SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Evidenčné číslo: FCHPT-113709-82048

Robotická optimalizácia

Bakalárska práca

2018 **Kristína Fedorová**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Evidenčné číslo: FCHPT-113709-82048

Robotická optimalizácia

Bakalárska práca

Študijný program: automatizácia, informatizácia a manažment v chémii a potravinárstve
Študijný odbor: 5.2.14. automatizácia, 5.2.52. priemyselné inžinierstvo
Školiace pracovisko: Ústav informatizácie, automatizácie a matematiky
Vedúci záverečnej práce: doc. Ing. Michal Kvasnica, PhD.
Konzultant: Ing. Peter Bakaráč

Bratislava 2018 **Kristína Fedorová**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2017/2018
Evidenčné číslo: FCHPT-113709-82048

STU FCHPT

ZADANIE BAKALÁRSKEJ PRÁCE

Študentka:	**Kristína Fedorová**
ID študenta:	82048
Študijný program:	automatizácia, informatizácia a manažment v chémii a potravinárstve
Kombinácia študijných odborov:	5.2.14. automatizácia, 5.2.52. priemyselné inžinierstvo
Vedúci práce:	doc. Ing. Michal Kvasnica, PhD.
Konzultant:	Ing. Peter Bakaráč

Názov práce: **Robotická optimalizácia**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je implementovať hardvérový systém na riešenie optimalizačných úloh, akými sú napr. nájdenie najteplejšieho miesta v budove za účelom rýchlej lokalizácie požiarov či zistenie zdroja kontaminácie vo vodných nádržiach. Študent v práci implementuje rôzne optimalizačné algoritmy (gradientová metóda, Newtonova metóda, Luus-Jaakola, Nelder-Mead, Extremum seeking, atď.), ktoré budú určovať robotom, kam majú ísť a merať požadované veličiny. Samotná implementácia bude realizovaná v Matlabe a experimentálne overená pomocou robotického plotra.

Rozsah práce: 30

Riešenie zadania práce od: 12. 02. 2018
Dátum odovzdania práce: 06. 05. 2018

Kristína Fedorová
študentka

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Poďakovanie

V prvom rade by som sa chcela poďakovať vedúcemu práce, doc. Ing. Michalovi Kvasnicovi, PhD. za všetky rady pri teoretickej časti tejto práce. Zároveň by som chcela poďakovať konzultantovi práce, Ing. Petrovi Bakaráčovi, za všetku pomoc pri teoretickej aj experimentálnej časti práce, za ochotu pomôcť, za všetok strávený čas pri riešení problémov, za každú rýchlu spätnú väzbu a neoceniteľné rady.

Kristína Fedorová
Bratislava, 2018

Abstrakt

Robotická optimalizácia ako taká zahŕňa využitie robotických zariadení pri riešení zadaného optimalizačného problému. Práca pojednáva o riešení problému hľadania minima bez možnosti analytického riešenia. Prvú časť tvorí teoretická časť, ktorá zahŕňa definíciu a úvod do všetkých aplikovaných numerických metód ako aj ich všeobecné algoritmy pre n-rozmerný priestor, od metód založených na náhodnom vyhľadávaní, cez metódy založené na odhade gradientu až po rozšírenie týchto metód o ohraničenia, teda prekážky v priestore. Druhú časť tvorí praktická časť práce, kde je priamo ukázaná implementácia metód na riešenie zadaného optimalizačného problému, ukazuje vhodné nastavenie parametrov pre dosiahnutie požadovaného výsledku a diskutuje o ich dôležitosti. Implementácia týchto metód bola realizovaná na robotickom zariadení 2D Plotter so senzorom svietivosti, ktorý vyhodnocoval funkčnú hodnotu bodov z kontúrového grafu, zobrazenom na tablete. Cieľom práce je oboznámenie sa s jednotlivými metódami, použitými na nájdenie minima v ohraničenom priestore, ich implementácia a nasledovné porovnanie na určenie najlepšej možnej techniky, ako aj úprava metód na zohľadnenie prekážok v priestore.

Kľúčové slová: robotická optimalizácia; Luus-Jaakola; Simulované žíhanie; Nelder-Mead; odhad gradientu; ohraničená optimalizácia; logaritmická bariéra

Abstract

Robotic optimization uses robotic devices to solve given optimization problem. The thesis deals with solving the problem of searching a minimum of a function without using any analytical solution. The first part consists of a theoretical section, which contains the definition and the introduction to all applied numerical methods. It also includes the general algorithms of these methods in n-dimensional space, starting from the methods based on iterations, through the methods based on the estimation of a gradient, to the extension of these methods by boundaries, i.e. obstacles in space. The second part, experimental section, contains an implementation of these methods to solve given problem. Further, it demonstrates an optimal set of parameters to achieve the desired result and discusses the signification of selected parameters. Discussed methods were experimentally implemented on the device 2D Plotter equipped with a photosensor. By using this photosensor, the device is able to evaluate the value of a function in a specific point. Investigated functions are displayed on a tablet as a contour graph. The aim of the thesis is to introduce, adjust and implement various methods for finding the extrema of a given function. These methods were compared to each other to determine the best performance of an extremum searching. The adaptations of the algorithms to take into account the constraints of the functions were applied as well.

Keywords: robotic optimization; Luus-Jaakola; Simulated annealing; Nelder-Mead; gradient estimation; constrained optimization; logarithmic barrier

Obsah

Úvod

Motiváciou riešenia optimalizačných problémov pomocou robotických zariadení je zefektívnenie dosiahnutia postačujúceho výsledku. V súčasnosti, v každom odvetví priemyslu preberajú ľudskú prácu roboty, pretože ich chybovosť je radikálne nižšia, práca precíznejšia a rýchlejšia. Nevýhodou robotických zariadení je neschopnosť reagovať na neočakávané situácie. V tomto má stále navrch ľudská pracovná sila. To ako sa robotické zariadenie správa pri rôznych situáciách závisí od typu implementovaného programu, podľa ktorého robot reaguje na zmenu podmienok. Základným problémom, ktorým sa zaoberá táto práca, je nájdenie miesta v priestore, ktoré sa výrazne líši svojimi fyzikálnymi vlastnosťami, akými sú teplota, hustota a tak ďalej. V reči matematiky riešime problém minimalizácie funkcie. Takýto problém je matematicky jednoducho riešiteľný analyticky, vďaka známemu gradientu funkcie. V prípade tejto práce je pre nás vývoj funkcie v priestore neznámy a teda predpis funkcie a jej derivácie pre náš prípad neexistujú. Naskytá sa otázka, ako riešiť takýto problém. Mnoho moderných optimalizačných metód si dokáže s takýmto problémom poradiť. V práci tieto metódy podrobnejšie rozoberáme a implementujeme na náš konkrétny problém. Všetky používané metódy sú metódy numerické a výsledkom je približne lokalizované minimum funkcie. Našim cieľom je teda okrem oboznámnia sa s rôznymi modernými metódami na riešenie takéhoto problému a ich aplikácie, aj využitie robotického zariadenia na čo najpresnejšiu a najrýchlejšiu detekciu extrému funkcie s neznámym predpisom, čo ovplyvňujeme rôznymi parametrami. Praktické využitie riešenia tejto problematiky spočíva v nájdení fyzikálneho extrému v priestore, kde človek nemá prístup, alebo v priestore, kde sú životu nebezpečné podmienky.

1 Robotická optimalizácia

Robotická optimalizácia je implementovanie hardvérového systému na riešenie optimalizačných úloh, akými sú napríklad nájdenie najteplejšieho miesta v budove za účelom rýchlej lokalizácie požiarov, zistenie zdroja kontaminácie vo vodných nádržiach, či lokalizácia najintenzívnejšieho úniku radiácie do okolia. Optimalizácia ako taká sa využíva pri hľadaní riešení v rôznych oblastiach priemyslu, ekonómie, v rôznych firmách či podnikoch za účelom maximalizácie zisku, či minimalizácie dopadov činnosti podniku na životné prostredie. Je založená na hľadaní optima funkcie predstavujúcej daný problém.

1.1 Základný problém

Problém, ktorým sa zaoberáme, je problém minimalizácie funkcie s ohraničeniami definovaný nasledovne:

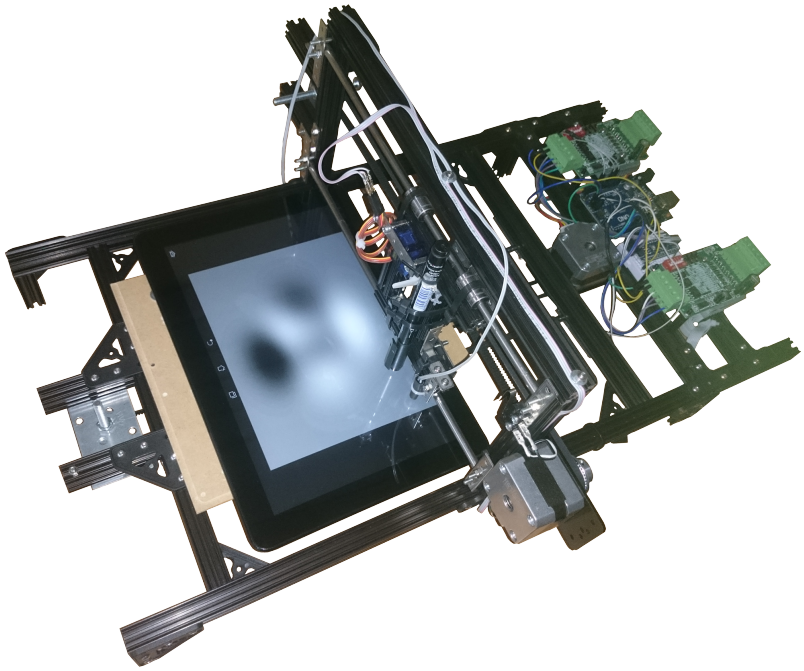
$$x^* = arg\ min\ f(x)$$

$$s.t.\ x\ \in\ \chi$$

Riešenie takéhoto problému sa dá dosiahnuť analyticky alebo numericky. Pri robotickej optimalizácii pracujeme s prípadom, že explicitne nepoznáme predpis ani derivácie skúmanej funkcie a tým pádom analytické riešenie neprichádza do úvahy. Na problém preto aplikujeme viacero numerických metód, ktoré použijeme na nájdenie možného riešenia, čiže minima, ale za rôzny čas a s rôznou presnosťou. Rozdeľujeme ich na dve základné skupiny, podľa toho či pri riešení využívame princíp gradientu – metódy odhadu gradientu, alebo pracujeme s náhodným generovaním bodov – bezgradientové metódy.

1.2 Implementácia

Vzhľadom na to, že predpis funkcie nepoznáme, pracujeme s vytvoreným farebným kontúrovým grafom funkcie na tablete, ktorý predstavuje meniace sa fyzikálne alebo chemické vlastnosti v priestore (teplota, hustota, atď.). V praxi by takýto graf mohol predstavovať výstup z termokamery. Na zisťovanie funkčných hodnôt funkcie využívame senzor svietivosti, ktorý nám v každom vygenerovanom bode vracia hodnotu merania, ktorú potom porovnávame s nasledujúcou. Ako sa menia farby (tmavnú, zosvetľujú sa), tak sa mení aj hodnota merania, ktorú vracia tento senzor pripojený na plotter (Obrázok 1). Algoritmus generuje náhodné body berúc do úvahy nami zadané parametre a ohraničenia, súradnice posiela riadiacej jednotke zariadenia, ktorá potom presúva senzor svietivosti do týchto vygenerovaných bodov a meria hodnotu svietivosti.



Obr. 1: Plotter so senzorom svietivosti

1.3 Komunikácia algoritmov s reálnym zariadením

Numerické metódy na hľadanie extrémov funkcie sú založené na vyhodnocovaní smeru konvergencie k extrému funkcie na základe funkčných hodnôt získaných v konkrétnych bodoch. To znamená, že zariadenie, na ktorom môžu byť aplikované metódy popísané v tejto práci, musí byť schopné odmerať funkčnú hodnotu funkcie v kontrétnom bode.

V rámci implementácie týchto algoritmov bolo použité zariadenie 2D Plotter [1]. Toto zariadenie je vybavené fotosenzorom, ktorý je upevnený na pohyblivej hlave. Táto hlava môže konať pohyb po ose x a po ose y. Funkciu reprezentuje kontúrový graf, zobrazený na tablete, ktorý je umiestnený pod fotosenzorom. Znamená to teda, že zariadenie je schopné pohnúť s fotosenzorom na presne určené súradnice nad tabletom a vyhodnotiť intenzitu osvetlenia z jeho obrazovky ako funkčnú hodnotu v tomto bode.

Ovládacou jednotkou tohoto zariadenia je programovateľný mikro-ovládač s procesorom ATmega328P. Ten je prepojený cez sériový port, t.j. USB kábel s počítačom. Navrhnuté numerické metódy boli naprogramované v programe MATLAB. V tomto výpočtovom programe je taktiež naprogramovaná trieda *Plotter* umožňujúca pripojenie k zariadeniu. V rámci tejto triedy sa nachádzajú viaceré metódy na komunikáciu so zariadením. Sú to metódy na získanie hodnoty intenzity osvetlenia z fotosenzora, ktorá je v rozsahu od 0

do 1023. Tento rozsah je daný 10-bitovým analógovo-digitálnym prevodníkom. Metóda $moveOn(x, y)$ slúži na poslanie príkazu do zariadenia, aby sa hlava s fotosenzorom presunula na súradnice $[x, y]$. Metóda $inMove$ vracia pravdivostnú hodnotu s informáciou, či sa hlava zariadenia pohybuje alebo nie.

Aby boli súradnice v rámci pracovnej oblasti pevne a vždy rovnako dané, metóda *goHome* presunie fotosenzor do východzej polohy. Každá metóda posunie informácie o príkaze vnútornej, privátnej metóde *send*, ktorá príkaz bajt po bajte odošle do zariadenia. Tu je znovu bajt po bajte zrekonštruovaný a aplikovaný. Dôležitými metódami sú metódy na kalibráciu počtu krokov a rozsahu funkcie $coefs$ a $setPos$. Hlava s fotosenzorom sa pohybuje po krokoch, ktoré sú tak malé a rýchle, že pohyb hlavy pôsobí ako spojitý a plynulý. Argumentom funkcie *moveOn* sú primárne tieto kroky. Ak je však známy počet krokov na jednotkový rozmer funkcie, je možné kalibráciou nastaviť, aby x a y reprezentovali súradnice v rámci funkcie samotnej.

