

Knižnica testovacích funkcií pre optimalizačné algoritmy v C/C++ jazyku

Library of Benchmark Functions for Optimization Algorithms in
C/C++ Language

Michal Vašek

Bakalárska práca
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal VAŠEK**
Osobní číslo: **A10985**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **kombinovaná**

Téma práce: **Knihovna testovacích funkcí pro optimalizační algoritmy v C/C++ jazyce**

Zásady pro vypracování:

1. Vyhledejte testovací funkce pro optimalizační úlohy včetně vzorců, prohledávaného intervalu a známých hodnot extrémů.
2. Porovnejte zápisy vzorců a hodnot z různých zdrojů.
3. Navrhněte vhodnou strukturu knihovny testovacích funkcí z pohledu vkládání parametrů, intervalů, nových funkcí.
4. Vytvořte knihovnu testovacích funkcí v C/C++ jazyce.
5. Ověřte snadnou připojitelnost k již vytvořeným evolučním algoritmům ve stejném prostředí a vhodně otestujte funkčnost a použitelnost knihovny s vybraným evolučním algoritmem.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN – technická literatura, 2009, 534 s. ISBN 978-80-7300-218-3.
2. ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. 1. vyd. Praha: BEN – technická literatura, 2002, 189 s. ISBN 8073000695.
3. ZELINKA, Ivan. Evolutionary algorithms and chaotic systems. Berlin: Springer, 2010, xxiv, 521 s. ISBN 978-3-642-10706-1.
4. OPLATKOVÁ, Zuzana. Metaevolution: synthesis of optimization algorithms by means of symbolic regression and evolutionary algorithms. Saarbrücken: Lambert Academic Publishing, c2009, 157 s. ISBN 978-3-8383-1808-0.
5. LAM, Hak-Keung, S LING a Hung T NGUYEN. Computational intelligence and its applications: evolutionary computation, fuzzy logic, neural network and support vector machine techniques. London: Imperial College Press, c2012, x, 307 s. ISBN 978-1-84816-691-2.

Vedoucí bakalářské práce:

Ing. Zuzana Komínková Oplatková, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

24. února 2013

Termín odevzdání bakalářské práce:

14. června 2013

Ve Zlíně dne 24. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Bakalárska práca sa zaoberá testovacími funkciami pre evolučné algoritmy. Teoretická časť obsahuje vybrané testovacie funkcie a porovnáva zápisy vzorcov a hodnôt získaných z rôznych zdrojov. Praktická časť implementuje knižnicu testovacích funkcií v prostredí C/C++, popisuje jej použitie a prepojitelnosť s už existujúcimi evolučnými algoritmami naprogramovanými v rovnakom prostredí.

Kľúčové slová: testovacie funkcie, evolučný algoritmus

ABSTRACT

Bachelor's thesis deals with test functions for evolutionary algorithms. The theoretical part contains selected test functions and compares them formulas and values obtained from various sources. The practical part implements the library of test functions in C/C++ language, describes its use and connectivity with existing evolutionary algorithms programmed in the same environment.

Keywords: testing functions, evolutionary algorithm

Na tomto mieste by som sa chcel poďakovať Ing. Zuzane Komínkovej Oplatkovej, Ph.D., pod ktorej vedením mi bolo umožnené vypracovanie mojej bakalárskej práce, za jej ochotu a trpezlivosť.

Prehlasujem, že

- beriem na vedomie, že odovzdaním bakalárskej práce súhlasím so zverejnením svojej práce podľa zákona č. 111/1998 Zb. o vysokých školách a o zmene a doplnení ďalších zákonov (zákon o vysokých školách), v znení neskorších právnych predpisov, bez ohľadu na výsledok obhajoby;
- beriem na vedomie, že bakalárska práca bude uložená v elektronickej podobe v univerzitnom informačnom systéme dostupná k prezenčnému nahliadnutiu, že jeden výtlačok bakalárskej práce bude uložený v príručnej knižnici Fakulty aplikovanej informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtlačok bude uložený u vedúceho práce;
- bol/a som zoznámený/á s tým, že na moju bakalársku prácu sa plne vzťahuje zákon č. 121/2000 Zb. o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon) v znení neskorších právnych predpisov, zejms. § 35 odst. 3;
- beriem na vedomie, že podľa § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzatvorenie licenčnej zmluvy o užití školského diela v rozsahu § 12 hl. 4 autorského zákona;
- beriem na vedomie, že podľa § 60 odst. 2 a 3 autorského zákona môžem užiť svoje dielo – bakalársku prácu alebo poskytnúť licenciú k jej využitiu len s predchádzajúcim písomným súhlasom Univerzity Tomáše Bati ve Zlíně, ktorá je oprávnená v takomto prípade odo mňa požadovať primeraný príspevok na úhradu nákladov, ktoré boli Univerzitou Tomáše Bati ve Zlíně na vytvorenie diela vynaložené (až do ich skutočnej výšky);
- beriem na vedomie, že pokiaľ bol na vypracovanie bakalárskej práce využitý software poskytnutý Univerzitou Tomáše Bati ve Zlíně alebo inými subjektmi iba na študijné a výskumné účely (teda iba k nekomerčnému využitiu), nie je možné výsledky bakalárskej práce využiť na komerčné účely;
- beriem na vedomie, že pokiaľ je výstupom bakalárskej práce akýkoľvek softwarový produkt, považujú sa za súčasť práce taktiež aj zdrojové kódy, poprípade súbory, z ktorých sa projekt skladá. Neodovzdanie týchto súčastí môže byť dôvodom k neobhájeniu práce.

Prehlasujem,

- že som na bakalárskej práci pracoval samostatne a použitú literatúru som citoval. V prípade publikácie výsledkov budem uvedený ako spoluautor.
- že odovzdaná verzia bakalárskej práce a verzia elektronická nahraná do IS/STAG sú totožné.

Ve Zlíně

.....
podpis diplomanta

ZOZNAM OBRÁZKOV	56
ZOZNAM TABULIEK	57
ZOZNAM PRÍLOH.....	58

ÚVOD

Princíp genetických algoritmov je inšpirovaný Darwinom. Evolučné Algoritmy sú postavené na teórii, že najlepší jedinci prezívajú a je im umožnené rozmnožovanie, tí slabší zahynú a ich gény tak nebudú zdedené nasledujúcimi generáciami. V každej generácii je vytvorená nová skupina jedincov (riešení) pomocou techník, ktoré napodobňujú evolučný proces (selekcia, kríženie, mutácia). Riešenia každej generácie majú podľa určitej funkcie pridelenú hodnotu. Takáto hodnota, ktorá sa nazýva fitness, určuje nakoľko riešenie vyhovuje zadaniu. Neúspešné riešenia sú potom zničené a z najlepších sa tvorí nová generácia. Tento proces sa opakuje určitý počet generácii alebo pokiaľ sa nájde dostatočne dobre riešenie problému.

Pri tvorbe evolučných algoritmov potrebujeme otestovať ich efektivitu a vhodnosť riešiť rôzne optimalizačné úlohy. Právě množina testovacích funkcií z rôznymi vlastnosťami poskytuje jeden zo spôsobov porovnania výkonnosti jednotlivých evolučných algoritmov.

V dnešnej dobe je možné nájsť veľa testovacích funkcií s rôznymi vlastnosťami. Medzi obľúbené funkcie patria napríklad Schwefelova, Rastriginova, Ackleyho. V tejto práci sú zobrazené testovacie funkcie v priestore, ktorý je daný tromi rozmermi, no v praxi sú funkcie rozšírené na n -rozmerný priestor a hľadanie optima napríklad pre dimenziu 30 sa stáva o to náročnejšie.

I. TEORETICKÁ ČASŤ

1 EVOLUČNÉ ALGORITMY

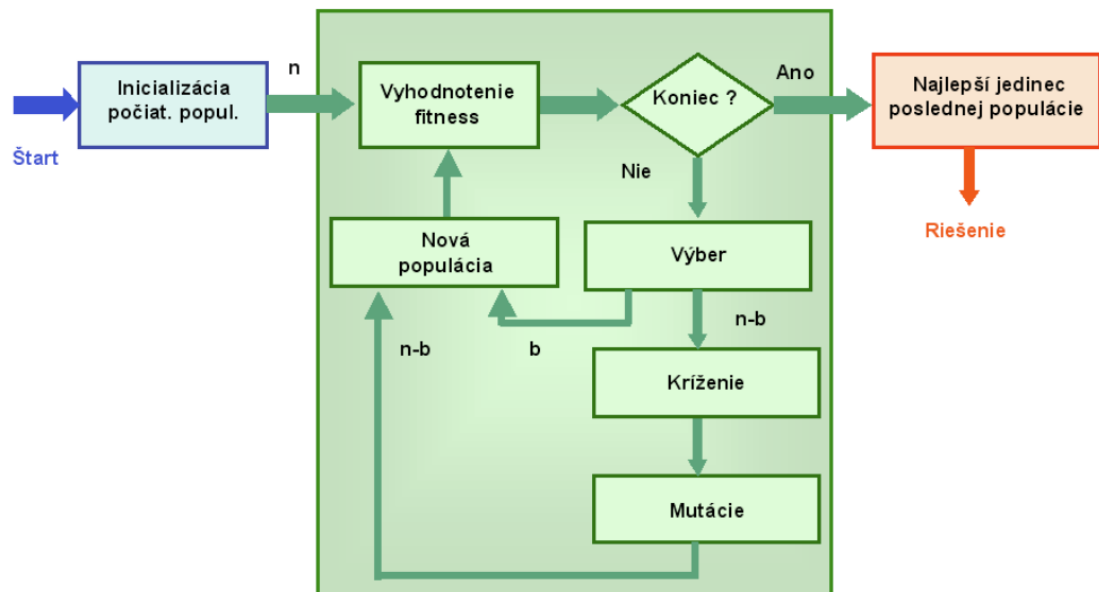
Inšpirovaní evolučným procesom začali matematici, biológovia, neskôr aj informatici čoraz častejšie nadošovať otázku, či by sa nedal tento „mechanizmus“ napodobňovať a využívať pri riešení vo všeobecnosti nebiologických problémov. Dvere týmto snahám, ako tomu bolo aj v mnohých iných oblastiach, otvorila až výkonná výpočtová technika. Bez nej by nebolo možné simulovať tisíce alebo milióny evolučných cyklov, ktoré prebiehali v prírode. V druhej polovici dvadsiateho storočia boli podľa vzoru prirodzenej evolúcie na rôznych pracoviskách vo svete pri riešení odlišných problémov vytvorené viaceré, ale v istých črtách podobné prístupy. V Nemecku v polovici šesťdesiatych rokov boli pri optimalizácii konštrukčných úloh I.Rechenbergom a H.P.Schwefelom vyvíjané tzv. „evolučné stratégie“. Lawrence Fogel v tom období v USA pri modelovaní a návrhu automatov vyvíjal techniku pod názvom „evolučné programovanie“. Za vznik „genetických algoritmov“, ktoré majú dnes široké použitie pri optimalizácii sú považované práce skupiny pod vedením Johna Hollanda z Michiganskej University v USA v 70. rokoch dvadsiateho storočia. Historicky mladšie „genetické programovanie“ je evolučný prístup určený najmä na automatizovaný vývoj a optimalizáciu programov, funkcionálnu regresiu alebo strojové učenie. Za jeho autora je považovaný John Koza (USA) na prelome 80. a 90. rokov. Všetky takéto smery sa dnes zvyknú zastrešovať pojmom “evolučné algoritmy”. [10]

1.1 Genetické algoritmy

Jedným z najvýznamnejších predstaviteľov evolučných algoritmov s veľmi širokým uplatnením sú genetické algoritmy. GA sú univerzálnym stochastickým prehľadávacím alebo optimalizačným prístupom, ktorý je v ohraničenom priestore prípustných riešení daného problému schopný nájsť alebo sa aspoň priblížiť ku globálnemu optimu. Uplatňuje sa pritom z prírody vypozerovaný princíp prežitia najsilnejších resp. najprispôsobivejších jedincov a nevyhnutnosť zániku najslabších, neživotaschopných resp. neprispôsobivých. [10]

Keby sme mali princíp GA vyjadriť dvomi vetami, mohli by sme povedať, že pracujú s populáciou viacerých riešení, kde pôsobením genetických operácií “kríženia” a “mutácie” priebežne vznikajú úspešnejšie a menej úspešné riešenia. Tie úspešné majú väčšiu šancu

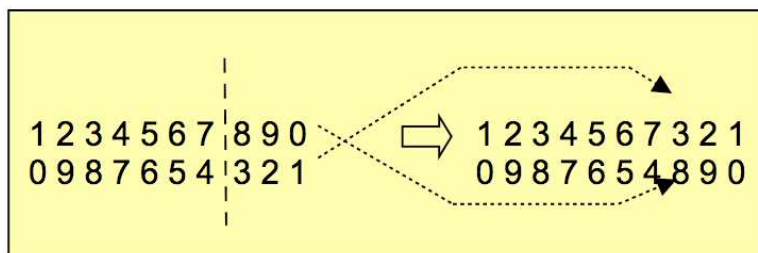
byť vybrané, postupovať v evolúcii ďalej a odovzdávať svoje vlastnosti ďalším generáciám, na úkor menej úspešných jedincov, ktoré zanikajú. [10]



[10]

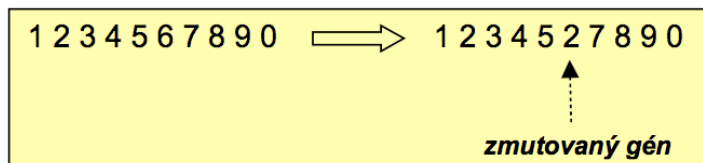
Obr. 1. Bloková schéma genetického algoritmu

Algoritmus pracuje so skupinou potenciálnych riešení – s tzv. “populáciou”. Každé potenciálne riešenie je pritom reprezentované usporiadanou množinou parametrov alebo hodnôt, ktoré charakterizujú jeho vlastnosti. Prvky tejto množiny sa nazývajú “gény” a môžu byť binárneho, celočíselného, reálnečíselného, symbolového alebo kombinovaného typu, v závislosti od charakteru daného problému. Sú usporiadané (zakódované) do postupnosti, ktorá sa nazýva “reťazec” alebo “chromozóm”. Počiatočná populácia reťazcov v prvom výpočtovom cykle (tzv. “generácii”) sa získa spravidla náhodným vygenerovaním ich génov v rámci uvažovaných ohraničení. Pre každé riešenie, ktoré sa dekoduje z reťazca do existujúceho počítačového modelu sa vyčíslí výpočtom, počítačovou simuláciou, atď. hodnota účelovej funkcie – tzv. “fitness”. Fitness je vlastne miera vhodnosti alebo úspešnosti daného reťazca alebo “jedinca”. Všetky jedince populácie sa navzájom porovnávajú a potom sa vyberie skupina jedincov, ktoré sa nezmenené dostanú do novej populácie. Tiež sa vyberie druhá skupina jedincov, ktorá je určená na inováciu. V tejto skupine sa vytvoria náhodné páry reťazcov, s ktorými sa uskutoční genetická operácia “križenie”. Potom sa na tejto skupine realizuje ešte “mutácia”. [10]



