VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ KATEDRA

Empirinis filtrų palyginimas

Filters

Bakalauro darbas

Contents

[IVADAS 5](#_Toc357417365)

[1. Kalmano filtras 6](#_Toc357417366)

[1. Matematinis Kalmano filtro modelis 6](#_Toc357417367)

[Kalmano filtro įgyvendinimas programoje 7](#_Toc357417368)

[Matematikos biblioteka 7](#_Toc357417369)

[Kalmano filtro modelis 8](#_Toc357417370)

[Kalmano filtro klasė 8](#_Toc357417371)

[Kalmano filtro pavyzdys 9](#_Toc357417372)

[Dalelių (particles) filtras 11](#_Toc357417373)

[Matematinis dalelių filtro modelis 11](#_Toc357417374)

[Dalelių filtro įgyvendinimas programoje 11](#_Toc357417375)

[Filtru palyginimas 13](#_Toc357417376)

[Voltmetro eksperimentas 13](#_Toc357417377)

[Judėjimo simuliacija 13](#_Toc357417378)

[Gps koordinačių filtravimas 13](#_Toc357417379)

[Android programėlė gps koordinačių fiksavimui 13](#_Toc357417380)

[Lėktuvo nusileidimo trajektorijos filtravimas 13](#_Toc357417381)

[isvados 14](#_Toc357417382)

[Literaturos sarasas 15](#_Toc357417383)

SANTRUMPA

SUMMARY

# IVADAS

# Kalmano filtras

Vienas iš geriausiai žinomų ir dažniausiai naudojamų matematinių įrankių triukšmingiems matavimams filtruoti yra Kalmano filtras. Pavadintas pagal Rudolph E. Kalman, kuris 1960 metais savo darbe (Kalman 1960) aprašė rekursinį diskrečios informacijos linijinių sistemų filtravimo sprendimą. Kalmano filtras tai matematinių formulių rinkinys aprašantis numatymo (prediction) ir pataisymo (correction) algoritmą, kuris optimaliai minimizuoja paklaidą, kai yra žinomos išankstinės prielaidos apie modelį. Viena iš filtro populiarumo priežasčių yra ta, kad jis veikia gerai, net kai išankstinės prielaidos apie modelį nėra žinomos. [1]

## Matematinis Kalmano filtro modelis

Kalmano filtras yra rekursinis procesas susidedantis iš 2 žingsnių: numatymo (prediction) ir pataisymo (correction). Numatymo žingsnyje algoritmas numato sistemos būseną ateityje. Pataisymo žingsnyje numatyta būsena yra pakoreguojama pagal gautą konkretų matavimą.



Numatymo lygtys:

(1)

(2)

Pataisymo lygtys:

var S = s.H \* s.P \* s.H.transpose() + s.Ez;

var K = s.P \* s.H.transpose() \* S.invert();

The ﬁrst task during the measurement update is to compute the Kalman gain, . Notice

that the equation given here as equation (4.11) is the same as equation (4.8). The next step

is to actually measure the process to obtain , and then to generate an a posteriori state

estimate by incorporating the measurement as in equation (4.12). Again equation (4.12) is

simply equation (4.7) repeated here for completeness. The ﬁnal step is to obtain an a

posteriori error covariance estimate via equation (4.13).

After each time and measurement update pair, the process is repeated with the previous a

posteriori estimates used to project or predict the new a priori estimates. This recursive

nature is one of the very appealing features of the Kalman ﬁlter—it makes practical

implementations much more feasible than (for example) an implementation of a Wiener

ﬁlter (Brown and Hwang 1996) which is designed to operate on all of the data directly for

each estimate. The Kalman ﬁlter instead recursively conditions the current estimate on all

of the past measurements. Figure 4.2 below offers a complete picture of the operation of

the ﬁlter, combining the high-level diagram of Figure 4.1 with the equations from table 4.1

and table 4.2.

