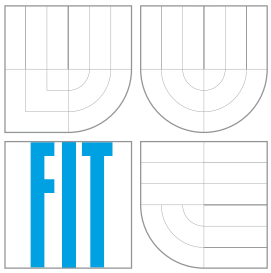
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ NÁVRH FILTRŮ PRO ZPRACOVÁNÍ SIGNÁLŮ

EVOLUTIONARY DESIGN OF FILTERS FOR SIGNAL PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ DOBIŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROLAND DOBAI, Ph.D.

BRNO 2016

Zadání bakalářské práce/16885/2015/xdobis00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Dobiš Tomáš**
Obor: Informační technologie
Téma: **Evoluční návrh filtrů pro zpracování signálů**
Evolutionary Design of Filters for Signal Processing
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s evolučním návrhem, problematikou zpracovaní signálu a adaptivních filtrů.
2. Navrhněte a implementujte metodu založenou na evolučním návrhu pro zpracování signálů.
3. Ověřte funkčnost implementované metody na zadaných úlohách.
4. Zhodnoťte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dobai Roland, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Kalmanův filter slouží na filtrovaní signálů na základe konfigurace filtra a odhadu hodnot. Jeho konfigurace je však obtížná a vyžaduje zkušenosti matematika. Tato práce se věnuje implementaci metody pro spracování signálů s využitím Kartézského genetického programování, kde výhodou je automatizovaná konfigurace filtra. Výsledná metoda je porovnána na vícerých testovacích příkladech s Kalmanovým filtrem. Z výsledků je možno usoudit, že implementovaná metoda funguje porovnatelne efektivne při filtrovaní periodických a exponenciálních vstupech, a při filtrování konstantních vstupů funguje mnohem efektivněji než Kalmanův filter.

Abstract

Kalman filter is used for signal filtering dependent on filter configuration and prediction of values. It's configuration is difficult and requires experiences of mathematician. This thesis deals with implementation of method for signal processing with use of Cartesian genetic programming, which advantage includes the automated configuration of filter. Final method is compared on multiple testing examples with Kalman filter. From results we can infer, that implemented method works comparatively efficient on periodic and exponential signal inputs, and works significantly better on constant signal inputs than Kalman filter.

Klíčová slova

adaptivní filter, evoluční návrh, Kalmanův filter, Kartézské genetické programování, umělá inteligence

Keywords

adaptive filter, evolutionary design, Kalman filter, Cartesian genetic programming, artificial intelligence

Citace

Tomáš Dobiš: Evoluční návrh filtrů pro zpracování signálů, bakalářská práce, Brno, FIT VUT v Brně, 2016

Evoluční návrh filtrů pro zpracování signálů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Rolanda Dobaia Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Dobiš
15. května 2016

Poděkování

Ďakujem Ing. Rolandovi Dobaiovi Ph.D. za vedenie, rady a trpezlivosť.

Obsah

Kapitola 1

Úvod

Evolučná teória, ktorá predstavila princíp prirodzeného výberu, má základy v knihe Pôvod druhov vydanej v roku 1859 Charlesom Darwinom [7]. Postupne sa o tejto teórii vytvoril vedný odbor evolučná biológia. V 20. storočí sa poznatky tohto odboru využili v počítačových vedách a vznikol odbor zaoberajúci sa aplikovaním princípov evolúcie pri vývoji software, nazvaný evolučný návrh [10].
Signály, ktoré predstavujú veličinu reálneho sveta sú často znečistené vonkajšími vplyvmi alebo nepresnosťou merania. Aby signály lepšie zodpovedali veličinám, ktoré v skutočnosti predstavujú filtrujeme ich rôznymi metódami. Kalmanov filter je algoritmus, ktorého cieľom je odfiltrovať zo vstupných dát neželané súčasti, a vytvoriť veličinu, odmeranými rôznymi nepresnými senzormi. Síce bol navrhnutý v 60. rokoch minulého storočia, ale je dodnes používaný vo viacerých odvetviach, napr. navigácia, počítačové videnie [6].
Táto práca sa zaoberá využitím evolučného návrhu pre spracovanie signálov, konkrétne filtrovaním signálov od nežiadaného šumu. K tomuto účelu v súčasnosti používame adaptívne filtre ako napríklad Kalmanov filter. Výhodou využitia evolučného návrhu je, že výsledná metóda bude automatizovaná.
Cieľom práce je vytvorenie a otestovanie metódy pre filtrovanie signálov a následné porovnanie metódy s Kalmanovým filtrom na viacerých typoch signálu.
V prvej kapitole 2 sa práca zaoberá signálmi a ich spracovaním. Popísaný je postup filtrovania signálov so zameraním na adaptívne filtre. Pozornosť je venovaná Kalmanovmu filtru, ktorý je použitý pre porovnanie výslednej metódy. Druhá kapitola 3 sa venuje evolučnému návrhu a jeho použitiu. Je v nej popísaný evolučný algoritmus a jeho súčasti. Podrobne je opísané Kartézske genetické programovanie. V nasledujúcej tretej kapitole 4 je opísaný návrh a implementácia evolučného algoritmu pre filtrovanie signálov. Kapitola sa tiež venuje implementačnému prostrediu a použitým knižniciam. V štvrtej kapitole 5 je zdokumentovaný priebeh testovania metódy a porovnanie výsledkov s Kalmanovým filtrom. Je použitých viacero typov vstupu a konfigurácií metódy. V kapitole 6 sú zhrnuté výsledky práce a je navrhnuté ďalšie možné smerovanie vývoja.

Kapitola 2

Signály a ich spracovanie

Podľa matematickej definície je signál funkcia, ktorá prevádza nezávislú premennú z množiny T na hodnoty množiny A [9].
Podľa charakteru množiny T delíme signály na:

- signály s diskrétnym časom, $n \in \mathbb{Z}$,
- signály so spojitým časom, $t \in \mathbb{R}$.

Pre túto prácu je ďalej podstatné delenie signálov podľa vlastností systému, ktorého je signál výstupom, a kde poznáme: analógové signály a digitálne signály.

2.1 Spracovanie signálov

Signál, ktorý nameriame, sa často odlišuje od reálneho signálu z dôvodov chýb merania spôsobených prostredím v ktorom sa signál šíri a nepresnými nástrojmi na meranie. K účelu úpravy signálov používame filtre. Funkciu filtra je úprava signálu s cieľom zdôrazniť, nájsť alebo potlačiť súčasti signálu [9]. Filtre delíme podľa rôznych vlastností. Podľa toho aký signál filtrujú, ich delíme na analógové a digitálne, respektíve spojité a diskrétne. Ďalšie vlastnosti filtrov ktoré odlišujeme sú:

- typ signálu - lineárny alebo nelineárny,
- pri spojitých filtroch - pasívne alebo aktívne,
- pri diskrétnych filtroch - s konečnou odozvou (FIR) alebo s nekonečnou odozvou (IIR).

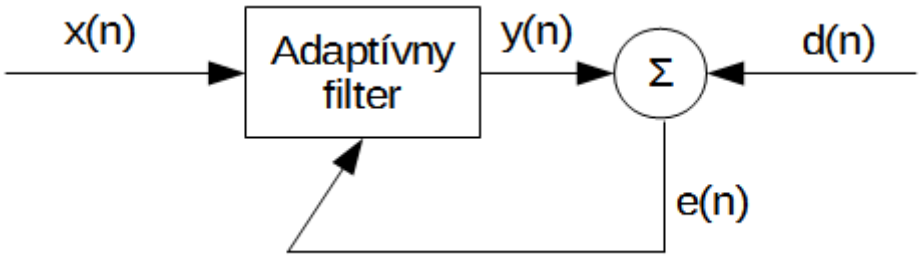
Využitie filtrov

Filtrovanie signálov sa využíva v mnohých oblastiach ako napríklad [14]:

- spracovanie zvuku - rozpoznávanie reči,
- spracovanie obrazu - kardiograf,
- echo lokácia - radar, sonar,
- telekomunikácie - kompresia signálov.

2.2 Adaptívne filtre

Adaptívne filtre sú špecifickou kategóriou filtrov, ktoré menia svoje parametre v čase v závislosti na vlastnostiach filtrovaného signálu [14]. Parametre ovplyvňuje optimalizačná funkcia. Vzhľadom na ich komplexnosť je väčšina adaptívnych filtrov digitálna. Jedným z použití adaptívnych filtrov je rušenie echa a šumu. Šum vzniká v dôsledku chýb merania spôsobených nepresnými nástrojmi a vplyvom prostredia, v ktorom sa signál šíri. Funkciou adaptívneho filtra je detekcia a potlačenie šumu v signále. Konkrétne filtre, ktoré sa k tomuto účelu používajú sú Kalmanov a Wienerov filter. Oblasti, kde sú tieto majoritne používané sú telekomunikácie a medicína [14] [13].



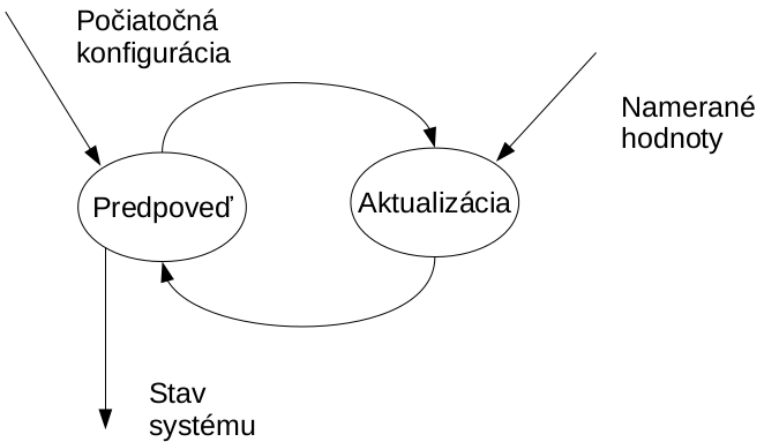
Obr. 1 Princíp fungovania adaptívneho filtra

Na Obr. 1 je vidieť signál $x(n)$, ktorý je vstupom pre adaptívny filter. Výstupom adaptívneho filtra je signál $y(n)$, ktorý je ovplyvnený parametrami filtra. Signál $d(n)$, je signál, ktorý je požadovaným výstupom pre filter. Rozdielom signálov $y(n)$ a $d(n)$, vypočítame chybový signál $e(n)$. Tento signál je spätnou väzbou pre filter, na základe ktorej filter upraví parametre filtrovania.

2.3 Kalmanov filter

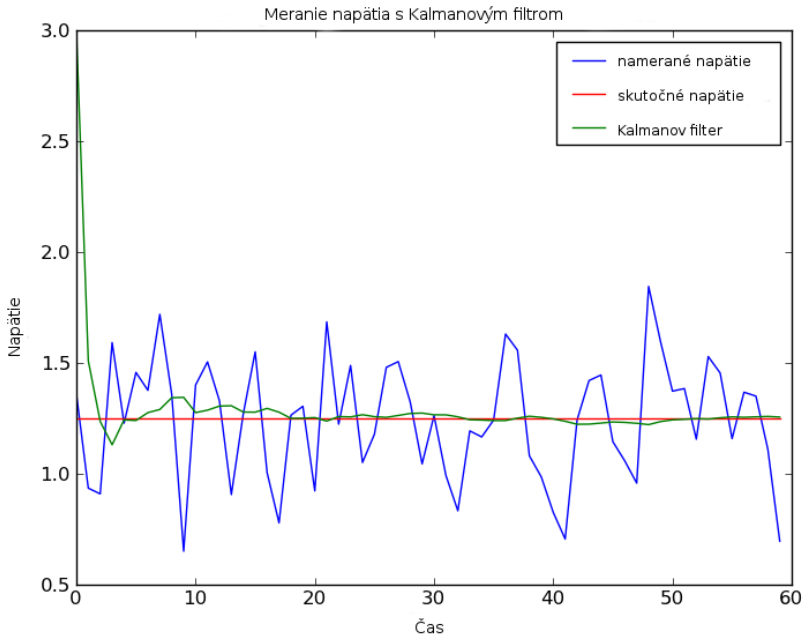
Kalmanov filter je adaptívny filter, ktorý je považovaný za optimálny pre 1-dimenzionálne lineárne diskrétne systémy z hľadiska nízkych výpočetných nárokov a presnosti [11]. Bol navrhnutý Rudolfom Emilom Kalmanom v 60. rokoch minulého storočia. Kalmanov filter je rekurzívny filter, ktorý odhaduje stav systému na základe vstupných hodnôt, ktoré sú ovplyvnené nepresnosťami merania a prostredím. Má rozsiahle použitie: sledovanie objektov, navigácia, ekonómia, počítačové videnie, etc [6].
Formálna definícia filtra hovorí, že to je algoritmus umožňujúci presné odvodenie pri lineárnom dynamickom systéme, ktorého Bayesianský model napodobňuje skrytý Markov model, ale stavový priestor latentných premenných je spojitý [11].
Môžeme povedať, že Kalmanov filter odstráni neželané súčasti signálu, na základe preddefinovaného modelu systému, ktorý zodpovedá danému systému.
Existujú modifikácie Kalmanovho filtra. Patrí medzi nich napríklad Rozšírený Kalmanov filter (Extended Kalman Filter), ktorý sa používa pri nelineárnych systémoch.
Algoritmus Kalmanovho filtra má 2 základné kroky, ktoré sa opakujú v cykle, ako je znázornené na Obr. 2. Na začiatku behu je definovaná začiatočná konfigurácia filtra, a je volaná funkcia `Predpoveď`, ktorej úlohou je odhadnúť skutočný stav systému. Následne sú namerané hodnoty o stave systému použité ako parameter funkcie `Aktualizácia`, ktorá modifikuje parametre filtra za účelom dosiahnuť presnejší odhad skutočného stavu systému.