[10]

Obr. 2. Príklad jednobodového kríženia dvoch reťazcov



[10]

Obr. 3. Príklad mutácie celočíselného reťazca

Takto zmodifikované jedince potom dokompletujú novú populáciu, ktorá sa stane objektom rovnakého postupu v ďalšej generácii. Pritom je dôležité, že pri výbere do oboch skupín majú najväčšiu pravdepodobnosť “prežitia” najúspešnejšie jedince, ale istú šancu majú aj menej úspešné jedince. Ak sa uvedený postup opakuje mnohokrát – počas mnohých generácií, riešenie konverguje k najlepšiemu riešeniu – ku globálnemu optimu. Pod pojmom “mnohokrát” si môžeme predstaviť číslo 100, 1000 alebo aj 1 milión. Závisí to od povahy a zložitosti riešeného problému. Beh algoritmu sa môže ukončiť po dosiahnutí požadovaného resp. prijateľného riešenia, alebo po ukončení určeného počtu generácií. [10]

Operácia kríženia náhodne skombinuje gény dvoch rodičov do jedného alebo viacerých potomkov. Pri bežnom spôsobe kríženia sa dva rodičovské reťazce rozdelia na jednom alebo viacerých náhodných miestach (oba reťazce na rovnakých) a potomkovia získajú striedavo každú druhú časť takto oddelených podreťazcov od každého z rodičov (Obr. 2). Operácia mutácie náhodne zmení náhodne zvolené gény náhodne vybraných jedincov (Obr. 3). Čitateľ nemôže prehliadnuť, že základnou črtou týchto algoritmov je náhodnosť. Poznamenajme ešte, že existujú viaceré typy kríženia aj mutácie. Spôsobov výberu jedincov do nových populácií je tiež niekoľko druhov, líšia sa mierou preferovania najúspešnejších jedincov oproti náhodne vybraným jedincom. Voľba genetických operácií môže byť ovplyvnená typom riešeného problému. [10]

Použitie EVT môže byť silným nástrojom pri optimalizácii. Známe je využitie GA pri optimalizácii motorov Boeingu 777, kedy sa zdanlivo malou konštrukčnou úpravou získala

na dané pomery mimoriadne významná úspora paliva až 2,5 %. Toto predstavuje pri jednom lietadle za rok úsporu 2 miliónov dolárov. [10]

2 TESTOVANIE EVOLUČNÝCH ALGORITMOV

Pre testovanie EA sa používajú dva rozdielne prístupy. V prvom obvykle vychádzame z už existujúcich príkladov, ktoré boli riešené inými algoritmami. Výsledky pravé testovaného EA potom porovnávame s výsledkami už existujúcimi. Druhý spôsob spočíva v tom, že použijeme množinu testovacích funkcií obsahujúcich funkcie s rôznymi vlastnosťami ako je nelinearita, rôzna patológia typu rovina okolo extrému apod. [1]

Vzhľadom k tomu, že sú známe analytické vzťahy, je u väčšiny testovacích funkcií veľmi jednoduché vypočítať pozíciu a hodnotu extrému pre ľubovoľnú dimenziu. Všetko, čo je treba vedieť, je hodnota extrému v 1D realizácii. Napríklad u tzv. Schwefelovej funkcie (Obr. 9) je v E_1 pozícia globálneho extrému na súradniciach $x_1 = 420,97$ a hodnota funkcie je $f(x_1) = -418,9829$. Pre výpočet hodnoty extrému v napr. E_{15} stačí vynásobiť hodnotu extrému v E_1 číslom dimenzie tj. číslom 15. V tomto prípade je hodnota globálneho extrému pre Schwefelovu funkciu v E_{15} rovná $f(x_1, x_2, \dots, x_{15}) = 15 \times (-418,9829) = -6284,7435$. Tento extrém leží na súradniciach $x_1, \dots, x_{15} = 420,97$. Rovnaký princíp platí aj pre ďalšie funkcie mimo už zmienenej funkcie. [1]

2.1 Účelová funkcia

Podmienkou použitia EA je počítačová reprezentácia optimalizovaného problému, čiže existencia matematického modelu realizovaného v počítači alebo počítačová simulácia daného procesu (deja), ktorého optimálne riešenie hľadáme. To znamená, že pre ľubovoľný bod prehľadávaného priestoru, teda pre ľubovoľné potenciálne riešenie vieme počítačom vyčíslieť hodnotu jeho účelovej funkcie – ohodnotiť ho z hľadiska úspešnosti, z hľadiska miery splnenia požadovaného cieľa. Pritom vôbec nezáleží na type daného procesu, ktorý môže byť fyzikálneho, chemického, biologického, ekonomického, alebo hoci sociologického charakteru. [10]

Účelová funkcia je matematicky model riešeného problému. Výrazom “účelová funkcia” budeme rozumieť funkciu, ktorej optimalizácia (nájdenie maxima alebo minima) povedie k nájdeniu optimálnych hodnôt jej argumentov. [1]

Na každú účelovú funkciu je možné pozerat' ako na geometrický problém, v jeho rámci sa hľadá najnižšia (minimum) či najvyššia (maximum) pozícia na ploche ležiacej v n -rozmernom priestore. Počet dimenzií je daný počtom optimalizovaných argumentov účelovej funkcie. [1]

Ak má optimalizovaná funkcia napr. šesť argumentov (nezávisle premenných), potom sa hľadá extrém na šesťrozmernej ploche v sedemrozmernom priestore, kde siedma dimenzia je návratová hodnota účelovej funkcie. [1]

Vo vzorci testovacej funkcie $f(x_1, x_2, \dots, x_D)$ sa používa symbol $\sum_{i=1}^D x_i$ pre súčet D sčítancov, $\prod_{i=1}^D x_i$ pre súčin D činiteľov, kde D je číslo dimenzie.

2.2 Vybrané testovacie funkcie

Testovacie funkcie je možné získať z viacerých zdrojov, niektoré vybrané zdroje som bližšie preskúmal a hodnoty porovnal.

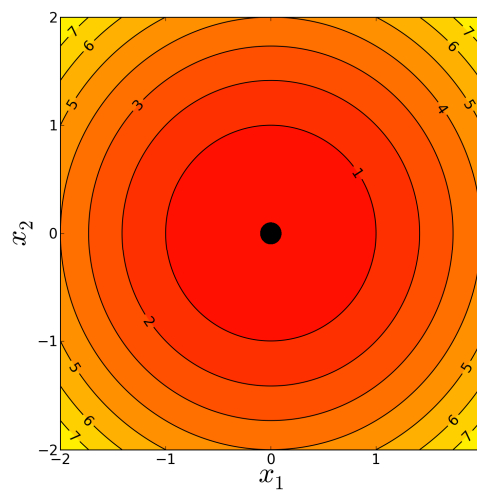
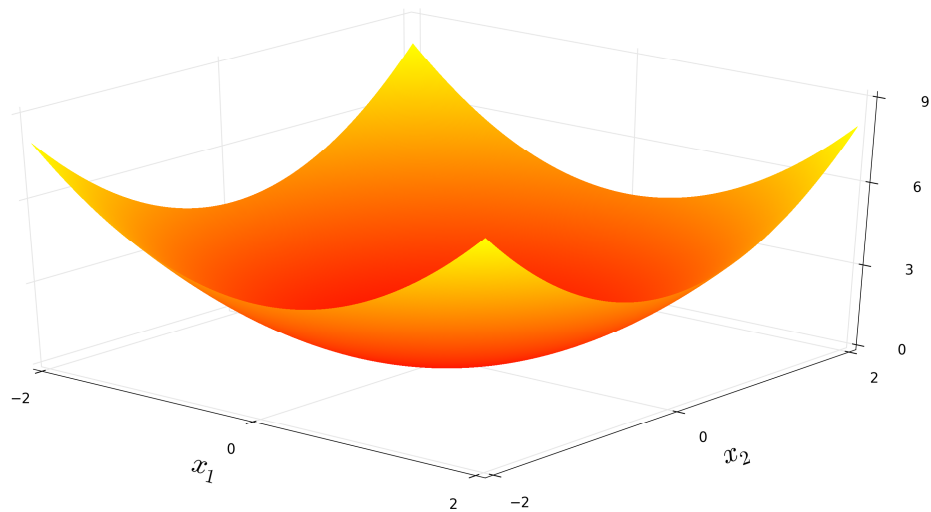
Pre testovacie funkcie uvádzam originálny názov, analytický zápis – vzorec $f(x_1, x_2, \dots, x_D)$, prehľadávaný interval, graf v E_3 , vrstevnicový graf v E_2 a hodnotu extrému v dimenzii D . Čierna oblasť v E_2 reprezentuje množinu bodov líšiacich sa od globálneho extrému x % v zmysle hodnoty účelovej funkcie.

Na generovanie grafov testovacích funkcií v E_3 a vrstevnicových grafov v E_2 som použil knižnicu matplotlib pre programovací jazyk python, ktorá používa podobnú syntax ako MATLAB a je voľne šíriteľná pod licenciou BSD.

2.2.1 Sphere function (First de Jong's function)

Účelová funkcia: $\sum_{i=1}^D x_i^2$ (1)

[1][2][3][4][5]



Obr. 4. Sphere function (1)

Prehľadovaný interval: $-5,12 \leq x_i \leq 5,12$; $i = 1, 2, \dots, D$ [2][3][5]

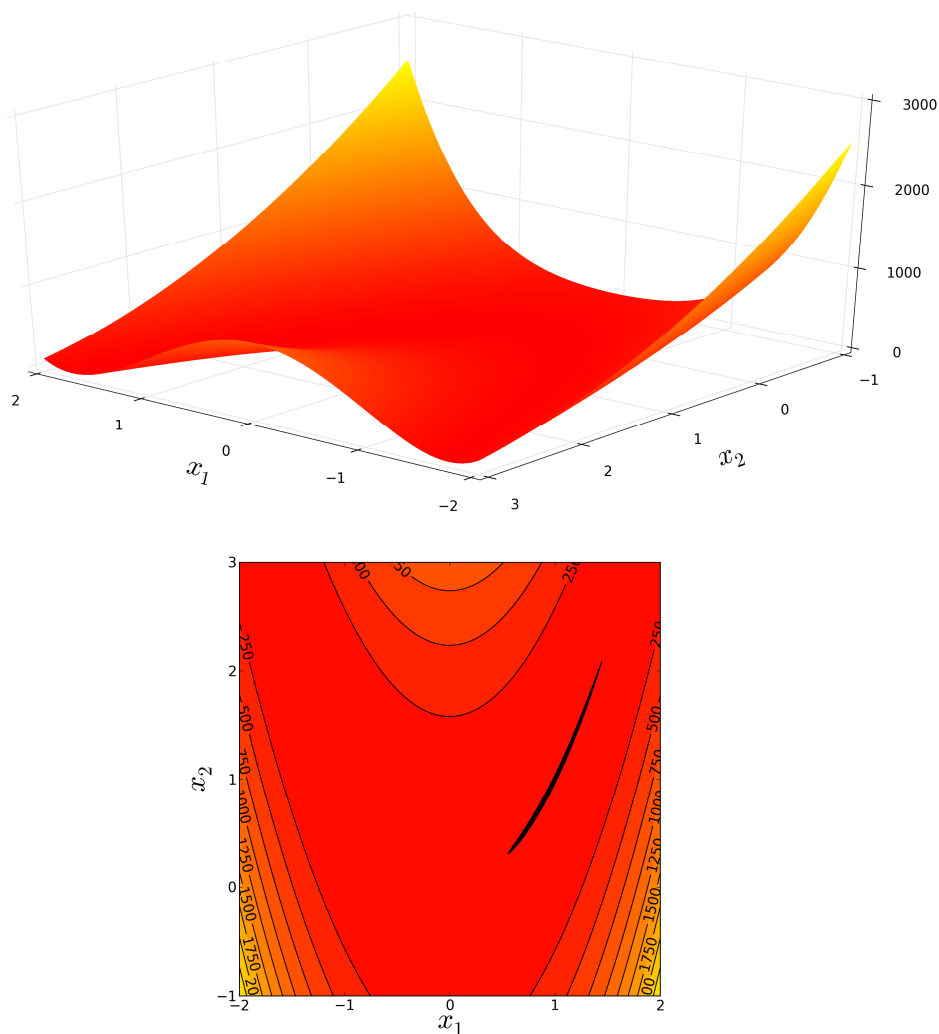
Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1][2][3][5]

2.2.2 Rosenbrock's function (Rosenbrock's valley, Second de Jong's function, Banana function)

Účelová funkcia:
$$\sum_{i=1}^{D-1} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \quad (2)$$

[1][2][3][4][5]



Obr. 5. Rosenbrock's function (2)

Prehľadovaný interval: $-2,048 \leq x_i \leq 2,048 ; i = 1, 2, \dots, D$ [2][5]

$-5 \leq x_i \leq 10 ; i = 1, 2, \dots, D$ [3]