## Kalmano filtro įgyvendinimas programoje

Kalmano filtras buvo įgyvendintas Dart programavimo kalboje. Filtrą sudaro 2 klasės priklausančios filters bibliotekai (library):

* class Kalman - esanti filters/kalman.dart faile
* class KalmanModel - esant filters/kalman\_model.dart faile

Šios klasės naudoja pagalbinė matematikos biblioteką math.

### Matematikos biblioteka

Kalmano filtrų skaičiavimai atliekami su matricomis ir vektoriais, o triukšmas yra atsitiktiniai dydžiai pasiskirstę pagal normalųjį skirstinį. Kadangi Dart programavimo kalba natūraliai nepalaiko matricų duomenų tipo ir turi galimybę generuoti tik tolygiai pasiskirsčiusius pseudo atsitiktinius skaičius, buvo iškeltas uždavinys sukurti klases darbui su matricomis ir normaliuoju skirstiniu. Kitoms programavimo kalboms tokio uždavinio spręsti nubūtų reikėję, nes tokios bibliotekos jau egzistuoja, bet Dart kalba yra labai jauna ir tokių bibliotekų rasti nepavyko.

Matrix klasė įgyvendina tik kelis iš visų matricoms taikomų veiksmų, kurių mums reikia atliekant Kalmano filtro skaičiavimus:

* Aritmetinės operacijos (sudėtis atimtis, daugyba, dalyba)
* Transpozicija - tai tokia matricos perstata kai kiekvienas matricos elementas sukeičiamas su elementu.
* Inversija - tai tokia matrica iš kurios padauginę pradinę matricą gauname vienetinę matrica I. Inversijos skaičiavimas įgyvendintas pagal Cramer‘io taisyklę.  
   - matricos A determinantas  
   – matricos A kofaktorių matrica. Matrica sudaroma vietoj kiekvieno matricos A elemento įrašant jo minorą.

Gaussian klasė yra skirta darbui su normaliuoju skirstiniu. Joje yra 2 funkcijos:

* randn(, ) - pavadinta pagal matlab funkcijos atitikmenį. Generuoja atsitiktinį skaičių pagal normalųjį skirstinį. Funkcija įgyvendinta naudojant Marsaglia polar metodą [2]. Imami du tolygiai pasiskirstę skaičiai U ir V  
  jeigu imami kiti skaičiai. Priešingu atveju skaičiuojami  
  Čia X ir Y yra 2 nepriklausomi atsitiktiniai dydžiai pasiskirstę pagal normalujį skirstinį su vidurkiu 0 ir standartiniu nuokrypiu 1. Šis algoritmas yra labai lengvai įgyvendinamas ir nereikalauja trigonometrinių funkcijų skaičiavimo, dėl ko tampa patrauklesnis už Box-Muller metodą.
* gaussian(x, ,) - skaičiuoja normaliojo skirstinio funkcijos reikšmę taške x.

### Kalmano filtro modelis

Kalmano filtro būsena yra aprašyta klasėje KalmanModel. Kiekvienas Kalmano filtras dirba su konkrečia KalmanModel klase ir ją modifikuoja laiko bėgyje. KalmanModel klasė atspindi Kalmano filtro būsena konkrečiu laiko momentu. Ją sudaro tokie kintamieji:

* x - būsena. Tai yra matrica su proceso būsena nusakančiais parametrais, pvz.: x koordinatė, x koordinatės kitimo greitis, posūkio kampas ir pan.
* out - kadangi algoritmas įgyvendintas taip, kad po filtravimo funkcijos x įgyja numatomos būsenos reikšmę kintajame out išsaugoma pakoreguotas matavimas.
* z – matavimas, netiksli sistemos būsena gauta iš matavimo.
* u – proceso valdymo vektorius, kai žinom kaip kinta procesas priklausomai nuo aplinkinių veiksnių.
* P - kovariacijos matrica
* K – Kalmano koeficientas
* Ex - sistemos proceso triukšmas
* Ez – sistemos matavimų triukšmas
* F – matrica aprašanti būsenos kitimą.
* H – matrica aprašanti stebimus matavimus. Kai sistemos būsena susideda iš daugiau knitamųjų negu yra stebima, ši matrica panaikina papildomus kintamuosius.
* B – matrica aprašanti sistemos būsenos priklausomybę nuo matricos u.