Stav systému je výstupom funkcie `Predpoveď`. Cyklus sa opakuje kým nie sú spracované všetky vstupné hodnoty.



Obr. 2 Diagram Kalmanov filter

Na Obr. 3 vidíme príklad, kde sa napätie mení v čase. Červeným je znázornené skutočná hodnota napätia, a modrým nameraná hodnota napätia. Zeleným je označený výstup Kalmanovho filtra, ktorý po začiatočných iteráciách pomerne správne odhadne skutočné napätie z nameraných hodnôt, tj. rozdiel medzi zelenou a červenou krivkou sa časom zmenšuje.



Obr. 3 Porovnanie vstupného a výstupného signálu Kalmanovho filtra

Konfigurácia Kalmanovho filtra

Konfigurácia Kalmanovho filtra spočíva v nastavení premenných:

- X - počiatočný odhad,
- F - matica prechodov stavov,
- H - funkcia merania,
- Q - neurčitosť procesu,
- P - matica kovariancie,
- R - neurčitosť merania.

Následne je spustený hlavný cyklus, v ktorom sa opakujú 2 kroky ako je znázornené na Obr. 2.
Konfigurácia Kalmanovho filtra je komplexná úloha, ktorá vyžaduje znalosť problematiky adaptívnych filtrov, a aj signálov ktoré ideme filtrovať. Preto sa v tejto práci budeme zaoberať návrhom a implementáciou samočinnej konfigurácie filtrov použitím evolučných metód.

Kapitola 3

Evolučný návrh a jeho využitie

Evolučný návrh je súčasťou odboru umelej inteligencie. Je založený na postupnom vývoji riešenia, ktoré po nastavení začiatočnej konfigurácie, cyklicky vylepšuje až do bodu v ktorom je dostatočné. Táto vlastnosť je inšpirovaná princípom prirodzeného výberu, ktorý sa vyskytuje v prírode.
Príkladom použitia evolučného návrhu v počítačovej vede je návrh rôznorodých pevných objektov širokého využitia [5].

3.1 Evolučný algoritmus

Evolučný algoritmus je všeobecný termín, ktorý popisuje stochastické prehľadávacie algoritmy, ktoré používajú populáciu kandidátnych riešení a biológiou inšpirované operátory za účelom tvorby nových kandidátnych riešení.
Pre efektivitu evolučného algoritmu sú podstatné vlastnosti: reprezentácia problému, konštrukcia fitness funkcie a zvolenie genetických operátorov.

Popis evolučného algoritmu

Pre nájdenie optimálneho riešenia evolučného algoritmu je nutné zvoliť vhodné parametre, a vhodne použiť genetické operátory. Prvým krokom je vytvorenie počiatočnej generácie jedincov. Populácia môže byť vytvorená náhodne, ale vhodnejšie je využiť znalosť daného problému. Veľkosť populácie môže byť statická alebo dynamická, znova záleží od konkrétneho problému, ktorý riešime [10].
Nasleduje priradenie fitness hodnôt jednotlivým jedincom populácie. Na tento účel použijeme fitness funkciu. Zvolenie vhodnej fitness funkcie je dôležité pre správne fungovanie evolučného algoritmu, aby vyhľadávanie pokračovalo v žiadanom smere, a aby boli nájdené vhodné kandidátne riešenia.
Po ohodnotení počiatočnej populácie môže začať hlavný cyklus algoritmu. Jeden prechod cyklom sa nazýva generácia. Na začiatku prechodu cyklom si použitím selekčného operátora, alebo ich kombináciou, zvolíme rodiča alebo rodičov z počiatočnej populácie. Noví jedinci sú vytvorení použitím genetických operátorov kríženia a mutácie.
Následne sú noví jedinci ohodnotení fitness funkciou, a vyberieme z nich jedincov do novej populácie. Jedincov vyberieme aj zo starej populácie, tj. nová populácia obsahuje potomkov, aj rodičov.

Vytvorením novej populácie sa končí generácia, a nasleduje kontrola podmienky, ktorá rozhodne či bude nasledovať ďalšia generácia. Podmienka je daná buď časom evolúcie, alebo konkrétnou vlastnosťou populácie ako napríklad hodnota fitness.

Reprezentácia problému

Pre zakódovanie kandidátneho riešenia používame termín chromozóm alebo tiež genotyp. Pri použití binárnej reprezentácie bude chromozóm reprezentovaný binárnym reťazcom, kde každý bit predstavuje jeden gén. Hodnota bitu, tj. nula alebo jedna sa nazýva alela. Pozícia bitu je označená ako locus. Pre zakódovanie jedinca je možné použiť aj iné reprezentácie, napríklad celo-číselná [10]. Dĺžka chromozómu môže byť pevná alebo premenlivá. Pre jednoduchosť sa používa najmä priame zobrazenie, kde každý gén predstavuje parameter riešenia.

Fitness funkcia

Fitness funkcia je spôsob určenia fitness hodnoty jedinca. Fitness hodnota vyjadruje zdatnosť jedinca v populácii. Existuje viacero spôsobov vyjadrenia fitness hodnoty [10]:

- Hrubá - hodnoty sú prirodzené doméne problému.
- Štandardizovaná - vyjadrenie hrubej fitness, kde je menšia hodnota žiadanejšia.
- Prispôsobená - prevrátená hodnota štandardizovanej fitness a čísla jeden.
- Normalizovaná - podiel hrubej fitness a sumy všetkých hrubých fitness v populácii.

Genetické operátory

Genetické operátory sa používajú pre výber a tvorbu nových jedincov v populácii [16]. Používajú sa genetické operátory: kríženie, mutácia a selekcia.

Kríženie

Operátor kríženia vymení časti génov medzi dvomi alebo viacerými chromozómami. Typicky prebieha medzi dvoma rodičmi, z ktorých vzniknú dvaja potomci. Ak je použitý operátor kríženia býva pravdepodobnosť kríženia vysoká, napríklad $p_c = 70\%$ [10]. V tomto prípade potom zvyšok novej populácie tvoria kópie a mutácie rodičov.

Mutácia

Tento operátor mutuje vybraného jedinca. Napríklad, pri binárnej reprezentácii sa realizuje ako negácia určitého počtu bitov. Mutácia býva typicky použitá ako doplnkový genetický operátor, preto pravdepodobnosť mutácie býva veľmi nízka, napríklad $p_m = 0.5\%$ [10].

Selekcia

Selekcia je operátor používaný pre výber jedincov k reprodukcii. Je nutné zabezpečiť splnenie dvoch protichodných cieľov. Sú to rôznorodosť populácie a preferovanie jedincov s najvyššou fitness hodnotou. Existujú nasledujúce druhy selekčných metód, ktoré sa v praxi kombinujú [10].

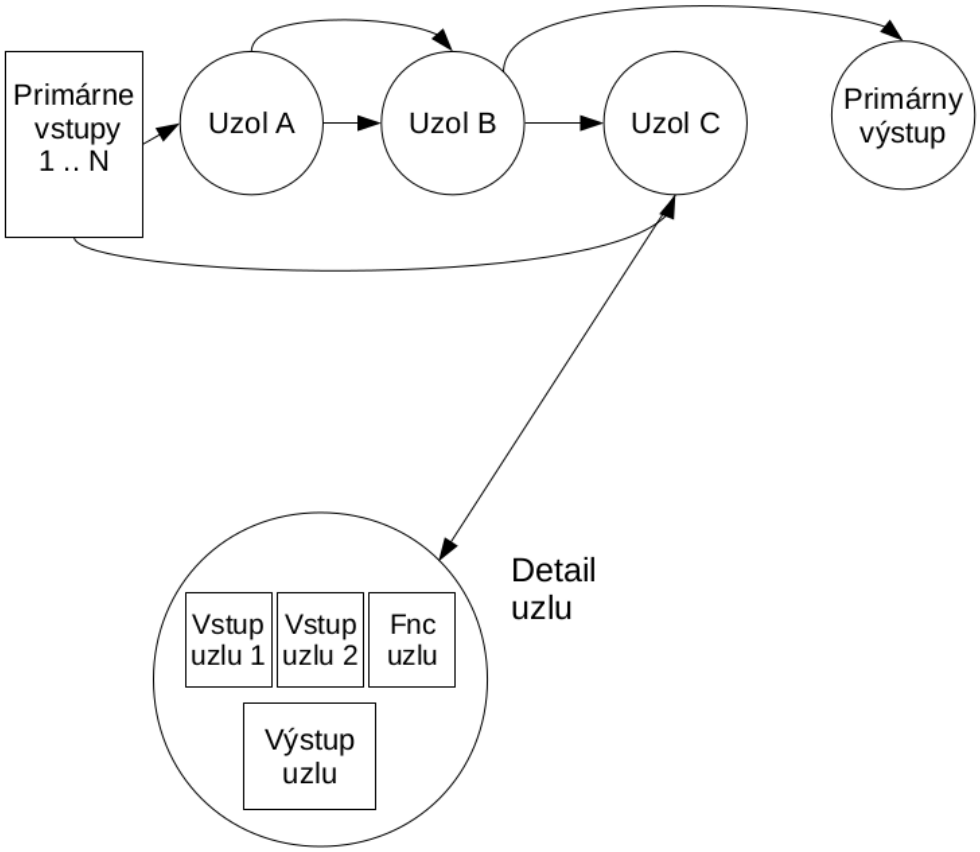
- Deterministická - výber určitého počtu jedincov s najvyššou fitness hodnotou.
- Turnajová - náhodný výber jedincov, z ktorých je následne vybraný najlepší.
- Proporcionálna - pravdepodobnosť výberu je priamo úmerná absolútnej fitness hodnote.
- Podla poradia - zoradí jedincov podla fitness a následne vyberá s cieľom zabrániť degenerácii (vývoj, ktorý nenapreduje k určenému cieľu).

3.2 Genetické programovanie

Genetické programovanie patrí do odboru umelej inteligencie. Je to metóda, v ktorej je program zakódovaný ako množina génov, a následne modifikovaný použitím evolučného algoritmu.

Kartézske genetické programovanie

Kartézske genetické programovanie je typ genetického programovania, kde kandidátne riešenia sú reprezentované celo-číselným reťazcom pevnej dĺžky, ktorý mapuje orientovaný graf pevnej veľkosti [17].
Kandidátne riešenie je zakódované ako pole programovateľných uzlov veľkosti $S \times R$, kde S je počet stĺpcov a R je počet riadkov. Každý uzol má pevný počet vstupov V_I a výstupov V_O. Typicky má uzol dva vstupy, jeden výstup, a implementuje jednu z preddefinovaných primitívnych funkcií. Každý vstup uzlu V_I môže privádzať výstup z predchádzajúcich L uzlov (tento parameter je v literatúre označený ako l-back) alebo primárny vstup.
Pole uzlov je implementované ako celo-číselné pole o veľkosti $S \times R \times (V_I + 1) + O$, kde O je počet primárnych výstupov [17]. Chromozóm, ktorý je reprezentovaný polom má pevnú dĺžku, ale vzhľadom na to, že nie všetky uzly musia byť použité, dĺžka fenotypu (veľkosť jedinca) je dynamická.
Pre učenie je použitá množina trénovacích vektorov T. Prvá časť vektora T_A je primárnym vstupom programu. Druhá časť vektora T_B je riešenie, ktoré chceme dosiahnuť postupným vývinom. Fitness funkcia počíta rozdiel medzi kandidátnym riešením a T_B, a túto hodnotu vyjadrí ako fitness hodnotu. Cieľom evolučného cyklu je minimalizovať rozdiel medzi fitness hodnotou a T_B. V klasickej implementácii je použitý iba genetický operátor mutácie.
Kartézske genetické programovanie je využité v mnohých oblastiach. Napríklad: riešenie problémov symbolickej regresie, návrh neurónových sietí, návrh číslicových filtrov, návrh logických obvodov [17].



Obr. 4 Kandidátne riešenie reprezentované prepojenými uzlami

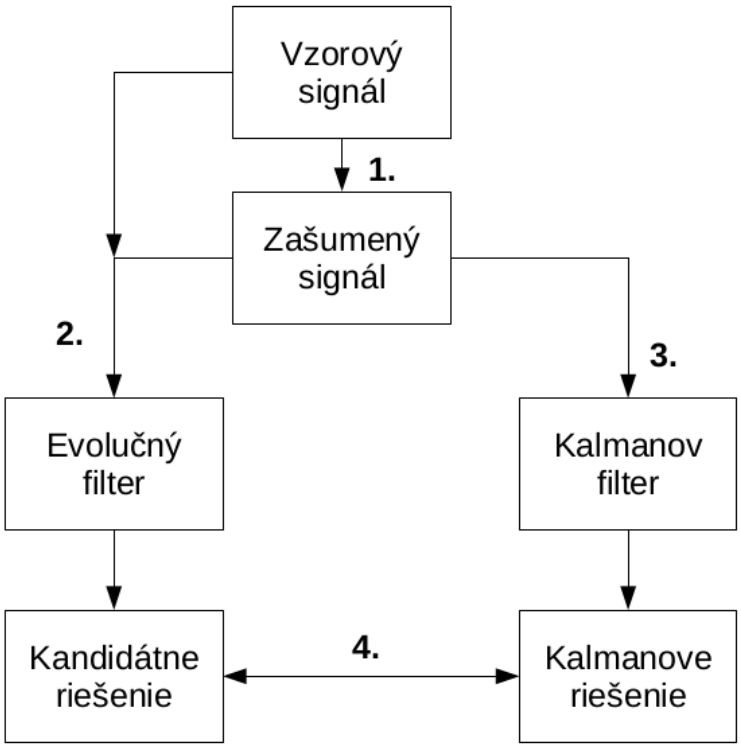
Príklad popisu kandidátneho rišenia

Na Obr. 4 je znázornený chromozóm s 3 uzlami (A, B, C) a 1 primárnym výstupom. Uzly sú usporiadané do konfigurácie 3 stĺpce $\times 1$ riadok. Každý uzol tvoria 2 vstupy (orientovaná hrana smerom k uzlu) a typ primitívnej funkcie. Okrem nich má uzol svoj výstup (orientovaná hrana smerujúca od uzla), ktorý je však implementovaný separátne. Tento chromozóm by sme implementovali ako reťazec o dĺžke (3 uzly $\times$(2 vstupy uzlu +1 výstup uzlu)) + 1 primárny výstup = 10.
Primárne vstupy sú zavedené do uzlov A a C. Uzol B má ako vstup pripojený výstup uzlu A, a podobne uzol C má na vstup pripojený výstup uzlu B. Na primárny výstup je napojený výstup uzlu B, z čoho vyplýva že uzol C je vynechaný a jeho hodnoty nie sú v tomto prípade použité.