Proces implementácie numerických algoritmov na reálnom zariadení spočíva vo viacerých krokoch. Najprv je potrebné vytvorenie inštancie triedy *Plotter* a nadviazanie komunikácie programu MATLAB so zariadením. Následne presun fotosenzora do východzej polohy zavolaním metódy goHome a kalibrácia krokov na jednotky funkcie. Algoritmus na vyhľadávanie extrémov funkcie si potom cyklicky žiada funkčné hodnoty funkcie v konkrétnych bodoch. To znamená, že zavolaním metódy $moveOn(x, y)$ sa fotosenzor presunie na požadované súradnice. Medzi zadaním pohybu a odmeraním intenzity osvetlenia sa algoritmus opakovane dožaduje informácie, či je hlava v pohybe. Až keď sa hlava zastaví, potom sa odmeria funkčná hodnota v danom bode a numerický algoritmus na základe tejto hodnoty vyhodnotí ďalšie súradnice, na ktoré sa má presunúť hlava s fotosenzorom. Výsledkom je konvergencia pohybu fotosenzora do extrému funkcie.

1.4 Testovacie funkcie

Pri riešení nášho problému sme pracovali so spomínaným kontúrovým grafom funkcie, predstavujúcim vývoj určitej fyzikálnej veličiny v priestore. Na otestovanie funkčnosti algoritmov sme použili viacero grafov funkcií, ktoré sa používajú na overovanie riešenia optimalizačných problémov. Použili sme napríklad Rosenbrockovú funkciu, Goldstein-Priceovú funkciu, Boothovú funkciu a Styblinski-Tangovú funkciu. Poslednú menovanú sme používali najčastejšie a poslúžila nám na ilustráciu presnosti našich algoritmov a závislosti nájdeného minima od počiatočného bodu. Preto je táto funkcia bližšie popísaná v nasledujúcej podkapitole.

1.4.1 Styblinski-Tangová funkcia

Pri aplikácii navrhutých algoritmov sme pracovali hlavne s kontúrovým grafom Styblinski–Tangovej funkcie (Obrázok 2). Táto funkcia je viacúčelová a má $2n$ lokálnych miním, kde n je počet dimenzií. Styblinski–Tangova funkcia je definovaná nasledovne:

$$f(x) = \frac{1}{2}\sum_{i=0}^{n-1} x_i^4 - 16x_i^2 + 5x_i$$

Doménou vyhľadávania extrémov tejto funkcie je interval $x \in \langle -5, 5\rangle$. V tejto oblasti nájdeme jedno globálne minimum, v ktorom je hodnota funkcie

$$f(x^*) = -39,16616570377016 \cdot n,$$

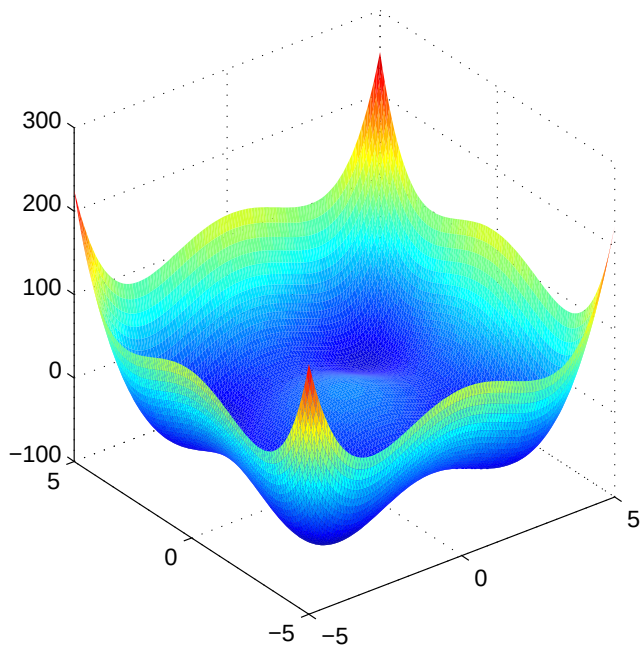
kde

$$x_i^* = -2,90353375790172752$$

pre $i = 1, 2, ...$ pre všetky dimenzie. V našom prípade používame predpis:

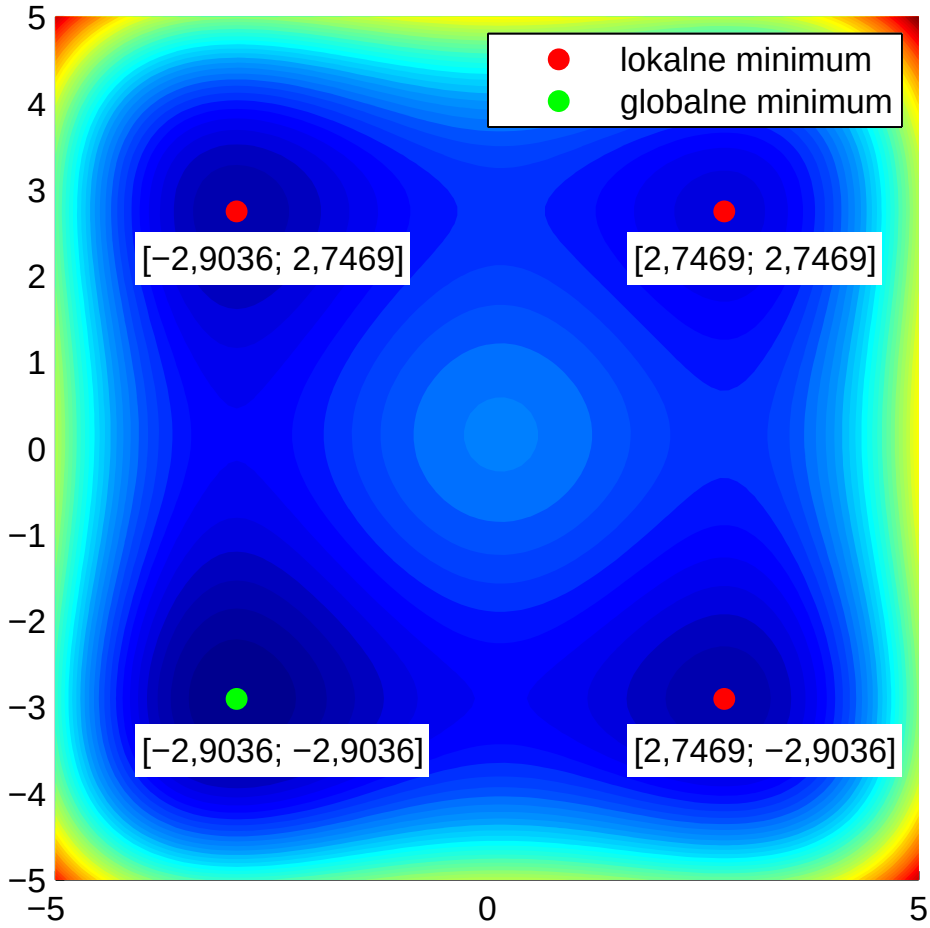
$$f(x) = \frac{(x_1^4 - 16x_1^2 + 5x_1) + (x_2^4 - 16x_2^2 + 5x_2)}{2}$$

Táto funkcia sa často využíva pri demonštrácii optimalizačných algoritmov, pretože má okrem jedného globálneho minima aj ďalšie tri lokálne minimá.[2]



Obr. 2: Styblinski–Tangova funkcia

Naším cieľom je nájsť minimum tejto funkcie pri použití iteračných alebo gradientových metód. Keďže táto funkcia má štyri minimá (Obrázok 3), z toho tri sú lokálne, nájdenie globálneho minima závisí nielen od použitej metódy, ale aj od počiatočného bodu. Pri každej z metód si zhodnotíme výsledky optimalizačného algoritmu.



Obr. 3: Minimá Styblinski-Tangovej funkcie

2 Bezgradientové metódy

Bezgradientové metódy sú metódy založené na hľadaní optima funkcie f, keď derivácia funkcie f je nedostupná, nespoľahlivá alebo zložitá na získanie a teda pracujú s náhodným vyhľadávaním a s iteráciami. Takýto typ techník zaraďujeme k stochastickým metódam. Náhodné vyhľadávanie je metóda numerickej optimalizácie, ktorá nevyžaduje gradient problému na nájdenie extrému a taktiež môže byť použitá na funkcie, ktoré nie sú spojite diferencovateľné. Náhodné vyhľadávanie pracuje iteračne na princípe posúvania sa do lepších bodov v priestore, ktorý je vytvorený ako „hypersféra“ obkolesujúca súčasnú pozíciu.

Náhodné vyhľadávanie teda funguje na iteráciach, ktoré program náhodne generuje na základe kľúča. Porovnávaním funkčných hodnôt v bodoch súčasnej x_k a predošlej x_{k-1} iterácie, pomocou algoritmu vyhodnotíme, či je ďalší vygenerovaný bod lepší (má menšiu funkčnú hodnotu) ako predošlý bod. Postupným zmenšovaním rozsahu generovania nových bodov, sa zvyšuje pravdepodobnosť nájdenia lokálneho minima funkcie. K takýmto technikám radíme metódy Luus-Jaakola a Simulované žíhanie, ktoré sme reálne implementovali a aplikovali na riešenie zadaného problému.[3]

2.1 Metóda Luus-Jaakola

Metóda Luus-Jaakola patrí medzi stochastické metódy hľadania extrému funkcie. Je to iteračná metóda generujúca sekvenciu bodov, ktorá konverguje do optimálneho riešenia, ak také existuje. Luus-Jaakola predstavuje heuristiku pre globálnu optimalizáciu. Táto heuristika nevyužíva gradient, čo robí túto metódu vhodnou na riešenie nediferencovateľných a nekonvexných problémov. Podľa pánov Luusa a Jaakolu, táto metóda generuje sekvenciu iterácii, kde nasledujúca iterácia je vybraná z okolia súčasnej pozície použitím rovnomerného rozdelenia. Každou iteráciou sa okolie zužuje a to zabezpečuje, že iterácie konvergujú do minima.

2.1.1 Algoritmus metódy

Majme funkciu $f : \mathbb{R}^n \to \mathbb{R}$, ktorú chceme minimalizovať. Nech $x \in \mathbb{R}^n$ predstavuje pozíciu alebo kandidáta na riešenie v priestore hľadania minima. Metóda Luus-Jaakola iteruje podľa nasledovných krokov:

- inicializácia $x_0 \in \langle b_L, b_U \rangle$ na jednoznačnú pozíciu v priestore, kde b_L a b_U sú dolné a horné ohraničenie

- nastavenie veľkosti okolia, v ktorom sa generujú náhodné body, aby pokrylo celý priestor, v ktorom hľadáme minimum:
  $$d = b_U - b_L$$
- pokiaľ nie je splnené stopovacie kritérium
  $$|f(x_0) - f(x)| < \varepsilon \ (\varepsilon = 10^{-3}),$$
  opakujujú sa nasledovné kroky:
  - náhodne sa vygeneruje x, v zadanom okolí
  - ak $f(x) < f(x_0)$, nastáva presun na novú pozíciu nastavením $x_0 = x$, v opačnom prípade dochádza k zmenšeniu okolia, v ktorom sa generujú body:
    $$d = \alpha \cdot d \ (\alpha = 0,95)$$
- na konci celej procedúry predstavuje x_0 najlepšiu nájdenú pozíciu (v konečnom dôsledku lokálne minimum funkcie) [4]

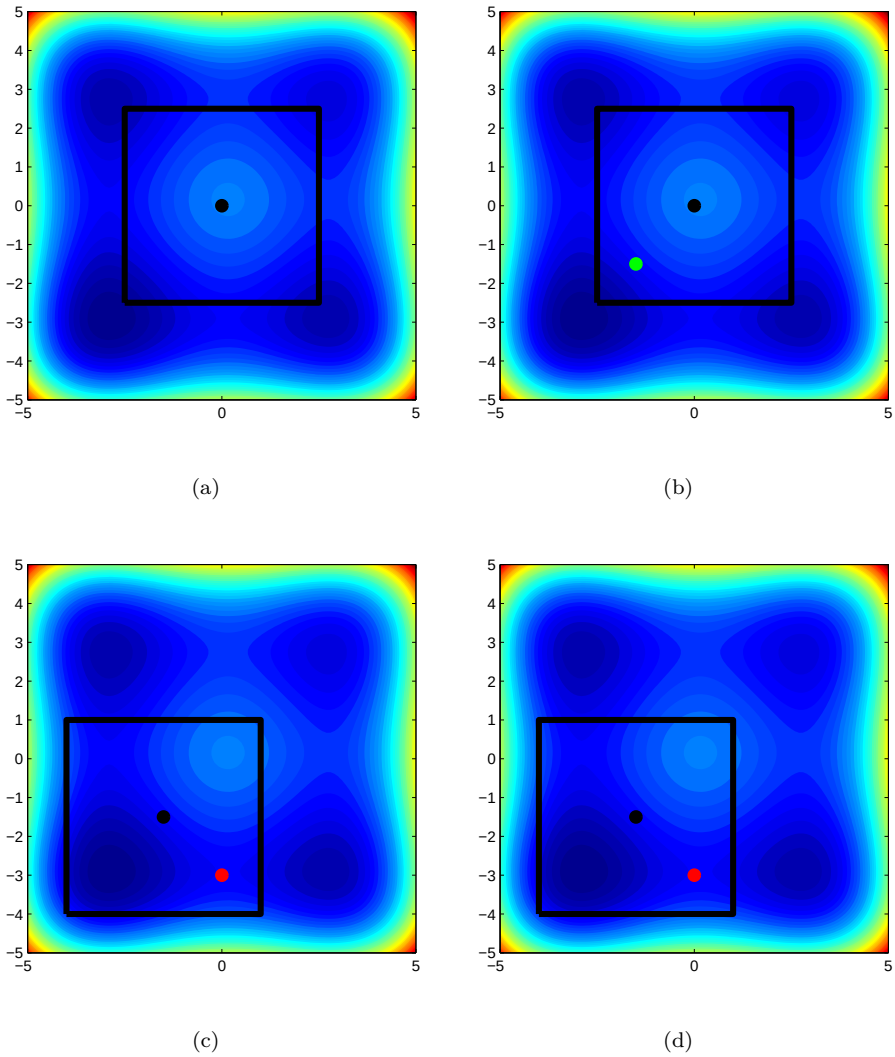
2.1.2 Parametre metódy

Každá technika pracuje s parametrami, ktoré musia byť vhodne zvolené pre riešený problém. Metóda Luus-Jaakola obsahuje dva parametre, ktoré môžeme voliť a tým spresniť polohu minima alebo skrátiť čas jeho nájdenia. Prvým parametrom je koeficient α určujúci mieru stiahnutia priestoru, v ktorom sa generuje bod pre ďalšiu iteráciu. Hovorí nám o tom, ako rýchlo sa bude priestor na generovanie nových bodov zmenšovať. Musí sa voliť opatrne, pretože veľmi veľká hodnota spôsobí nárast počtu iterácii a tým pádom nárast času nájdenia minima. Ak sa naopak zvolí veľmi malá hodnota parametra α, môže dôjsť k veľmi rýchlemu skončeniu programu bez nájdenia extrému funkcie. Druhým parametrom je tolerancia ε, zahrnutá v stopovacom kritériu algoritmu. Počas behu algoritmu sledovaný bod postupne konverguje do minima a v každej iterácii sa porovnávajú funkčné hodnoty. Hodnota parametra tolerancie určuje najväčší rozdiel funkčných hodnôt, pri ktorom je nový bod akceptovaný ako lokálne minimum. Opatrnosť voľby je znova taká istá ako pri predošlom parametri. Musíme voliť efektívne a bilancovať rýchlosť a presnosť nájdenia extrému funkcie.

