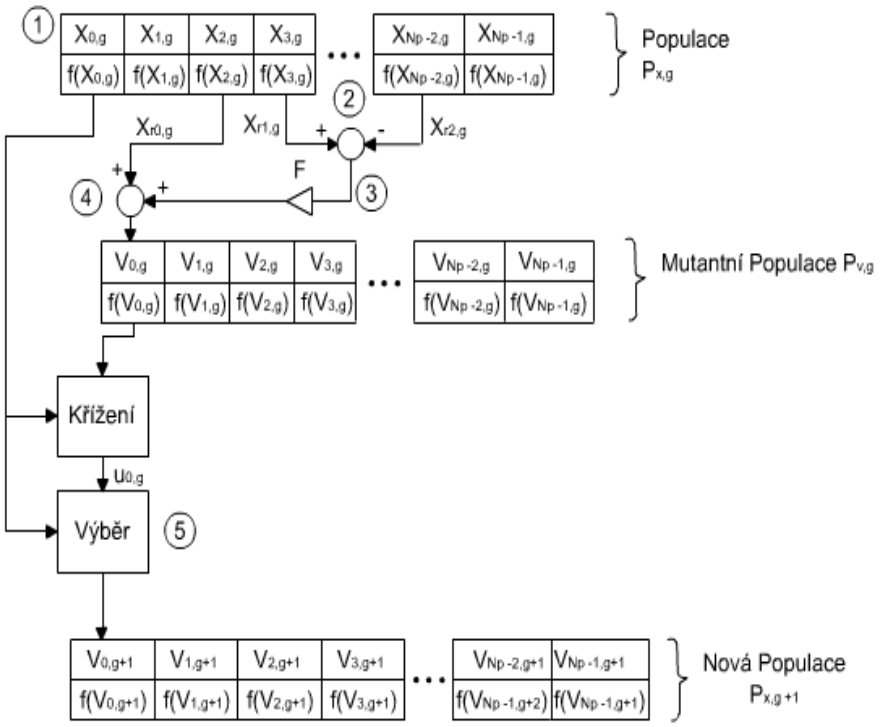
- Np NP >3 - počet jedinců, velikost populace
- F F <0,2> - mutační konstanta
- CR CR <0,10> - pravděpodobnost křížení

5.2.3 Varianty algoritmu

Z důvodu neustálého vývoje a zlepšování optimalizačního výkonu existuje algoritmus v mnoha variantách. Nynějším populárním výzkumným trendem je zavedení schémat pro rušivé a adaptující parametry, které probíhají během optimalizace[7].

5.2.4 Popis implementace

Diferenciální evoluční algoritmus je velmi populačně založený, stochasticky funkcionální minimalizátor, který je velmi výkonný. Diferenciální evoluční algoritmus má potenciál stát se nejlepším geneticky orientovaným algoritmem[7].



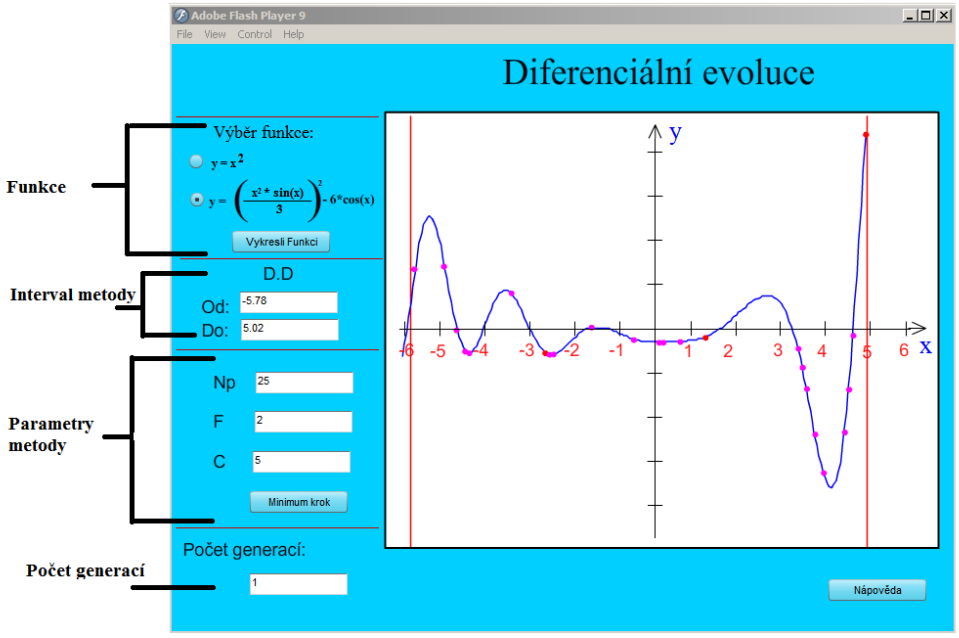
Obr. 10 Blokové schéma algoritmu Diferenciální evoluce.