Kapitola 4

Návrh a implementácia metódy evolučného návrhu pre vývoj filtrov

Navrhnutá metóda má za cieľ spracovať vstupný signál tak, že z neho odstráni neželaný šum. Vstupný signál je generovaný použitím matematických funkcií. Navrhnutá metóda využíva princípy evolučného algoritmu a Kartézskeho genetického programovania. Výsledky metódy sú porovnané s konvenčným adaptívnym filtrom, konkrétne Kalmanovým filtrom.



Obr. 5 Všeobecný návrh metódy

1. **Generovanie trénovacích dát**
   Generovanie trénovacích dát je prvým vykonávaným krokom v navrhovanej metóde, znázornenej na Obr. 5. Podľa definovaných parametrov sa vygenerujú dvojice hodnôt.

Prvá hodnota A bude vstupná hodnota navrhnutej metódy a aj Kalmanovho filtra. Druhá hodnota B bude použitá len v navrhnutej metóde, a bude to hodnota, ktorú treba dosiahnuť aplikovaním metódy na hodnotu A. Hodnota A bude dosiahnutá sčítaním hodnoty B, a náhodnej hodnoty od 0 do 1 podľa normálneho rozdelenia, čo reprezentuje šum, teda B je zašumené A. Hodnoty B budú vygenerované podľa predvolenej matematickej funkcie. Okrem typu funkcie je možné zvoliť v navrhnutej metóde začiatok, koniec a krok intervalu, v ktorom sú hodnoty generované.

2. **Spracovanie signálu navrhnutou metódou**
   Druhým krokom je filtrovanie vstupného signálu evolučným filtrom, ako je vidieť na Obr. 5. Na základe trénovacích dát a daných parametrov sa prefiltruje signál, s využitím evolučného algoritmu a Kartézskeho genetického programovania. Konfigurovateľné parametre sú veľkosť chromozómu (počet riadkov a stĺpcov), parameter l-back (udáva počet predchádzajúcich stĺpcov v chromozóme, ktorých výstupy môžu byť použité ako vstupy nasledujúcich stĺpcov), veľkosť populácie, počet generácií, počet použitých primitívnych funkcií a maximálny počet mutácii v chromozóme na jednu generáciu. Výsledkom metódy je pole hodnôt o veľkosti pola vstupných dát.

3. **Aplikácia Kalmanovho filtra**
   Nasleduje filtrovanie vstupného signálu Kalmanovým filtrom (Obr. 5). Vstupom filtra je pole hodnôt A vygenerované v prvej časti. Konfigurácia filtra je manuálna. Výsledkom je pole hodnôt o veľkosti pola vstupných hodnôt reprezentujúce odfiltrovaný signál.

4. **Porovnanie výsledných hodnôt**
   Polia, v ktorých sú uložené výsledky navrhnutej metódy a Kalmanovho filtra sú porovnané graficky, a numericky na základe ich rozdielu voči vstupným hodnotám A a B, ako je znázornené na Obr. 5.

Zvolené parametre pre navrhnutú metódu sú uvedené neskoršie.

4.1 Implementačné prostredie

Metóda je implementovaná v jazyku Python verzie 2.7.11. Python je imperatívny interpretovaný skriptovací jazyk. Používa silne dynamické typovanie. Metóda je navrhnutá s použitím objektovej orientácie.

Použité knižnice

Zo štandardných Python knižníc boli použité [3]:

- `random` - generovanie náhodných čísel pri tvorbe počiatočnej populácie,
- `json` - uloženie výsledných riešení vo formáte `json`,
- `math` - matematické funkcie pri generovaní trénovacích dát,
- `copy` - kopírovanie objektov pri operátore mutácie,
- `time` - systémové volania na meranie času,

- `os` - práca s adresármi a súbormi pri ukladaní výsledkov
- `threading` - práca s vláknami pri paralelizácii,
- `argparse` - spracovanie argumentov.

Ostatné použité knižnice:

- `NumPy` - knižnica pre vedecké výpočty použitá pre súčet dvoch signálov [15],
- `Matplotlib` - knižnica pre 2D kreslenie grafov s riešeniami [8],
- `FilterPy` - knižnica, ktorá implementuje Bayesianské filtre, menovite Kalmanov filter a jeho variácie [4].

4.2 Popis implementácie

Implementácia je rozdelená do 6 fáz. Parametre sú v prípade ich prítomnosti prečítané z príkazového riadku, inak sú použité predvolené hodnoty. Výstupom programu je adresár obsahujúci:

- súbor s príponou `.txt` s výpismi pre všetky vlákna,
- súbory s príponou `.txt` s výpismi z jednotlivých vlákien,
- súbor s príponou `.json` s vstupnými hodnotami (A),
- súbor s príponou `.json` s žiadanými hodnotami (B),
- súbor s príponou `.json` s riešením prednastaveného Kalmanovho filtra,
- súbory s príponou `.json` s riešeniami jednotlivých vlákien.

Zoznam tried a funkcií

Pomocné funkcie:

- `rangeIter(start, finish, step)` - iteruje po danom intervale,
- `getDiff(sol1, sol2)` - porovná dve polia čísel a ich rozdiel vyjadrí ako sumu $\sqrt{|sol1[n] - sol2[n]|}$,
- `generateData(ideal, noised, setup)` - podla parametru setup vygeneruje dve polia čísel a uloží ich do polí `ideal` a `noised`,
- `avgNum(a, b)` - vypočíta priemer čísel a a b.

Triedy:

- `BestThread()` - štatistická trieda zdieľaná vláknami,
- `Median()` - uloženie štatisticky významných hodnôt,
  - `getStats(data)` - nájdenie štatisticky významných hodnôt,
- `evolThread(threading.Thread)` - zapúzdrenie vlákna,

  - `run()` - spustenie vlákna,
- `chromPossibles()` - platné prepojenia chromozómu,
- `Individual()` - jedinec populácie evolučného algoritmu,
- `Stats()` - štatistická trieda špecifická pre vlákno,
  - `export()` - exportuje štatistické dáta do súboru,
- `evolSignal()` - navrhnutá filtrovacia metóda,
  - `evolve()` - filtrovanie signálu,
  - `fitness()` - výpočet fitness hodnoty,
  - `evaluate()` - výpočet nových výstupov chromozómu,
  - `mutate()` - mutácia jedinca,
- `Param()` - trieda s parametrami programu.

Rozčlenenie implementácie

Implementácia je rozdelená do nasledujúcich fáz:

1. spracovanie parametrov a príprava prostredia,
2. generovanie trénovacích dát,
3. príprava použitia vlákien,
4. inicializácia triedy `evolSignal()`,
5. vytvorenie a ohodnotenie počiatočnej populácie,
6. evolučný cyklus,
7. použitie Kalmanovho filtra,
8. porovnanie výsledných dát.

Spracovanie parametrov a príprava prostredia

V prvej fáze behu programu sa inicializuje trieda `Param`, ktorá obsahuje všetky údaje o konfigurácii programu. Najprv sú spracované argumenty príkazového riadku. Ak nie sú prečítané žiadne, sú použité predvolené konfiguračné hodnoty. Hodnoty, ktoré sa dajú nastaviť parametrami sú:

- počet riadkov a stĺpcov chromozómu,
- l-back chromozómu,
- veľkosť populácie,
- maximálny počet mutácií v rámci jedinca na jednu generáciu,
- počet použitých primitívnych funkcií,

- prefix názvu výstupného adresára,
- počet vlákien v ktorých beží metóda,
- typ funkcie generovaných dát,
- začiatok, koniec a krok intervalu trénovacích dát.

Trieda `Param` okrem toho obsahuje polia pre hodnoty trénovacích dát.
Ďalším krokom je vytvorenie adresára pre výstupy programu, a v rámci neho súboru pre výpisy zdieľaného všetkými vláknami. Do tohto súboru sa zapíše konfigurácia programu daná parametrami. Zapisovať sa doňho budú základné údaje o behoch jednotlivých vlákien.

Generovanie trénovacích dát

Druhá fáza sa zaoberá generovaním trénovacích dát. Dáta sú uložené v 2 poliach, ktoré sú v triede `Param`. V poliach sú uložené hodnoty typu `float`. Samotné generovanie implementuje funkcia `generateData`. Táto funkcia na základe zvolených parametrov zvolí interval, krok generácie a typ generovanej funkcie. Pre iteráciu po generačnom intervale je použitá funkcia `rangeIter`. Podporované typy generácie sú:

- sin - matematická funkcia sínus,
- cos - matematická funkcia cosínus,
- const - konštantná funkcia v hodnote 3,22,
- custom - neperiodická funkcia, ktorá je prvú polovicu intervalu lineárna $(2x - 1)$ a druhú kvadratická $((x - 8)^2)$,
- exp - exponenciálna funkcia podľa Salustowiczovho benchmarku [2].

Vygenerované hodnoty B sú uložené do pola `pure`. Pričítaním pola náhodných hodnôt od 0 do 1, rozdelených podla normálneho rozdelenia, je vytvorené druhé pole `polluted` obsahujúce hodnoty A. Pre uloženie a sčítavanie polí je použitá knižnica Numpy.

Príprava použitia vlákien

V tretej fáze je vytvorená trieda `BestThread`, ktorá obsahuje priebežne aktualizované údaje o najlepšom vlákne, súčet času všetkých behov v sekundách, a pole obsahujúce finálne fitness hodnoty všetkých vlákien. Následne sú inicializované vlákna triedy `evolThread`, ktoré obsahujú triedy `Param` a `BestThread`. Okrem toho obsahujú názov zdieľaného súboru pre výpisy `stat`. Je ošetrené manipulovanie so zdieľaným súborom pomocou zamykania, aby nedošlo k synchronizačným problémom. Po inicializácii `evolThread` sú spustené jednotlivé vlákna. Vlákna vykonajú inicializáciu triedy `evolSignal` argumentami `Param`, `BestThread` a `stat`. Následne je spustená metóda `evolve` triedy `evolSignal`, ktorá implementuje filtrovanie signálu. Po spustení počtu vlákien daného parametrom, program čaká, kým všetky vlákna ukončia svoju činnosť.

Inicializácia triedy `evolSignal`

Trieda `evolSignal`, ktorá beží v rámci `evolThread`, implementuje navrhnutú metódu filtrovania signálov použitím evolučného algoritmu a Kartézskeho genetického programovania. Trieda je inicializovaná nasledujúcimi parametrami:

- názov súboru pre výpisy,
- názov vlákna, ktoré zapúzdruje triedu,
- trieda `BestThread`,
- trieda `Param`.

Jej členom je aj trieda `Stats`, do ktorej sú zaznamenávané informácie o behu v danom vlákne. Okrem údajov o trénovacích dátach, trieda `Stats` obsahuje hodnoty počiatočnej a finálnej fitness hodnoty, a dĺžku evolučného behu v sekundách. Pre vykonávanie metódy `evolve` sú podstatné 2 polia objektov. Sú to polia `population` a `possibles`.
Pole `population` obsahuje parametrom daný počet jedincov triedy `Individual`. Členmi triedy `Individual` sú:

- `iID` - identifikácia jedinca,
- `chrom` - celo-číselné pole predstavujúce chromozóm,
- `output` - pole s výstupmi uzlov chromozómu,
- `solution` - pole hodnôt výstupného signálu,
- `fitVal` - celková fitness hodnota jedinca.

Prvkami pola `possibles` sú platné prepojenia uzlov chromozómu implementované triedou `chromPossibles`. Každá instancia triedy obsahuje:

- `pID` - identifikácia uzla,
- `value` - počet platných prepojení,
- `items` - pole s identifikáciou uzlov, ktoré je možné prepojiť.

V rámci inicializácie triedy `evolSignal`, sa po vytvorení premenných a polí, skopírujú potrebné štruktúry z triedy `Param`, a vypočítajú premenné pre zjednodušenie výpočtov. Sú to premenné: počet uzlov, súčet vstupov a výstupov uzla, veľkosť stĺpca a index primárneho výstupu jedinca.
Ďalším krokom je inicializovať a vypočítať platné prepojenia pre všetky stĺpce, na základe týchto premenných a parametru l-back. Parameter l-back určuje koľko predchádzajúcich stĺpcov je platných pre prepojenie. Každému stĺpcu je priradená instancia triedy `chromPossibles` opísaná vyššie.