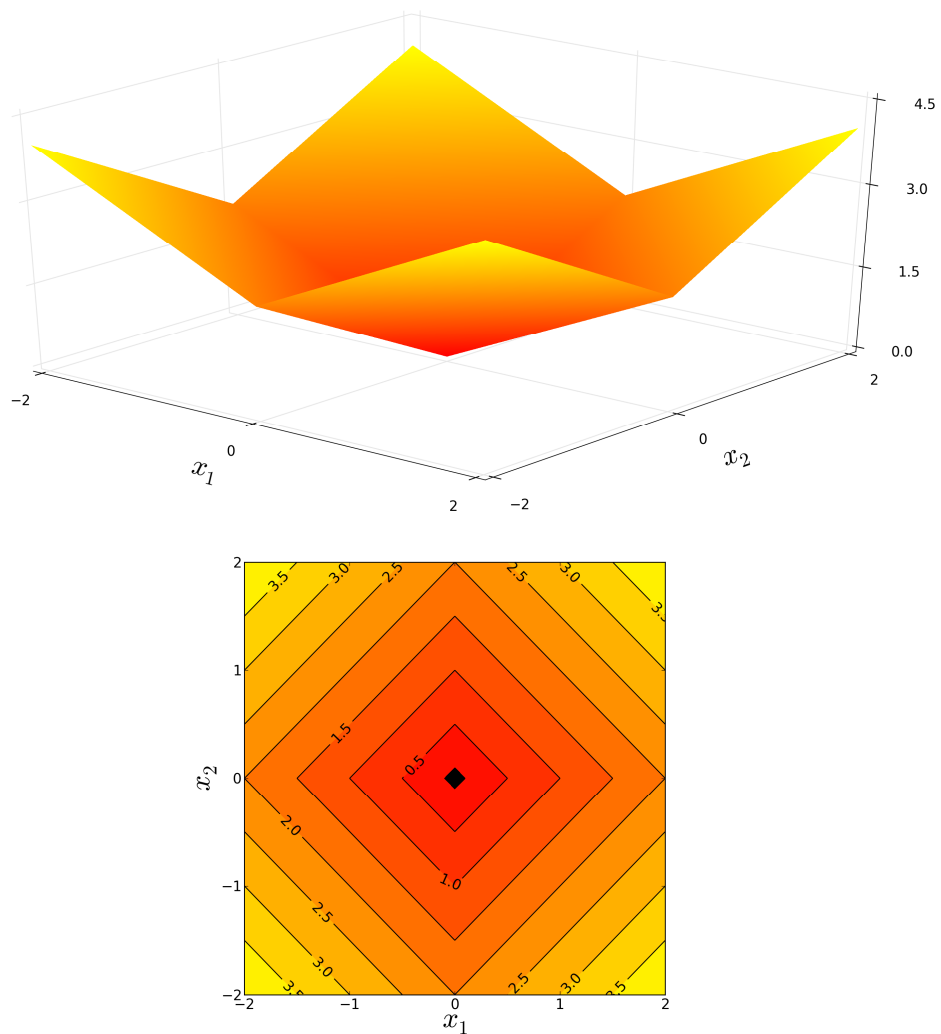
Globálne minimum v E_2 : $f(x_1, x_2) = (1, 1) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (1, 1, \dots, 1) = 0 \times D = 0$ [1][2][3][5]

2.2.3 Third de Jong's function

Účelová funkcia: $\sum_{i=1}^D |x_i|$ (3)

[1][4]



Obr. 6. Third de Jong's function (3)

Prehľadovaný interval: $-2,048 \leq x_i \leq 2,048 ; i = 1, 2, \dots, D$ [4]

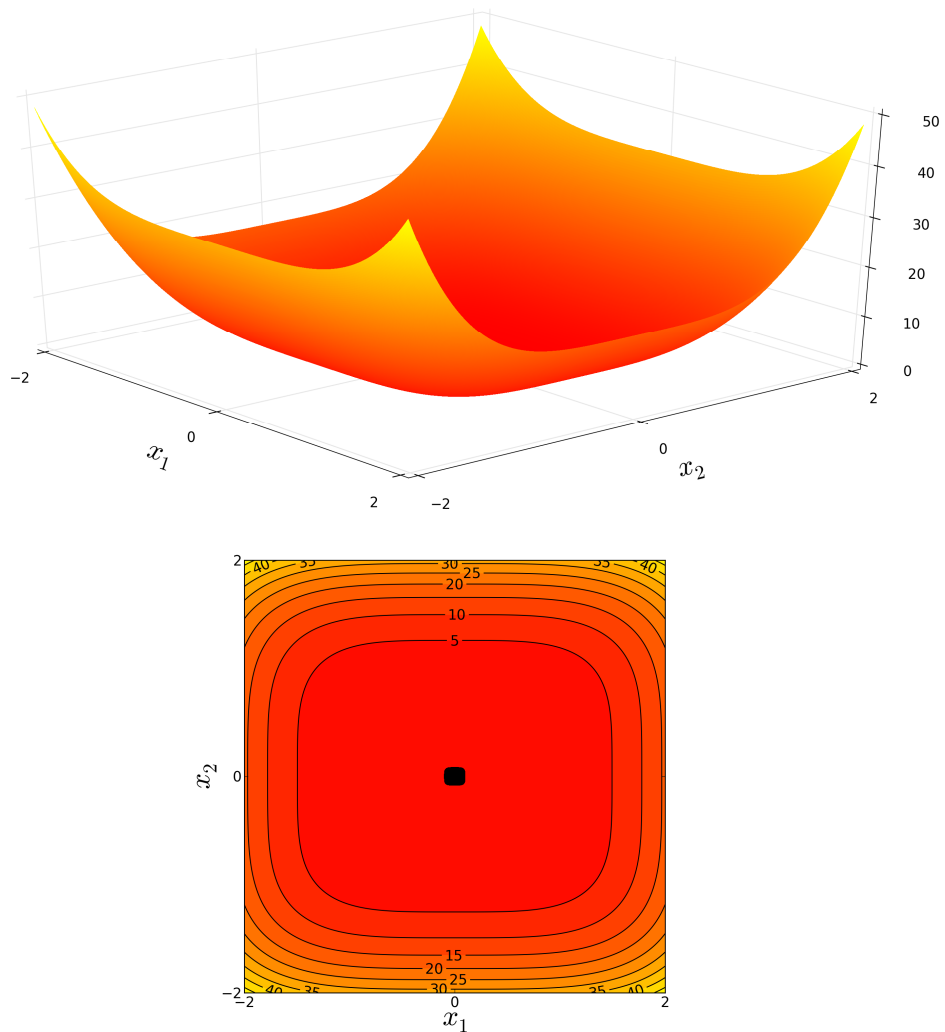
Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1]

2.2.4 Fourth de Jong's function

Účelová funkcia: $\sum_{i=1}^D ix_i^4$ (4)

[1][4]



Obr. 7. Fourth de Jong's function (4)

Prehľadávaný interval: $-1,28 \leq x_i \leq 1,28$; $i = 1, 2, \dots, D$ [4]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1]

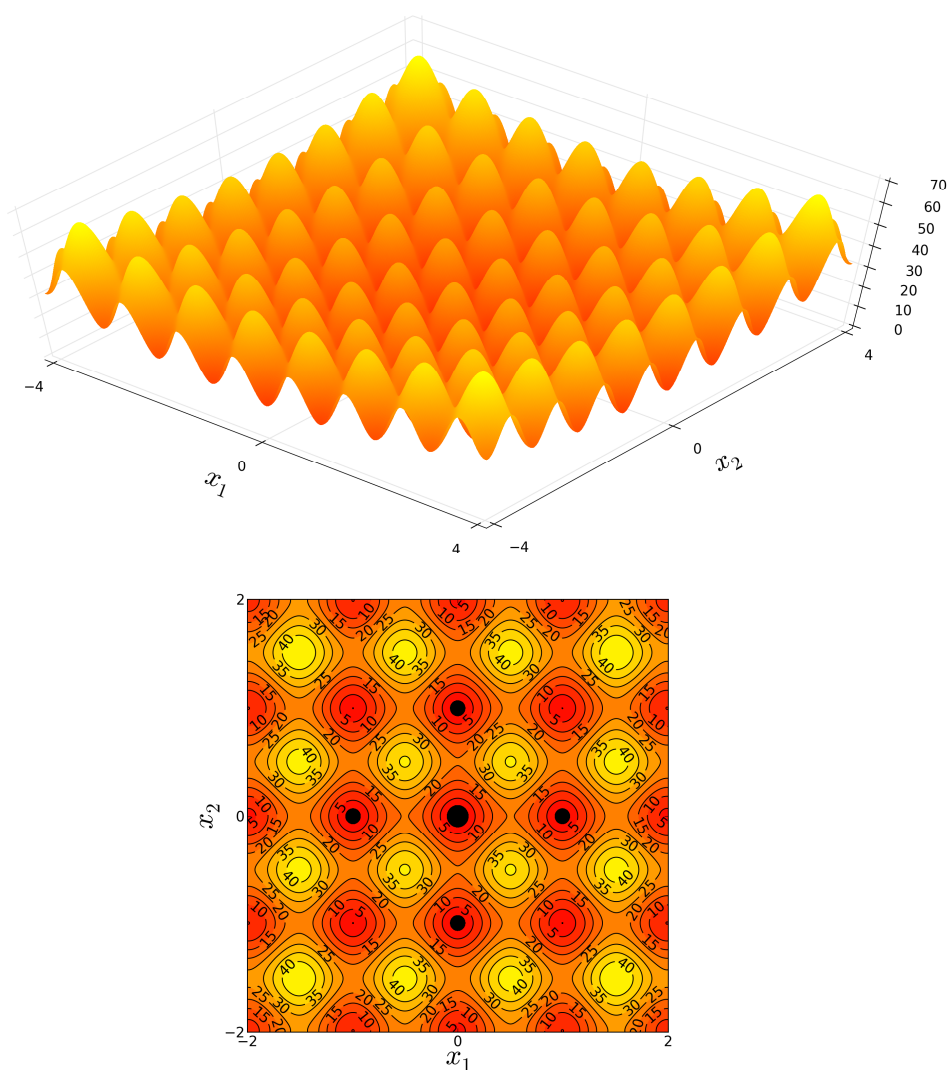
2.2.5 Rastrigin's function

Účelová funkcia: $2D \sum_{i=1}^D x_i^2 - 10 \cos(2\pi x_i)$ (5)

[1]

$$10D + \sum_{i=1}^D x_i^2 - 10 \cos(2\pi x_i) \quad (6)$$

[2][3][4][5]



Obr. 8. Rastrigin's function (6)

Prehľadovaný interval: $-5,12 \leq x_i \leq 5,12$; $i = 1, 2, \dots, D$ [2][3][4][5]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = -200 \times D$ [1]

$$f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0 \quad [2][3][5]$$

Zdroj [1] uvádza vo vzorci (5) násobenie $2D$ a tiež uvádza nesprávnu pozíciu globálneho minima v E_D , ktoré je správne $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = -40 \times D$. Zobrazenie v E_3 , ktoré uvádza zdroj [1] nie je ani podľa vzorca (5), ani podľa (6) a pravdepodobne bola použitá funkcia $f(x_1, x_2) = 2 \times 10 \sum_{i=1}^2 x_i^2 - 10 \cos(2\pi x_i)$, ktorej zodpovedá uvedené globálne minimum $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = -200 \times D$.

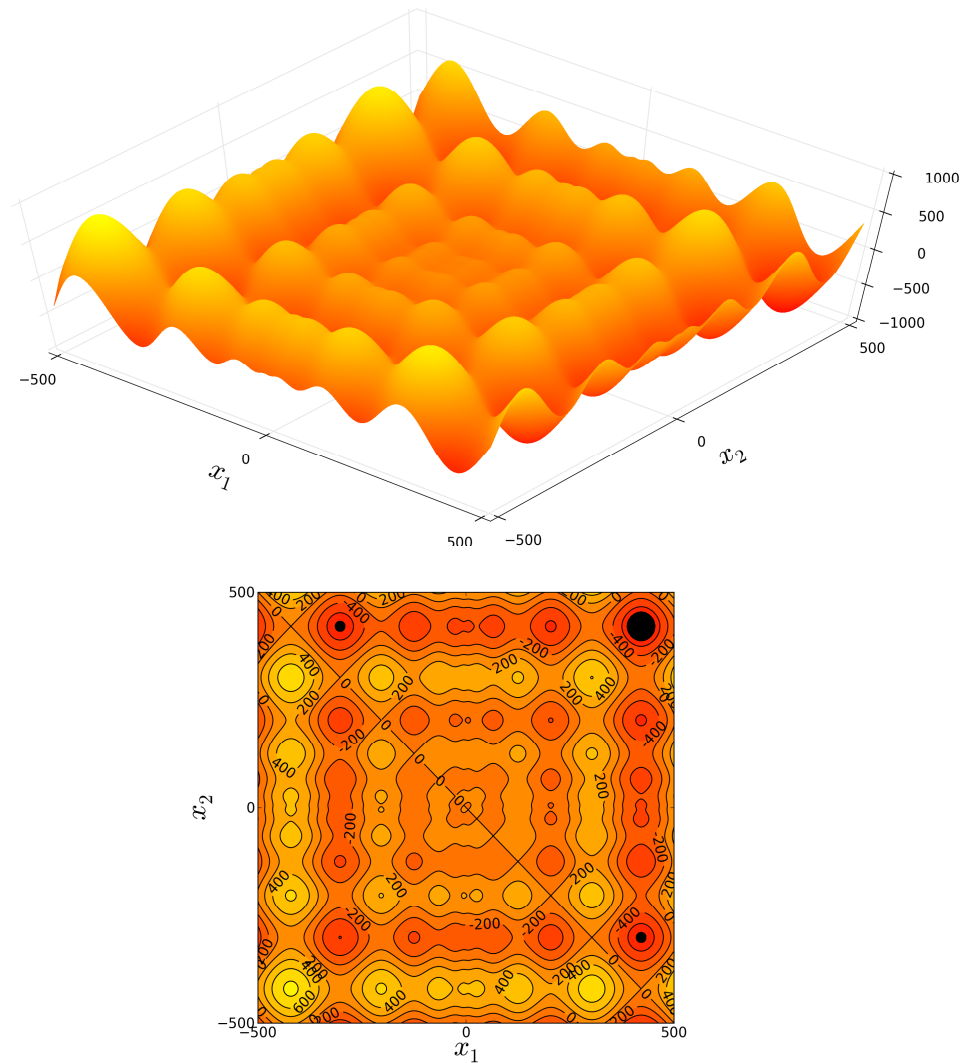
2.2.6 Schwefel's function

$$\text{Účelová funkcia: } - \sum_{i=1}^D x_i \sin(\sqrt{|x_i|}) \quad (7)$$

[1][2][4]

$$418,9829D - \sum_{i=1}^D x_i \sin(\sqrt{|x_i|}) \quad (8)$$

[3][5]



Obr. 9. Schwefel's function (7)

Prehľadovaný interval: $-500 \leq x_i \leq 500$; $i = 1, 2, \dots, D$ [2][3]

$-512 \leq x_i \leq 512$; $i = 1, 2, \dots, D$ [4][5]

Globálne minimum v E_2 : $f(x_1, x_2) = (420,9687; 420,9687) = -837,9658$

v E_D :

$f(x_1, x_2, \dots, x_D) = (420,9687; 420,9687; \dots; 420,9687) = -418,9829 \times D$ [1][2]

$f(x_1, x_2, \dots, x_D) = (420,9687; 420,9687; \dots; 420,9687) = 0 \times D = 0$ [5]