2.1.3 Implementácia metódy

Na začiatku programu sme inicializovali hodnoty parametrov, ktoré sme vhodne zvolili po predošlých pokusoch. Priestor, v ktorom sa generujú nové body sme zmenšovali na 95% pôvodnej veľkosti a toleranciu sme zistili skúšaním behu programu na zariadení. Generovanie bodov beží jednoduchým algoritmom vzhľadom na ohraničenia funkcie, ale aj

vzhľadom na ohraničenie priestoru v okolí súčasnej pozície. Implementácia je jednoducho ukázaná na obrázku 4.



(a) (b)

(c) (d)

Obr. 4: Implementácia metódy Luus-Jaakola: (a) nami zadaný bod a okolie, v ktorom sa generuje nasledovná pozícia, (b) vygenerovanie nového bodu a jeho prijatie (zelený bod) za stred okolia z dôvodu nižšej funkčnej hodnoty v danom mieste, (c) vygenerovanie nového bodu a nasledovne jeho zamietnutie (červený bod) z dôvodu vyššej funkčnej hodnoty v danom bode, (d) stiahnutie okolia smerom k stredu z dôvodu zamietnutia prijatia vygenerovaného bodu za stred.

2.1.4 Výsledky aplikácie metódy Luus-Jaakola

Ako sme už spomínali, metóda Luus-Jaakola pracuje s náhodným generovaním bodov, preto sme predpokladali, že je vyššia pravdepodobnosť konvergencie do globálneho minima ako pri použití gradientových metód. Samozrejme, takýto výsledok sa prejaví na čase trvania algoritmu. Výsledky aplikácie tejto metódy na reálnom zariadení pri rôznych počiatočných bodoch sú prezentované v nasledujúcej tabuľke.

Tabuľka 1: Výsledky implementácie metódy Luus-Jaakola

Počiatočný bod[1]	**Čas [s]**	**Iterácie**	**Nájdené minimum**[1]	**Vzdialenosť**[2]	**Typ minima**
[0; 0]	72,6	68	[-3,0953; -2,9779]	0,2056	globálne
[0; 0]	80,5	65	[-3,3221; -3,0198]	0,4343	globálne
[0; 0]	35,6	25	[2,8153; 2,9370]	0,2021	lokálne
[1,5; 1,5]	65,8	30	[2,7177; -2,4956]	0,4090	lokálne
[1,5; 1,5]	70,0	49	[2,8578; -2,7771]	0,1683	lokálne
[1,5; 1,5]	65,4	68	[2,8896; 2,8914]	0,2032	lokálne
[-1,5; -1,5]	52,3	28	[-3,4744; -3,2165]	0,6509	globálne
[-1,5; -1,5]	33,2	13	[-3,0983; -3,4760]	0,6046	globálne
[-1,5; -1,5]	40,2	16	[-2,5893; -2,4358]	0,5636	globálne

Z týchto výsledkov je zrejmé, že takýto algoritmus trvá priemerne približne jednu minútu. Samozrejme, čas sa tým predlžuje, čím väčší priestor prehľadávame, pretože robotovi trvá, kým sa presunie na vygenerovanú pozíciu. Ak si v tabuľke porovnáme vzdialenosť pozície identifikovaného minima programom a reálnej pozície minima, môžeme presnosť metódy považovať za uspokojivú. Vzhľadom na danú vzorku experimentov by sa mohlo zdať, že nájdenie globálneho minima závisí aj od počiatočného bodu. Táto závislosť však nebola ďalšími experimentami potvrdená. Touto vzorkou sme chceli poukázať hlavne na časovú oblasť a pozíciu nájdeného minima v porovnaní so skutočnou pozíciou.

2.2 Simulované žíhanie

Simulované žíhanie je pravdepodobnostná technika na dosiahnutie globálneho optima danej funkcie f. Považuje sa taktiež za heuristiku aproximácie optima. Často sa táto metóda používa pri vyhľadávaní v prostredí, ktoré je diskrétne (napr. problém obchodného cestujúceho). Pre problémy, kde nájdenie globálneho optima je dôležitejšie ako nájdenie presného lokálneho minima v danom čase, môže byť táto technika perfektnou alternatívou ku gradientovým metódam.

[1] Bod reprezentovaný ako $[a;b]$, kde a je x-ová a b y-ová súradnica

[2] Vzdialenosť nájdeného minima n a najbližšieho reálneho minima r počítaná ako $\|n - r\|$

Názov a inšpirácia pochádzajú od skutočného žíhania v priemysle. Táto technika zahŕňa ohrievanie a kontrolované chladenie materiálu na zvýšenie počtu kryštálov a redukciu ich defektov. Obidva sú atribútmi materiálu, ktoré záležia na termodynamickej energii. Simulácia žíhania môže byť použitá na nájdenie globálneho minima pre funkciu s vysokým počtom premenných. [5]

2.2.1 Algoritmus metódy

Predstava pomalého chladenia implementovaná v algoritme simulovaného žíhania je interpretovaná ako pomalé znižovanie pravdepodobnosti akceptácie horšieho riešenia ako aj priestor pre nájdenie riešenia. Akceptovanie horšieho riešenia je základná vlastnosť heuristiky, pretože to dovoľuje rozsiahlejšie vyhľadávanie globálneho optima. Vo všeobecnosti, algoritmus Simulovaného žíhania pracuje nasledovne.

V každom kroku algoritmus náhodne vyberá riešenie v blízkosti súčasného, zmeria jeho kvalitu a potom sa rozhodne, či sa doň presunie alebo ostane v súčasnom riešení. Toto rozhodnutie je založené na pravdepodobnosti výpočítanej na základe funkčnej hodnoty v bode merania a hodnoty teplotnej funkcie v súčasnej iterácii. Počas vyhľadávania sa teplota progresívne znižuje od pôvodnej pozitívnej hodnoty k nule a tým ovplyvňuje pravdepodobnosť. S meniacou sa pravdepodobnosťou sa mení aj spôsob akceptácie generovaných bodov za novú vhodnú pozíciu. To znamená, že na základe pravdepodobnostnej funkcie sa na začiatku algoritmu príjmu aj body s vyššou funkčnou hodnotou za vhodné, ale postupne sa pravdepodobnosť ich prijatia zmenšuje, vďaka čomu sa algoritmus nezasekne v lokálnom minime a postupnými krokmi konverguje do extrému funkcie.

2.2.2 Pravdepodobnosť akceptácie riešenia

Pravdepodobnosť presunu zo súčasného stavu s do nového stavu s' je špecifikovaná akceptačnou pravdepodobnostnou funkciou $P(e, e', T)$, ktorá záleží na energiách obidvoch stavov

$$E = f(e)\ a\ E' = f(e')$$

a na teplote T. Stavy s nižšou energiou sú lepšie ako tie s energiou vyššou. Pravdepodobnostná funkcia P musí byť pozitívna, ak e' je väčšie ako e. Táto podmienka je prevenciou, aby metóda nezostala zaseknutá v lokálnom minime, ktoré je horšie ako globálne minimum. Ak sa T blíži k nule

$$\mathrm{T} \rightarrow 0,$$

pravdepodobnosť sa musí blížiť k nule

$$\mathrm{P(e,\ e',\ T)} \rightarrow 0,$$

ak platí, že

$$e' > e,$$

a k pozitívnym hodnotám v opačnom prípade. V originálnom popise simulovaného žíhania je pravdepodobnosť rovná jednej ak

$$e' < e \;\Rightarrow\; P(e, e', T) = 1,$$

čiže systém bude preferovať pohyby, ktoré budú smerovať do miest s nižšou energiou bez ohľadu na teplotu. Veľa popisov a implementácií, ako aj tá naša využíva túto podmienku ako súčasť definície simulovaného žíhania. Napriek tomu, táto podmienka nie je nutná, aby metóda správne pracovala. Pravdepodobnostná funkcia sa zvyčajne vyberá tak, aby pravdepodobnosť akceptácie nového stavu klesala, keď rozdiel $e' - e$ narastá. Ani táto podmienka však nie je striktne potrebná. V konečnom dôsledku teplota T hrá dôležitú rolu v kontrolovaní vývoja stavu s systému s ohľadom na jeho citlivosť na rôzne energie systému. [6]

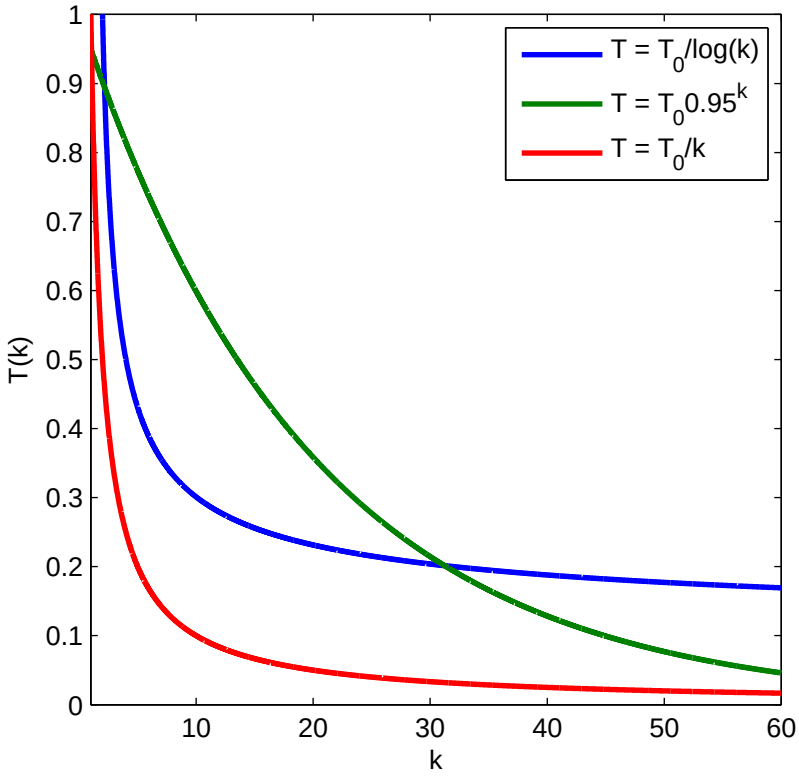
V prípade našej praktickej aplikácie simulovaného žíhania, definujeme pravdepodobnostnú funkciu nasledovne:

$$P = e^{\frac{-f(e')-f(e)}{T}},$$

ktorá závisí od funčných hodnôt nameraných v súčasnom a predošlom stave, čo značí, že zo začiatku pravdepodobnostná funkcia príjme aj bod s horšou funkčnou hodnotou za lepší, ale v priebehu programu, kedy teplota klesá, klesá aj pravdepodobnosť prijatia bodu s horšou funkčnou hodnotou. Pokles teploty T vzhľadom na počet iterácii k v našom programe je znázornený na obrázku 5 modrou farbou.

2.2.3 Parametre metódy

Pri Simulovanom žíhaní, tak ako aj pri predošlej metóde, musíme opatrne pracovať s parametrami. Prvým parametrom je pravdepodobnostná funkcia, ktorej priebeh si ľubovoľne môžeme voliť, ale musíme pritom myslieť na to, ako to ovplyvní hodnotu výsledku. To platí aj pre parameter teploty. Keďže teplota je pri žíhaní dôležitým parametrom, lebo aj v reálnom procese od nej závisí pohyb častíc, je dôležité voliť rýchlosť jej klesania vhodným spôsobom. Na obrázku 5 môžeme vidieť viacero typov závislosti teploty od počtu iterácii. Oba tieto parametre, ale menia iba kvalitu získavaného minima. Pri Simulovanom žíhaní je stopovacím kritériom počet iterácii, čiže rýchlosť nájdenia optima môžeme meniť iba týmto parametrom. Veľa iterácii, predlžuje beh programu, ale môže navýšiť presnosť vyhľadávania a málo krokov naopak skracuje čas behu programu, ale strácame kvalitu výsledku. [7]



Obr. 5: Rôzne funkcie závislosti teploty T od počtu iterácii k

2.2.4 Implementácia metódy

Simulované žíhanie je veľmi zaujímavé na implementáciu, pretože je dôležité voliť vhodné generovanie bodov. Použili sme na to algoritmus, kde nasledujúci bod x_{k+1} generujeme vzhľadom na predošlý bod x_k a jeho okolie definované dolným x_L a horným x_U ohraničením ako

$$x_L = x_k - d \cdot 0,95^k$$

$$x_U = x_k + d \cdot 0,95^k,$$

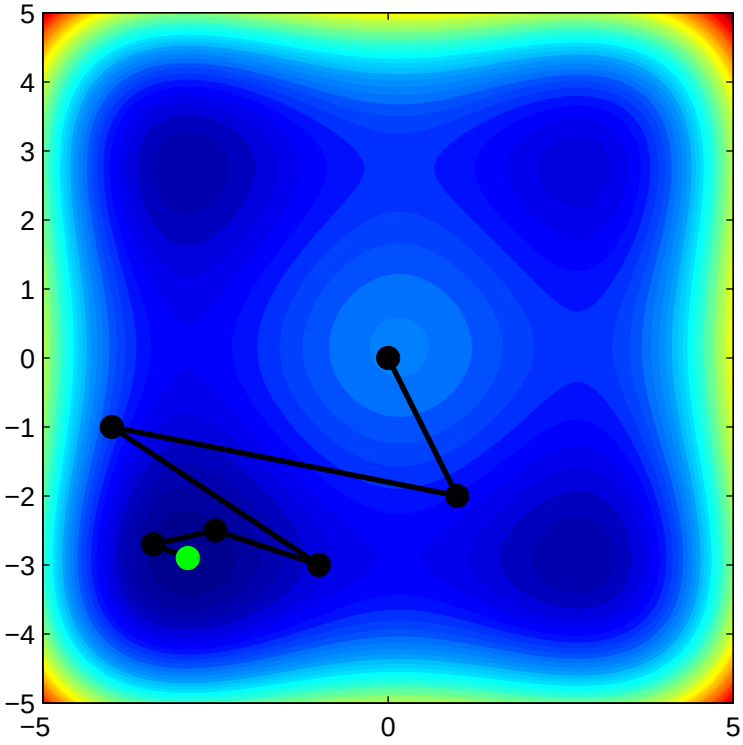
kde d je šírka domény, v rámci ktorej je funkcia prehľadávaná, v našom prípade $d = 10$, lebo sa pohybujeme v intervale $\langle -5, 5 \rangle$ a parameter umocnený na hodnotu súčasnej iterácie volíme podľa očakávanej rýchlosti zmenšovania okolia generovania. Vidíme, že tieto ohraničenia závisia od iterácie, v ktorej sa nachádzame. Behom programu sa teda zmenšuje priestor okolo súčasnej pozície na generovanie nasledovného kroku, čo súhlasí s

reálnym procesom. Nový bod sa generuje nasledovne:

$$x_{k+1} = |x_U - x_L| \cdot rand() + x_L,$$

kde funkcia $rand()$ vracia hodnoty v intervale $\langle 0, 1 \rangle$, čím dosiahneme udržanie bodu vo vypočítanom okolí.