Vytvorenie a ohodnotenie počiatočnej populácie

Primárna funkcia triedy `evolSignal`, tj. filtrovanie vstupného signálu je zapúzdrená v metóde `evolve`. Prvým krokom metódy je vytvorenie počiatočnej populácie o počte danom parametrom. Implementácia je realizovaná naplnením pola `population` instanciami triedy `Individual`. Následne je každému prvku pola `population` vygenerované pole `chrom`. Generovanie prebieha náhodným výberom z pola `possibles` pre príslušný stĺpec. Ďalším krokom metódy `evolve` je ohodnotenie a modifikovanie počiatočnej populácie `fitness` funkciou. Implementácia je zapúzdrená vo funkcii `fitness`.
Funkcia `fitness` najprv nastaví `fitness` hodnotu na 0, každému okrem najlepšieho jedinca, ktorý v prvom ohodnotení nie je. Následne vonkajší cyklus iteruje po trénovacích dvojiciach A a B, a vnútorný cyklus po jedincoch populácie. Ak nájde jedinca s najlepšou `fitness` hodnotou, pokračuje v ďalšej iterácii. Pri prvom ohodnotení nakopíruje do polí `output` jedincov hodnoty A. Pri ďalších behoch sa kopírovanie deje s 50% pravdepodobnosťou určenou na základe experimentov. Potom je volaná metóda `evaluate`, ktorá vypočíta nové hodnoty do pola `output`.
Metóda `evaluate` iteruje po uzloch v poli `chrom`. Na základe hodnôt z pola `chrom` daného jedinca určí z ktorých výstupov uzlov (pole `output`) vyberie hodnoty. Okrem nich výberom určí použitú primitívnu funkciu. Sú implementované tieto primitívne funkcie:

- priradenie konštanty (0,25; 0,5; 0,75),
- súčet vstupov,
- rozdiel vstupov,
- súčin vstupov,
- identita jedného zo vstupov,
- násobenie jedného zo vstupov konštantou (0,99; 1,01).

Týmto končí metóda `evaluate` a pokračuje metóda `fitness` tým, že uloží riešenie z primárneho výstupu jedinca do pola `solution`. Ďalej vypočíta `fitness` hodnotu podla vzorca $\sqrt{|S - B|}$, kde S je hodnota kandidátneho riešenia a B je hodnota ku ktorej sa chceme čo najviac priblížiť. Posledným krokom v cykle metódy `fitness` je pripočítanie fitness hodnoty jednej iterácie k celkovej `fitness` hodnote jedinca `fitVal`.
Následne pokračuje vo vykonávaní metóda `evolve` výberom najlepšieho jedinca populácie, tj. jedinca s najnižšou fitness hodnotou. Jeho `ID` a `fitness` hodnota sú uložené do triedy `Stats`. Pred začatím evolučného cyklu je zrealizované vytvorenie súboru pre výpisy daného vlákna. Do súboru `Stat` sú zapísané základné parametre o evolučnom behu v danom vlákne.

Evolučný cyklus

Evolučný cyklus beží pevný počet iterácií daný parametrom `maxGen`. Prvým krokom, ktorý sa vykoná v iterácii, je že na základe najlepšieho jedinca sú zmutovaný všetci ostatní. Genetický operátor mutácie je implementovaný metódou `mutate`.
Jedincovi, vybranému pre mutáciu, sa náhodne určí počet mutovaných génov (uzlov) v intervale 0 až `maxMut`, ktoré je dané parametrom evolúcie. Následne sú vybraté gény pre mutáciu. Sú použité 3 typy mutácie:

- mutácia vstupu uzla,
- mutácia primitívnej funkcie uzla,
- mutácia primárneho výstupu jedinca.

Mutácia prebieha ako zmena aktuálnej hodnoty na inú platnú hodnotu. Cykly s podmienku zabezpečujú, že je vybraná iná než pôvodná hodnota určená k mutácii. Cyklus metódy `mutate` sa opakuje podľa počtu génov určených k mutácii. Po ukončení cyklu sa vracia vykonávanie do evolučného cyklu metódy `evolve`. Znova je volaná metóda `fitness`, pre ohodnotenie a modifikovanie populácie, ale tentoraz s parametrom identifikujúcim najlepšieho jedinca, ktorý nebude hodnotený alebo modifikovaný.
Nasleduje cyklus, ktorý iteruje po jedincoch populácie, a hľadá jedincov ktorý majú `fitness` hodnotu rovnakú, alebo nižšiu než je aktuálna najnižšia `fitness` hodnota. `Fitness` hodnota sa minimalizuje, lebo čím je nižšia, tým je menší rozdiel oproti želaným hodnotám, tj. ideálna `fitness` hodnota je 0. Títo jedinci sú označení ako najlepší pre aktuálnu generáciu, a do súboru sa zapíše ich `fitness` hodnota, spolu s generáciu v ktorej sa vyskytli. Na konci cyklu iterujúcom po jedincoch populácie je najlepší jedinec populácie vybraný, a určený ako najlepší jedinec pre ďalšiu generáciu, tj. bude použitý ako vzor pri mutácii.
Tento proces sa opakuje pevný počet generácií. Následne je finálna `fitness` hodnota, a čas behu evolúcie zapísaný do triedy `Stat`. Okrem toho sú tieto údaje zapísané do súboru v výstupmi, špecifického pre dané vlákno. Riešenie vygenerované metódou, uložené v poli `solution`, je uložené vo formáte json do adresára.
Posledným krokom metódy `evolve` je volanie metódy `export` triedy `Stat`. Táto metóda aktualizuje vo svojej triede, zdieľanej všetkými vláknami, údaje o najlepšom vlákne, sumu času všetkých evolučných behov v sekundách, a vloží do pola `fitness` hodnôt `fitness` hodnotu vykonávaného vlákna. Nakoniec metóda zapíše, do zdieľaného súboru pre výpisy, čas evolúcie, počiatočnú a finálnu fitness hodnotu, a percentuálny rozdiel týchto dvoch hodnôt.

Použitie Kalmanovho filtra

Siedma fáza behu programu aplikuje na vstupné hodnoty A Kalmanov filter z knižnice FilterPy. Nastavenie premenných Kalmanovho filtra prebieha vzhľadom na svoju komplexnosť manuálne [12]. Následne je spustený cyklus, ktorý iteruje po 3 krokoch:

- predict - predpoveď,
- update - aktualizácia,
- write - zápis hodnoty do pola.

Po ukončení cyklu filtra je vypočítaný celkový rozdiel výsledných hodnôt, voči žiadaným hodnotám B, metódou akou je počítaná fitness funkcia hodnota v navrhnutej metóde, tj. $sqrt(abs(K - B))$, kde K sú hodnoty riešenia Kalmanovho filtra. Tento výpočet je implementovaný vo funkcii `getDiff`. Posledným krokom tejto fáze je zápis výsledných hodnôt Kalmanovho filtra do súboru vo formáte `json`.

Porovnanie výsledných dát

Posledná fáza metódy sa zaoberá porovnaním výsledných riešení navrhnutej metódy a riešenia Kalmanovho filtra. Je inicializovaná trieda `Median`, a volaním jej metódy `getStats`, sú zistené štatisticky významné `fitness` hodnoty. Sú to hodnoty:

- dolný kvartil - 25 percentil `fitness` hodnôt,
- stredný kvartil (medián) - 50 percentil `fitness` hodnôt,
- horný kvartil - 75 percentil `fitness` hodnôt.

V metóde `getStats` je použitá funkcia `avgNum` pre výpočet priemeru dvoch desatinných čísel. Následne sú prečítané hodnoty zo súborov s najlepším a mediánovým riešením. Tieto riešenia sú, spolu s hodnotami A a B, zakreslené do grafu využitím knižnice Matplotlib. Posledným krokom je zápis štatistík celkového behu programu do zdieľaného súboru. Okrem údajov o kvartiloch, je zapísaný rozdiel Kalmanovho filtra voči hodnotám B, a priemerný čas behu vlákna v sekundách.

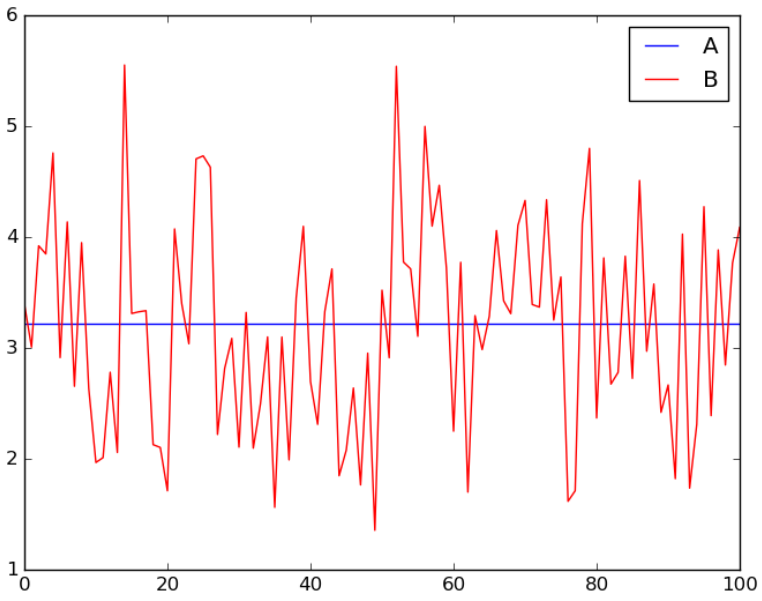
Kapitola 5

Overenie metódy evolučného návrhu pre vývoj filtrov

Implementovaná metóda bola overená na vybraných úlohách. Fitness hodnoty sú zaokrúhlené na 2 desatinné miesta a časové hodnoty na celé sekundy.

5.1 Konštantná funkcia

Na základné overenie navrhnutej metódy bola vybraná konštantná funkcia s hodnotou 3,22 na intervale <0;10> s krokom 0,1.



Obr. 6 Vstupný signál konštantnej funkcie

Na Obr. 6 je signál A, ktorý má konštantnú hodnotu 3,22. Signál B je zašumený signál A.

Riešenie pomocou Kalmanovho filtra

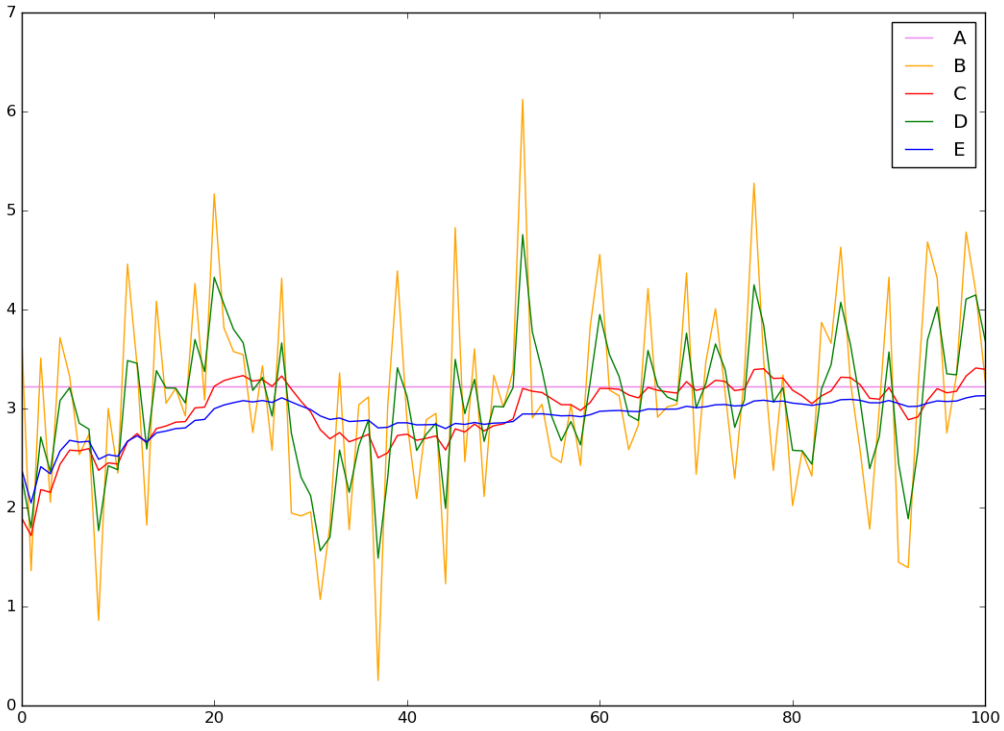
Boli vybrané 3 konfigurácie Kalmanovho filtra znázornené v Tab. 1. Jednotlivé parametre boli zvolené náhodne a na základe testovaní, z ktorých boli vybraté najlepšie riešenia.

Konfigurácia	Parametre Kalmanovho filtra						
	X	F	H	Q	P	R	Fitness
1	0,0	1,0	1,0	0,1	10,0	10,0	47,55
2	0,0	1,0	1,0	0,6	1,0	1,0	65,52
3	1,0	1,0	1,0	0,3	1000,0	1000,0	54,08

Tab. 1 Konfigurácie a výsledky Kalmanovho filtra pri použití na konštantný signál

Na Obr. 7 je znázornených 5 signálov, ktoré sú označené:

- A - signál podla konštantnej funkcie,
- B - zašumený konštantný signál,
- C - konfigurácia 1 Kalmanovho filtra,
- D - konfigurácia 2 Kalmanovho filtra,
- E - konfigurácia 3 Kalmanovho filtra.