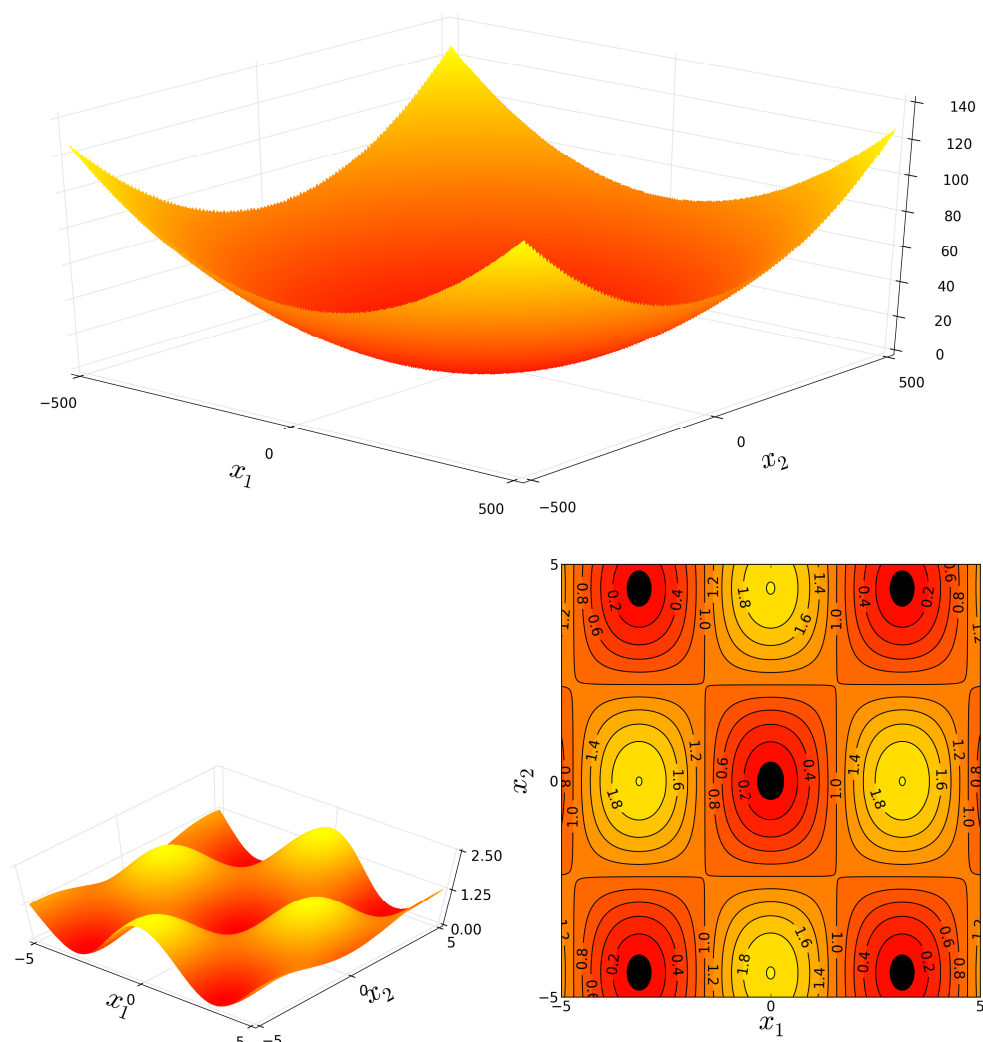
$f(x_1, x_2, \dots, x_D) = (1, 1, \dots, 1) = 0 \times D = 0$ [3]

Zdroje uvádzejú v podstate ten istý vzorec, rozdiel je v tom, že [3] a [5] upravujú vzorec (8) tak, aby dostali globálne minimum $f = (x_1, x_2, \dots, x_D) = 0$. Naviac zdroj [3] uvádza nesprávnu pozíciu globálneho minima v E_D na súradniciach $(1, \dots, 1)$.

2.2.7 Griewangk's function

$$\text{Účelová funkcia: } 1 + \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (9)$$

[1][2][3][4][5]



Obr. 10. Griewangk's function (9)

Prehľadávaný interval: $-600 \leq x_i \leq 600$; $i = 1, 2, \dots, D$ [2][3][4]

$-512 \leq x_i \leq 512$; $i = 1, 2, \dots, D$ [5]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1][2][3][5]

2.2.8 Schaffer's F6 function (Sine envelope sine wave function)

Účelová funkcia:
$$-\sum_{i=1}^{D-1} \left(\frac{\sin^2(x_i^2 + x_{i+1}^2 - 0,5)}{(0,001(x_i^2 + x_{i+1}^2) + 1)^2} + 0,5 \right)$$
 (10)

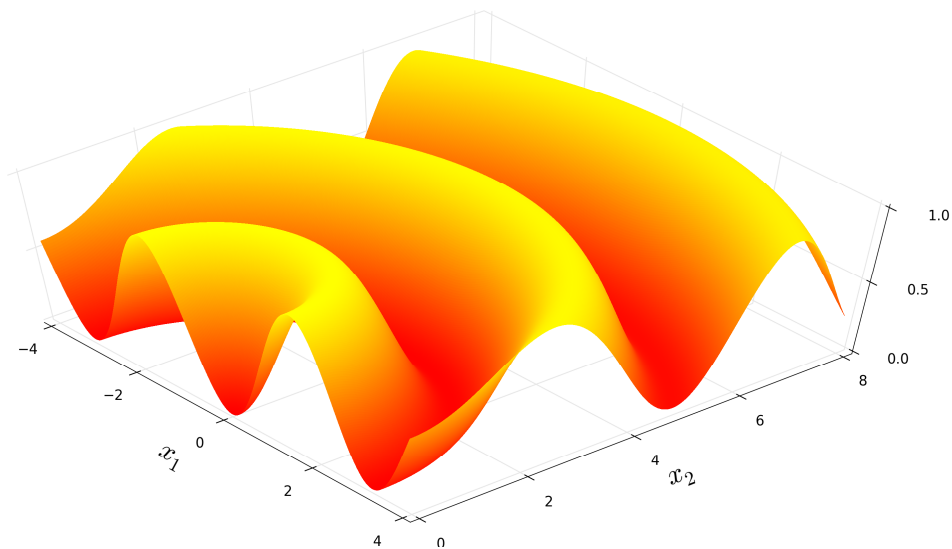
[1]

$$-\sum_{i=1}^{D-1} \left(\frac{\sin^2(\sqrt{x_i^2 + x_{i+1}^2} - 0,5)}{(0,001(x_i^2 + x_{i+1}^2) + 1)^2} + 0,5 \right)$$
 (11)

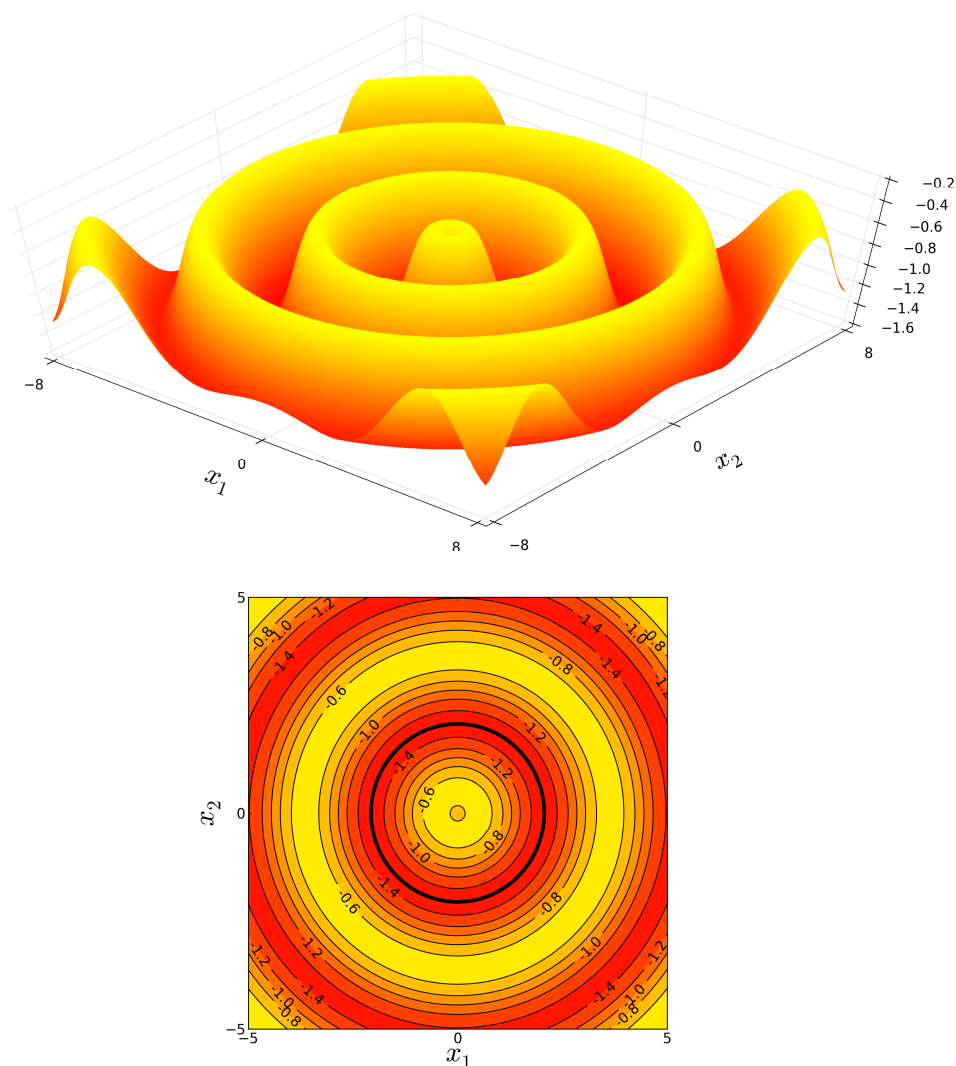
[4]

pre $D = 2$:
$$f(x_1, x_2) = \frac{\sin^2(\sqrt{x_1^2 + x_2^2} - 0,5)}{(0,001(x_1^2 + x_2^2) + 1)^2} + 0,5$$
 (12)

[5][6]



Obr. 11. Schaffer's F6 function (12)



Obr. 12. Sine envelope sine wave function (11)

Prehľadovaný interval: $-100 \leq x_i \leq 100$; $i = 1, 2, \dots, D$

[4][5]

Globálne minimum v E_1 : $f(x_1) = (\pm 2,06668) = -1,4915$

v E_2 : kružnica $r = 2,06668 = \sqrt{x_1^2 + x_2^2}$

$$f(x_1, x_2) = -1,4915$$

v E_D : kružnica v E_D $r = 2,06668 = \sqrt{x_1^2 + x_2^2 + \dots + x_D^2}$

$$f(x_1, x_2, \dots, x_D) = -1,4915 \times (D - 1) \quad [1]$$

$$f(x_1, x_2) = (0, 0) = 0 \quad [5]$$

Zdroj [1] nemá v čitatel'ovi (10) odmocninu, globálne minimum uvádza podľa (10), no zobrazenie v E_3 podľa (11). Realizáciu funkcie som našiel aj pod názvom Schaffer's F6 function (12), v tomto vzorci nie je $-0,5$ súčast'ou funkcie sínus v čitatel'ovi a globálne minimum je 0.

2.2.9 Schaffer's F7 function (Stretched V sine wave function)

Účelová funkcia:
$$\sum_{i=1}^{D-1} \left(\sqrt[4]{x_i^2 + x_{i+1}^2} \sin^2 \left(50^{10} \sqrt{x_i^2 + x_{i+1}^2} \right) + 1 \right) \quad (13)$$

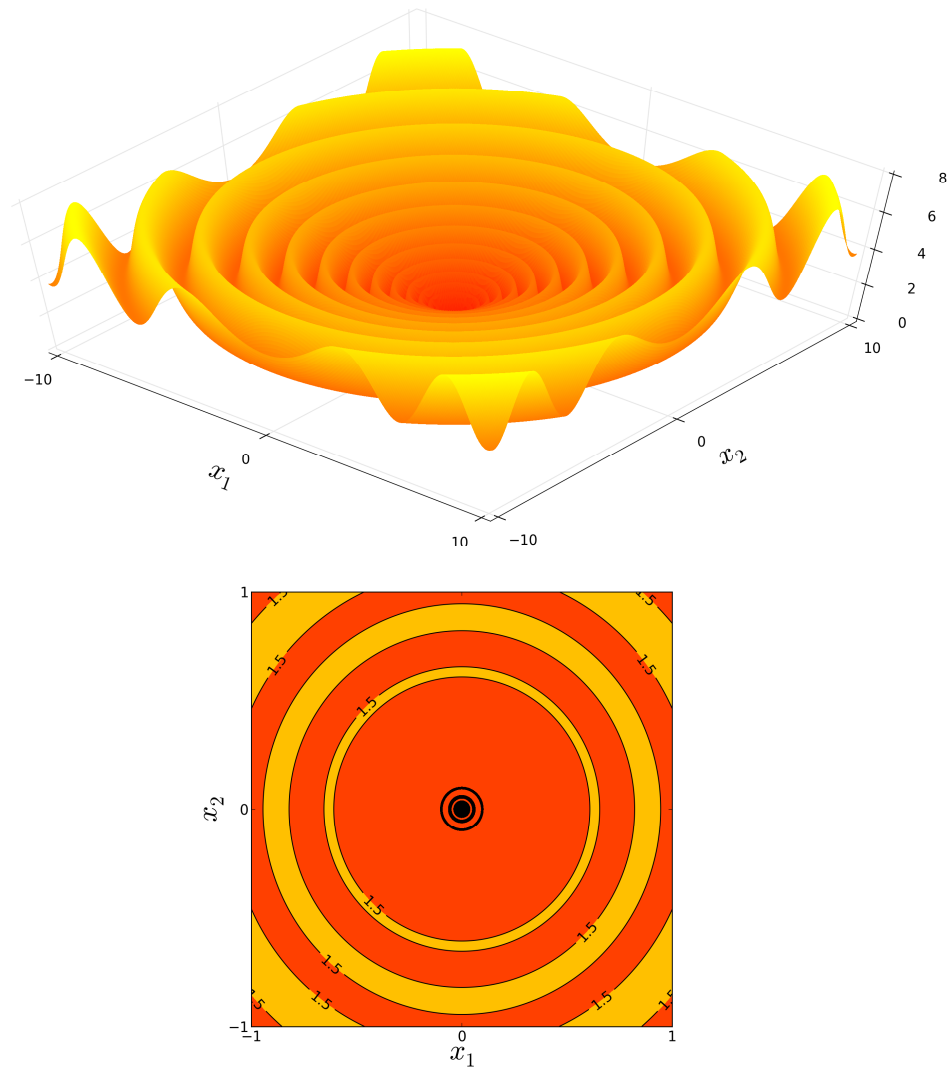
[1]

$$\sum_{i=1}^{D-1} \sqrt[4]{x_i^2 + x_{i+1}^2} \left(\sin^2 \left(50^{10} \sqrt{x_i^2 + x_{i+1}^2} \right) + 1 \right) \quad (14)$$

[4]

pre $D = 2$:
$$f(x_1, x_2) = \sqrt[4]{x_1^2 + x_2^2} \left(\sin^2 \left(50^{10} \sqrt{x_1^2 + x_2^2} \right) + 1 \right) \quad (15)$$

[7]



Obr. 13. Schaffer's F7 function (15)

Prehľadovaný interval: $-10 \leq x_i \leq 10$; $i = 1, 2, \dots, D$ [4]

$-1 \leq x_i \leq 1$; $1 \leq i \leq 2$ [7]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1]

Zdroj [1] vo vzorci (13) násobí $\sqrt[4]{x_i^2 + x_{i+1}^2}$, až potom pripočíta 1, no zobrazenie v E_3 uvádza podľa (14), ide pravdepodobne len o chybu zátvorky.