### Kalmano filtro klasė

Kalman klasė įgyvendiną vieną funkciją skirtą filtravimui ir aprašančią algoritmą pateiktą viršuje. Pirmos 6 eilutės aprašo matavimo atnaujinimo algoritmą, 7-10 būsenos spėjimą. Iš pradžių yra suskaičiuojama pagalbinė matrica S (1), kad nereikėtų jos skaičiuoti 2 kartus. Apskaičiuojamas Kalmano koeficientas (2). Gautas matavimas paverčiamas į matricą, kad būtų galima atlikti skaičiavimus su juo (3). Apskaičiuojamas skirtumas tarp matavimo ir spėjimo (4). Apskaičiuojama patikslinta sistemos būsena s.x (5). Kuo didesnis Kalmano koeficientas, tuo didesnę reikšmę patikslintai būsenai turi apskaičiuotas y. Kalmano koeficientas lemia, kaip labai filtras pasitiki matavimais. Išsaugom patikslintą sistemos būsenos matavimą (6). Apskaičiuojam patikslintą kovariacijos matricą P (7). Numatom kitą sistemos būseną (8)-(9). Numatom kitą sistemos kovariacijos matricą (10).

Funkcijos kodas:

var S = s.H \* s.P \* s.H.transpose() + s.Ez; (1)

var K = s.P \* s.H.transpose() \* S.invert(); (2)

var Z = new Matrix.fromList(measurement); (3)

var y = Z - (s.H \* s.x); (4)

s.x = s.x + (K \* y); (5)

s.out = s.x.clone(); (6)

s.P = (s.I - (K \* s.H)) \* s.P; (7)

s.x = s.F \* s.x; (8)

s.x = s.x + s.B \* s.u; (9)

s.P = s.F \* s.P \* s.F.transpose() + s.Ex; (10)

Priešingai nei matematiniame modelyje, algoritmas iš pradžių pataiso mūsų matavimą, po to numato kitą būseną. Taip pasielgta dėl natūralesnio duomenų padavimo algoritmui. Iš pradžių filtras neturi jokios būsenos, todėl negali nuspėti tolimesnės. Algoritmo veikimo pradžioje būsena nustatoma į bet kokią ir pateikiamas labai didelis pasitikėjimas matavimu. Po pirmo filtravimo pradinė būsena tiesiog tampa matavimu. Tai labai gerai tinka sistemoms, kurios neturi visų prielaidų apie sistemą. Sistemoms turinčios pradines prielaidas tiesiog nustatome tinkamą būseną ir kovariacijos matricą. Algoritmas baigęs šią veiksmų seką savyje turi numatomą būseną, todėl papildomai reik išsisaugoti ją s.out kintamajame.

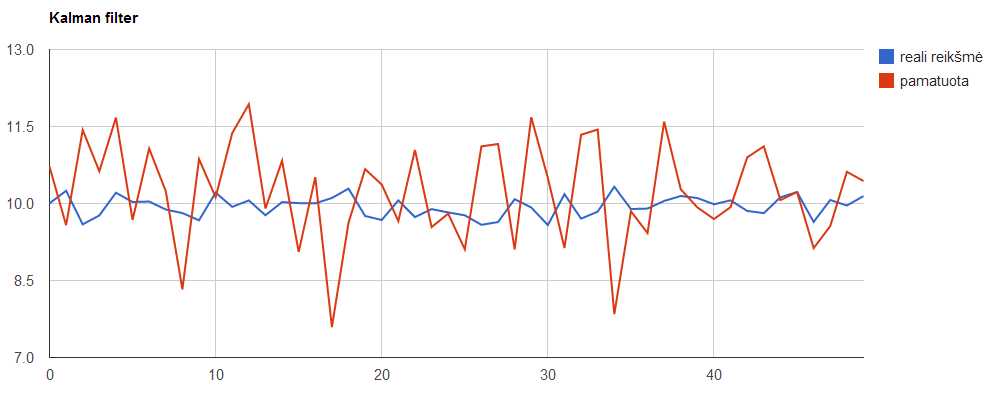
Visi skaičiavimai atliekami su matricomis ir vektoriais, kur vektorius yra ne kas kita kaip 1xN arba Nx1 matrica. Tam naudojama pagalbinė autoriaus parašyta matematikos biblioteka.