Zároveň používame teplotnú funkciu znázornenú na obrázku 5. Prvým nastaveným parametrom je počet iterácii tejto techniky. Najvhodnejším počtom je číslo 60, pretože pri takomto počte iterácií sme dosiahli najpresnejšiu lokalizáciu minima. Po vygenerovaní nového bodu sa zistí jeho funkčná hodnota a tá sa potom vzhľadom na pravdepodobnosť porovnáva s náhodne vygenerovaným číslom. Na obrázku 6 môžeme vidieť implementáciu simulovaného žíhania.



Obr. 6: Implementácia metódy Simulovaného žíhania: Body na grafe predstavujú body prijaté vzhľadom na prepočítanú pravdepodobnosť a postupnosť iterácii až po dosiahnutie minima (zelený bod).

2.2.5 Výsledky aplikácie metódy simulovaného žíhania

Metóda simulovaného žíhania je ďalšia z ciest náhodného vyhľadávania, kde predpokladáme porovnateľne dlhý čas behu algoritmu ako pri predošlej metóde. Výsledky experimentov sú zhrnuté v nasledovnej tabuľke.

Tabuľka 2: Výsledky implementácie metódy simulovaného žíhania

Počiatočný bod	**Čas [s]**	**Iterácie**	**Nájdené minimum**	**Vzdialenosť**	**Typ minima**
[0; 0]	102,5	60	[-2,8034; -2,8255]	0,1270	globálne
[0; 0]	117,1	60	[-2,8473; 3,0558]	0,3140	lokálne
[0; 0]	103,1	60	[2,8310; -2,6574]	0,2602	lokálne
[1,5; 1,5]	125,2	60	[3,0131; 3,2796]	0,5956	lokálne
[1,5; 1,5]	123,4	60	[-2,9746; -3,2051]	0,3097	globálne
[1,5; 1,5]	118,2	60	[-3,1783; 3,1353]	0,4757	lokálne
[-1,5; -1,5]	100,9	60	[-2,5197; -3,1589]	0,4610	globálne
[-1,5; -1,5]	111,4	60	[-2,3950; -2,7013]	0,5474	globálne
[-1,5; -1,5]	105,5	60	[-2,9412; -2,6790]	0,2277	globálne

Prvým zaujímavým faktom je dlhý čas trvania programu. Je to spôsobené tým, že počet iterácii je pevne daný a radíme ho k parametrom tejto metódy, ktorého vhodný počet sme nastavili pomocou experimentov. Zároveň je dobre vidieť, že nezávisí od voľby počiatočného bodu, pretože v prípade $x_0 = [1,5; 1,5]$ sa nachádzajú dve lokálne minimá bližšie, ale vďaka náhodnému generovaniu bodov, sme pri tejto vzorke dosiahli vzdialenejšie globálne minimum. Presnosť pre reálne zariadenie považujeme za vyhovujúcu.

3 Metódy odhadu gradientu

Metódy, ktorým sa budeme venovať v tejto kapitole, sú metódami, ktoré sa snažia aproximovať skutočný gradient určitými matematickými operáciami. Gradient je definovaný ako stupeň sklonu, alebo miera stúpania či klesania funkcie. Poznáme viacero techník, ktoré sa snažia aproximovať gradient viac, či menej presne. Jednou z najznámejších je technika Nelder-Mead, ktorej sa budeme podrobnejšie venovať. Všeobecne sú tieto metódy založené na porovnávaní funkčných hodnôt bodov v n-rozmernom priestore a vzhľadom na túto informáciu následné vytváranie vektora, reprezentujúceho odhad gradientu. V prípade, že algoritmus pozná smer najväčšieho rastu alebo klesania funkcie, vie určiť smer pre ďalšiu iteráciu pri hľadaní extrému. Okrem metódy Nelder-Mead sme na hľadanie optima v dvojrozmernom priestore využili aj odhad gradientu pomocou n-uholníkov.

3.1 Metóda Nelder-Mead

Metóda Nelder-Mead, nazývaná aj simplexová metóda, je často aplikovaná numerická metóda, používaná na nájdenie optima funkcie v multidimenziálnom priestore. Procedúra sa aplikuje na nelineárne optimalizačné problémy, ktorých derivácie nemusia byť známe. Technika Nelder-Mead bola predstavená Johnom Nelderom a Rogerom Meadom v roku 1965. Táto metóda používa koncept simplexu, ktorým je špeciálny mnohosten o $n + 1$ vrcholoch v n dimenziách. Ak uvažujeme dvojdimenziálny priestor, tak je týmto mnohostenom trojuholník, v prípade trojrozmerného hovoríme o štvorstene. Technika Nelder-Mead aproximuje lokálne optimum problému s n premennými, keď sa funkcia mení plynule.

Základný princíp tejto techniky spočíva v tom, že metóda sa v n dimenziách skladá zo setu $n + 1$ testovaných bodov vytvárajúcich simplex. To potom extrapoluje správanie sa funkcie meranej v každom z testovaných bodov, aby sa našiel nový testovací bod a nahradili sa staré testovacie body novými a takto algoritmus ďalej pokračuje. Najjednoduchší prístup je nahradenie najhoršieho bodu bodom reflektovaným cez stred trojuholníka. Ak je tento bod lepší ako najlepší z doterajších bodov, potom sa môžeme pokúsiť posunúť tento bod po priamke smerom od pôvodného trojuholníka. Na druhej strane, ak nový bod nie je oveľa lepší ako predošlý, potom prechádzame cez „údolie“ funkcie, čiže zúžime simplex smerom k lepšiemu bodu. Dnes už poznáme mnoho vylepšení tejto techniky. Najčastejšie používanou variantou je tá, ktorá používa malý simplex o konštantnej veľkosti, ktorý hrubo kopíruje gradientový smer. Najjednoduchšou predstavou je malý trojuholník na mape presúvajúci sa dole údolím do lokálneho dna. [8]

3.1.1 Všeobecný algoritmus metódy Nelder-Mead

Minimalizujeme funkciu $f(x)$, kde $x \in \mathbb{R}^n$. Súčasné testovacie body sú $x_1, ..., x_{n+1}$:

- vytvorenie poradia funkčných hodnôt vo vrcholoch simplexu:
  $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$
- vypočítanie x_0, stredu všetkých bodov
- reflexia (Obrázok 7): vyrátanie reflektovaného bodu
  $x_r = x_0 + \alpha(x_0 - x_{n+1}),$
  pričom $\alpha > 0$, ak je reflektovaný bod lepší ako druhý najhorší, ale nie lepší ako najlepší bod, platí:
  $f(x_1) \leq f(x_r) < f(x_n),$
  potom sa získava nový simplex nahradením najhoršieho bodu x_{n+1} s reflektovaným bodom x_r a dochádza k návratu do prvého bodu
- expanzia (Obrázok 7): ak je reflektovaný bod doteraz najlepším bodom
  $f(x_r) < f(x_1),$
  vypočítava sa expandovaný bod
  $x_e = x_0 + \gamma(x_r - x_0),$
  pričom $\gamma > 1$. Ak je expandovaný bod lepší ako reflektovaný
  $f(x_e) < f(x_r),$
  potom sa získava nový simplex nahradením najhoršieho bodu x_{n+1} s expandovaným bodom x_e a dochádza k návratu do prvého bodu, v opačnom prípade sa vytvára nový simplex nahradením najhoršieho bodu x_{n+1} s reflektovaným bodom x_r a potom dochádza k návratu do prvého bodu
- kontrakcia (Obrázok 8): ak je reflektovaný bod horším bodom
  $f(x_n) < f(x_r),$
  vypočítava sa bod kontrakcie
  $x_c = x_0 + \rho(x_{n+1} - x_0),$
  kde $0 < \rho \leq 0,5$. Ak je bod kontrakcie lepší ako najhorší bod
  $f(x_c) < f(x_{n+1}),$
  získava sa nový simplex nahradením najhoršieho bodu x_{n+1} bodom kontrakcie x_c a dochádza k návratu do prvého bodu
- zmrštenie (Obrázok 8): v prípade, že nie je splnená žiadna z podmienok, všetky body

okrem najlepšieho x_1 sa nahradzujú podľa rovnice

$$x_i = x_1 + \sigma(x_i - x_1)$$

a nastáva návrat do prvého bodu [9]

V prípade reflexie vieme, že x_{n+1} je bod s najväčšou funkčnou hodnotou, takže môžeme očakávať nájdenie nižšej funkčnej hodnoty reflexiou tohto bodu cez stred simplexu. Ak uvažujeme o expanzii a ak bod reflexie x_r je novým minimom vzhľadom na ostatné vrcholy simplexu, môžeme nájsť lepšie hodnoty v smere od x_0 ku x_r. Pokiaľ ide o kontrakciu, ak

$$f(x_r) > f(x_n),$$

môžeme očakávať, že lepšia hodnota bude vo vnútri simplexu vytvoreného z vrcholov x_i. Nakoniec, zmrštenie sa zaoberá zriedkavým prípadom, že sa trojuholník stiahne smerom k bodu s najnižšou hodnotou v očakávaní nájdenia jednoduchšieho priestoru na spracovanie.

3.1.2 Parametre metódy

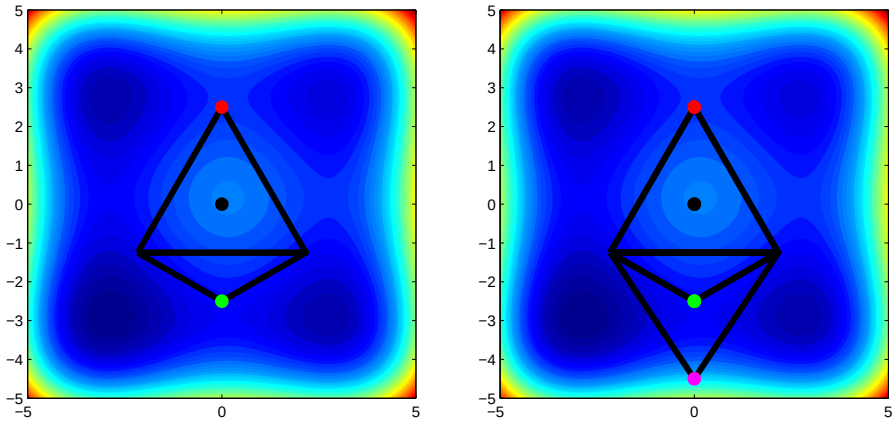
α, β, γ, σ sú postupne koeficienty reflexie, expanzie, kontrakcie a zmrštenia. Štandardné hodnoty sú $\alpha = 1$, $\beta = 2$, $\gamma = \frac{1}{2}$ a $\sigma = \frac{1}{2}$. Tieto koeficienty predstavujú parametre techniky Nelder-Mead, ktoré meníme ľubovoľne podľa riešeného problému. Pri vypracovaní algoritmu k tejto práci sme začínali so štandardnými hodnotami týchto koeficientov a potom, ich postupným obmieňaním sme sa dopracovali k hodnotám pre riešený problém. V neposlednom rade, je to počiatočná veľkosť simplexu, ktorej voľbu si rozoberieme neskôr. Posledným parametrom je parameter tolerancie ε, ktorý má tú istú funkciu ako to bolo pri metóde Luus-Jaakola.

Voľba počiatočného simplexu je veľmi dôležitá. V skutočnosti príliš malý simplex môže viesť k lokálnemu vyhľadávaniu, teda Nelder-Mead algoritmus môže ľahšie uviaznuť v lokálnom optime. Takže voľba počiatočného simplexu by mala záležať na povahe problému. V našom prípade generujeme simplex vzhľadom na zadaný stred, pričom vzdialenosť medzi vrcholmi je fixne daná. Všeobecne je preferovaný presne takýto simplex.

Nelder-Mead používa štandardné odchýlky vzorky funkčných hodnôt súčasného simplexu. Ak odchýlka klesne pod určitú toleranciu, cyklus sa zastaví a bod s najmenšou hodnotou v simplexe prislúcha optimu funkcie.

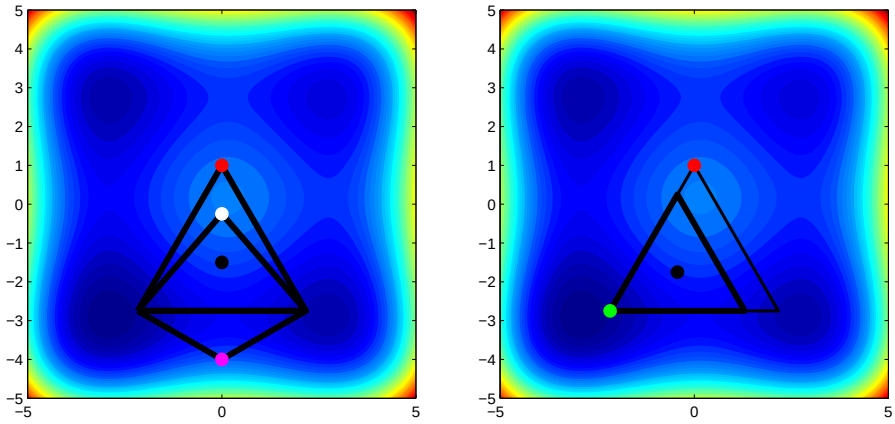
3.1.3 Implementácia algoritmu

Prvým problémom, ktorý sme museli vyriešiť, bol spôsob generovania pôvodného simplexu. Keďže sa jedná o trojuholník, zvolili sme výpočet jeho vrcholov cez kružnicu, teda sme vytvorili rovnostranný trojuholník. Pri prvom behu programu sme parametre tejto metódy nastavili na štandardné hodnoty (uvedené vyššie) a až na jeden sa ukázali byť efektívne. Parameter zmršťovania sme museli zväčšiť oproti pôvodnému z dôvodu, že trojuholník sa rýchlo zmenšoval a tým pádom nebol schopný skonvergovať do minima. Samozrejme, v rámci aplikácie tejto metódy na reálnom zariadení musel byť parameter tolerancie *epsilon* experimentálne nastavený, vzhľadom na presnosť meraných hodnôt zo senzora.