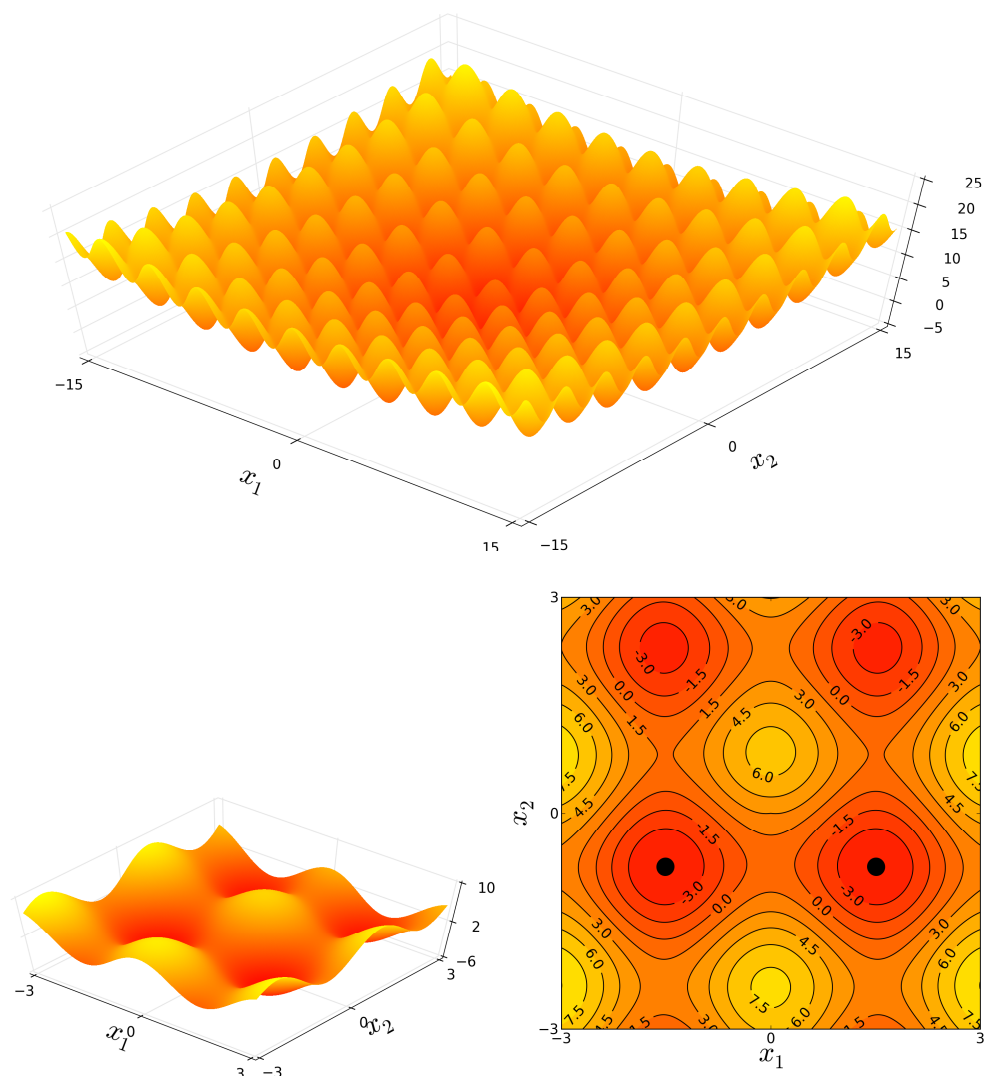
2.2.10 Ackley I

Účelová funkcia:
$$\sum_{i=1}^{D-1} 3(\cos(2x_i) + \sin(2x_{i+1})) + \frac{\sqrt{x_i^2 + x_{i+1}^2}}{e^5} \quad (16)$$

[1]

$$\sum_{i=1}^{D-1} 3(\cos(2x_i) + \sin(2x_{i+1})) + \frac{\sqrt{x_i^2 + x_{i+1}^2}}{\sqrt[5]{e}} \quad (17)$$

[4]



Obr. 14. Ackley I (17)

Prehľadovaný interval: $-30 \leq x_i \leq 30 ; i = 1, 2, \dots, D$ [4]

Globálne minimum v E_2 : $f(x_1, x_2) = (\pm 1,50236; -0,754865) = -4,5901$

$$\text{v } E_3: \quad f(x_1, x_2, x_3) = (1,51563; -1,10937; -0,747245) = -7,54276$$

v E_D (približne):

$$f(x_1, x_2, \dots, x_D) = (1,51563; -1,1151; -1,10972; \dots, -1,10972; -0,747245)$$

$$f(x_1, x_2, \dots, x_D) = -7,54276 - 2,91867 \times (D - 3) \quad [1]$$

Zdroj [1] vo vzorci (16) uvádza e^5 , no zobrazenie v E_3 uvádza podľa vzorca (17), v ktorom je $\sqrt[5]{e}$.

2.2.11 Ackley's function

Účelová funkcia:

$$\sum_{i=1}^{D-1} \left(a + e - \frac{a}{e^{\sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}}} - e^{\frac{\cos(cx_i) + \cos(cx_{i+1})}{2}} \right) = \sum_{i=1}^{D-1} \left(a + e - ae^{-b\sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}} - e^{\frac{\cos(cx_i) + \cos(cx_{i+1})}{2}} \right) \quad (18)$$

[1]

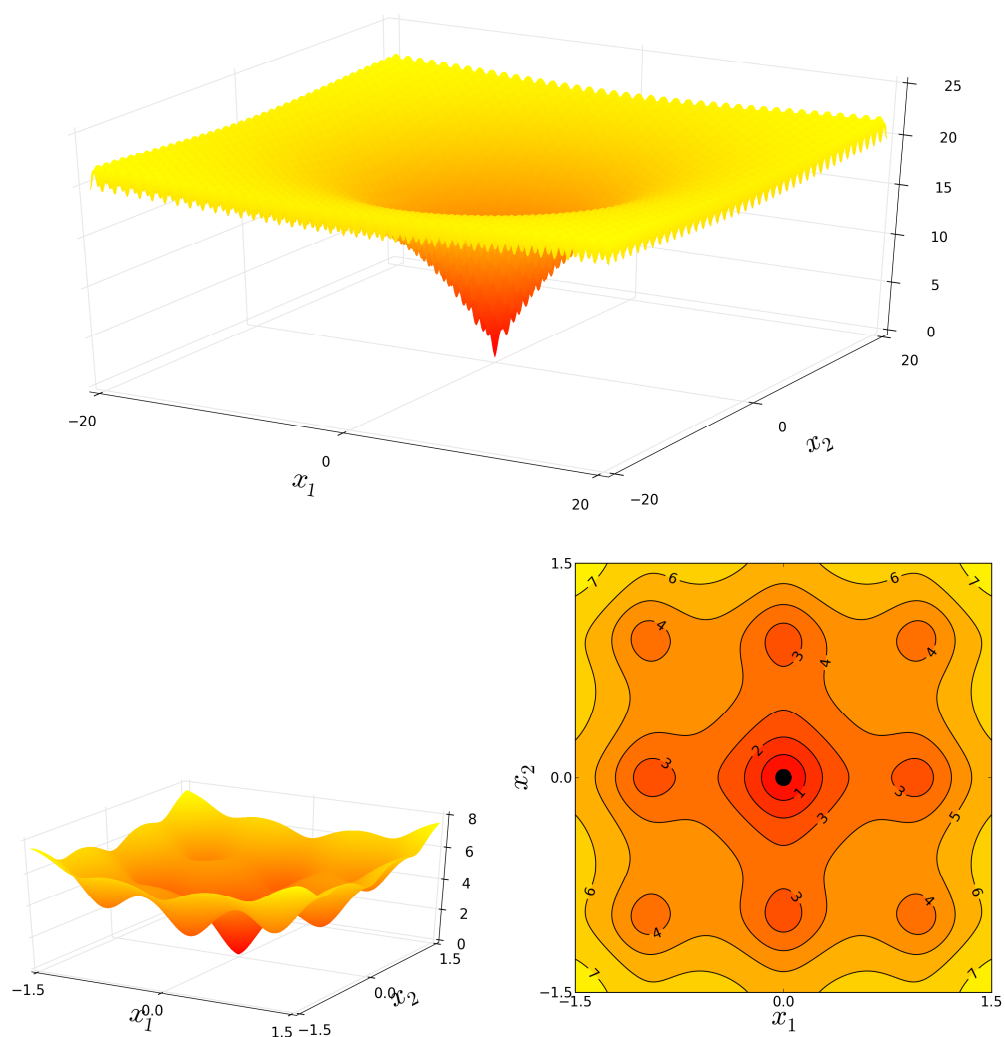
$$\sum_{i=1}^{D-1} \left(a + e^{-a} e^{-b\sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}} - e^{\frac{\cos(cx_i) + \cos(cx_{i+1})}{2}} \right) \quad (19)$$

[4]

$$a + e - ae^{-b\sqrt{\frac{\sum_{i=1}^D x_i^2}{D}}} - e^{\frac{\sum_{i=1}^D \cos(cx_i)}{D}} \quad (20)$$

[2][3][5]

Doporučené nastavenie hodnôt: $a = 20$; $b = 0,2$; $c = 2\pi$



Obr. 15. Ackley's function (20)

Prehľadávaný interval: $-1 \leq x_i \leq 1 ; i = 1, 2, \dots, D$ [2]

$-15 \leq x_i \leq 30 ; i = 1, 2, \dots, D$ [3]

$-30 \leq x_i \leq 30 ; i = 1, 2, \dots, D$ [4]

$-32 \leq x_i \leq 32 ; i = 1, 2, \dots, D$ [5]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = 0$

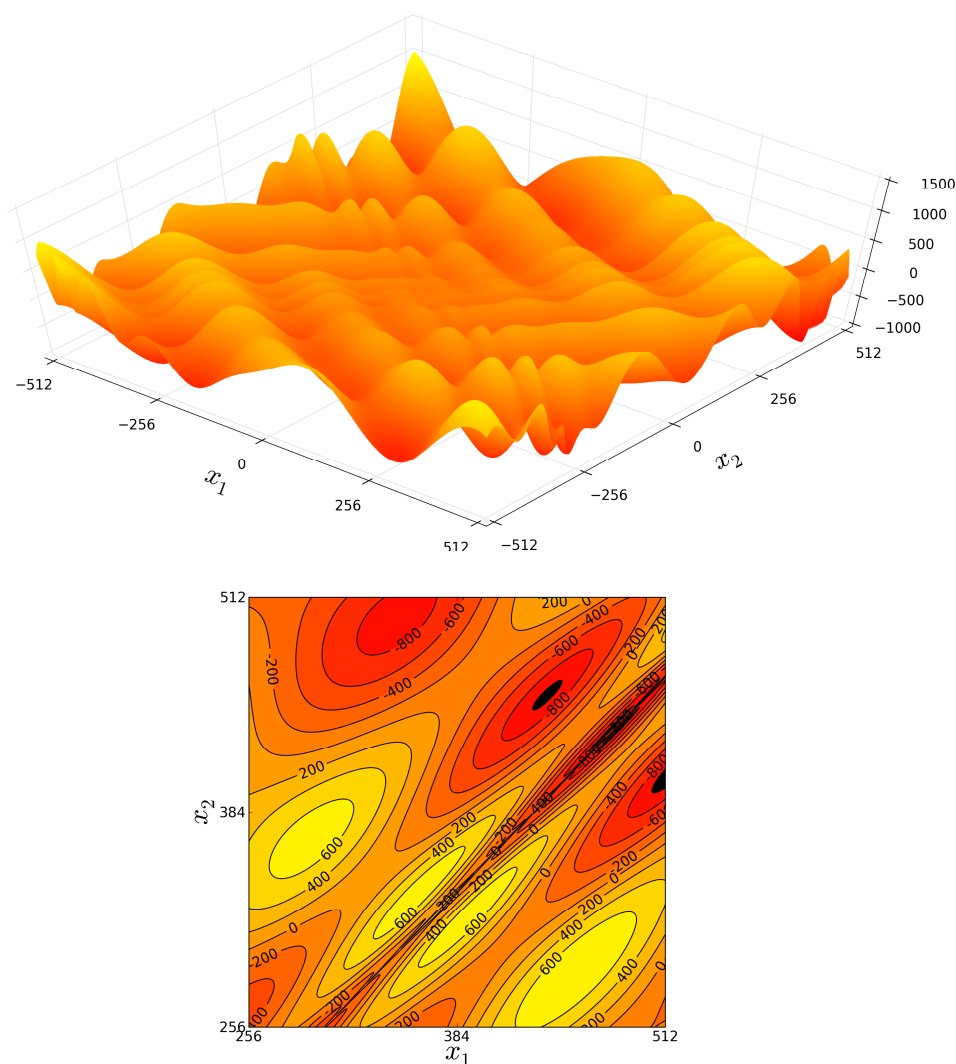
v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = 0 \times D = 0$ [1][2][3][5]

Zdroj [1] vo vzorci (18) zapisuje Ackleyho podľa (20) pre $D = 2$, v inej dimenzii budú hodnoty účelovej funkcie rôzne pre (18) a pre (20). Zdroj [4] má vo vzorci (19) podstatný rozdiel e^{-a} .