Obr. 7 Výsledky Kalmanovho filtra pri použití na zašumený konštantný signál

Na Obr. 7 je vidieť že nastavenie Kalmanovho filtra výrazne ovplyvňuje výsledný signál. Signál A, je konštantný signál, ktorý chceme dosiahnuť. Signál B je zašumený signál, ktorý je vstupom Kalmanovho filtra. Ostatné signály predstavujú výsledky Kalmanovho filtra s rôznymi konfiguráciami. Pre porovnanie bude použitá konfigurácia 3, lebo má najmenší rozdiel oproti signálu, ktorý chceme dosiahnuť filtrovaním.

Navrhnutá metóda

Hodnoty v Tab. 2 boli konštantné vo všetkých konfiguráciách. Veľkosť populácie 5 bola zvolená podľa evolučnej stratégie 1+4 [1]. Počet generácií 250 000 bol vybraný, podľa prvých testov, ako dostatočný pre vývin riešenia. Počet riadkov je pri tomto type testovania statický. Primitívnych funkcií je navrhnutých 12 a boli použité všetky. Počet testovacích behov 30 bol vybraný ako dostatočne vysoký pre získanie štatisticky významných údajov. Ostatné parametre vyplývajú z typu úlohy, ktorá je riešená.

Veľkosť populácie	5
Počet generácií	250 000
Počet riadkov	1
Počet funkcií	12
Počet vstupov	1
Počet výstupov	1
Počet vstupov uzlu	2
Počet výstupov uzlu	1
Počet testovacích behov	30

Tab. 2 Konštantné parametre evolučnej metódy

Bolo otestovaných 5 konfigurácií evolučnej metódy na konštantný vstupný signál. Tab. 3 obsahuje premenné konfiguračné parametre a výsledné hodnoty. Počet stĺpcov bol zvyšovaný v priebehu testovania, aby bol zistený jeho vplyv na riešenie. Maximálny počet mutácií sa pohybuje v intervale <10%;33%>. Bola použitá konfigurácia Kalmanovho filtra č. 3, z Tab. 1.

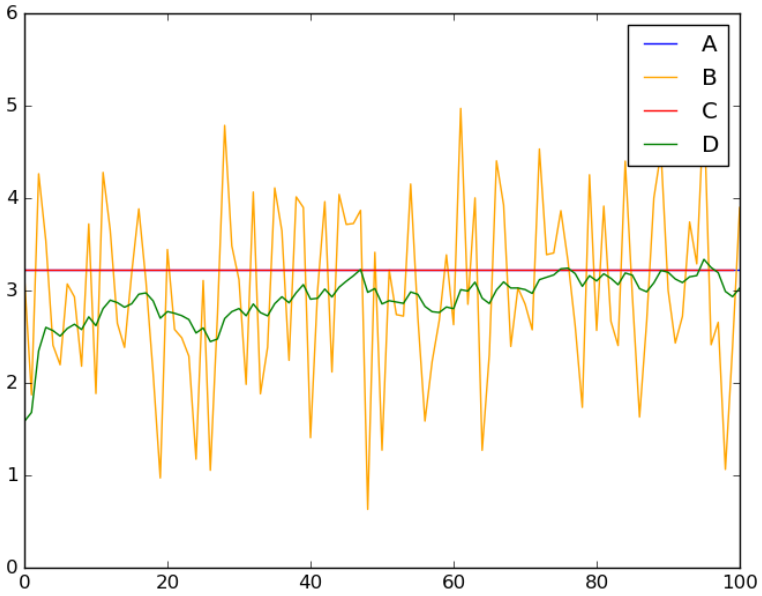
Konfigurácia	1	2	3	4	5
Počet stĺpcov	20	30	40	40	50
Maximálny počet mutácií	4 (20%)	10 (33%)	10 (25%)	5 (12,5%)	5 (10%)
Najlepšia fitness	0,0	0,0	0,0	0,0	0,0
Dolný kvartil	0,5	0,0	0,0	0,12	2,26
Mediánová fitness	9,61	0,06	0,1	0,65	14,10
Horný kvartil	76,65	0,31	8,7	39,06	52,47
Priemerný čas trénovania	3209s	4238s	5642s	5130s	6782s
Kalmanov filter	42,28	52,86	48,48	47,74	48,84

Tab. 3 Premenné konfiguračné parametre a výsledné hodnoty

Na každom z Obr. 8, Obr. 9 a Obr. 10 sú znázornené 4 signály, ktoré sú označené:

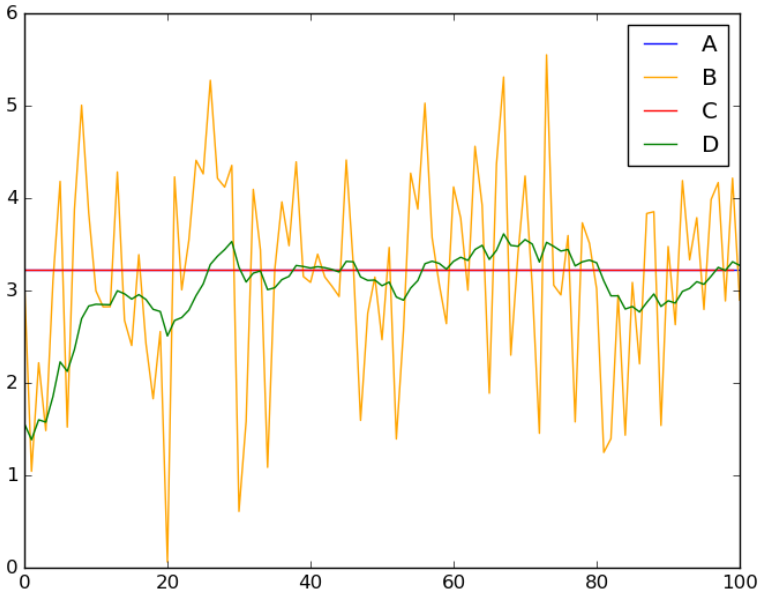
- A - signál podla konštantnej funkcie,

- B - zašumený konštantný signál,
- C - mediánové riešenie danej konfigurácie navrhnutej metódy,
- D - riešenie Kalmanovho filtra.

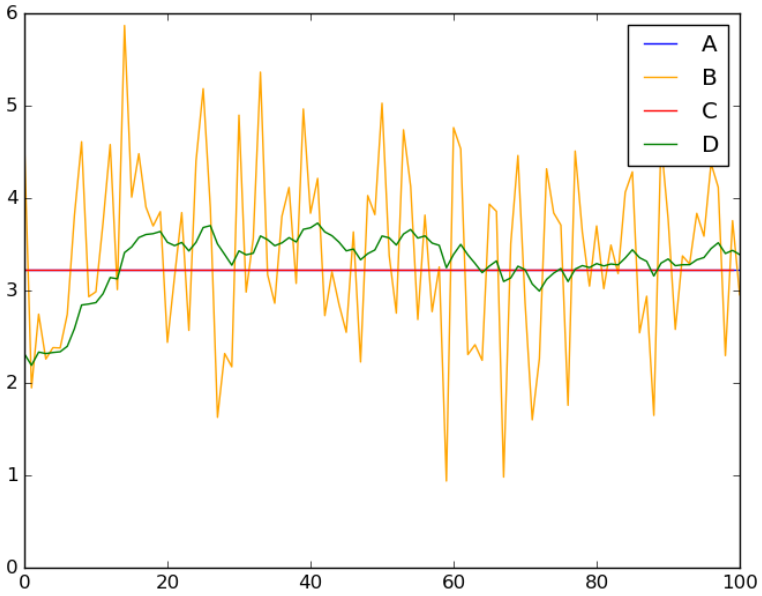


Obr. 8 Porovnanie mediánového riešenia konfigurácie 2 a Kalmanovho filtra

Na Obr. 8 je vidieť, že vstupný konštantný signál A, sa prekrýva s riešením evolučnej metódy C. Riešenie Kalmanovho filtra D dokáže odfiltrovať zašumený vstup B, tiež uspokojivo, ale menej presne ako navrhnutá metóda.



Obr. 9 Porovnanie mediánového riešenia konfigurácie 3 a Kalmanovho filtra



Obr. 10 Porovnanie mediánového riešenia konfigurácie 4 a Kalmanovho filtra

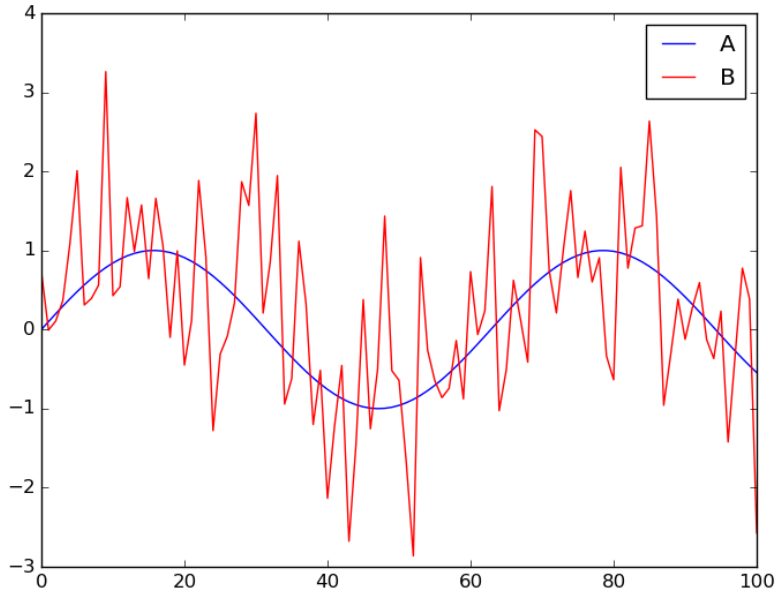
Na Obr. 9 a Obr. 10 je vidieť, že aj v iných konfiguráciách dokáže evolučná metóda najefektívnejšie filtrovať zašumený signál B. Výsledky Kalmanovho filtra sú vo všetkých prípadoch menej presné.

Zhodnotenie výsledkov

Evolučná metóda funguje efektívne pre tento typ signálu. V porovnaní s Kalmanovým filtrom sú výsledky takmer vždy lepšie. V každej konfigurácii bol beh, ktorý dokázal presne prefiltrovať vstupný signál. Pre efektívne filtrovanie je potrebné, aby tabuľka mala aspoň 30 stĺpcov, a maximálny počet mutácií by mal byť 20-35% z celkového počtu génov. Čas behu sa zvyšuje úmerne so zvyšujúcim sa počtom génov. Metóda je vhodná na filtrovanie tohto signálu, ak pomerne dlhý čas trénovania nie je podstatný a chceme eliminovať nutnosť zložitého nastavenia Kalmanovho filtra.

5.2 Periodická funkcia

Ďalším príkladom zvoleným pre overenie správnosti navrhnutej metódy je periodická funkcia sínus na intervale <0;10> s krokom 0,1.



Obr. 11 Vstupný signál typu sínus

Na Obr. 11 je signál A, ktorý opisuje matematickú funkciu sínus. Signál B je zašumený signál A.

Riešenie pomocou Kalmanovho filtra

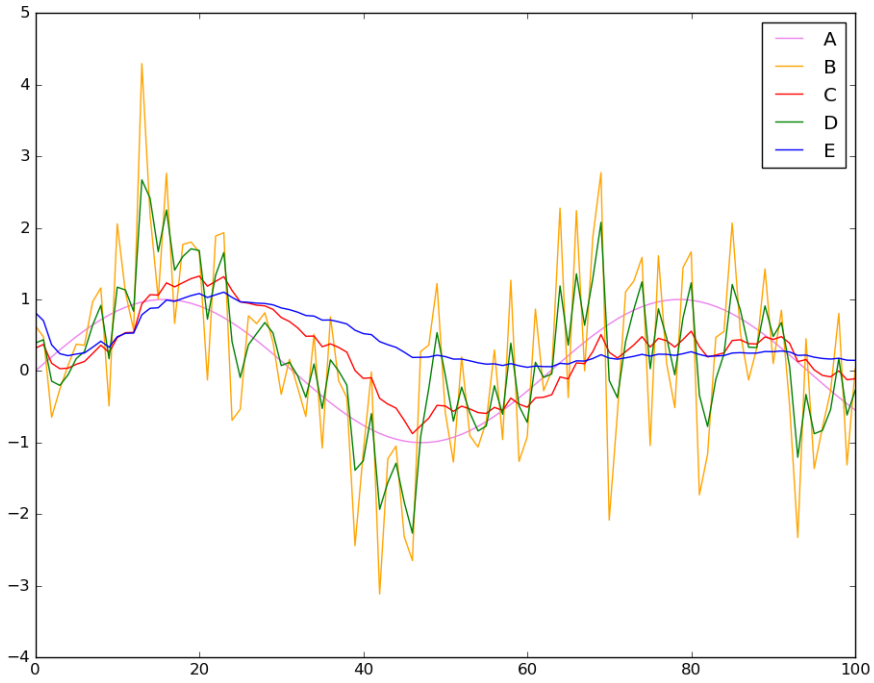
Boli vybrané 3 konfigurácie Kalmanovho filtra znázornené v Tab. 4. Jednotlivé parametre boli zvolené náhodne a na základe testovaní, z ktorých boli vybraté najlepšie riešenia.

Konfigurácia	Parametre Kalmanovho filtra						
	X	F	H	Q	P	R	Fitness
1	0,0	1,0	1,0	0,1	10,0	10,0	58,72
2	0,0	1,0	1,0	0,6	1,0	1,0	65,90
3	1,0	1,0	1,0	0,3	1000,0	1000,0	71,06

Tab. 4 Konfigurácie a výsledky Kalmanovho filtra pri použití na periodický signál

Na Obr. 12 je znázornených 5 signálov, ktoré sú označené:

- *A* - signál podla periodickej funkcie sínus,
- *B* - zašumený periodický signál,
- *C* - konfigurácia 1 Kalmanovho filtra,
- *D* - konfigurácia 2 Kalmanovho filtra,
- *E* - konfigurácia 3 Kalmanovho filtra.