(a) (b)

Obr. 7: Metóda Nelder-Mead - reflexia, expanzia: (a) k bodu s najvyššou funkčnou hodnotou (červený bod) bol vygenerovaný reflektovaný bod (zelený bod), ktorý má nižšiu funkčnú hodnotu, v takomto prípade sa pokúsime reflektovaný bod posunúť ešte ďalej od pôvodného (b) a zistiť jeho funkčnú hodnotu, ktorá v tomto prípade je vyššia (ružový bod) a preto je bod zamietnutý a vraciame sa späť do reflektovaného bodu, ktorý príjmame za nový vrchol trojuholníka



(a) (b)

Obr. 8: Metóda Nelder-Mead - kontrakcia, zmrštenie: (a) v prípade, že sa reflexiou nenájde bod s nižšou funkčnou hodnotou (ružový bod), bod s najvyššou funkčnou v pôvodnom trojuholníku (červený bod) prejde kontrakciou a stiahne sa do vnútra trojuholníka (biely bod), ak ani po tejto operácii nenameriame nižšiu funkčnú hodnotu, dochádza k zmršteniu (b) celého trojuholníka smerom k bodu s najnižšou funkčnou hodnotou v rámci trojuholníka (zelený bod)

3.1.4 Výsledky aplikácie Nelder-Mead metódy

Metóda Nelder-Mead je prvá z gradientových metód, ktorú sme aplikovali na reálne zariadenie, teda na 2D Plotter si senzorom svietivosti. Z definície gradientu a znalosti vývoja funkcie môžeme predpokladať vysokú závislosť nájdenia globálneho minima od voľby počiatočného bodu. Tento predpoklad odôvodňujeme tým, že gradient smeruje vždy v smere najväčšieho lokálneho poklesu funkcie. Ak sa teda bod nachádza v blízkosti lokálneho minima namiesto globálneho, algoritmus zostane v tomto lokálnom minime zaseknutý. Zároveň, keďže sa neprehľadáva celý priestor, ale pohybujeme sa v smere poklesu funkcie dochádza k skráteniu času trvania programu a teda aj počtu iterácii. Predpoklady si môžeme overiť v tabuľke uskutočnených experimentov.

Tabuľka 3: Výsledky implementácie metódy Nelder-Mead

Počiatočný bod	Čas [s]	Iterácie	Nájdené minimum	Vzdialenosť	Typ minima
[0; 0]	37,3	20	[-2,6751; -3,0136]	0,2536	globálne
[0; 0]	31,5	12	[-2,6751; -2,8704]	0,2309	globálne
[0; 0]	30,6	11	[-2,6366; -2,4852]	0,4963	globálne
[1,5; 1,5]	35,9	11	[-2,8013; 3,5833]	0,8426	lokálne
[1,5; 1,5]	31,1	12	[-2,1482; 3,2936]	0,9325	lokálne
[1,5; 1,5]	35,9	13	[-4,0137; -2,5167]	1,1756	globálne
[-1,5; -1,5]	39,5	11	[-2,8472; -2,4877]	0,4197	globálne
[-1,5; -1,5]	40,5	14	[-2,8597; -2,4628]	0,4430	globálne
[-1,5; -1,5]	40,5	16	[-2,7358; -2,5053]	0,4322	globálne

Naše odhady sme touto vzorkou experimentov potvrdili. V každom jednom prípade bolo nájdené minimum, pričom to, či šlo o lokálne alebo globálne záviselo od počiatočného bodu. V prípade gradientových metód ide o preddefinované kroky, čiže pri každom spustení by sme sa mali dostať do rovnakej pozície. Pretože meranie z fotosenzora je zaťažené šumom a ovplyvnené rôznymi svetlnými podmienkami v okolí zariadenia, nastáva situácia, kedy v jednom bode nenameriame totožné funkčné hodnoty. Vďaka tomuto sa algoritmus nezasekol v blízkom lokálnom minime, ale skonvergoval do vzdialenejšieho globálneho pri uvažovaní počiatočného bodu $x_0 = [1,5; 1,5]$

3.2 Odhad gradientu pomocou trojuholníka

Princíp odhadu gradientu pomocou troch bodov v 2D priestore spočíva v porovnávaní funkčných hodnôt v týchto bodoch (vrcholoch trojuholníka). Vrchol, v ktorom zistíme najmenšiu funkčnú hodnotu bude predstavovať smer, ktorým sa posunie spojnica zvyšných dvoch bodov. Táto metóda je jednou z tých nepresnejších, ktoré sú založené na odhade gradientu, pretože vytvorený vektor od stredu spojnice dvoch horších bodov po ten najlepší, nereprezentuje skutočný maximálny pokles funkcie, ale iba pokles funkcie. Akokoľvek, aj takáto metóda postupnými krokmi. a teda posunmi trojuholníka, dokáže skonvergovať do lokálneho minima.

3.2.1 Algoritmus metódy

Minimalizujeme funkciu $f(x)$, kde $x_0 \in \mathbb{R}^2$, predstavuje preddefinovaný stred trojuholníka v priestore, nasledovne:

- vygenerovanie troch bodov v okolí stredu x_0, pričom vrcholy sú od seba fixne vzdialené

- vytvorenie poradia funkčných hodnôt vo vrcholoch trojuholníka:
  $$f(x_1) \leq f(x_2) \leq f(x_3)$$
- výpočet normály N (odhadu gradientu) ku spojnici dvoch horších bodov v smere najlepšieho bodu (v prípade rovnostranného trojuholníka sa N počíta pomocou stredu trojuholníka x_0)
- výpočet nového bodu
  $$x' = x_0 + t \cdot N,$$
  kde koeficient t predstavuje optimálnu veľkosť kroku, ktorá sa v priebehu programu počíta pomocou metódy "backtracking", ktorá je bližšie popísaná v ďalšej podkapitole
- ak platí $f(x') < f(x_0)$, príjme sa $x_0 = x'$ a dochádza k návratu do prvému kroku
- v prípade, že $f(x_0) < f(x_1)$, teda v trojuholníku dochádza ku poklesu funkcie, trojuholník sa zmrští na 90% pôvodnej veľkosti a dochádza k návratu do prvému kroku
- zastavovacím kritériom behu programu je porovnanie funkčných hodnôt novo vygenerovaného bodu a súčasného najlepšieho bodu:
  $$|f(x') - f(x_1)| < \varepsilon_1$$
  a zároveň veľkosť normály N nesmie byť zanedbateľne malá:
  $$\|N\| < \varepsilon_2.$$

3.2.2 Backtracking

Backtracking je metóda vyhľadávania na určenie maximálneho posunu jedným konkrétnym smerom. Zahŕňa to začiatok s relatívne veľkým odhadom veľkosti kroku pohybu pozdĺž vyhľadávacieho smeru a iteračne zmenšuje veľkosť kroku, pokiaľ nepozorujeme zníženie funkčnej hodnoty účelovej funkcie korešpodujúce klesaniu, ktoré je očakávané na základe gradientu funkcie.

Backtracking funguje na základe danej začiatočnej pozície x_0 a smeru vyhľadávania Δx. Úlohou priamkového vyhľadávania je určiť veľkosť kroku t, ktorý adekvátne redukuje účelovú funkciu $f : \mathbb{R}^n \to \mathbb{R}$, t.j. nájdenie hodnoty t, ktorá redukuje

$$f(x_0 + t \cdot \Delta x),$$

príbuznej ku $f(x)$. Avšak, zvyčajne je nežiadúce venovať značné zdroje hľadaniu hodnoty t, ktorá presne minimalizuje f. Je to spôsobené tým, že výpočtové zdroje potrebné na presnejšiu lokalizáciu minima v jednom konkrétnom smere, by sa mohli použiť na identifikáciu lepšieho smeru vyhľadávania. Keď sa identifikuje lepší východiskový bod,

ďalšie vyhľadávanie začne v inom smere. Cieľom je nastaviť hodnotu t tak, aby vytvorilo rozumné zlepšenie účelovej funkcie namiesto toho, aby sa našla skutočná minimalizačná hodnota parametra t.

Zadefinujeme lokálny sklon funkcie vzhľadom na smer vyhľadávania ako $\Delta x^T \nabla f(x)$. Predpokladá sa, že Δx je jednotkový vektor v smere lokálneho klesania funkcie, čiže sa tým pádom predpokladá

$$\Delta x^T \nabla f(x) < 0.$$

Na základe vybraného kontrolného parametra $\alpha \in (0, 1)$ sa testuje vhodnosť kroku pri presune zo súčasnej pozície x_0 na pozíciu $x_0 + t \cdot \Delta x$, či sa dosiahol dostatočný pokles funkcie. Podmienka je splnená, ak platí:

$$f(x_0 + t\Delta x) < f(x_0) + \alpha t N^T \Delta x.$$

Ak je táto podmienka (tzv. Armijo-Goldsteinová podmienka) použitá priamo vo vyhľadávaní, môže zaistiť, že veľkosť kroku nebude nadmerne veľká. Avšak, táto podmienka sama o sebe nezaisťuje, že veľkosť kroku je optimálna, pretože akékoľvek t, ktoré je dostatočne malé splní túto podmienku. Preto technika takéhoto vyhľadávania začína s relatívne veľkým krokom a opakovane sa zmenšuje podľa koeficientu $\beta \in (0, 1)$, pokiaľ Armijo-Golsteinová podmienka nie je splnená. [10]

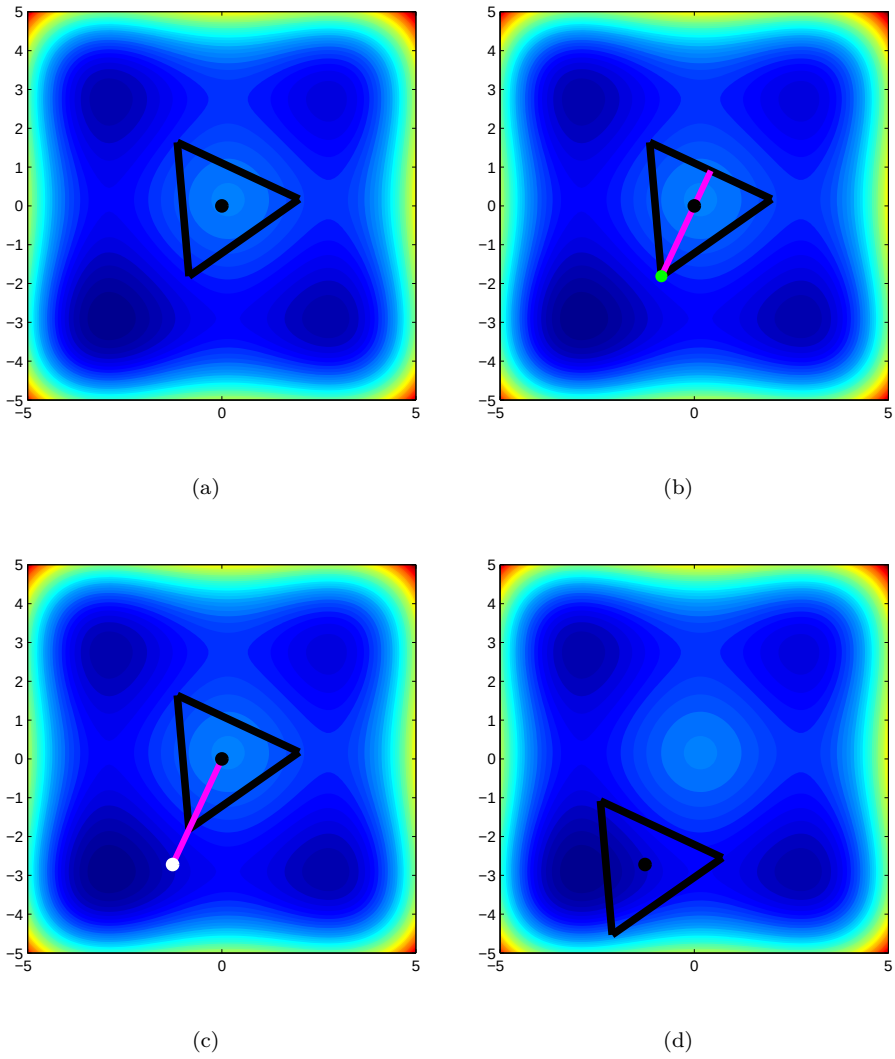
3.2.3 Parametre metódy

Parametrami, ktorými môžeme ovplyvňovať rýchlosť a presnosť riešenia, pri tejto metóde sú koeficienty využívané pri metóde backtracking, ktoré nám ovplyvňujú generovanie nového bodu. Zároveň je parametrom aj počiatočná veľkosť trojuholníka a jeho zmenšovanie pri behu programu. V poslednom rade je to tolerancia prijatia vygenerovaného bodu za optimum. Všetky tieto parametre si volíme s úmyslom dosiahnutia vhodnej dĺžky behu algoritmu a presnosti výsledku.

3.2.4 Implementácia metódy

Pri implementácii tejto metódy (Obrázok 9) používame ako odhad gradientu normálu, ako to bolo vysvetlené v predošlých podkapitolách. Táto normála sa však môže podstatne líšiť od skutočného gradientu, lebo nepriemerujeme vplyv horších bodov na jej orientáciu. Napriek tomu, táto metóda funguje a podobne ako pri technike Nelder-Mead generujeme trojuholník pomocou kružnice (rovnostranný trojuholník). Na začiatku sme ale zmenili jeho natočenie oproti trojuholníku použitému v Nelder-Mead, aby sa nám ľahšie pracovalo s výpočtom normály. Najdôležitejší je ale backtracking, ktorý vyhľadáva možné nové body vhodným krokom v danom smere poklesu funkcie. Po viacerých pokusoch sa nám podarilo

experimentálne určiť parametre, pomocou ktorých bol algoritmus schopný nájsť riešenie s dostatočnou presnosťou.



Obr. 9: Aplikácia metódy odhadu gradientu pomocou trojuholníka: (a) počiatočný bod s vygenerovanými troma bodmi tvoriacimi trojuholník, (b) nájdenie bodu s najnižšou funkčnou hodnotou v rámci trojuholníka (zelený bod) a následne vypočítaný gradient v danom smere (ružová úsečka), (c) výpočet nového stredu trojuholníka vzhľadom na gradient (biely bod), (d) presun trojuholníka do nového stredu

3.2.5 Výsledky aplikácie odhadu gradientu pomocou trojuholníka

Odhad gradientu pomocou trojuholníka na prvý pohľad pôsobí ako metóda Nelder-Mead, ale už pri prvých iteráciách vidíme rozdiel. Zbavili sme sa totiž spomaľujúcich matematických operácii a tým sme znížili počet iterácii algoritmu. Tým pádom by malo teoreticky dôjsť aj k skráteniu času behu programu. Výsledky experimentov sú v tabuľke.