2.2.12 Egg holder

Účelová funkcia:
$$-\sum_{i=1}^{D-1} x_i \sin\left(\sqrt{|x_i - x_{i+1} - 47|}\right) + (x_{i+1} + 47) \sin\left(\sqrt{\left|\frac{x_i}{2} + x_{i+1} + 47\right|}\right) \quad (21)$$

[1][4]



Obr. 16. Egg holder (21)

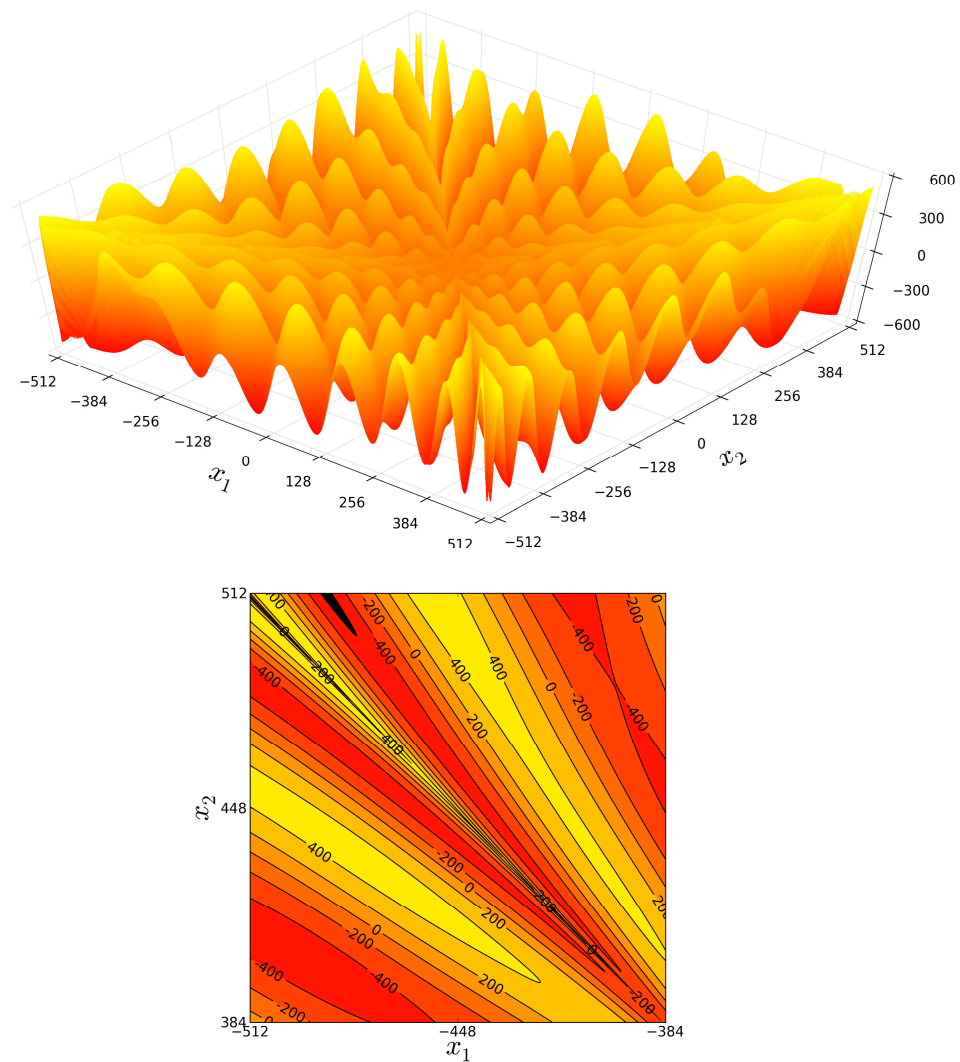
Prehľadávaný interval: $-512 \leq x_i \leq 512 ; i = 1, 2, \dots, D$ [4]

Globálne minimum v E_2 : $f(x_1, x_2) = (512; 404, 2319) = -959,640662711$ [8][9]

2.2.13 Rana's function

Účelová funkcia:
$$\sum_{i=1}^{D-1} \left(x_i \sin\left(\sqrt{|-x_i + x_{i+1} + 1|}\right) \cos\left(\sqrt{|x_i + x_{i+1} + 1|}\right) + (x_{i+1} + 1) \cos\left(\sqrt{|-x_i + x_{i+1} + 1|}\right) \sin\left(\sqrt{|x_i + x_{i+1} + 1|}\right) \right) \quad (22)$$

[1][4][5]



Obr. 17. Rana's function (22)

Prehľadávaný interval: $-500 \leq x_i \leq 500 ; i = 1, 2, \dots, D$ [4]

$-512 \leq x_i \leq 512 ; i = 1, 2, \dots, D$ [5]

Globálne minimum v E_2 : $f(x_1, x_2) = (-488, 6326; 512) = -511,73$ [5]

Globálne minimum v E_2 podľa zdroja [5] platí pre $-512 \leq x_1 \leq 512 ; -512 \leq x_2 \leq 512$.

2.2.14 Pathological function

Účelová funkcia:

$$\sum_{i=1}^{D-1} \left(\frac{\sin^2 \left(\sqrt{100x_i^2 - x_{i+1}^2} \right) - 0,5}{0,001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2 + 1} + 0,5 \right) = \sum_{i=1}^{D-1} \left(\frac{\sin^2 \left(\sqrt{100x_i^2 - x_{i+1}^2} \right) - 0,5}{0,001(x_i - x_{i+1})^4 + 1} + 0,5 \right) \quad (23)$$

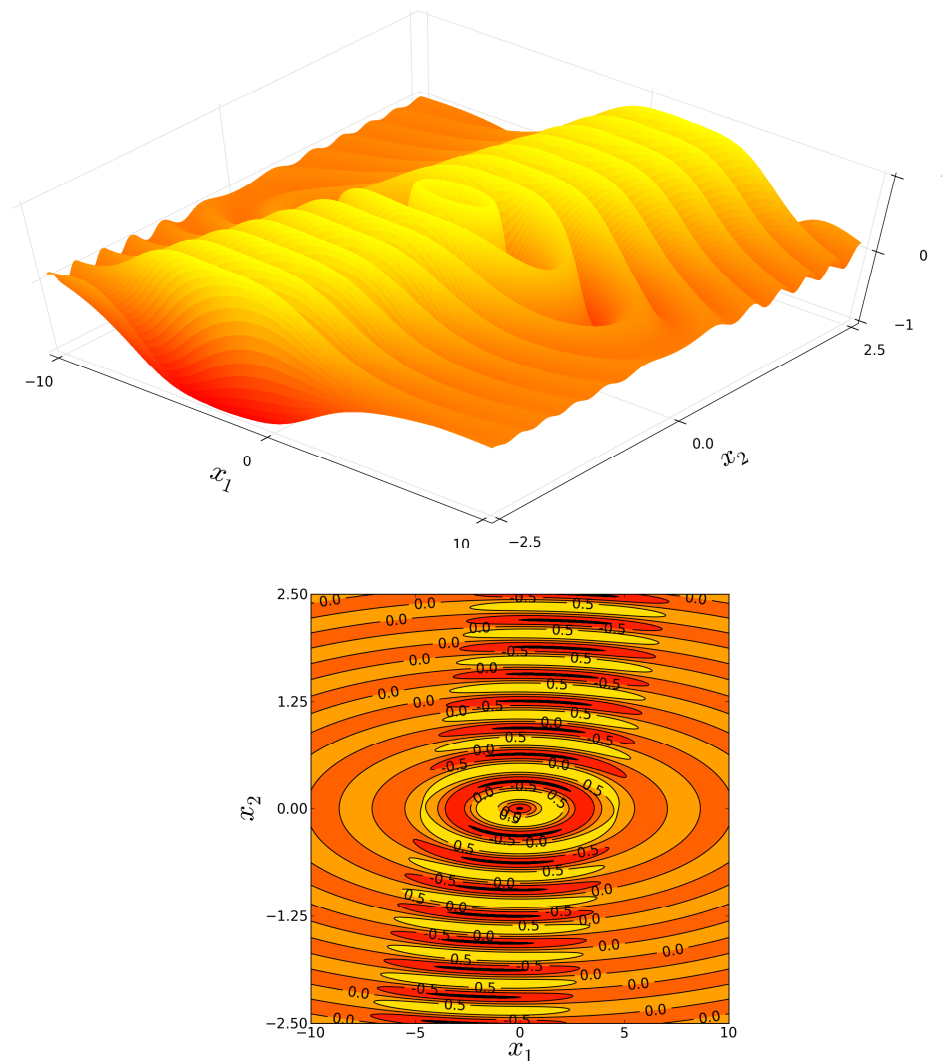
[1]

$$\sum_{i=1}^{D-1} \left(\frac{\sin^2 \left(\sqrt{100x_i^2 + x_{i+1}^2} \right) - 0,5}{0,001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2 + 1} + 0,5 \right) = \sum_{i=1}^{D-1} \left(\frac{\sin^2 \left(\sqrt{100x_i^2 + x_{i+1}^2} \right) - 0,5}{0,001(x_i - x_{i+1})^4 + 1} + 0,5 \right) \quad (24)$$

[4]

$$\sum_{i=1}^{D-1} \frac{\sin^2 \left(\sqrt{x_i^2 + 100x_{i+1}^2} \right) - 0,5}{0,001(x_i - x_{i+1})^4 + 0,5} \quad (25)$$

[9]



Obr. 18. Pathological function (25)

Prehľadávaný interval: $-100 \leq x_i \leq 100$; $i = 1, 2, \dots, D$ [4]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = -1,99600798403$ [9]

Vo vzorci (23) podľa zdroja [1] je možné dosiahnuť záporné číslo pod odmocninou. Zdroj [9] uvádza nesprávne globálne minimum v E_2 , podľa (25) po dosadení do vzorca je hodnota $f(x_1, x_2) = (0, 0) = -1$ a globálnych miním je veľa, to platí aj pre (24) s posunutím na 0.

2.2.15 Michalewicz's function

Účelová funkcia:
$$-\sum_{i=1}^{D-1} \sin(x_i) \sin^{2m}\left(\frac{x_i^2}{\pi}\right) + \sin(x_{i+1}) \sin^{2m}\left(\frac{2x_i^2}{\pi}\right) \quad (26)$$

[1]

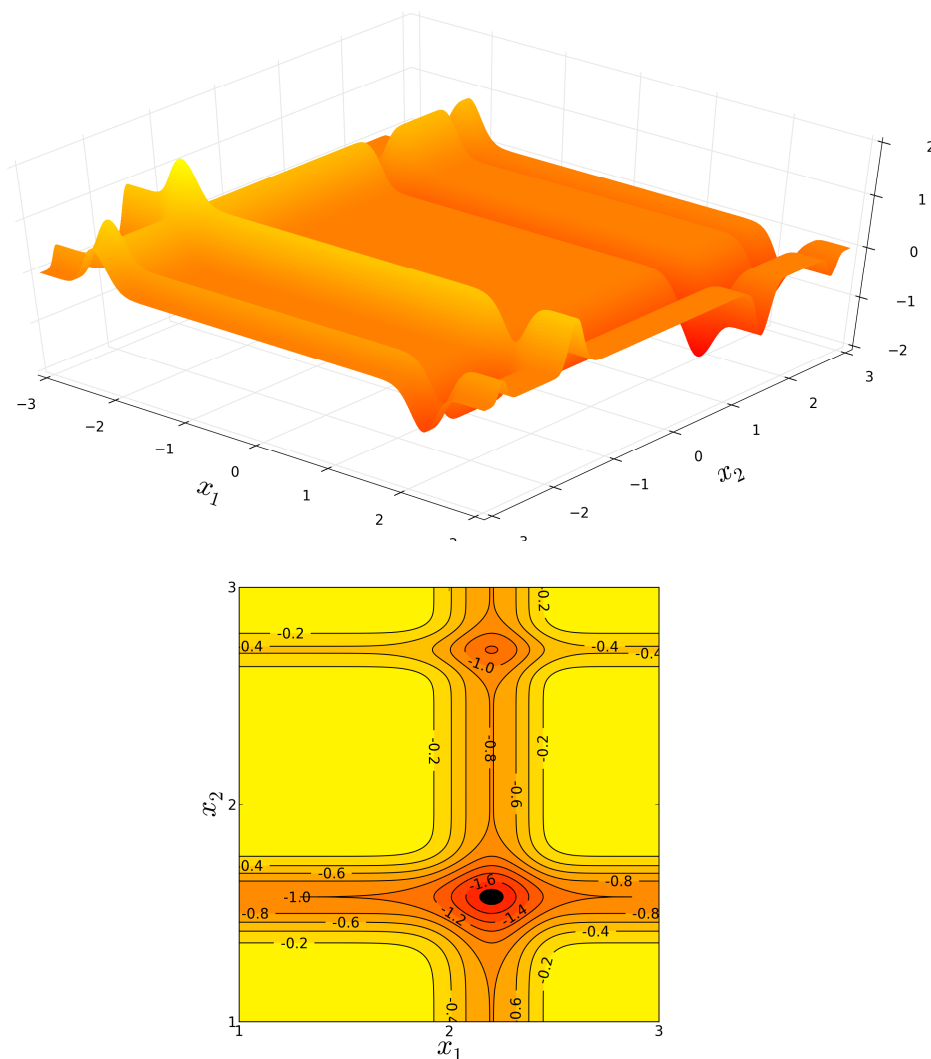
$$\sum_{i=1}^{D-1} \sin(x_i) \sin^{2m}\left(\frac{x_{i+1}^2}{\pi}\right) + \sin(x_{i+1}) \sin^{2m}\left(\frac{2x_{i+1}^2}{\pi}\right) \quad (27)$$

[4]

$$-\sum_{i=1}^D \sin(x_i) \left(\sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \right) \quad (28)$$

[2][3]

Zmenou hodnoty m určujeme „strmost“ hrán, vyššia hodnota spôsobí náročnejšie hľadanie optima. Bežné nastavenie hodnoty: $m = 10$



Obr. 19. Michalewicz's function (28)

Prehľadovaný interval:

$$0 \leq x_i \leq \pi ; i = 1, 2, \dots, D$$

[2][3][4]

Globálne minimum v E_2 : $f(x_1, x_2) = (2, 20291; 1, 57096) = -1, 8013$ [1][3][9]

v E_D :

$$f(x_1, x_2, \dots, x_D) = (2, 20291; 1, 57104; \dots; 1, 57104) = 1, 00098 \times (D - 2) \quad [1]$$

Zdroj [1] uvádza vo vzorci (26) v jednom menovateľovi x_i^2 a v druhom $2x_i^2$, podobne zdroj [4] vo vzorci (27) uvádza x_{i+1}^2 a $2x_{i+1}^2$, takéto rozloženia nezodpovedajú vzorcu (28), ktorý má v E_2 menovateľov x_i^2 a $2x_{i+1}^2$. No zobrazenie v E_3 , ktoré uvádza zdroj [1], je podľa vzorca (28).