1) Výběr počáteční populace řešení
2) Náhodná volba dvou populačních členů
3) Výpočet váženého diferenčního vektoru
4) Přidání k základnímu vektoru
5) Výběr mezi řešením mutantní populace a řešením počáteční populace

5.2.5 Popis aplikace

Tato aplikace demonstruje grafické znázornění metody diferenciální evoluce na dvou funkcích. K vytvoření této aplikace byl použit Adobe Flash CS3 se skriptovacím jazykem Acionscript 3. Účelem programu je demonstrovat studentům tuto metodu v přehledné grafické úpravě.

5.2.6 Ukázka aplikace



Obr. 11 Ukázka aplikace metody Diferenciálního evolučního algoritmu.

- **Funkce –** slouží k výběru funkce
- **Interval metody** - určuje interval, ve kterém bude metoda pracovat
- **Parametry metody** nastavení jednotlivých parametrů metody
- **Počet generací –** zobrazuje aktuální generaci řešení metody
- **Nápověda –** nápověda s podrobným popisem metody

5.3 Simulované žíhání

Inspirace a jméno pochází z žíhání oceli v metalurgii, kde tato technika spočívá v zahřívání a kontrolovaném zchlazování materiálu. V materiálu se zvětšují zrna krystalů a redukují se tak jejich defekty. Atomy se teplem uvolní z jejich počátečních energetických hodnot a následně náhodně putují do stavů s vyšší energií. Pomalé ochlazení poskytuje vyšší šanci najít takovou konfiguraci s nižší interní energií než je počáteční energie.

Napodobením těchto fyzikálních procesů do algoritmu simulovaného žíhání nám poskytuje metodu prohledávání stavového prostoru, která eliminuje uvíznutí v lokálním minimu tím, že za určitých okolností je schopna přijmout i horší řešení než řešení optimální[8].

Metodu nezávisle na sobě popsali Scott Kirkpatrick, C. Daniel Gelatt, Mario P.Vecchi v roce 1983 a Vlado Černý v roce 1985. Metoda je adaptací Metropolisova-Hastingsova algoritmu a metody Monte Carla ke generování vzorkových stavů termodynamického systému objeveného N. Metropolisem v roce 1953 [8].

5.3.1 Volba parametrů

Aby metoda mohla správně řešit daný problém, musíme vhodně zvolit následující parametry:

- Kmax
- Tmax
- Tmin
- Alfa

Výběr správných parametrů má výrazný vliv na efektivitu metody. Bohužel nelze aplikovat univerzální nastavení těchto parametrů pro všechny problémy [8].

Existuje několik způsobů, jak zvolit vhodné parametry podle [8]:

- Diametrální prohledávání grafu
- Efektivní generování kandidátů
- Vyhýbání bariér
- Plánované ochlazování

5.3.2 Popis algoritmu

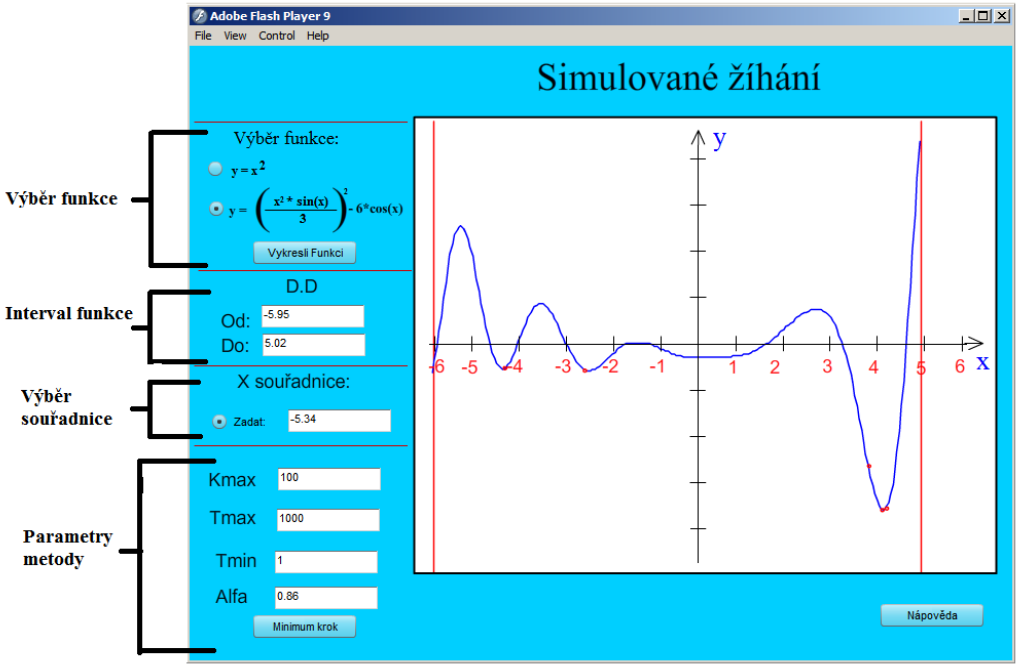
Zápis celé metody v pseudokódu:

```
Procedure    Simulovane_Zihani(Tmin, Tmax, Kmax, Alfa) {

            Xini = námi zvolený bod;

         While (Tmax > Tmin) { // Podmínka ochlazování

//Metropolisovo kriterium

Xini=Metropolis_Algoritm(xini,Kmax,Tmax);

Tmax = alfa*Tmax; // snižování (ochlazování teploty)

}

Xmin = xini; // výsledek

}

Procedure Metropolis_Algoritm(xini, Kmax, Tmax){

            K = 0;

            X = xini;

            While( K< Kmax) { //  Podmínka kristalizace

             K ++;

  X' = Opert(x);// Operátor pertuberance (generátor poruchy)

            Pr = min(1,exp(-(f(X')-f(X))/Tmax);

           if random_number <Pr {

           X = X'

     }

}

return X;

}
```