Tabuľka 4: Výsledky implementácie odhadu gradientu pomocou trojuholníka

Počiatočný bod	**Čas [s]**	**Iterácie**	**Nájdené minimum**	**Vzdialenosť**	**Typ minima**
[0; 0]	49,9	14	[-2,8918; 3,0115]	0,2649	lokálne
[0; 0]	38,8	9	[-3,3772; -2,7417]	0,5005	globálne
[0; 0]	43,9	10	[-2,9352; -2,6013]	0,3039	globálne
[1,5; 1,5]	46,5	10	[3,2478; 3,2621]	0,7186	lokálne
[1,5; 1,5]	38,4	5	[2,7398; 2,6918]	0,0556	lokálne
[1,5; 1,5]	29,8	4	[2,5873; 3,2178]	0,4972	lokálne
[-1,5; -1,5]	19,2	3	[-3,5588; -3,1605]	0,7038	globálne
[-1,5; -1,5]	57,9	19	[-3,1926; -2,5410]	0,4637	globálne
[-1,5; -1,5]	34,4	7	[-3,2788; -3,0692]	0,4101	globálne

Presnosť riešenia je podobná ako v prípade metódy Nelder-Mead. Zaujímavejší je ale čas trvania programu, vzhľadom na predošlú metódu. Predpokladali sme skrátenie času, ale čas je približne rovnaký, alebo dokonca dlhší ako pri Nelder-Mead metóde. Zbavili sme sa totiž matematických operácii, ale zároveň sme pridali metódu backtracking, ktorá prepočítava a skúma najvýhodnejší nasledovný krok. To sa prejavuje v časovej oblasti a zároveň aj v zložitosti algoritmu. Napriek tomu nedochádza k výrazným odchýlkam od stanoveného cieľa.

3.3 Odhad gradientu pomocou kružnice

Princíp tejto metódy je založený na vytvorení kružnice pomocou n-bodov v dvojrozmernom priestore. Táto technika pracuje s väčším počtom bodov (pre našu prácu $n = 12$), čo zaručuje presnejší odhad smeru najväčšieho poklesu funkcie. Skúmaním funkčných hodnôt v bodoch kružnice zistíme najlepšiu a najhoršiu funkčnú hodnotu v r okolí bodu x_0, ktorý predstavuje stred kružnice. Vytvorením spojnice týchto bodov dostaneme priamku s najväčšou smernicou, čiže priamku určujúcu najväčší sklon funkcie. Vzhľadom na to, že hľadáme minimum funkcie, si vytvoríme vektor v smere jej poklesu. Tým pádom dostávame vektor, ktorý môžeme označiť za odhadnutý gradient.

3.3.1 Algoritmus metódy

Minimalizujeme funkciu $f(x)$, kde $x_0 \in \mathbb{R}^2$, predstavuje preddefinovaný stred kružnice v priestore, nasledovne:

- vygenerovanie n bodov v okolí stredu x_0, pričom tieto body sú od seba fixne vzdialené
- nájdenie najväčšej $f(x_n)$ a najmenšej $f(x_1)$ funkčnej hodnoty spomedzi hodnôt bodov kružnice a im prislúchajúce súradnice:
  $f(x_1) \leq ... \leq f(x_n)$
- výpočet odhadu gradientu vzhľadom na bod s najvyššou a najnižšou funkčnou hodnotou:
  $N = x_1 - x_n$
- výpočet nového bodu
  $x' = x_0 + tN,$

  kde koeficient t predstavuje optimálnu veľkosť kroku, ktorá sa v priebehu programu počíta cez metódu backtracking
- ak platí $f(x') < f(x_0)$, príjma sa $x_0 = x'$ a dochádza k návratu do prvého kroku
- v prípade, že $f(x_0) < f(x_1)$, teda v kružnici dochádza ku poklesu funkcie, priemer kružnice sa zmenšuje $r = 0,9 \cdot r$ a dochádza k návratu do prvého kroku
- zastavovacím kritériom behu programu je porovnanie funkčných hodnôt v novo vygenerovanom bode a súčasného najlepšieho bodu:
  $|f(x') - f(x_1)| < \varepsilon_1$

  a zároveň veľkosť odhadovaného gradientu N nesmie byť zanedbateľne malá:
  $\|N\| < \varepsilon_2$.

3.3.2 Parametre metódy

Pre metódu vyhľadávania optima pomocou odhadu gradientu kružnicou vhodne volíme hodnoty parametrov. Tieto parametre sú definované tak, ako pre predošlú metódu v backtracking, zároveň si môžeme voliť počiatočný polomer kružnice ako aj koeficient zmenšovania tohto polomeru. Tak ako pri každej metóde s výnimkou simulovaného žíhania, potrebujeme zvoliť aj vhodnú toleranciu prijatia bodu za hľadané optimum. Rozdiel oproti predošlým metódam je, že počet bodov po obvode kružnice, tiež môžeme efektívne meniť.

3.3.3 Implementácia algoritmu

Algoritmus vytvorený pre túto metódu sa opiera o spomínané parametre. Hodnoty týchto parametrov boli opäť experimentálne stanovené tak, aby bol extrém funkcie lokalizovaný s dostatočnou presnosťou a rýchlosťou. Začínali sme na polomere kružnice s hodnotou $0,5$, ktorá sa počas programu postupne zmenšovala. Ako môžeme v algoritme ďalej vidieť, definovali sme počet bodov mnohostena na 12, pretože zariadenie bolo schopné pomerne rýchlov prejsť tento počet bodov a zároveň sa kružnica pohybovala správnym smerom. Hodnota tolerancie bola zvolená na základe pokusov na reálnom zariadení, pričom stopovacím kritériom je aj veľkosť kružnice, pretože pri veľmi malej kružnici dochádza už iba k minimálnemu posunu vzhľadom na funkciu. Základný princíp funkcie algoritmu je znázornený na obrázku (Obrázok 10).

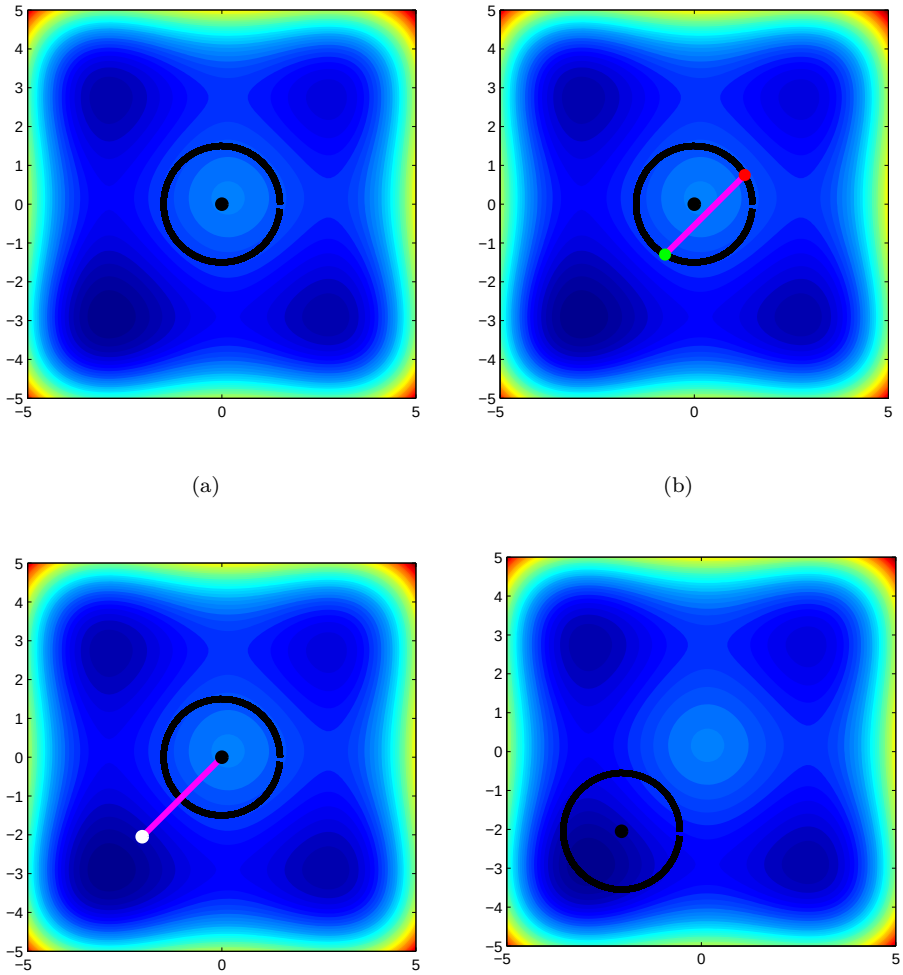
3.3.4 Výsledky aplikácie odhadu gradientu pomocou kružnice

Aplikovaním tejto metódy očakávame zrýchlenie aj spresnenie vyhľadávania minima. Tieto očakávania vyplývajú z lepšej aproximácie gradientu a tým pádom kvalitnejšiemu posunu v rámci funkcie. Keďže znova používame backtracking, môžeme očakávať približne rovnaké trvanie behu programu v porovaní s metódou odhadu gradientu pomocou trojuholníka.

Tabuľka 5: Výsledky implementácie odhadu gradientu pomocou kružnice

Počiatočný bod	**Čas [s]**	**Iterácie**	**Nájdené minimum**	**Vzdialenosť**	**Typ minima**
[0; 0]	33,5	6	[-3,1651; -2,4821]	0,4960	globálne
[0; 0]	37,5	7	[-2,6740; -2,6986]	0,3078	globálne
[0; 0]	32,8	6	[-2,5490; -3,0490]	0,3833	globálne
[1,5; 1,5]	24,2	4	[2,8660; 3,5000]	0,7625	lokálne
[1,5; 1,5]	28,4	5	[2,6205; 3,2243]	0,4938	lokálne
[1,5; 1,5]	18,4	3	[2,6830; 3,0490]	0,3088	lokálne
[-1,5; -1,5]	27,5	5	[-2,9910; -2,8325]	0,1127	globálne
[-1,5; -1,5]	26,7	5	[-2,8995; -2,7990]	0,1047	globálne
[-1,5; -1,5]	17,2	3	[-2,6830; -3,0490]	0,2642	globálne

Keďže stále hovoríme o gradientovej metóde, závislosť nájdenia globálneho minima stále závisí od počiatočného stavu. Zároveň vidíme, že došlo k zníženiu počtu iterácii, pričom čas zostal nezmenený v porovnaní s predošlými gradientovými metódami, čo je spôsobené využívaním spomínanej metódy backtracking. Napriek tomu, táto metóda sa pri experimentoch ukázala ako najvýhodnejšia, pretože pri reálnom probléme (únik radiácie, nebezpečného plynu) je pre nás rovnako dôležité nájdenie lokálneho aj globálneho minima.



(a) (b)

(c) (d)

Obr. 10: Aplikácia odhadu gradientu pomocou kružnice na problém: (a) počiatočný bod s vygenerovanými bodmi vytvárajúcimi kružnicu, (b) nájdenie bodu s najnižšou (zelený bod) a najvyššou (červený bod) funkčnou hodnotou v rámci kružnice a následne vypočítaný gradient v danom smere (ružová úsečka), (c) výpočet nového stredu kružnice vzhľadom na gradient (biely bod), (d) presun kružnice do nového stredu

4 Ohraničená optimalizácia

V matematickej optimalizácii je ohraničená optimalizácia proces hľadania extrémov účelovej funkcie vzhľadom na nejaké premenné, za prítomnosti obmedzení na týchto premenných. Ohraničené optimalizačné problémy sa vyskytujú vo viacerých aplikáciach v oblasti vedy, inžinierstva, sociálnych vied a medicíny. Všeobecný nelineárny problém môže byť daný nasledovne:

$$\min_x f(x)$$

$$s.t.\ l \leq \begin{Bmatrix} x \\ A_L x \\ c(x) \end{Bmatrix} \leq u,$$

kde $f(x)$ je spojitá nelineárna účelová funkcia, A_L je matica ohraničení a $c(x)$ je vektor spojitých nelineárnych funkcií ohraničení. V konečnom dôsledku sa hlavne skúma nasledujúci problém:

$$\min_x f(x)$$

$$s.t.\ c(x) \leq 0,$$

kde f a c majú spojitú druhú deriváciu v blízkosti riešenia. Výsledok minimalizácie takejto účelovej funkcie sa označuje x^*. Bod x, pre ktorý platí $c(x) > 0$, sa považuje za prijateľný a ohraničenie c_i sa považuje za porušené v x, ak $c_i(x) < 0$. Výsledok minimalizácie je za daných podmienok unikátny. [11]

Riešiť tento problém môžeme rôznymi objavenými metódami alebo postupmi. Ak uvažujeme použité iteračné metódy v tejto práci (metóda Luus-Jaakoly a metóda Simulovaného žíhanie), boli by sme schopní upraviť ich algoritmy, aby sme jednoducho body nevyhovujúce ohraničeniam vylúčili z procedúry. Takýmto spôsobom by sme sa dopracovali k požadovanému výsledku, ale nedošlo by k urýchleniu celého procesu, pretože body by boli iteračne vygenerované, ale následne cez podmienku zamietnuté. Matematicky prijateľnejším riešením je využívanie metód, ktoré sa snažia problém transformovať na neohraničený podproblém. Takýmito metódami sú napríklad metóda penalizovanej funkcie, generalizované metódy redukovaného gradientu alebo bariérová funkcia. Poslednú menovanú sme použili na riešenie nášho problému.

4.1 Bariérová metóda

Bariérové metódy sú navrhnuté na riešenie problému tak, že riešia sekvenciu špeciálne skonštruovaných optimalizačných problémov. Pri bariérovej metóde predpokladáme, že máme daný počiatočný bod x_0, ktorý leží vo vnútri skúmanej funkcie vzhľadom na

ohraničenia a ukladáme penalizáciu (umelo zvyšujeme funkčnú hodnotu) na body, ktoré ležia bližšie k ohraničeniam, čím sa vytvorí bariéra na opustenie tejto časti skúmaného priestoru. Základná myšlienka bariérovej metódy je odradiť body x od priblíženia k ohraničeniu preskúmavaného priestoru. Dosiahneme to rozšírením pôvodného minimalizačného problému

$$\min_x f_0(x)$$
$$s.t.\ f_i(x) \leq 0,\ i = 1, ..., m,$$

kde m predstavuje počet obmedzení, o súčet indikátorových funkcií zahŕňajúcich ohraničenia, teda prekážky, v priestore:

$$\min_x f_0(x) + \sum_{i=1}^{m} I_-(f_i(x))$$

Indikátorová funkcia nadobúda hodnoty vzhľadom na ohraničenia

$$I_-(z) = \begin{cases} 0 & z \leq 0 \\ \infty & z > 0 \end{cases}$$

Problémom je, že indikátorová funkcia nie je spojite diferencovateľná. Tento problém nám rieši práve bariérová funkcia, ktorá aproximuje indikátorovú funkciu podobnou funkciou, ktorá je spojite diferencovateľná.