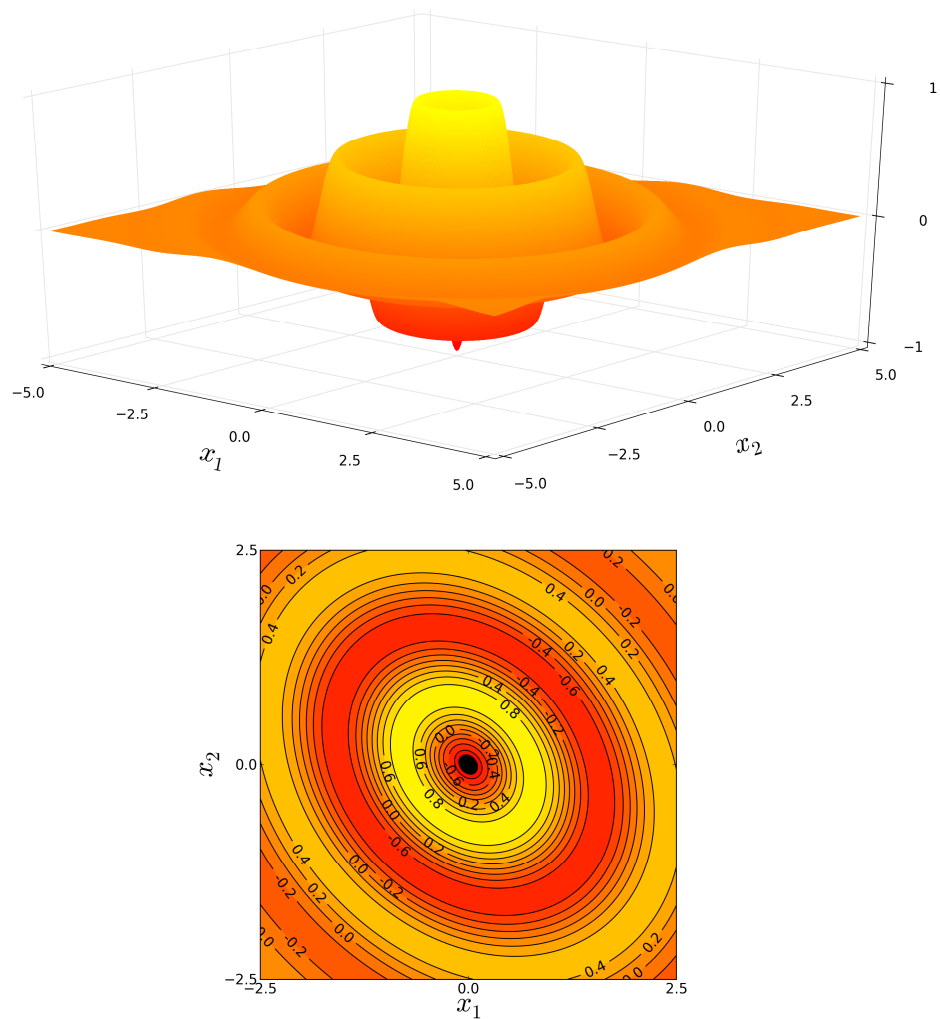
2.2.16 Master's cosine wave function

$$\text{Účelová funkcia: } \sum_{i=1}^{D-1} e^{-\frac{1}{8}\left(x_i^2 + \frac{1}{2}x_i x_{i+1} + x_{i+1}^2\right)} \cos\left(4\sqrt{x_i^2 + \frac{1}{2}x_i x_{i+1} + x_{i+1}^2}\right) \quad (29)$$

[1]

$$-\sum_{i=1}^{D-1} e^{-\frac{1}{8}\left(x_i^2 + \frac{1}{2}x_i x_{i+1} + x_{i+1}^2\right)} \cos\left(4\sqrt{x_i^2 + \frac{1}{2}x_i x_{i+1} + x_{i+1}^2}\right) \quad (30)$$

[4]



Obr. 20. Master's cosine wave function (30)

Prehľadávaný interval: $-5 \leq x_i \leq 5$; $i = 1, 2, \dots, D$ [4]

Globálne minimum v E_2 : $f(x_1, x_2) = (0, 0) = -1$

v E_D : $f(x_1, x_2, \dots, x_D) = (0, 0, \dots, 0) = -1 \times D$ [1]

Zdroj [1] uvádza vzorec (29), ktorý nie je určený pre hľadanie minimálnej, ale pre hľadanie maximálnej hodnoty funkcie. No zobrazenie v E_3 , ktoré uvádza [1], je podľa vzorca (30).

II. PRAKTICKÁ ČASŤ

3 IMPLEMENTÁCIA KNIŽNICE V JAZYKU C++

3.1 Implementované testovacie funkcie

Na základe informácií získaných z viacerých zdrojov v teoretickej časti práce som zostavil tabuľku testovacích funkcií, podľa ktorej navrhнем jednotlivé triedy v C++.

Trieda	Vzorec	Interval
cSphere	$\sum_{i=1}^D x_i^2$	$-5,12 \leq x_i \leq 5,12$
cRosenbrock	$\sum_{i=1}^{D-1} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	$-2,048 \leq x_i \leq 2,048$
cThirdDeJong	$\sum_{i=1}^D x_i $	$-2,048 \leq x_i \leq 2,048$
cFourthDeJong	$\sum_{i=1}^D ix_i^4$	$-1,28 \leq x_i \leq 1,28$
cRastrigin	$10D + \sum_{i=1}^D x_i^2 - 10 \cos(2\pi x_i)$	$-5,12 \leq x_i \leq 5,12$
cSchwefel	$-\sum_{i=1}^D x_i \sin(\sqrt{ x_i })$	$-512 \leq x_i \leq 512$
cGriewangk	$1 + \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-512 \leq x_i \leq 512$
cSchafferF6	$\frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0,5}{(0,001(x_1^2 + x_2^2) + 1)^2} + 0,5$	$-100 \leq x_i \leq 100$
cSchafferF7	$\sqrt[4]{x_1^2 + x_2^2} \left(\sin^2\left(50\sqrt[4]{x_1^2 + x_2^2}\right) + 1 \right)$	$-10 \leq x_i \leq 10$
cAckley1	$\sum_{i=1}^{D-1} 3(\cos(2x_i) + \sin(2x_{i+1})) + \frac{\sqrt{x_i^2 + x_{i+1}^2}}{\sqrt[5]{e}}$	$-30 \leq x_i \leq 30$
cAckley	$a + e - ae^{-b \sqrt{\frac{\sum_{i=1}^D x_i^2}{D}}} - e^{-\frac{\sum_{i=1}^D \cos(cx_i)}{D}}$	$-32 \leq x_i \leq 32$
cEggHolder	$-\sum_{i=1}^{D-1} \left(x_i \sin\left(\sqrt{ x_i - x_{i+1} - 47 }\right) + (x_{i+1} + 47) \sin\left(\sqrt{\left \frac{x_i}{2} + x_{i+1} + 47\right }\right) \right)$	$-512 \leq x_i \leq 512$

Trieda	Vzorec	Interval
cRana	$\sum_{i=1}^{D-1} \left(\frac{x_i \sin(\sqrt{ -x_i + x_{i+1} + 1 }) \cos(\sqrt{ x_i + x_{i+1} + 1 }) + (x_{i+1} + 1) \cos(\sqrt{ -x_i + x_{i+1} + 1 }) \sin(\sqrt{ x_i + x_{i+1} + 1 })}{(x_{i+1} + 1) \cos(\sqrt{ -x_i + x_{i+1} + 1 }) \sin(\sqrt{ x_i + x_{i+1} + 1 })} \right)$	$-512 \leq x_i \leq 512$
cPathological	$\sum_{i=1}^{D-1} \frac{\sin^2(\sqrt{x_i^2 + 100x_{i+1}^2}) - 0,5}{0,001(x_i - x_{i+1})^4 + 0,5}$	$-100 \leq x_i \leq 100$
cMichalewicz	$-\sum_{i=1}^D \sin(x_i) \left(\sin^{2m} \left(\frac{ix_i^2}{\pi} \right) \right)$	$0 \leq x_i \leq \pi$
cMastersCosine	$-\sum_{i=1}^{D-1} e^{-\frac{1}{8} \left(x_i^2 + \frac{1}{2} x_i x_{i+1} + x_{i+1}^2 \right)} \cos \left(4 \sqrt{x_i^2 + \frac{1}{2} x_i x_{i+1} + x_{i+1}^2} \right)$	$-5 \leq x_i \leq 5$

Tab. 1. Implementované testovacie funkcie

3.2 Základná trieda cTestFun

Deklarácie a definície jednotlivých tried pre názornosť uvádzam neoddelené. Tieto triedy majú potom v zdrojových súboroch deklaráciu v hlavičkovom súbore testfun.h a definíciu v kompilačnej jednotke testfun.cpp. Základnou triedou pre testovacie funkcie je abstraktná trieda cTestFun. Z nej sú dedené všetky triedy testovacích funkcií.

```
typedef long double t_real;
typedef unsigned char t_byte;

/* cTestFun class */
class cTestFun
{
private:
    t_real domainMin;
    t_real domainMax;

protected:
    void SetDomainMin( t_real aDomainMin ) { domainMin = aDomainMin; }
    void SetDomainMax( t_real aDomainMax ) { domainMax = aDomainMax; }

public:
    cTestFun() {
        domainMin = -512; // default domain minimum
        domainMax = 512; // default domain maximum
    }
    virtual ~cTestFun() {};
    t_real GetDomainMin() { return domainMin; }
    t_real GetDomainMax() { return domainMax; }
    bool OutOfDomain( t_byte aN, t_real *aX ) {
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            if( aX[ii] < domainMin ) return true;
            if( aX[ii] > domainMax ) return true;
        }
        return false;
    }
};
```

```

    }
    t_real GetRandomX() {
        t_real rValue;
        rValue = (domainMax-domainMin)*((t_real)rand()/RAND_MAX)+domainMin;
        return rValue;
    }
    virtual t_real GetFitness( t_byte aN, t_real *aX ) = 0;
};

```

3.2.1 Metóda GetFitness

```

public:
    virtual t_real GetFitness( t_byte aN, t_real *aX ) = 0;

```

Číro virtuálna funkcia GetFitness definuje základnú triedu cTestFun ako abstraktnú, to znamená, že funkcia nemá implementáciu a táto trieda nemôže byť preto inšancovateľná. Aby sa zdedené triedy tiež nestali abstraktnými, musia mať virtuálnu funkciu cGetFitness implementovanú. Funkcia GetFitness implementuje účelovú funkciu, musí byť uvedená v každej zdedenej triede.

3.2.2 Metódy SetDomainMin a SetDomainMax

```

protected:
    void SetDomainMin( t_real aDomainMin );
    void SetDomainMax( t_real aDomainMax );

```

Konštruktor zdedenej triedy vždy volá konštruktor základnej triedy, v tejto základnej triede nastavuje konštruktor hodnoty domény nasledovne:

```

domainMin = -512; // default domain minimum
domainMax = 512; // default domain maximum

```

Každá zdedená trieda môže tieto hodnoty zmeniť volaním metód SetDomainMin alebo SetDomainMax vo svojom konštruktore, metódy sú chránené a teda použiteľné len pre potomkov, okolie ich nepozná.

3.2.3 Metódy GetDomainMin a GetDomainMax

```

public:
    t_real GetDomainMin();
    t_real GetDomainMax();

```

Ak je potrebné zistiť rozsah prehl'adáwanej oblasti testovacej funkcie, evolučný algoritmus môže zavolať metódy GetDomainMin a GetDomainMax, ktoré sú verejné. Pri testovaní s mojim GA som napríklad použil tieto metódy na vykreslenie prehl'adáwanej oblasti.

3.2.4 Metóda OutOfDomain

```
public:
    bool OutOfDomain( t_byte aN, t_real *aX );
```

Metódou je možné zistiť, či je daný jedinec mimo oblasť prípustných riešení. Toto zisťovanie môže byť potrebné pri vzniku nových indivíduí, napríklad pri krížení alebo mutácii.

3.2.5 Metóda GetRandomX

```
public:
    t_real GetRandomX();
```

Metóda vráti náhodné číslo v rozsahu domény prehl'adáwanej oblasti. Táto verejná metóda sa používa napríklad pri inicializovaní počiatocnej populácie.

3.3 Odvodené triedy – testovacie funkcie

```
typedef long double t_real;
typedef unsigned char t_byte;

t_real const PI = 3.141592653589793;
t_real const EXP = 2.718281828459045;

/* cSphere class */
class cSphere : public cTestFun {
public:
    cSphere() {
        SetDomainMin( -5.12 );
        SetDomainMax( 5.12 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += aX[ii]*aX[ii];
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cRosenbrock class */
class cRosenbrock : public cTestFun {
public:
    cRosenbrock() {
        SetDomainMin( -2.048 );
        SetDomainMax( 2.048 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = aX[ii]*aX[ii]-aX[ii+1];
            t_real s2 = 1-aX[ii];
            sum += 100*s1*s1+s2*s2;
        }
    }
};
```

```

        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cThirdDeJong class */
class cThirdDeJong : public cTestFun {
public:
    cThirdDeJong() {
        SetDomainMin( -2.048 );
        SetDomainMax( 2.048 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += abs(aX[ii]);
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cFourthDeJong class */
class cFourthDeJong : public cTestFun {
public:
    cFourthDeJong() {
        SetDomainMin( -1.28 );
        SetDomainMax( 1.28 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += aN*pow(aX[ii],4);
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cRastrigin class */
class cRastrigin : public cTestFun {
public:
    cRastrigin() {
        SetDomainMin( -5.12 );
        SetDomainMax( 5.12 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += aX[ii]*aX[ii]-10*cos(2*PI*aX[ii]);
        }
        t_real rValue = 10*aN+sum;
        return( rValue );
    }
};

/* cSchwefel class */
class cSchwefel : public cTestFun {
public:
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;

```



```

        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += aX[ii]*sin(sqrt(abs(aX[ii])));
        }
        t_real rValue = -sum;
        return( rValue );
    }
};

/* cGriewangk class */
class cGriewangk : public cTestFun {
public:
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        t_real pro = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum += aX[ii]*aX[ii];
            pro += cos(aX[ii]/sqrt(ii+1.));
        }
        t_real rValue = 1+(1/4000)*sum-pro;
        return( rValue );
    }
};

/* cSchafferF6 class */
class cSchafferF6 : public cTestFun {
public:
    cSchafferF6() {
        SetDomainMin( -100 );
        SetDomainMax( 100 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real s1 = sin(sqrt(aX[0]*aX[0]+aX[1]*aX[1]));
        t_real s2 = s1*s1-.5;
        t_real s3 = (.001*(aX[0]*aX[0]+aX[1]*aX[1]))+1;
        t_real s4 = s3*s3;
        t_real rValue = s2/s4+.5;
        return( rValue );
    }
};

/* cSchafferF7 class */
class cSchafferF7 : public cTestFun {
public:
    cSchafferF7() {
        SetDomainMin( -10 );
        SetDomainMax( 10 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real s1 = aX[0]*aX[0]+aX[1]*aX[1];
        t_real s2 = pow(s1,(t_real).25);
        t_real s3 = sin(50*pow(s1,(t_real).1));
        t_real s4 = s3*s3+1;
        t_real rValue = s2*s4;
        return( rValue );
    }
};

/* cAckley1 class */
class cAckley1 : public cTestFun {
public:
    cAckley1() {
        SetDomainMin( -30 );