Obr. 12 Výsledky Kalmanovho filtra pri použití na zašumený periodický signál

Na Obr. 12 je vidieť že nastavenie Kalmanovho filtra výrazne ovplyvňuje výsledný signál. Signál *A*, je periodický signál sínus, ktorý chceme dosiahnuť. Signál *B* je zašumený signál, ktorý je vstupom Kalmanovho filtra. Ostatné signály predstavujú výsledky Kalmanovho filtra s rôznymi konfiguráciami. Pre porovnanie bude použitá konfigurácia 3, z Tab. 4, lebo má najmenší rozdiel oproti signálu, ktorý chceme dosiahnuť filtrovaním.

Navrhnutá metóda

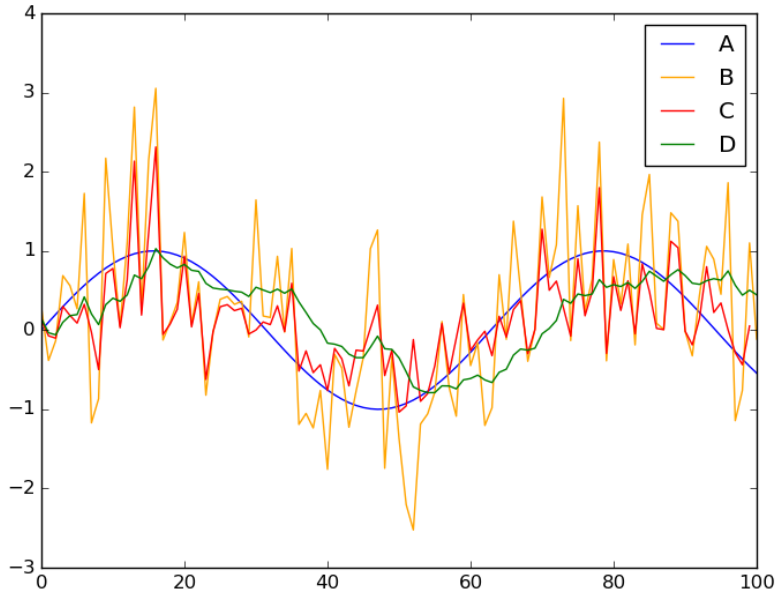
Pri testovaní boli použité parametre uvedené v Tab. 2. Bolo otestovaných 5 konfigurácií evolučnej metódy na periodický vstupný signál. Tab. 6 obsahuje premenné konfiguračné parametre a výsledné hodnoty. Počet stĺpcov bol zvyšovaný v priebehu testovania, aby bol zistený jeho vplyv na riešenie. Maximálny počet mutácií sa pohybuje v intervale <5%;36%>. Bola použitá konfigurácia Kalmanovho filtra č. 3, z Tab. 4.

Konfigurácia	1	2	3	4	5
Počet stĺpcov	20	30	40	40	50
Maximálny počet mutácií	5 (25%)	4 (13%)	2 (5%)	10 (25%)	18 (36%)
Najlepšia fitness	55,84	58,66	56,67	59,23	58,20
Dolný kvartil	56,88	59,22	58,66	60,03	59,39
Mediánová fitness	57,22	59,43	58,93	60,68	60,17
Horný kvartil	57,64	60,05	59,81	61,07	60,85
Priemerný čas trénovania	3034s	4495s	5621s	5255s	7051s
Kalmanov filter	62,85	58,93	62,10	61,81	61,98

Tab. 5 Premenné konfiguračné parametre a výsledné hodnoty

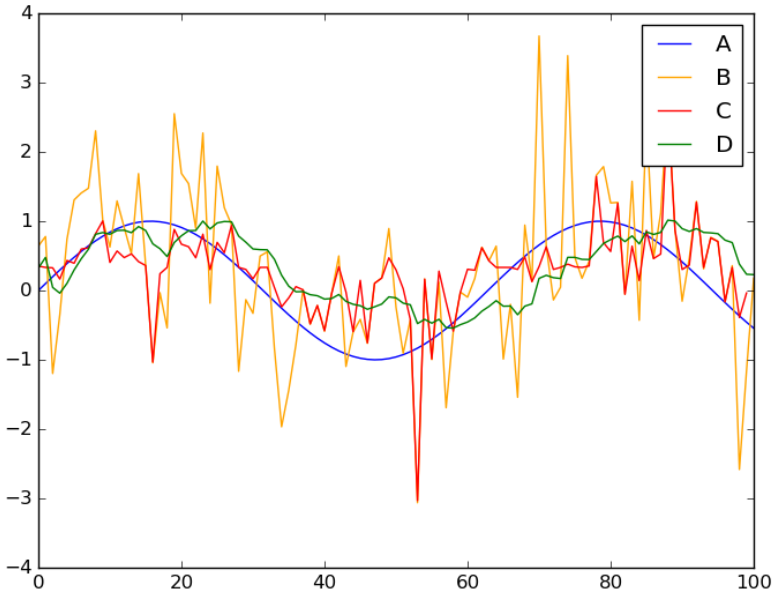
Na každom z Obr. 13, Obr. 14 a Obr. 15 sú znázornené 4 signály, ktoré sú označené:

- *A* - signál podla funkcie sínus,
- *B* - zašumený periodický signál,
- *C* - mediánové riešenie danej konfigurácie navrhnutej metódy,
- *D* - riešenie Kalmanovho filtra.

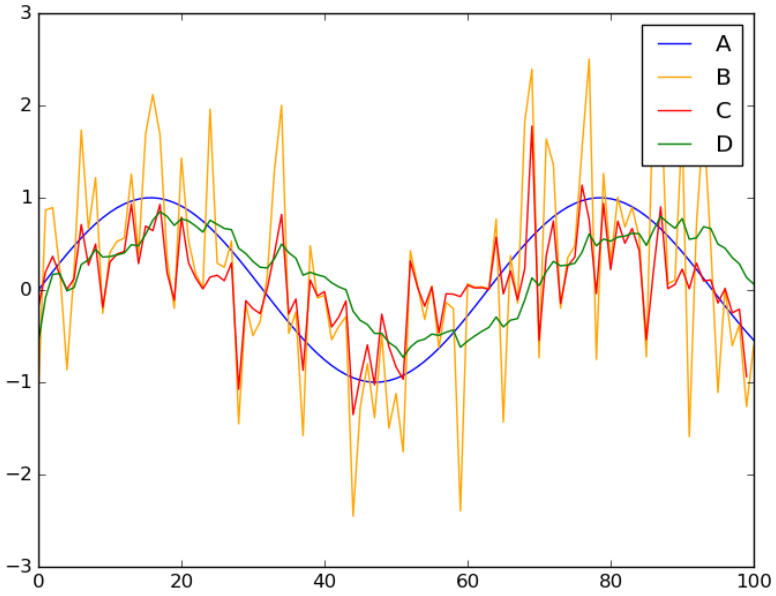


Obr. 13 Porovnanie mediánového riešenia konfigurácie 3 a Kalmanovho filtra

Na Obr. 13 je vidieť konfiguráciu č. 3 označenú *C*, ktorej výsledné riešenie, je porovnateľné s riešením Kalmanovho filtra *D*. Maximálna chyba evolučnej metódy je väčšia, kopíruje krivku periodickej funkcie *A*, bez oneskorenia spôsobeného algoritmom Kalmanovho filtra, ktoré vidíme v intervale <60;80>.



Obr. 14 Porovnanie mediánového riešenia konfigurácie 4 a Kalmanovho filtra



Obr. 15 Porovnanie mediánového riešenia konfigurácie 5 a Kalmanovho filtra

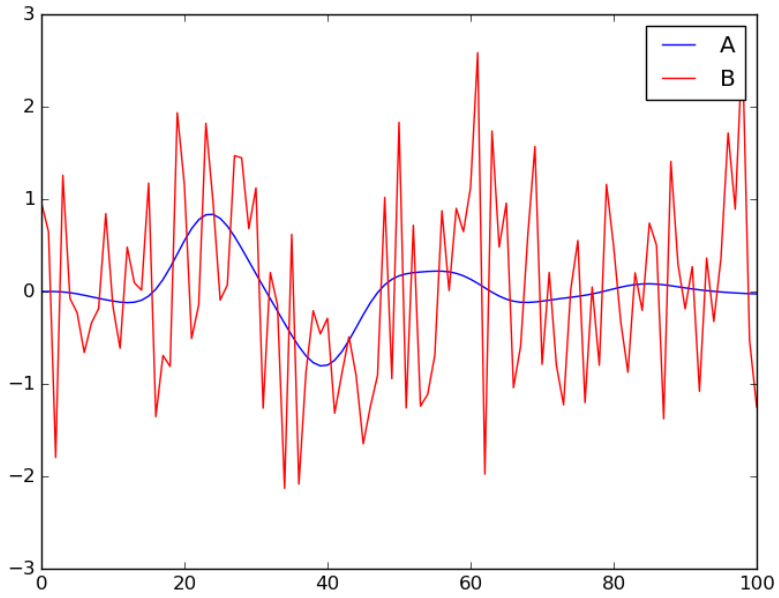
Z Obr. 14 a Obr. 15 je možné usúdiť, že so zvyšujúcim sa počtom stĺpcov sa nezlepšuje výsledné riešenie. Riešenie C na Obr. 14 má ten istý počet stĺpcov ako riešenie na Obr. 13, a vyššie percento mutácie. Táto zmena nepriniesla pozitívne výsledky, keďže všetky priemerné fitness hodnoty sú vyššie. Na Obr. 15 vidíme riešenie C s väčším počtom stĺpcov, ale ani táto modifikácia konfigurácie nepriniesla zlepšenie výsledkov.

Zhodnotenie výsledkov

Evolučná metóda funguje pomerne efektívne pre tento typ signálu. Jej výsledky sú porovnateľné s Kalmanovým filtrom. Počet stĺpcov ovplyvňoval výsledky menej ako pri konštantnom signále. Zvyšovanie počtu stĺpcov nad 40 nemalo pozitívny vplyv na riešenie. Podstatný bol počet maximálnych mutácií, ktorého optimálne nastavenie sa pohybovalo v intervale 10-20%. Čas behu sa znova zvyšoval úmerne s počtom génov. Oproti Kalmanovmu filtru spočíva hlavná výhoda použitia metódy v jednoduchšej počiatočnej konfigurácii.

5.3 Exponenciálna funkcia

Ďalším príkladom zvoleným pre overenie správnosti navrhnutej metódy je exponenciálna funkcia podľa Salustowiczovho benchmarku na intervale <0;10> s krokom 0,1. Tento benchmark má predpis:
$f(x) = e^{-x}x^3\cos(x)\sin(x)(\cos(x)\sin^2(x) - 1)$.



Obr. 16 Vstupný signál Salustowiczovho benchmarku

Na obrázku je signál A, ktorý reprezentuje Salustowiczov benchmark. Signál B je zašumený signál A.

Riešenie pomocou Kalmanovho filtra

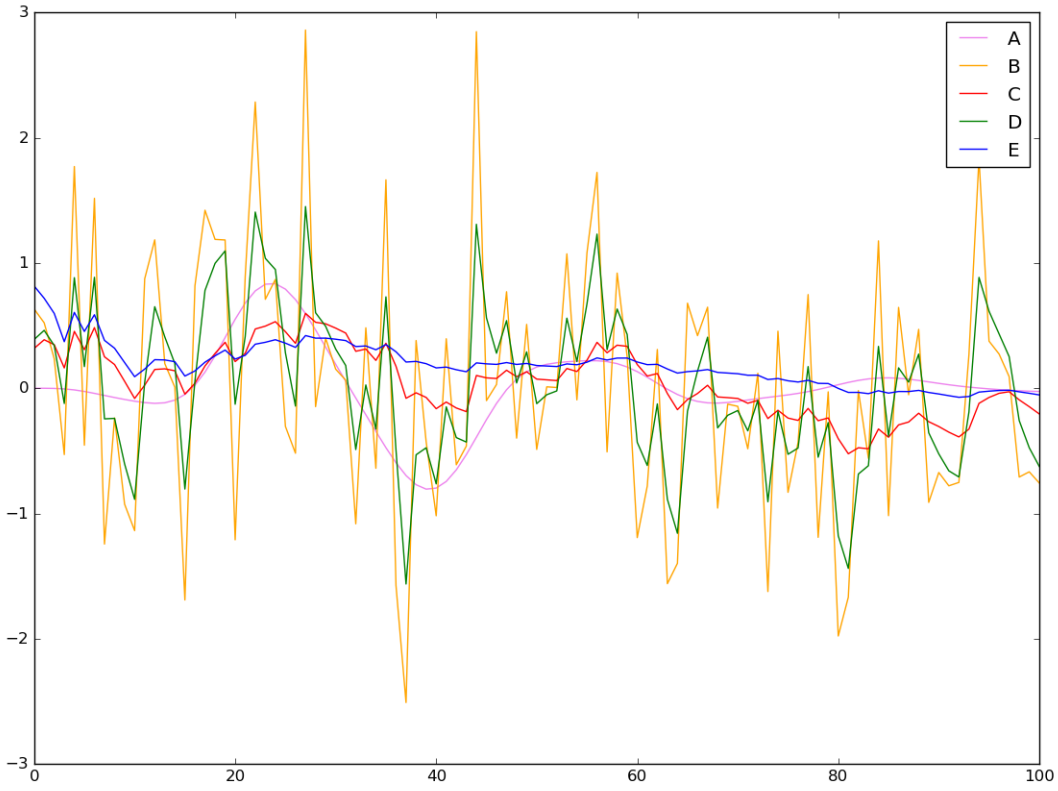
Boli vybrané 3 konfigurácie Kalmanovho filtra znázornené v Tab. 7. Jednotlivé parametre boli zvolené náhodne a na základe testovaní, z ktorých boli vybraté najlepšie riešenia.