Proces hľadania minima s ohľadom na prítomnosť prekážok vo veľkej miere závisí od vhodnej aproximácie. Čím je aproximácia presnejšia, tým lepšie výsledky dostávame. Presnosť sa dosahuje na úkor rýchlosti. Preto sa pri nahradení indikátorovej funkcie aproximovanou používa aj parameter t, ktorý nám túto funkciu v priebehu programu mení. Nastavujeme si ho spočiatku na nejakú vysokú hodnotu, ktorá nám nezabezpečí úplne vhodnú aproximáciu, ale posunieme sa v smere poklesu funkcie. V momente, keď sa nájde minimum takejto funkcie, parameter t sa zníži, určí sa nový počiatočný bod v nájdenom minime a celý program beží od znova. Znižovaním parametra t dochádza k presnejšej aproximácii účelovej funkcie a keďže sa mení postupne, dokážeme minimum lokalizovať pomocou menšieho počtu iterácii. Možností na výber bariérovej funkcie je viacero a volí sa podľa toho, ktorá nám lepšie aproximuje danú indikátorovú funkciu. Najpoužívanejšou je logaritmická bariéra, ktorú využívame aj pri našom riešení.

4.1.1 Logaritmická bariérová metóda

Rieši problém

$$\min_x f_0(x) + t\phi(x)$$

s logaritmickou bariérovou funkciou

$$\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x)),$$

kde f_i predstavuje ohraničenia. V závislosti od toho v akej časti skúmaného priestoru sa bod nachádza, podľa toho nadobúda aj funkčnú hodnotu, ktorá môže byť kladná (bod sa nachádza v ohraničení), záporná (bod sa nachádza mimo ohraničenia) alebo nulová (bod sa nachádza na hranici). V prípade, že sa bod blíži k ohraničeniu, jeho funkčná hodnota vzhľadom na omedzenia f_i ide limitne do nuly a teda aj záporná hodnota f_i ide limitne do nuly. Preto platí

$$-f_i(x) \to 0 \Rightarrow log(-f_i(x)) \to \infty,$$

čo predstavuje základnú myšlienku bariérovej metódy. Teda, ak sa blížime k prekážke, detekujeme vyššiu funkčnú hodnotu v danom bode ako je reálna, čím algoritmus vyhodnotí tento smer za nevyhovujúci. V hraničnom prípade, ak algoritmus vygeneruje bod v ohraničení, dostávame nedefinovaný logaritmus zápornej hodnoty, ktorý nahrádzame nekonečnom. Dochádza teda k pozmeneniu účelovej funkcie, vzhľadom na ohraničenia, do ktorých sa nesmieme dostať.

4.2 Implementácia

Oboznámili sme sa už so základným princípom všetkých gradientových aj bezgradientových metód, s ktorými v tejto práci pracujeme. Na tie najlepšie sme aplikovali aj bariérovú metódu a teda sme kód rozšírili o prípadné ohraničenia.

4.2.1 Logaritmická bariérová funkcia a metóda Luus-Jaakola

Rozšírenie klasickej iteračnej metódy Luus-Jaakola o zohľadnenie ohraničení funguje nasledovne:

- inicializácia $x_0 \in \langle b_L, b_U \rangle$ na jednoznačnú pozíciu v priestore, kde b_L a b_U sú dolné a horné ohraničenie
- nastavenie veľkosti okolia, v ktorom sa generujú náhodné body, aby pokrylo celý priestor, v ktorom hľadáme minimum:
  $$d = b_U - b_L$$
- zadefinovanie funkcií ohraničení, v našom prípade v tvare kruhu, so stredom z_i a polomerom r_i nasledovne:
  $$f_i = r_i + \|x - z_i\|$$
- pokiaľ nie je splnené stopovacie kritérium
  $$|F(x_0) - F(x)| < \varepsilon \ (\varepsilon = 10^{-3}),$$
  opakujú sa nasledovné kroky:

- náhodne sa vygeneruje x v zadanom okolí
- vypočítanie logaritmickej bariérovej funkcie z funkcií ohraničení

  $$\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x)),$$

  pričom, ak je $f_i > 0$, logaritmus je nedefinový a preto mu algoritmus priradí hodnotu nekonečno

  $$\log(-f_i(x)) = \infty$$

- pripočítanie hodnoty logaritmickej bariérovej funkcie k funkčnej hodnote v bode x:

  $$F(x) = f(x) + t\phi(x),$$

  kde t je parameter predstavujúci mieru presnosti aproximácie indikátorovej funkcie
- ak $F(x) < F(x_0)$, potom nastáva presun na novú pozíciu nastavením $x_0 = x$, v opačnom prípade sa okolie, v ktorom generujeme body, zmenší:

  $$d = \alpha \cdot d \ (\alpha = 0,95)$$

- x_0 predstavuje najlepšiu nájdenú pozíciu (v konečnom dôsledku lokálne minimum funkcie)

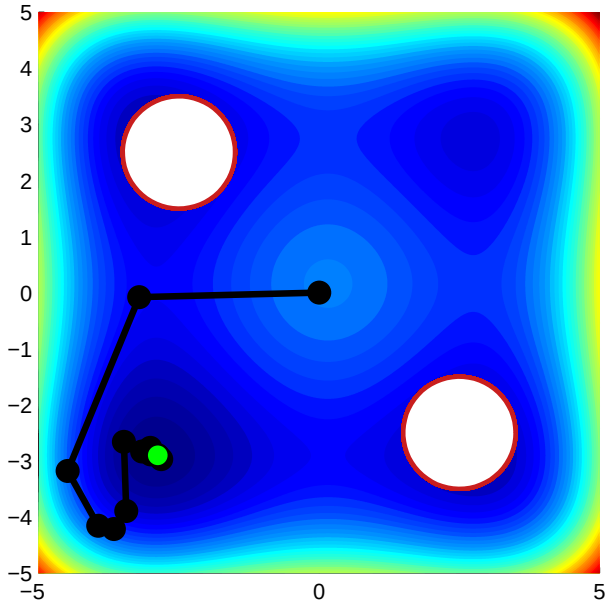
4.2.2 Výsledky aplikácie logaritmickej bariérovej metódy na metódu Luus-Jaakola

Vzhľadom na už nabudubnuté vedomosti o logaritmickej bariérovej metóde sme schopní posúdiť správnosť jej implementácie. To, do ktorého minima Styblinski-Tangovej funkcie algoritmus skonverguje, závisí od polohy prekážok v priestore. Pri našich experimentoch prekážky predstavovali kružnice, ktorých poloha bola presne daná ich stredom. Na demonštráciu funkčnosti metódy sme použili dva grafy, pričom na každom z nich boli prekážky lokalizované na inom mieste v priestore. V prvom prípade sme použili graf s dvoma kruhovými prekážkami so stredmi v [-2,5; 2,5] a [2,5; -2,5]. Obe prekážky sa teda nachádzali v lokálnych minimách funkcie. V tabuľke 6 sú zobrazené výsledky experimentov vzhľadom na už spomínané obmedzenia.

Tabuľka 6: Výsledky implementácie logaritmickej bariérovej metódy na metódu Luus-Jaakola s kruhovými prekážkami so stredmi v bodoch [-2,5; 2,5] a [2,5; -2,5]

Počiatočný bod	Čas [s]	Iterácie	Nájdené minimum	Vzdialenosť
[0; 0]	59,5	43	[-2,5996; -3,2993]	0,4990
[0; 0]	50,0	37	[-2,8715; -3,3780]	0,4755
[0; 0]	48,5	35	[-2,9564; -3,0874]	0,1912

Keďže sme aplikovali iteračnú metódu, algoritmu trvalo dlhšie, kým sa dopracoval k výsledku. Napriek tomu, je vždy výstupom z programu nájdené globálne minimum s uspokojivou presnosťou. Počas behu programu došlo k matematickému prepočítavaniu účelovej funkcie pomocou logaritmicej bariérovej metódy, ktorá zabezpečila vyhnutie sa prekážkam. Konvergencia do extrému pomocou tejto metódy je zobrazený na obrázku 11.



Obr. 11: Konvergenciu do minima pri použití logaritmickej bariérovej metódy na metódu Luus-Jaakola so stredmi prekážok v bodoch [-2,5; 2,5] a [2,5; -2,5]

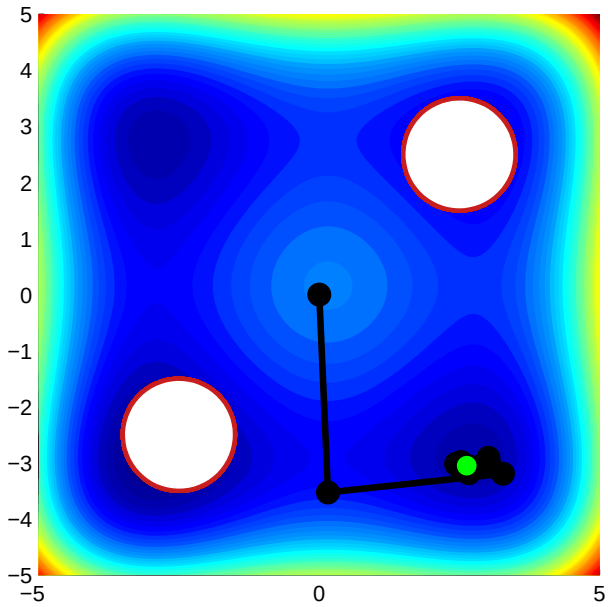
V druhom prípade sme taktiež pracovali s grafom s dvoma kruhovými prekážkami, ale stredy boli lokalizované v bodoch [-2,5; -2,5] a [2,5; 2,5]. Takéto umiestnenie prekážok znamená, že globálne minimum Styblinsky-Tangovej funkcie sa nachádza presne na mieste

prekážky. V tomto prípade teda výsledkom nikdy nebude dosiahnutie globálneho minima, ale použitím logaritmickej bariérovej funkcie sa dostávame do jedného z lokálnych miním. Výsledky experimentov uvažujúcich práve takéto prekážky sú spracované v tabuľke 7.

Tabuľka 7: Výsledky implementácie logaritmickej bariérovej metódy na metódu Luus-Jaakola s kruhovými prekážkami so stredmi v bodoch [-2,5; -2,5] a [2,5; 2,5]

Počiatočný bod	**Čas [s]**	**Iterácie**	**Nájdené minimum**	**Vzdialenosť**
[0; 0]	52,8	40	[2,6315; -3,0525]	0,1884
[0; 0]	59,6	37	[2,4556; -2,9821]	0,3017
[0; 0]	55,2	38	[3,0139; -2,8263]	0,2780

Počet iterácii a dĺžka trvania nájdenia minima sú stále primárne ovplyvňované použitou Luus-Jaakola. Ako sme už spomínali, nájdené minimum je lokálne, pretože ku globálnemu sa algoritmus nevie dostať z dôvodu obmedzení. Napriek tomu, toto lokálne minimum bolo nájdené s vysokou presnosťou. Pre lepšiu predstavu je táto konvergencia zobrazená na obrázku 12.



Obr. 12: Konvergenciu do minima pri použití logaritmickej bariérovej metódy na metódu Luus-Jaakola so stredmi prekážok v bodoch [-2,5; -2,5] a [2,5; 2,5]

4.2.3 Logaritmická bariérová funkcia a metóda odhadu gradientu pomocou kružnice

Rozšírenie metódy odhadu gradientu pomocou kružnice o zohľadnenie ohraničení funguje nasledovne:

- vygenerovanie n bodov v okolí stredu x_0, pričom tieto body sú od seba fixne vzdialené
- zadefinovanie funkcií ohraničení, v našom prípade v tvare kruhu, so stredom z_i a polomerom r_i nasledovne:
  $$f_i = r_i + \|x - z_i\|$$
- vypočítanie logaritmickej bariérovej funkcie z funkcií ohraničení
  $$\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x)),$$
  pričom, ak je $f_i > 0$, logaritmus je nedefinový a preto mu algoritmus priradí nekonečnú hodnotu
  $$-\log(-f_i(x)) = \infty$$
- pripočítanie hodnoty logaritmickej bariérovej funkcie k funkčnej hodnote v bode x
  $$F(x) = f(x) + t\phi(x),$$
  kde t predstavuje parameter zabezpečujúci vhodnú aproximáciu indikátorovej funkcie
- nájdenie najväčšej $F(x_n)$ a najmenšej $F(x_1)$ funkčnej hodnoty spomedzi hodnôt bodov kružnice a im prislúchajúce súradnice:
  $$F(x_1) \leq ... \leq F(x_n)$$
- výpočet odhadu gradientu vzhľadom na bod s najvyššou a najnižšou funkčnou hodnotou:
  $$N = x_1 - x_n$$
- výpočet nového bodu
  $$x' = x_0 + tN,$$
  kde koeficient t predstavuje optimálnu veľkosť kroku, ktorá sa v priebehu programu počíta cez metódu backtracking
- ak platí $F(x') < F(x_0)$ príjma sa $x_0 = x'$ a dochádza k návratu do prvého kroku
- v prípade, že $F(x_0) < F(x_1)$, teda v kružnici dochádza k poklesu funkcie, priemer kružnice sa zmenší $r = 0,9r$ a dochádza k návratu do prvého kroku

- zastavovacím kritériom behu programu je porovnanie funkčných hodnôt v novo vygenerovanom bode a súčasnom najlepšom bode:

  $$|F(x') - F(x_1)| < \varepsilon_1$$

  a zároveň veľkosť odhadovaného gradientu N nesmie byť zanedbateľne malá:

  $$\|N\| < \varepsilon_2.$$

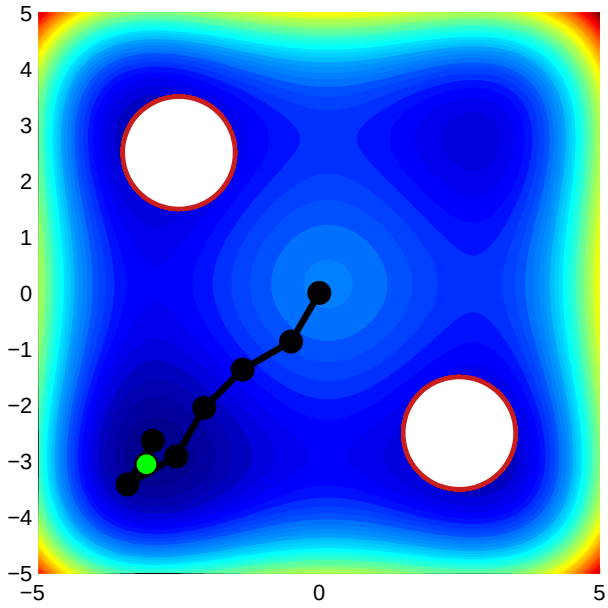
4.2.4 Výsledky aplikácie logaritmickej bariérovej metódy na metódu odhadu gradientu pomocou kružnice

Princíp gradientovej metódy už poznáme. Pri experimentoch s touto metódou sme postupovali tak, ako pri predošlej aplikácii logaritmickej bariéry na metódu Luus-Jaakola. To znamená, že sme znova pracovali s dvoma kontúrovými grafmi, pričom každý z nich mal prekážky na iných pozíciach. Najprv si ukážeme výsledky použitia metódy na problém s dvoma prekážkami so stredmi v bodoch [-2,5; 2,5] a [2,5; -2,5] (Tabuľka 8).