```

```

        SetDomainMax( 30 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        t_real s3 = pow(EXP,(t_real).2);
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = 3*(cos(2*aX[ii])+sin(2*aX[ii+1]));
            t_real s2 = sqrt(aX[ii]*aX[ii]+aX[ii+1]*aX[ii+1]);
            t_real s4 = s2/s3;
            sum += s1+s4;
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cAckley class */
class cAckley : public cTestFun {
private:
    t_real a;
    t_real b;
    t_real c;

public:
    cAckley( t_real aA = 20, t_real aB = .2, t_real aC = 2*PI ) {
        SetDomainMin( -32 );
        SetDomainMax( 32 );
        a = aA;
        b = aB;
        c = aC;
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum1 = 0;
        t_real sum2 = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            sum1 += aX[ii]*aX[ii];
            sum2 += cos(c*aX[ii]);
        }
        t_real s1 = exp(-b*sqrt(sum1/aN));
        t_real s2 = exp(sum2/aN);
        t_real rValue = a+EXP-a*s1-s2;
        return( rValue );
    }
};

/* cEggHolder class */
class cEggHolder : public cTestFun {
public:
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = sin(sqrt(abs(aX[ii]-aX[ii+1]-47)));
            t_real s2 = sin(sqrt(abs((aX[ii]/2)+aX[ii+1]+47)));
            sum += aX[ii]*s1+(aX[ii+1]+47)*s2;
        }
        t_real rValue = -sum;
        return( rValue );
    }
};

/* cRana class */
class cRana : public cTestFun {

```

```

public:
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = sqrt(abs(-aX[ii]+aX[ii+1]+1));
            t_real s2 = sqrt(abs(aX[ii]+aX[ii+1]+1));
            t_real s3 = aX[ii]*sin(s1)*cos(s2);
            t_real s4 = (aX[ii+1]+1)*cos(s1)*sin(s2);
            sum += s3+s4;
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cPathological class */
class cPathological : public cTestFun {
public:
    cPathological() {
        SetDomainMin( -100 );
        SetDomainMax( 100 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = sin(sqrt(aX[ii]*aX[ii]+100*aX[ii+1]*aX[ii+1]));
            t_real s2 = s1*s1-.5;
            t_real s3 = .001*pow(aX[ii]-aX[ii+1],4)+.5;
            sum += s2/s3;
        }
        t_real rValue = sum;
        return( rValue );
    }
};

/* cMichalewicz class */
class cMichalewicz : public cTestFun {
private:
    t_real m;

public:
    cMichalewicz( t_real aM = 10 ) {
        SetDomainMin( 0 );
        SetDomainMax( PI );
        m = aM;
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN ; ++ii ) {
            t_real s1 = sin((ii+1)*aX[ii]*aX[ii]/PI);
            sum += sin(aX[ii])*pow(s1,2*m);
        }
        t_real rValue = -sum;
        return( rValue );
    }
};

/* cMastersCosine class */
class cMastersCosine : public cTestFun {
public:
    cMastersCosine() {
        SetDomainMin( -5 );
    }
};

```

```
        SetDomainMax( 5 );
    }
    t_real GetFitness( t_byte aN, t_real *aX ) {
        t_real sum = 0;
        for( t_byte ii = 0 ; ii < aN-1 ; ++ii ) {
            t_real s1 = aX[ii]*aX[ii]+.5*aX[ii]*aX[ii+1]+aX[ii+1]*aX[ii+1];
            sum += exp(-.125*s1)*cos(4*sqrt(s1));
        }
        t_real rValue = -sum;
        return( rValue );
    }
};
```

4 PREPOJENIE KNIŽNICE S EA A JEJ POUŽITIE

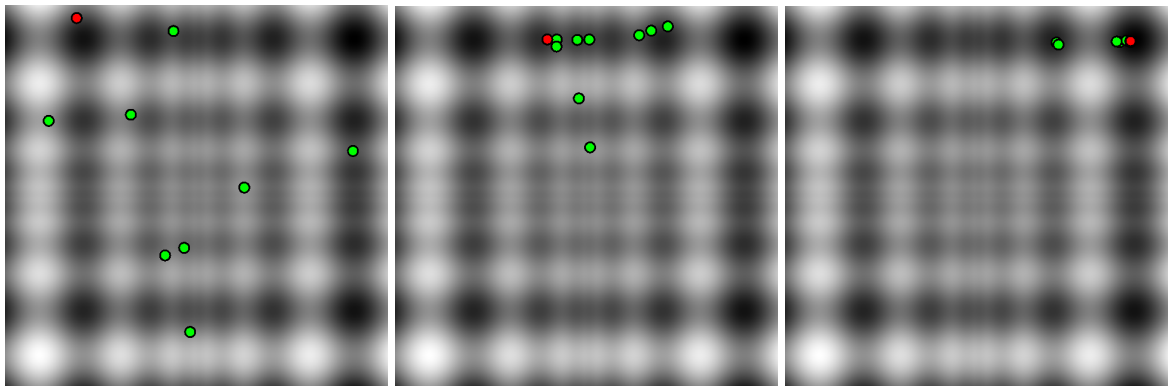
Použitie knižnice evolučnými algoritmi je jednoduché. Na otestovanie prepojitelnosti a použiteľnosti som vytvoril GA, ktorý testuje knižnicu na hľadanie minimálnej hodnoty v E_2 a priebeh graficky zobrazuje. Vizualizácia je naprogramovaná pomocou grafickej knižnice Allegro 5, ktorá patrí medzi slobodný software a má otvorený zdrojový kód.

Pri demonštrácii pracoval program s populáciou desiatich jedincov a hľadal minimum Schwefelovej testovacej funkcie pre dimenziu $D = 2$ na doméne $-512 \leq x_i \leq 512$, testovanie malo nasledujúci priebeh:

Generation: 1
Best solution: $f(-320.43, 475.69) = -346.04$

Generation: 10
Best solution: $f(-104.67, 421.14) = -494.52$

Generation: 20
Best solution: $f(412.10, 416.68) = -825.81$



Obr. 21. Zobrazenie priebehu GA (1., 10. a 20. generácia)

Teraz popíšem tie časti programu, ktoré volajú jednotlivé metódy knižnice testovacích funkcií, celý program je umiestnený v hlavnom súbore `ga.cpp`. a pomocných súboroch `parameters.h`, `fnctns.h`, `vision.h`.

Vytvoríme si inštanciu (objekt) triedy `cSchwefel`, ktorú nazveme `testFun`. S týmto objektom budeme ďalej pracovať.

```
cSchwefel testFun;
```

Pomocou metódy `GetRandomX` priradíme počiatočnej populácii náhodné hodnoty z intervalu $-512 \leq x_i \leq 512$. Tento interval má objekt `testFun` nastavený hneď po vytvorení, vďaka svojmu konštruktoru.

```

#define POP_CNT 10 // population count of individuals

enum eAXIS {
    eX1_AXIS = 0,
    eX2_AXIS = 1,
    eFITNESS = 2
};

t_real population[(POP_CNT)][3];

for( int ii = 0 ; ii < [(POP_CNT) ; ++ii ) {
    population[ii][eX1_AXIS] = testFun.GetRandomX();
    population[ii][eX2_AXIS] = testFun.GetRandomX();
}

```

Následne zavolaním metody GetFitness priradíme fitness všetkých jedincov.

```

for( int ii = 0 ; ii < [(POP_CNT) ; ++ii ) {
    population[ii][eFITNESS] = testFun.GetFitness( 2, population[ii] );
}

```

Metódu OutOfDomain využijeme počas kríženia dvoch jedincov population[a] a population[b], aby sme zaistili, že potomok tempNew bude v oblasti prípustných riešení, v opačnom prípade sa kríženie bude opakovať s inou hodnotou crossLine, ktorá je náhodne generovaná funkciou RandReal.

```

t_real RandReal( t_real aMin, t_real aMax ) {
    t_real rValue = (aMax-aMin)*((t_real)rand())/RAND_MAX+aMin;
    return rValue;
}

...

t_real crossLine;
t_real tempNew[2];
do {
    crossLine = RandReal( 0, 1 );
    tempNew[eX1_AXIS] = (0+crossLine)*population[a][eX1_AXIS]+
        (1-crossLine)*population[b][eX1_AXIS];
    crossLine = RandReal( 0, 1 );
    tempNew[eX2_AXIS] = (0+crossLine)*population[a][eX2_AXIS]+
        (1-crossLine)*population[b][eX2_AXIS];
} while( testFun.OutOfDomain( 2, tempNew) );

```

Metóda OutOfDomain sa využije podobne aj pri mutácii jedinca.

ZÁVER

Cieľom práce bolo priblížiť najznámejšie testovacie funkcie pre optimalizačné algoritmy, vyhľadať ich vzorce z rôznych zdrojov, vrátane prehľadávaného intervalu a známych hodnôt extrémov. Na základe týchto zdrojov bola vytvorená knižnica a jej funkcionálnosť bola overená na vybranom evolučnom algoritme.

V praktickej časti som opísal volanie jednotlivých metód knižnice na existujúcom evolučnom algoritme, ktorý hľadal globálne minimum Schwefelovej funkcie v dvojrozmernom priestore.

Vytvorená knižnica testovacích funkcií poskytuje širokú škálu testov, je jednoducho použiteľná a prepojitelná s evolučnými algoritmi naprogramovanými v jazyku C/C++. V prípade potreby, je možné knižnicu jednoducho rozšíriť o ďalšie užívateľské testovacie funkcie.

ZÁVER V ANGLIČTINE

The purpose of this thesis was to bring well-known test functions for optimization algorithms, find their formula from a variety of sources, including intervals and the known values of extremes. Based on these sources was created library and its functionality has been verified with the selected evolutionary algorithms.

In the practical part I described methods of the created library and connectivity to the existing evolutionary algorithm, which found global minimum on Schwef's function in two-dimensional space.

Created library of test functions provides a wide range of tests, its simple to use and connectable with evolutionary algorithms programmed in C/C++. If necessary, the library can be easily extended to other test functions by user.

ZOZNAM POUŽITEJ LITERATÚRY

- [1] ZELINKA, Ivan - OPLATKOVÁ Zuzana - ŠEDA Miloš – OŠMERA Pavel – VČELARĚ František: *Evoluční výpočetní techniky – principy a aplikace*. 1. české vydanie, Praha: BEN – technická literatura, 2008. 534 strán. ISBN 978-80-7300-218-3
- [2] *GEATbx: Example Functions*. The Generic and Evolutionary Algorithm Toolbox. [online]. 2006 [cit. 2013-05-28]. Dostupné na internete: <http://www.geatbx.com/docu/fcnindex-01.html>
- [3] HEDAR, Abdel-Rahman. *Test Functions for Unconstrained Global Optimization*. System Optimization Laboratory, Kyoto University. [online]. [cit. 2013-05-28]. Dostupné na internete: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm
- [4] BOTEV Zdravko – KROESE Dirk – LIU Jenny – NARIAI Sho – TAIMRE Thomas. *Cross-Entropy Toolbox*. University of Queensland Australia. [online]. 2004-12-17 [cit. 2013-05-28]. Dostupné na internete: <http://www.maths.uq.edu.au/CEToolBox/node3.html>
- [5] HOLTSCHULTE, Neal. *Benchmark Functions*. University of New Mexico. [online]. [cit. 2013-05-28]. Dostupné na internete: <http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction/index.html>
- [6] DIETERICH, M. Johannes – HARTKE Bernd: *Empirical review of standard benchmark functions using evolutionary global optimization*. [online]. 2012-07-18 [cit. 2013-06-04]. Dostupné na internete: <http://arxiv.org/pdf/1207.4318.pdf>
- [7] NGUYEN, Ngoc Thanh: *Computational Collective Intelligence IV*. Springer-Verlag New York Inc 2011. e-ISBN 978-3-642-21884-2
- [8] OLDENHUIS, Rody. *Many testfunctions for global optimizers*. MathWorks. [online]. 2009-02-28 (Updated 2010-02-11) [cit. 2013-06-01]. Dostupné na internete: <http://www.mathworks.com/matlabcentral/fileexchange/23147-many-testfunctions-for-global-optimizers>

- [9] Multidimensional Test Functions. Global Optimization Benchmarks and AMPGO. [online]. [cit. 2013-06-01]. Dostupné na internete: http://infinity77.net/global_optimization/test_functions.html
- [10] SEKAJ, Ivan. *Evolúcia v počítači alebo evolučné a genetické algoritmy*. Portál pre odborné publikovanie ISSN 1338-0087. [online]. 2009 [cit. 2013-05-26]. Dostupné na internete: <http://www.posterus.sk/?p=3364>

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

- EA Evolučný algoritmus, evolučné algoritmy
- GA Genetický algoritmus, genetické algoritmy
- EVT Evolučné výpočtové techniky

ZOZNAM OBRÁZKOV

Obr. 1. Bloková schéma genetického algoritmu	12
Obr. 2. Príklad jednobodového kríženia dvoch reťazcov	13
Obr. 3. Príklad mutácie celočíselného reťazca	13
Obr. 4. Sphere function (1)	17
Obr. 5. Rosenbrock's function (2)	18
Obr. 6. Third de Jong's function (3)	19
Obr. 7. Fourth de Jong's function (4)	20
Obr. 8. Rastrigin's function (6).....	21
Obr. 9. Schwefel's function (7).....	23
Obr. 10. Griewangk's function (9).....	24
Obr. 11. Schaffer's F6 function (12).....	25
Obr. 12. Sine envelope sine wave function (11).....	26
Obr. 13. Schaffer's F7 function (15).....	28
Obr. 14. Ackley I (17).....	29
Obr. 15. Ackley's function (20)	31
Obr. 16. Egg holder (21)	32
Obr. 17. Rana's function (22)	33
Obr. 18. Pathological function (25)	35
Obr. 19. Michalewicz's function (28).....	36
Obr. 20. Master's cosine wave function (30).....	38
Obr. 21. Zobrazenie priebehu GA (1., 10. a 20. generácia)	49

ZOZNAM TABULIEK

Tab. 1. Implementované testovacie funkcie.....	41
--	----

ZOZNAM PRÍLOH

P I: Elektronická príloha na CD. Obsahuje samotnú prácu vo formáte PDF, ďalej zdrojové kódy testovacej knižnice pre optimalizačné algoritmy a kódy evolučného algoritmu použitého na testovanie knižnice naprogramované v jazyku C++. Príloha obsahuje aj zdrojové kódy na vytvorenie grafov testovacích funkcií použitých v tejto práci naprogramovaných v jazyku Python.