Konfigurácia	Parametre Kalmanovho filtra						
	X	F	H	Q	P	R	Fitness
1	0,0	1,0	1,0	0,1	10,0	10,0	45,50
2	0,0	1,0	1,0	0,6	1,0	1,0	61,95
3	1,0	1,0	1,0	0,3	1000,0	1000,0	45,56

Tab. 6 Konfigurácie a výsledky Kalmanovho filtra pri použití na Salustowiczov benchmark

Na Obr. 17 je znázornených 5 signálov, ktoré sú označené:

- A - signál podla Salustowiczovho benchmarku,
- B - zašumený signál,
- C - konfigurácia 1 Kalmanovho filtra,
- D - konfigurácia 2 Kalmanovho filtra,
- E - konfigurácia 3 Kalmanovho filtra.



Obr. 17 Výsledky Kalmanovho filtra pri použití na zašumený Salustowiczov benchmark

Konfigurácia 2 je neefektívna v porovnaní s ostatnými dvoma. Konfigurácie 1 a 3 fungujú pomerne efektívne a prefiltrujú podstatnú časť šumu. Pre porovnanie bude použitá konfigurácia 3, z Tab. 7, z dôvodu mierne presnejšieho výsledku.

Navrhnutá metóda

Pri testovaní boli použité parametre uvedené v Tab. 2. Bolo otestovaných 8 konfigurácií evolučnej metódy na exponenciálny vstupný signál. Tab. 9 obsahuje premenné konfiguračné parametre a výsledné hodnoty. Počet stĺpcov bol zvyšovaný v priebehu testovania, aby bol zistený jeho vplyv na riešenie. Maximálny počet mutácií sa pohybuje v intervale <12,5%;40%>. Bola použitá konfigurácia Kalmanovho filtra č. 3, z Tab. 6. V konfiguráciách 5 až 8 je použitý polovičný interval testovacích dát, tj. konfigurácie 5 a 6 majú interval <0,0;5,0>, a konfigurácie 7 a 8 interval <5,0;10,0>.

Konfigurácia	1	2	3	4
Počet stĺpcov	30	40	50	40
Maximálny počet mutácií	10 (33%)	16 (40%)	10 (20%)	5 (12,5%)
Najlepšia fitness	35,71	34,88	35,42	35,53
Dolný kvartil	36,14	35,31	36,16	35,67
Mediánová fitness	36,64	35,72	36,32	36,10
Horný kvartil	36,39	35,90	36,54	36,30
Priemerný čas trénovania	4101s	5301s	6411s	5203s
Kalmanov filter	53,91	52,35	46,60	49,55

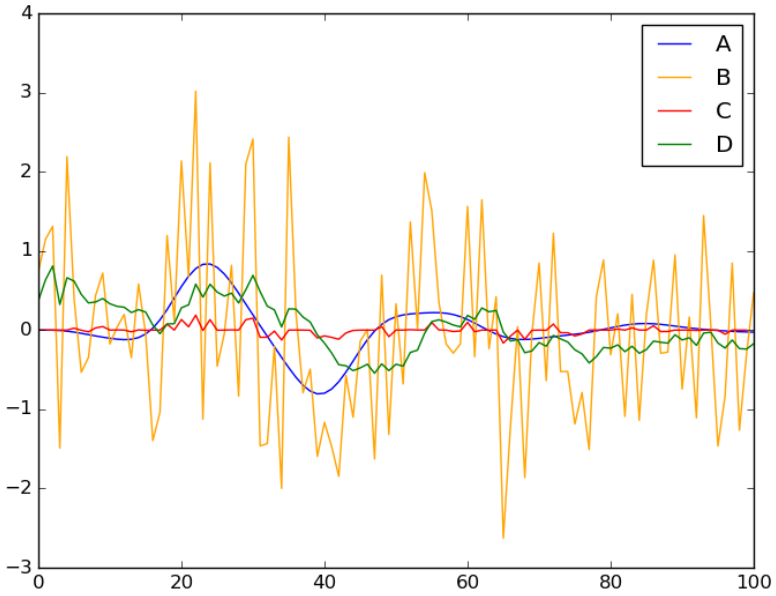
Tab. 7 Premenné konfiguračné parametre a výsledné hodnoty v intervale <0,0;10,0>

Konfigurácia	5	6	7	8
Počet stĺpcov	40	40	40	40
Maximálny počet mutácií	10 (25%)	5 (12,5%)	10 (25%)	5 (12,5%)
Najlepšia fitness	20,66	22,04	11,98	11,50
Dolný kvartil	22,16	22,43	12,20	11,92
Mediánová fitness	22,39	22,95	12,28	12,06
Horný kvartil	22,81	23,13	12,52	12,13
Priemerný čas trénovania	2757s	2727s	2759s	2687s
Kalmanov filter	26,14	27,79	18,49	23,45

Tab. 8 Premenné konfiguračné parametre a výsledné hodnoty v rozdelenom intervale

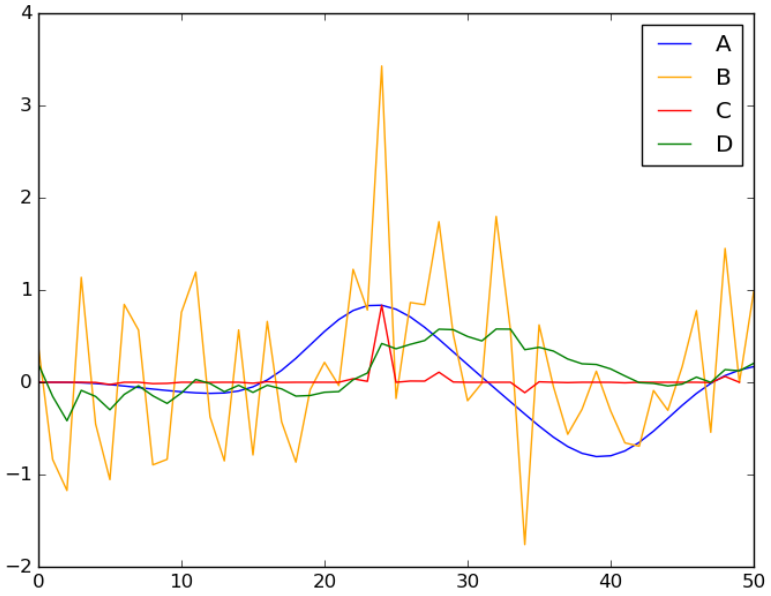
Na každom z Obr. 18, Obr. 19 a Obr. 20 sú znázornené 4 signály, ktoré sú označené:

- *A* - signál podla Salustowiczovho benchmarku,
- *B* - zašumený signál,
- *C* - mediánové riešenie danej konfigurácie navrhnutej metódy,
- *D* - riešenie Kalmanovho filtra.



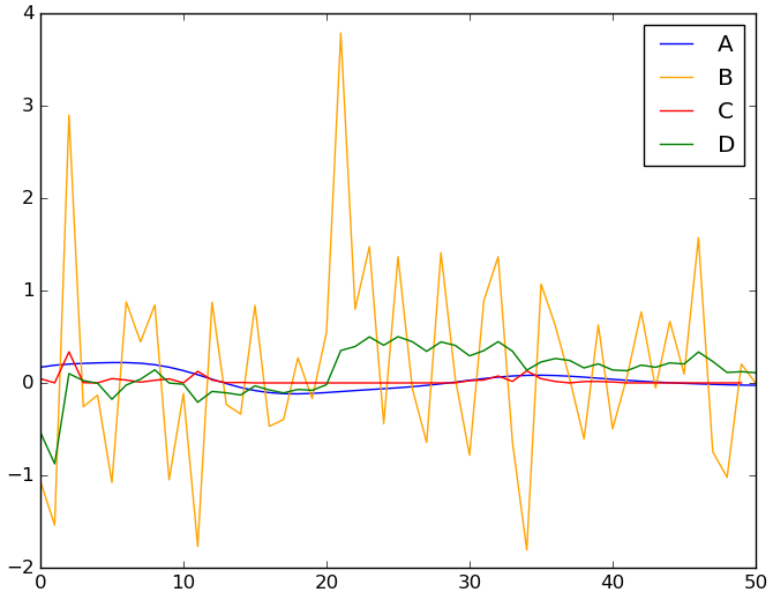
Obr. 18 Porovnanie mediánového riešenia konfigurácie 2 a Kalmanovho filtra

Z Tab. 9 vyplýva, že evolučné riešenie C je presnejšie ako riešenie Kalmanovho filtra D, ale z grafického znázornenia, ktoré je na Obr. 18 je vidieť, že celý signál je ovplyvnený druhou časťou signálu, ktorej hodnoty sú blízke nule. Riešenie Kalmanovho filtra presnejšie opisuje krivku vstupnej funkcie.



Obr. 19 Porovnanie mediánového riešenia konfigurácie 7 a Kalmanovho filtra

Na Obr. 19 vidíme konfiguráciu 7, v ktorej je filtrovaná prvá polovica intervalu z predchádzajúcich konfigurácií. Riešenie C je mierne presnejšie ako v prípade celého intervalu, ale nie je to významné zlepšenie.



Obr. 20 Porovnanie mediánového riešenia konfigurácie 8 a Kalmanovho filtra

Obr. 20 znázorňuje filtrovanie druhej polovice bežného filtrovacieho intervalu. Na tomto intervale je evolučné riešenie C podstatne presnejšie ako riešenie Kalmanovho filtra D.

Zhodnotenie výsledkov

Filtrovanie tohto signálu dokáže evolučná metóda zvládnuť porovnateľne s Kalmanovým filtrom. Numerické výsledky sú v každom testovacom prípade lepšie, ale grafické znázornenie ukazuje, že metóda nedokáže korektne filtrovať signál s meniacou sa amplitúdou. Testy, v ktorých bol testovací interval rozdelený ukazujú, že prišlo k zlepšeniu presnosti v druhej polovici intervalu. Počet stĺpcov a parameter mutácie mali malý vplyv na presnosť riešenia. Pre filtrovanie tohto typu signálu je podstatné, či chceme eliminovať konfiguráciu Kalmanovho filtra, alebo výpočtovú dobu v evolučnej metóde.

5.4 Ďalšie príklady

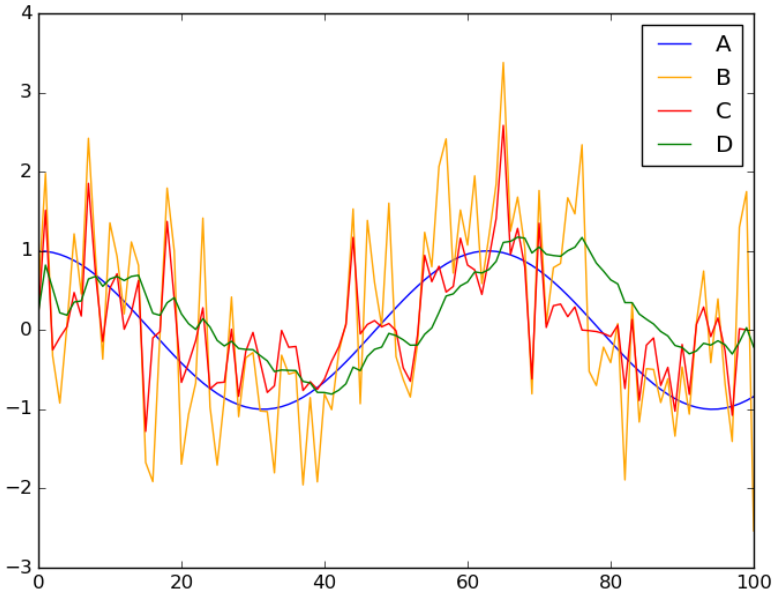
Periodická funkcia cosínus

Testovanie bolo s výnimkou použitia cosínusu namiesto sínusu identické s prvým testovaním periodického signálu.

Na Obr. 21 sú znázornené 4 signály, ktoré sú označené:

- A - periodický signál cosínus,
- B - zašumený signál,

- C - mediánové riešenie vybranej konfigurácie navrhnutej metódy,
- D - riešenie Kalmanovho filtra.



Obr. 21 Porovnanie vstupu typu cosínus

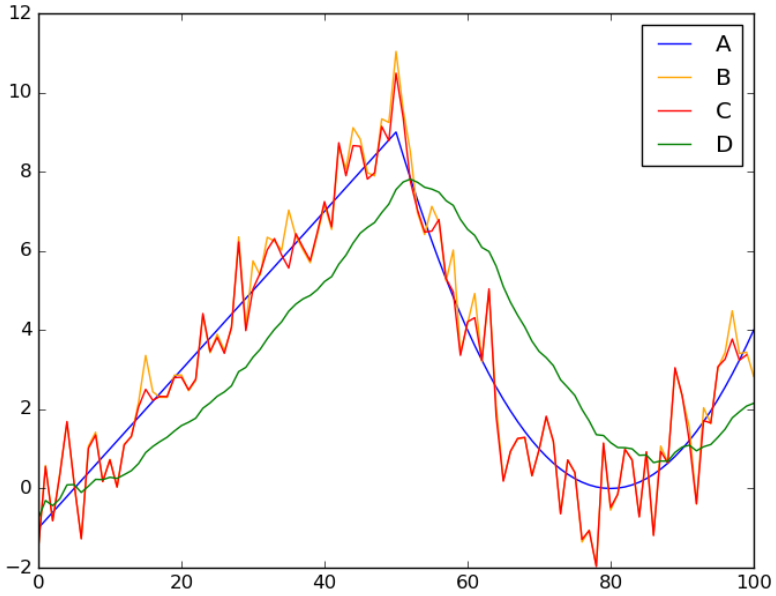
Ako je vidieť na Obr. 21 výsledky testovania sú porovnateľné s prvým testovaním periodického signálu typu sínus.