Tabuľka 8: Výsledky implementácie logaritmickej bariérovej metódy na metódu odhadu gradientu pomocou kružnice s kruhovými prekážkami so stredmi v bodoch [-2,5; 2,5] a [2,5; -2,5]

Počiatočný bod	Čas [s]	Iterácie	Nájdené minimum	Vzdialenosť
[0; 0]	67,2	14	[-2,7261; -2,6982]	0,2715
[0; 0]	43,5	8	[-3,0887; -3,0964]	0,2673
[0; 0]	46,3	9	[-3,0776; -3,0555]	0,2310

Keďže používame gradientovú metódu, celý proces sa nám značne urýchlil. Vidieť to môžme na kratšom čase trvania programu a menšom počte iterácii. Už vieme, že je to spôsobené posunom bodov v smere najväčšiieho klesania funkcie. Vďaka bariérovej funkcii bol algoritmus schopný vyhnúť sa prekážkam a skonvergovať do globálneho minima, ako je to znázornené na obrázku 13. V prípade zmeny počiatočného bodu, algoritmus vždy skonvergoval do minima mimo prekážky a to, aký typ minima bol na výstupe, záviselo od polohy počiatočného bodu v priestore.



Obr. 13: Konvergenciu do minima pri použití logaritmickej bariérovej metódy na metódu odhadu gradientu pomocou kružnice so stredmi prekážok v bodoch [-2,5; 2,5] a [2,5; -2,5]

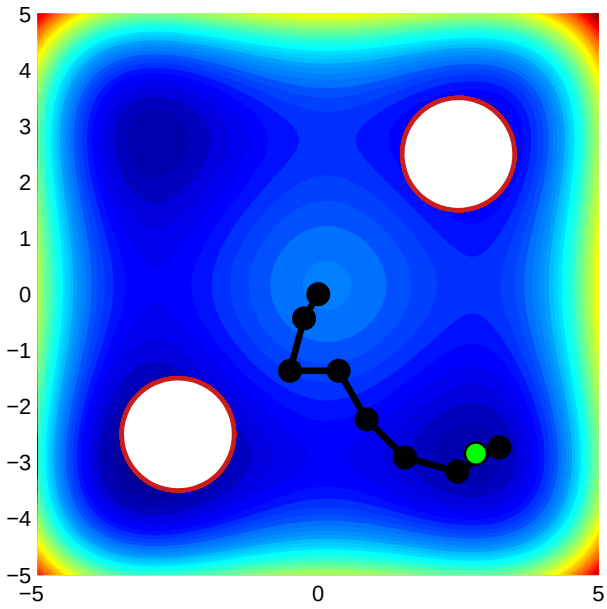
Druhý prípad, kedy stredy prekážok sú lokalizované v bodoch [-2,5; -2,5] a [2,5; 2,5], je zaujímavejší. Dôvodom je, že táto technika je založená na odhade gradientu a ten sa vždy vytvára na základe najväčšieho klesania funkcie v priestore. Preto bude mať algoritmus tendenciu presúvať body v smere ku globálnemu minimu. V tabuľke 8 sú zhrnuté výsledky aplikácie tejto metódy.

Tabuľka 9: Výsledky implementácie logaritmickej bariérovej metódy na metódu odhadu gradientu pomocou kružnice s kruhovými prekážkami so stredmi v bodoch [-2,5; -2,5] a [2,5; 2,5]

Počiatočný bod	**Čas [s]**	**Iterácie**	**Nájdené minimum**	**Vzdialenosť**
[0; 0]	48,8	10	[2,8122; -2,8446]	0,0880
[0; 0]	45,8	9	[3,0393; -2,8991]	0,2924
[0; 0]	69,8	15	[2,6900; -2,8805]	0,0614

Ako môžeme vidieť, výstupom z programu je vždy uspokojivo presne lokalizované lokálne minimum. I keď spočiatku gradient smeroval ku globálnemu minimu, vďaka matematickým

operáciam implementovaným logaritmickou bariérovou metódou, algoritmus nakoniec skonvergoval do lokálneho extrému (Obrázok 14).



Obr. 14: Konvergenciu do minima pri použití logaritmickej bariérovej metódy na metódu odhadu gradientu pomocou kružnice so stredmi prekážok v bodoch [-2,5; -2,5] a [2,5; 2,5]

5 Výsledky

Výsledkom práce je konečné porovnanie všetkých použitých metód s cieľom výberu tej najvhodnejšej na implementáciu na robotické zariadenie. Postupne sme aplikovali rôzne numerické metódy na riešenie optimalizačného problému, počínajúc metódou Luus-Jaakola a metódou Simulovaného žíhania, ktoré radíme medzi stochastické metódy, teda metódy využívajúce náhodné vyhľadávanie pomocou iterácií. Tieto metódy, vzhľadom na voľbu vhodných parametrov, dosiahli presné riešenie, pričom bilancia rýchlosti a presnosti nájdeného riešenia bola uspokojivá. Výhodou obidvoch metód je prehľadávanie celého priestoru v rámci ohraničení, čo zabezpečuje to, že tieto metódy nezostanú zaseknuté v lokálnom minime, ale sú schopné skonvergovať až do globálneho minima. Vzájomným porovnaním stochastických metód môžeme prehlásiť, že technika Luus-Jaakola je rýchlejšia, ale má väčšiu tendenciu zaseknúť sa v lokálnom minime. V prípade Simulovaného žíhania, si rýchlosť volíme ako parameter, ale keďže pri vysokých teplotách prehľadáva celý priestor, takmer v každom prípade bolo výsledkom globálne minimum.

V prípade metód odhadujúcich gradient, sme začali všeobecne známym algoritmom Nelder-Mead, ktorý pracuje s troma bodmi vytvárajúcimi trojuholník a ich rôznou úpravou. Na podobnom princípe funguje aj metóda odhadu gradientu pomocou trojuholníka, ktorú sme si vytvorili, ale líši sa behom programu. Algoritmus Nelder-Mead ale pracuje s viacerými matematickými operáciami, čo spôsobuje, že čas konvergencie do minima sa značne predĺži vzhľadom na našu podobnú metódou, založenú na porovnávaní funkčných hodnôt troch bodov. Oba algoritmy sú schopné nájsť minimum, ale často hovoríme o lokálnom minime. Zistili sme, že to, do akého minima sa dostaneme behom algoritmu, závisí od počiatočnej polohy trojuholníka. Poslednou technikou je odhad gradientu pomocou kružnice, ktorá je oveľa presnejšia ako predošlé dve, pretože má k dispozícii merania z viacerých bodov, a tým dokáže presnejšie odhadnúť smer lokálneho klesania. Problém zaseknutia je aktuálny pre všetky metódy odhadu gradientu, keďže sa posúvajú smerom najväčšieho lokálneho poklesu, ktorý nemusí predstavovať cestu ku globálnemu minimu.

Celkové porovnanie použitých metód pri neohraničenej optimalizácii sa nachádza v tabuľke 10. Časovo najmenej vyhovujúca sa ukázala metóda Simulovaného žíhania, ale jej presnosť bola najlepšia. Naopak, najkratší beh programu sme zaznamenali pri metóde Nelder-Mead a metóde odhadu gradientu pomocou kružnice. Algoritmus Nelder-Mead trvá síce o niečo kratšie, ale naopak, vyžaduje viac iterácii a teda matematických operácii. V prípade odhadu gradientu pomocou kružnice je čas o niečo dlhší z dôvodou použitia metódy backtracking, kedy senzor potreboval čas na presun do prepokladaného kroku a jeho preskúmanie. Presnosť všetkých metód bola primeraná vzhľadom na veľkosť a farebné rozlíšenie kontúrového grafu. Vzhľadom na experimenty, vykonané na 2D Plottri,

za najvhodnejšiu vyberáme práve metódu odhadu gradientu pomocou kružnice.

Tabuľka 10: Výsledne porovnanie použitých metód v neohraničenej optimalizácii

Metóda	**Čas [s]**	**Iterácie**	**Vzdialenosť**
Luus-Jaakola	72,6	68	0,2056
Simulované žíhanie	102,5	60	0,1270
Nelder-Mead	31,5	12	0,2309
Odhad gradientu pomocou trojuholníka	49,9	14	0,2649
Odhad gradientu pomocou kružnice	37,5	7	0,3078

V prípade, že aplikujeme problém do praxe, je pre nás uspokojivé nájdenie aj lokálneho minima. Takéto miesta môžu predstavovať lokálne úniky radiácie, lokálne požiare alebo viacero lokálnych miest kontaminácie a ich lokalizácia je pre nás rovnako dôležitá ako lokalizácia globálneho miesta úniku, kontaminácie, požiaru. Ak sa na to dívame z tohto pohľadu, najlepšou možnosťou pre nás je riešenie pomocou odhadu gradientu kružnicou. Bilancia času a presnosti pri tejto technike vychádza najvýhodnejšie. V prípade, že cieľom je striktné dosiahnutie globálneho minima funkcie, v tom prípade sme získali výsledky, ktoré za najlepšiu metódu určujú metódu Simulovaného žíhania. Samozrejme, každá metóda je závislá na svojich parametroch a jej efektivita závisí aj od typu problému, ktorý riešime. Dôležitý je teda priebeh skúmanej metódy. Pri implementácii do praxe, ale preferujeme riešenie pomocou odhadu gradientu.

V prípade, že v priestore, ktorý preskúmavame, sa nachádzajú prekážky, kam sa nevieme fyzicky dostať a merať údaje, potrebujeme sa takýmto prekážkam vyhnúť. Kvôli spomínanému problému sme aplikovali bariérovú metódu ohraničenej optimalizácie, ktorá nám všetky metódy povýšila o kategóriu vyššie. Vďaka tejto metóde, sme boli schopní dosiahnuť minimum a zároveň vyhnúť sa prekážkam v priestore, výsledkom čoho bol nový algoritmus implementovateľný do reálneho prostredia. Zhrnutie výsledkov použitia logaritmickej bariérovej metódy máme možnosť vidieť v tabuľke 11. Lokalizáciu minima sme rýchlejšie dosiahli po použití tejto techniky na metódu odhadu gradientu pomocou kružnice, pretože sa neprehľadáva celý priestor. Vidíme, že aj iteračne je to menej náročný algoritmus. V prípade presnosti vyšla lepšie Luus-Jaakola, ale obe metódy sú dostatočne presné.

Tabuľka 11: Výsledne porovnanie použitých metód v ohraničenej optimalizácii

Metóda	**Čas [s]**	**Iterácie**	**Vzdialenosť**
Luus-Jaakola	48,5	35	0,1912
Odhad gradientu pomocou kružnice	46,3	9	0,2310

Záver

Cieľom práce bolo lokalizovanie minima v ohraničenom priestore bez poznania predpisu funkcie vývoja určitej vlastnosti, ktorú v priestore sledujeme. Aplikovali sme na tento problém päť rôznych metód riešenia, pričom sme získali vhodné parametre každej jednej techniky vzhľadom na používané robotické zariadenie. Vďaka týmto metódam, sme nadobudli informáciu o lokalizácii minima v prehľadávanom priestore a zároveň sme zhodnotili, ktorá z metód vracia najlepšie výsledky berúc do úvahy bilanciu rýchlosti a presnosti riešenia problému ohraničeného stanovenými podmienkami. Toto hodnotenie vyplýva z uskutočnených experimentov na zariadení, ktorým bol 2D Plotter so senzorom svietivosti pohybujúcim sa ponad kontúrový graf funkcie zobrazenom na grafe. Dokázali sme metódy prispôsobiť na riešenie problému aj v priestore, kde sa nachádzajú prekážky zabraňujúce merať funkčné hodnoty bodov v danom mieste. Vďaka matematickým prepočtom sa 2D Plotter dokázal vyhnúť obmedzeniam a presne a rýchlo lokalizovať extrém funkcie. Odhliadnuc od prekážok, každá z použitých metód skonvergovala do minima s určitou presnosťou. Znalosti z tejto práce by sme mohli využiť pri riešení reálnych problémov v prevádzkach, akými sú úniky plynov, tepelné úniky, či lokalizácia požiarov.

Literatúra

[1] Juraj Oravec, Martin Kalúz, Peter Bakaráč, and Monika Bakošová. Improvements of educational process of automation and optimization using 2d plotter. *IFAC-PapersOnLine*, 49(6):16 – 21, 2016. 11th IFAC Symposium on Advances in Control Education ACE 2016.

[2] Schumacher A., Vietor Th., Fiebig S., Bletzinger K.-U., and Maute K. *Advances in Structural and Multidisciplinary Optimization*. Springer International Publisher, 2017.

[3] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, Jul 2013.

[4] R. Luus. Use of luus–jaakola optimization procedure for singular optimal control problems. *Nonlinear Analysis: Theory, Methods & Applications*, 47(8):5647 – 5658, 2001. Proceedings of the Third World Congress of Nonlinear Analysts.

[5] Mathieu Gerber and Luke Bornn. Convergence results for a class of time-varying simulated annealing algorithms. *Stochastic Processes and their Applications*, 128(4):1073 – 1094, 2018.

[6] S. Anily and A. Federgruen. Simulated annealing methods with general acceptance probabilities. *Journal of Applied Probability*, 24(3):657–667, 1987.

[7] Mohamed El Yafrani and Belaïd Ahiod. Efficiently solving the traveling thief problem using hill climbing and simulated annealing. *Information Sciences*, 432:231 – 244, 2018.

[8] Marco A. Luersen and Rodolphe Le Riche. Globalized Nelder–Mead method for engineering optimization. *Computers & Structures*, 82(23):2251 – 2260, 2004. Computational Structures Technology.

[9] Kuo-Hao Chang. Stochastic Nelder–Mead simplex method – A new globally convergent direct search method for simulation optimization. *European Journal of Operational Research*, 220(3):684 – 694, 2012.

[10] Bingbing Li, Li Ding, Mark Rajai, Di Hu, and Shengzi Zheng. Backtracking algorithm-based disassembly sequence planning. *Procedia CIRP*, 69:932 – 937, 2018. 25th CIRP Life Cycle Engineering (LCE) Conference, 30 April – 2 May 2018, Copenhagen, Denmark.

[11] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Recent developments in constrained optimization. *Journal of Computational and Applied Mathematics*, 22(2):257 – 270, 1988.