## Kalmano filtro pavyzdys

Šiame pavyzdyje nagrinėsime labai paprastą sistemą, kurią galime pavadinti voltmetro matavimų. Sistemą apibrėžia vienas kintamasis x, kurio reikšmė yra konstanta, šiuo atveju lygi atsitiktinai pasirinktam skaičiui 10. Taigi x priklausomybė nuo t yra lygtis:

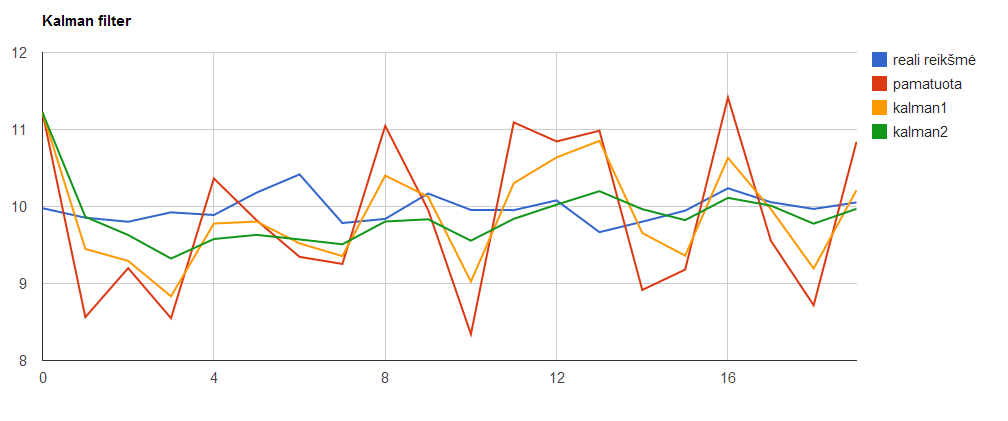
Sistemai galima parinkti triukšmą, kuris yra pagal normalųjį skirstinį pasiskirstęs atsitiktinis dydis Ex su vidurkiu 0 ir standartiniu nuokrypiu , kurį galima keisti. Kai sistemos lygtis tampa:

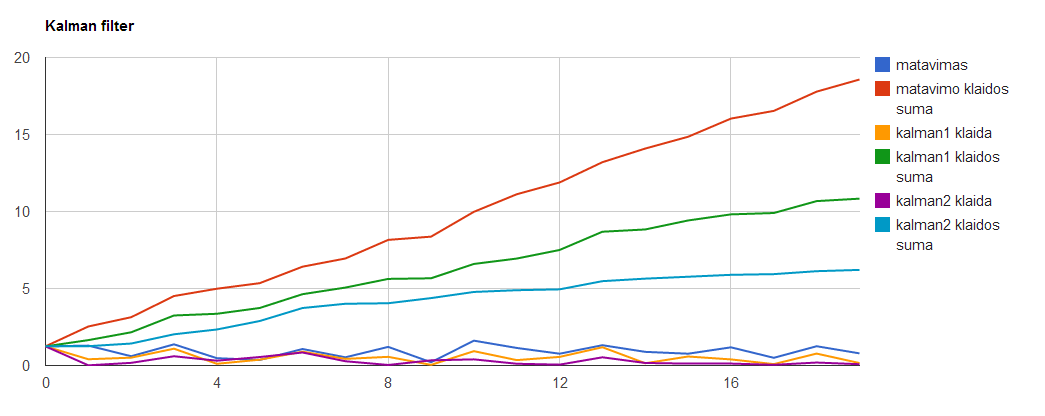
Matavimus aprašo lygtys:



1 pav.