Lineárno-kvadratická funkcia

Testovaný signál má svoju prvú polovicu podľa lineárneho predpisu $(2x - 1)$, a druhá časť je podľa predpisu $(x - 8)^2$.

Na Obr. 22 sú znázornené 4 signály, ktoré sú označené:

- A - lineárno-kvadratický signál,
- B - zašumený signál,
- C - mediánové riešenie vybranej konfigurácie navrhnutej metódy,
- D - riešenie Kalmanovho filtra.



Obr. 22 Porovnanie vstupu typu lineárno-kvadratická funkcia

Z grafu na Obr. 22 môžeme usúdiť, že ani Kalmanov filter, ani evolučná metóda nemajú uspokojivé výsledky. Tento výsledok sa dá očakávať od Kalmanovho filtra, ktorý je vhodný pre lineárne signály. Navrhnutá metóda má problémy pri trénovaní tohto typu signálu z dôvodu meniaceho sa predpisu.

5.5 Porovnanie časovej efektivity

Čas, ktorý trvá trénovanie evolučných filtrov, sa úmerne zvyšuje so zvyšujúcim sa počtom generácií a veľkosťou kandidátneho riešenia. Od veľkosti kandidátneho riešenia závisí trvanie jednej generácie v evolučnom cykle. Čas jednej generácie sa pohybuje rádovo v stotinách sekúnd (0,01-0,05s). Táto hodnota je porovnateľná s časovým trvaním filtrovania pomocou Kalmanovho filtra (0,05-0,07s).
Z tohto dôvodu je treba pri výbere metódy nutné zvážiť, či je vhodnejšie použiť evolučnú metódu, ktorá vytrénuje riešenie automaticky, alebo sa zaoberať hlbšou analýzou problému a vhodným nastavením Kalmanovho filtra.

Kapitola 6

Záver

Práca je zameraná na spracovanie signálov s využitím evolučného algoritmu a genetického programovania. Konkrétne je to využitie Kartézskeho genetického programovania za účelom vývoja automatizovaných filtrov, ktoré sú porovnané s Kalmanovým filtrom, ktorý patrí medzi adaptívne filtre.
Cieľom práce bolo navrhnúť, implementovať a overiť funkčnosť pre spracovanie signálov. Metóda bola implementovaná v skriptovacom jazyku Python verzie 2.7.
Funkčnosť metódy bola overená na niekoľkých typoch príkladov. Podrobne boli opísané príklady konštantného, periodického a exponenciálneho vstupu. Výsledky týchto spracovaní boli porovnané s Kalmanovým filtrom. Konfigurácia Kalmanovho filtra je komplexný problém, ktorý vyžaduje hlbšiu znalosť problematiky adaptívnych filtrov, z čoho vyplýva hlavná výhoda evolučného prístupu, ktorý požaduje iba znalosť vhodne reprezentovať daný problém. Evolučný prístup je vhodný v prípadoch, keď nie je dôležitá výpočtová doba trénovania. Metóda funguje efektívnejšie ako Kalmanov filter v prípade vstupu typu konštantný signál. V niektorých behoch dokázala prefiltrovať signál úplne, tj. odstrániť všetok šum. Pri periodickom druhu signálu metóda dosiahla porovnateľné výsledky s Kalmanovým filtrom a preto je treba brať do úvahy komplexnejšiu konfiguráciu Kalmanovho filtra oproti dlhšej výpočtovej dobe evolučnej metódy. V prípade exponenciálnej vstupnej funkcie boli výsledky porovnateľné s Kalmanovým filtrom. Filtrovanie však bolo presné len v druhej polovici testovaného intervalu z dôvodu princípu fungovania metódy. Taktiež bola testovaná lineárno-kvadratická funkcia, ale na tomto type vstupu nemala evolučná metóda ani Kalmanov filter uspokojivé výsledky.
Ďalší vývoj metódy by mohol riešiť niekoľko problémov. Patrí sem možnosť experimentovania s genetickým operátorom kríženia, ktorý nie je klasickou súčasťou Kartézskeho genetického programovania. Tiež existuje možnosť vývoja, ktorý by poskytol riešenia pre exponenciálne signály, alebo zložené signály (lineárno-kvadratický). Iným smer vývoja je snaha skrátiť výpočtovú dobu. Pre tento účel by mohla byť použitá heuristika pre vynechávanie nepoužitých génov pri výpočte alebo použitie paralelizácie.

Literatura

[1] *Cartesian Genetic Programming.* [Online; navštíveno 7.5.2016].
URL http://www.cartesiangp.co.uk

[2] *DEAP 1.1.0 documentation.* [Online; navštíveno 7.5.2016].
URL http://deap.readthedocs.org/en/master/api/benchmarks.html#deap.benchmarks.gp.salustowicz_1d

[3] *Dokumentácia k jazyku Python verzie 2.7.* [Online; navštíveno 7.5.2016].
URL https://docs.python.org/2.7/

[4] *Dokumentácia knižnice filterpy.* [Online; navštíveno 7.5.2016].
URL http://pythonhosted.org/filterpy/

[5] *Generic Evolutionary Design.* [Online; navštíveno 7.5.2016].
URL http://www0.cs.ucl.ac.uk/staff/ucacpjb/BEWAC5.pdf

[6] *Kalman Filter Applications.* 2008.
URL http://www.cs.cornell.edu/courses/cs4758/2012sp/materials/mi63slides.pdf

[7] C. Darwin: *The Origin of Species.* John Murray, 1859.
URL
http://literature.org/authors/darwin-charles/the-origin-of-species/

[8] Hunter, J. D.: *Matplotlib: A 2D graphics environment.* Computing In Science & Engineering, *ročník 9, 2007: s. 90–95.*

[9] J. Černocký: Slajdy k predmetu Signály a systémy. [Online; navštíveno 7.5.2016]. URL http://www.fit.vutbr.cz/study/courses/ISS/public/

[10] L. Sekanina a kolektiv: Evoluční hardware. Academia, 2009, ISBN ISBN 978-80-200-1729-1.

[11] R. Faragher: Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. IEEE Signal Processing Magazine, *ročník 29, 2012: s. 128–132. URL http://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf*

[12] Roger R. Labbe Jr.: Kalman and Bayesian Filter in Python. 2015, [Online; navštíveno 7.5.2016].
URL https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

[13] *Scott C. Douglas: Introduction to Adaptive Filters. 1999, [Online; navštíveno 7.5.2016].*
URL http://www.ece.mcmaster.ca/faculty/reilly/coe4tl4/adaptive%20filters%20Scot%20Douglas.PDF

[14] *S.W. Smith: The Scientist and Engineer's Guide to Digital Signal Processing. [Online; navštíveno 7.5.2016].*
URL http://www.dspguide.com/pdfbook.htm/

[15] *Stéfan van der Walt, S. C. C.; Varoquaux, G.: The Numpy Array:A Structure for Efficient Numerical Computation.* Computing In Science & Engineering*, ročník 13, 2011: s. 22–30.*

[16] *Z. Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs. Springer, 1999, ISBN ISBN 3-540-60676-9.*

[17] *Z. Vašíček: Cartesian Genetic Programming. 2012, [Online; navštíveno 7.5.2016].*
URL http://www.fit.vutbr.cz/~vasicek/cgp/

Prílohy

Zoznam príloh

Príloha A

Obsah CD

- `code` - adresár so skriptami
- `test` - adresár s výsledkami testov
- `latex` - adresár s textovou časťou bakalárskej práce

Príloha B

Manuál

Filtrovacia metóda opísaná v práci
./filterEvol.py [-h] [–row ROW] [–col COL] [–lback LBACK] [–popSize POPSIZE] [–maxGen MAXGEN] [–maxMut MAXMUT] [–maxFunc MAXFUNC] [–tCount TCOUNT] [–fnc FNC] [–start START] [–finish FINISH] [–step STEP] [–stat STAT]

Popis argumentov:

- -h, –help - správa help,
- –row ROW - počet riadkov chromozómu ROW,
- –col COL - počet stĺpcov chromozómu COL,
- –lback LBACK - LBACK parameter chromozómu,
- –popSize POPSIZE - počet jedincov populácie POPSIZE,
- –maxGen MAXGEN - počet generácií MAXGEN,
- –maxMut MAXMUT - maximálny počet mutácii na 1 generáciu MAXMUT,
- –maxFunc MAXFUNC - počet použitých primitívnych funkcií MAXFUNC,
- –tCount TCOUNT - počet vlákien v ktorých prebieha filtrovanie TCOUNT,
- –fnc FNC - funkcia FNC použitá pre generovanie trénovacích dát,
- –start START - začiatok intervalu generovania dát START,
- –finish FINISH - koniec intervalu generovanie dát FINISH,
- –step STEP - krok generovania trénovacích dát STEP,
- –stat STAT - prefix adresára a súboru s výpismi.

Príklad použitia:
./filterEvol.py –col 30 –lback 30 –popSize 5 –maxGen 250000 –maxMut 4 –tCount 30 –fnc "sin"–stat "sin0"

V tomto použití skriptu bude 1 riadok (default), 30 stĺpcov a parameter l-back bude rovný

počtu stĺpcov. V populácii bude 5 jedincov, a evolučný cyklus bude mat 250 000 generácií. Nebudú prebiehať viac ako 4 mutácie na jedinca za generáciu, a je použitých všetkých 12 primitívnych funkcií. Metóda bude vykonaná v 30 vláknach, a zdieľaný súbor bude mat prefix "sin0", doplnený o systémový čas v ktorom začína beh skriptu. Trénovacie dáta budú vygenerované podľa funkcie sínus, v prednastavenom intervale <0,0;10,0>.

Výpis v súbore špecifickom pre vlákno:
Training Data: vectors: 100
inputs: 1 outputs: 1
Evolution parameters:
Rows: 1 Columns: 30 L-Back: 30
Node Inputs: 2 Node Output: 1
Population Size: 5 Functions: 12
Max Mutated Genes: 4 Max Generations: 250000
Initial Fitness: 70.630046313
Evolution starting
Generation: 93 Fitness: 70.4065400134

...

Generation: 127946 Fitness: 60.1970461043
Evolution finished
Time passed: 4506.94903708
Final Fitness: 60.1970461043

Prvé riadky každého súboru pre výpisy obsahujú popis trénovacích dát a parametre evolučnej metódy, ktorá bude spustená. Nasleduje hodnota začiatočnej fitness hodnoty. Ďalej je označený začiatok evolučného behu, a každé zlepšenie fitness hodnoty v priebehu generácií. Po ukončení evolučného cyklu je zapísaný čas, ktorý trval beh, a konečná fitness hodnota.

Výpis v zdieľanom súbore:
Training data type: sin
Training data sample size: 100
CGP rows: 1
CGP cols: 30
CGP L-Back: 30
CGP functions: 12
CGP generations: 250000
CGP mutations: 4
CGP population size: 5

Thread-1
Evolution runtime: 4506.94903708
Initial Fitness: 70.630046313
Final Fitness: 60.1970461043

Percentual improvement: 14.7713342314%

...

Best Thread: Thread-6
Median Thread: Thread-7
Average Run Time: 4495.28266963

Difference of Kalman filter: 58.9260300352
Best Fitness: 58.6609862372
Lower Quartile Fitness: 59.2207385853
Median Fitness: 59.4260996619
Upper Quartile Fitness: 60.0499032082

Zdieľaný súbor pre výpisy obsahuje na svojom začiatku popis trénovacích dát, a parametre evolučnej metódy, ktorá bude spustená vo všetkých vláknach. Nasledujú údaje o jednotlivých behoch: čas behu, začiatočná fitness, konečná fitness, a percentuálny rozdiel týchto dvoch hodnôt. Na konci súboru sú zapísané údaje, ktoré označujú vlákno s najlepšou fitness, vlákno s fitness, ktorá je najbližšia mediánu a priemerný čas behu vlákna. Ďalej sú tu fitness hodnoty: najlepšia, dolný kvartil, stredný kvartil (medián) a horný kvartil. Okrem nich je tu údaj predstavujúci rozdiel riešenia Kalmanovho filtra oproti cieľovému riešeniu.

Vytvorený adresár obsahuje súbory:

- `default-Solution.json` - súbor s riešením Kalmanovho filtra,
- `pure.json` - súbor s vygenerovaným signálom,
- `polluted.json` - súbor s modifikovaným generovaným signálom,
- `sin0-stats.txt` - zdieľaný súbor pre výpisy,
- `Thread-N-solution.json` - súbor s riešením N-tého vlákna,
- `Thread-N.txt` - súbor s výpismi N-tého vlákna.

Pomocná metóda pre vykreslenie a porovnanie riešení

./plotSolution.py [-h] [–f F]

Popis argumentov:

- -h, –help - správa help,
- –f F - súbor typu json, v ktorom je uložené riešenie k vykresleniu.

Príklad použitia:
./plotSolution.py –f ./sin0123456789/Thread-1-solution.json

Pomocná metóda vykreslí použitím knižnice Matplotlib riešenie vybrané argmentom skriptu. Okrem neho sú zakreslené aj: generovaný signál, modifikovaný generovaný signál a riešenie Kalmanovho filtra. Ak nenájde v adresári tieto súbory skript sa ukončí chybou.