5.3.3 Popis aplikace

Tato aplikace demonstruje grafické znázornění metody simulovaného žíhání na dvou funkcích. K vytvoření této aplikace byl použit Adobe Flash CS3 se skriptovacím jazykem Acionscript 3. Účelem programu je demonstrovat studentům tuto metodu v přehledné grafické úpravě. Studenti mohou sledovat nalezení minima funkce po jednotlivých krocích.

5.3.4 Ukázka aplikace



Obr. 12 Ukázka aplikace Simulovaného žíhání.

- **Výběr funkce –** slouží k výběru funkce
- **Interval metody** - určuje interval, ve kterém bude metoda pracovat
- **Výběr souřadnice** – slouží k nastavení počáteční souřadnice
- **Parametry metody** nastavení jednotlivých parametrů metody
- **Nápověda –** nápověda s podrobným popisem metody

ZÁVĚR

V této bakalářské práci bylo vytvořeno několik interaktivních aplikací, které se zabývají vybranými matematickými a soft-computingovými metodami. Prezentace těchto metod pak následně poslouží jako interaktivní výukové aplikace, které lze využít k názorné demonstraci vybraných metod ve výuce.

Matematické metody, které byly vybrány:

- Newtonova metoda
- Metoda půlení intervalu

Soft-computingové metody, které byly vybrány:

- Náhodné prohledávání
- Diferenciální evoluce
- Simulované žíhání

Všechny aplikace názorně demonstrují vybrané metody na jednotlivých funkcích. Programy byly psány v Adobe Flash CS3 a jako programovací jazyk byl použit Actionscript 3. Tyto aplikace jsou popsány v přiložené e-dokumentaci, která se nachází jako příloha na CD u bakalářské práce.

SEZNAM POUŽITÉ LITERATURY

[1] Ondřej Suchomel, ing. Tomaš Hyhlik. *Optimalizační metody v CFD – diferenciální evoluce.* [PDF dokument]. Fakulta strojní ČVUT, - Odbor mechaniky tekutin a termodynamiky [cit. 4.5.2010]. Dostupný z: <http://www.suchomel.org/files/Optimalizace-v-CFD.pdf>

[2] Y. Jin. Short Definition of Soft Computing [online]. [cit. 4.5.2010]. Dostupné z: <http://www.soft-computing.de/def.html>

[3] Wikipedia. Adobe Flash [online]. [cit. 7.5.2010]. Dostupné z: <http://en.wikipedia.org/wiki/Adobe_Flash>

[4] Wikipedia. ActionScript[online]. [cit. 7.5.2010]. Dostupné z: <http://en.wikipedia.org/wiki/ActionScript>

[5] Wikipedia. Newton's method[online]. [cit. 10.5.2010]. Dostupné z: <http://en.wikipedia.org/wiki/Newton's_method>

[6] Wikipedia. Differential evolution[online]. [cit. 12.5.2010]. Dostupné z: <http://en.wikipedia.org/wiki/Differential_evolution>

[7] Rainer Storn. Differential evolution[online]. [cit. 12.5.2010]. Dostupné z: <http://www.icsi.berkeley.edu/~storn/code.html>

[8] Wikipedia. [online]. Simulated annealing [cit. 12.5.2010]. Dostupné z: <http://en.wikipedia.org/wiki/Simulated_annealing>

[O1] Wikipedie.[online].The maximum of a paraboloid. Dostupné z: <http://en.wikipedia.org/wiki/Optimization_(mathematics)>

[O2] Adobe Flash.[online].Percentage of internet enabled PC's. Dostupné z: <http://www.adobe.com/products/player_census/flashplayer/>

PŘÍLOHA

CD s elektronickou verzí bakalářské práce, e-dokumentace.