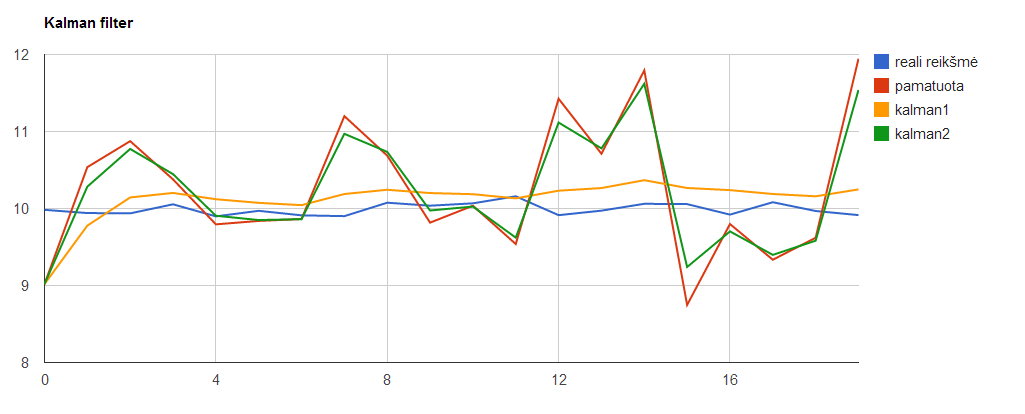
Kalmano filtras veikia optimaliai, jeigu yra žinomi visi sistemos modelio parametrai, tai yra žinoma tiek sistemos modelio lygtis, tiek proceso, bei matavimų paklaidos. Viena iš Kalmano filtro populiarumo priežasčių yra ta, kad net ir nežinant šių dydžių filtras veikia pakankamai gerai.





Pav x ir pav y grafikai vaizduoja 2 skirtingų parametrų kalmano filtrus.

Filtro parametrai Ex ir Ez leidžia nustatyti Kalmano filtro pasitikėjimo matavimais ir modeliu proporcijas. Jeigu yra tiksliai žinoma, kokia yra sistemos būsenos lygtis, naudojame mažą Ex ir didelį Ez. Tokia atveju filtras mažiau atsižvelgia į matavimus. Jeigu nežinome tikslaus modelio naudojame didelį Ex nusakantį, kad Kalmano filtras turi labiau orientuotis į matavimų parodymus



Pav w. Kalman1 filtro Ex / Ez santykis mažas, todėl filtras ne taip sipriai reaguoja į matavimus. Iš grafiko matyti, kad į matavimus filtras vis tiek atsižvelgia. Kalman2 filtras turi didelį Ex / Ez santykį ir bando kuo tiksliau prisitaikyti prie matavimų. Filtrai, kurių Ez ir Ex reikšmės yra skirtingos, bet jų santykis yra vienodas elgiasi identiškai.

# Dalelių (particles) filtras

Dalelių filtras, dar kitaip kartais ne visiškai teisingai vadinamas kaip nuoseklus Monte Carlo metodas, kuris apima daugiau panašių algoritmų, yra labai populiarus dinaminių sistemų filtravimo algoritmas. Viena iš jo populiarumo priežasčių yra ta, kad jis yra labai lankstus ir priešingai nei Kalmano filtras tinka ne tik linijinėms sistemoms ir gali filtruoti ne tik pagal normalųjį skirstinį pasiskirsčiusį triukšmą. Pagrindinė filtro algoritmo mintis yra naudoti sugeneruotus sistemos būsenos mėginius (samples) dar kitaip vadinamus dalelėmis (particles), kad simuliuoti sistemos būseną ateityje įvertinant iki šiol turimus netikslius sistemos būsenos matavimus. Po kiekvieno naujo matavimo filtras atnaujina sugeneruotus būsenos mėginius, kad kuo tiksliau numatytų tikrąją sistemos būseną. Filtro tikslumas labiausiai priklauso nuo 2 faktorių:

* Generuojamų dalelių skaičiaus.
* Funkcijos, kuri aprašo stebimą sistemos modelį ir yra naudojamos dalelių būsenos atnaujinimui.

Nagrinėjamame filtro variante šie du faktoriai yra konstantos ir nekinta. Yra filtro variantų, kur šie dydžiai kinta taip gaunant didesnį tikslumą ir geresnį algoritmo našumą. Našumas tiesiogiai priklauso nuo generuojamų dalelių skaičiaus, bet tikslumas tiesiogiai nuo to nepriklauso. Didinant dalelių skaičių tikslumas artėja prie ribos, kur net padvigubinus dalelių skaičių tikslumas beveik nepakinta. Galimybė filtrui prisitaikyti ir nebegeneruoti neprasmingų dalelių padeda sumažinti didelį skaičiavimų skaičių, vieną iš silpnųjų filtro pusių. [3]

## Matematinis dalelių filtro modelis

Dalelių filtro algoritmą sudaro 4 žingsniai:

* Pradinių dalelių sugeneravimas
* Dalelių būsenos atnaujinimas
* Dalelių svorių perskaičiavimas
* Dalelių atrinkimas (resampling)

Pirmasis žingsnis atliekamas tik vieną kartą filtravimo pradžioje, vėliau yra kartojami 2-4 žingsniai.

## Dalelių filtro įgyvendinimas programoje

Dalelių filtras, kaip ir Kalmano filtras, įgyvendintas Dart programavimo kalboje. Algoritmą įgyvendina 2 klasės esančios filters bibliotekoje:

* Particles - filters/particles.dart
* ParticlesModel - filter/particles\_model.dart

Bendra filtro logika yra aprašyta Particles klasėje ir atskirta nuo konkrečios sistemos modelio, kuris aprašomas sukuriant ParticlesModel objektą. Kadangi palyginimuose modeliuojame sistemas su baltu pagal normalųjį skirstinį pasiskirsčiusiu triukšmu, filtras naudoja pagalbinę math bibliotekos klasę Gaussian, kurios aprašymas pateiktas anksčiau.

Particles klasė aprašo tik skeletines filtro funkcijas, kurios kviečia funkcijas aprašytas konkrečioje filtro modelio klasėje ParticlesModel, nes kiekvienas sistema gali turėti skirtingas sistemą aprašančias funkcijas. Sukuriant ParticleModel klasės objektą reikia aprašyti ir konstruktoriui paduoti šias funkcijas:

* Generate – funkcija skirta sugeneruoti pradinį dalelių rinkinį. Dažnu atveju ši funkcija generuoja dalelias pasiskirsčiusiais pagal normalųjį skirstinį aplink pirmą gautą matavimą.
* Move – funkcija skirta atnaujinti dalelės būseną. Ši funkcija yra skirtinga visoms sistemoms. Paprastai sistemai, kurią nusako judėjimo lygtis x = x + v, kur x – objekto koordinatė, o v – koordinatės kitimo greitis, ši funkcija būtų panaši į:  
  KODAS
* Weight – funkcija suskaičiuojanti dalelės svorį. Dalelės svoris – tai tikimybė, kad stebint dabartinį matavimą tikroji reikšmė yra būtent tokia, kokią turi ši dalelė. Funkcija priklauso nuo triukšmo funkcijos ir sistemos dimensijos (stebimų parametrų skaičiaus). Paprastai judėjimo sistemai ši funkcija grąžina normaliojo skirstinio reikšmę P.
* Mean – funkcija skirta apskaičiuoti patikslintai matavimo reikšmei. Dažniausiai ši reikšmė skaičiuojama imant atrinktų dalelių matmenų vidurkį. Iš čia ir paimtas šis nevisiškai teisingai funkcijos prasmę atspindintis pavadinimas, nes patikslintos vertės skaičiavimui galima rinktis ir kitokį algoritmą.
* Clone – pagabinė funkcija skirta dalelės objektą atitinkančiai struktūrai.
* aNoise – proceso triukšmą generuojanti funkcija.
* mNoise – matavimų triukšmą generuojanti funkcija.

### Triukšmo generavimo funkcijos

Dalelių filtras neturi ribojimo optimaliai veikti tik su linijinėmis baltą, pagal normalųjį skirtinį pasiskirsčiusį, triukšmą turinčiomis sistemomis, tačiau kompiuteriu generuoti galime tik tolygiai pasiskirsčiusius atsitiktinius skaičius. Kyla klausimas kaip generuoti atsitiktinius skaičius, pasiskirsčiusius pagal kažkokią funkciją f(). Kad išspręsti čia problemą mes pasinaudojame kita atsitiktinius skaičius generuojančia funkcija g(), dažnu atveju tinkamas yra normalusis skirstinys. Pagal žinomą funkciją g() sugeneruojame savo skaičius. Pav x. Kur stulpelių dažnumas atspindi skaičiaus sugeneravimo tikimybę. Šios skaičius pasveriame pagal mus dominančią funkciją f() pav y. Stulpelių aukštis atspindi skaičiaus svarbą. Taip gaunamas rezultatas, kur retai pasikartojantis skaičius m turi būti dažnas funkcijos f() skirstinyje, todėl jo svoris yra didelis. Iš grafiko matosi, kad kuo daugiau tokių skaičių generuojame tuo tiksliau perteikiame norimą funkciją g().

<http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam09-particle-filter-4.pdf>

### Filtro veikimo pavyzdys ne linijinei sistemai

Šiame pavyzdyje bus nagrinėjama ne linijinė sistema, kurią aprašo funkcija:

– sistemos būsena laiko momentu t (būsena realiu laiku),

– ankstesnė sistemos būsena,

t - laikas,

Ex – yra sistemos proceso triukšmas.

# Filtru palyginimas

## Voltmetro eksperimentas

Pirmajam filtrų palyginimui buvo iškeltas uždavinys palyginti filtrus filtruojant menamą voltmetrą. Tam reikėjo sukurti voltmetro parodymų simuliacinę programą, kuri imituotų netikslius menamo voltmetro matavimus. Šitame uždavinyje voltmetras apibrėžtas kaip prietaisas, stebintis 10 voltų ne idealiai veikiančią sistemą. Sistemos procesas turi triukšmą, kuris yra pagal normalųjį skirstinį pasiskirstęs atsitiktinis triukšmas su vidurkiu 0 ir standartiniu nuokrypiu, kurį turi būti galimybė keisti. Voltmetro parodymai irgi nėra tikslūs, jie turi tokį patį triukšmą su kitu irgi redaguojamu standartiniu nuokrypiu. Taip pat turi būti galimybė keisti eksperimento trukmę, kiek stebėjimų darysime, Kalmano ir dalelių filtro parametrus.

### Kalmano filtro modelis

### Dalelių filtro modelis

## Judėjimo simuliacija

### Gps koordinačių filtravimas

Dažnu atveju mes nežinome visų duomenų apie sistemą arba tie duomenys nėra idealiai matematiškai tikslūs. Daug procesų vykstančių gamtoje priklauso nuo daugybės skirtingų veiksnių ir jų visų apibrėžti sistemos modelyje negalime. Tai reikalautų labai sudėtingų fizikinių formulių kas labai padidintų skaičiavimams reikalingų veiksmų skaičių ir reikalautų daug resursų iš skaičiavimus atliekančios įrangos. Taip pat reikėtų matuoti visus sistemai įtaką darančius veiksnius. Net paprastam judančiam objektui tokių veiksnių yra labai daug: trintis, oro pasipriešinimas, priklausantis nuo slėgio ir t. t. Visų jų išmatuoti negalime. Visi šie nematuojami veiksniai yra laikomi triukšmu, o medelis dažniausiai yra suprastinamas į idealiosios fizikos modelį.

Buvo iškeltas uždavinys palyginti, kuris iš filtrų yra tikslesnis lyginant realiomis sąlygomis gautus duomenis. Kad įgyvendinti šitą uždavinį buvo pasirinkta tokia darbų eiga:

* sukurti GPS koordinates registruojančią programėlę Android telefonui
* surinkti duomenis
* pritaikyti filtrus surinktiems duomenims

### Android programėlė gps koordinačių fiksavimui

GPS koordinates fiksuojančių programėlių galima rasti jau parašytų, tačiau nėra žinoma kaip jos veikia. Gali būti, kad jos jau savyje turi integruotus filtravimo algoritmus ir tai gali daryti įtakos mūsų gautiems rezultatams lyginant filtrus. Todėl buvo nutarta patiems parašyti primityvią programėlę šiai užduočiai įgyvendinti.

Programėlės vartotojo sąsają sudaro 2 mygtukai ir vienas įvesties laukas pav z

* Start – pradeda registruoti koordinates į nurodytą failą.
* Stop – baigia koordinačių registravimą.
* Įvesties laukas – failo vardas, kuriame bus registruojamos koordinatės

Programėlė tekstinį failą išsaugo SD kortelėje kataloge Downloads. Taip daroma dėl Android saugumo reikalavimų, kurie kitu atveju failą leistų matyti tik jo savininkui, tai yra pačiai programėlei, ir mes negalėtumėm jo išsisaugoti kompiuteryje. Kartais šis failas vis tiek yra nematomas prijungus telefoną prie kompiuterio, tokiu atveju rekomenduojama naudoti AirDroid programėlę, su kuria galima failą pasiekti ir išsisaugoti.

Išsaugoto duomenų failo formatas yra cvs. Kiekvienas naujas matavimas yra išsaugomas naujoje eilutėje, o skirtingi matavimo parametrai atskiriami kableliais. Eilutėje eilės tvarka yra išsaugomi:

FAILO EILUTE

1. 0 arba 1 – matavimo tipas. 0 – matavimas gautas iš GPS sensoriaus. 1 – matavimas gautas iš Wi-Fi ir mobiliojo interneto lokacijos.
2. Laikas. Išreikštas milisekundėmis, bet milisekundžių dalis yra visada nulinė. Gali priklausyti nuo telefono.
3. Platuma
4. Ilguma
5. Aukštis. Aukštis gaunamas tik iš GPS sensoriaus.
6. Greitis. Šis dydis nėra matuojamas, jis yra skaičiuojamas iš praeitos koordinatės atimant dabartinę.
7. Tikslumas. Sensorius pateikia pateiktų koordinačių tikslumą. Kyla klausimas ar tikslumas yra tikslus.

Matavimai daromi taip dažnai, kaip leidžia naudojamas telefonas. Naudojant Sony Ericsson Arc S šis intervalas yra viena sekundė.

Programėlė yra labai primityvi ir neišbaigta, bet atlieka savo funkciją teisingai naudojant. Koordinatės išsaugomos tik paspaudus mygtuką stop, todėl išjungus programą registravimo failas lieka tuščias. Dėl šios priežasties programėlė duomenų neišsaugo kai yra pakeičiama telefono oriantacija, jis paverčiamas arba telefonas užsirakina. Taip yra dėl Android architektūros. Programėlės instaliacinis paketas pateiktas prie kodu bin kataloge. Norint įsirašyti programėlę iš instaliacinio paketo reikia telefone įjungti debug režimą ir aktyvuoti programėlių instaliaciją iš trečiųjų šalių.

## Lėktuvo nusileidimo trajektorijos filtravimas

## 

# isvados

# Literaturos sarasas

1. <http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf>
2. <http://en.wikipedia.org/wiki/Marsaglia_polar_method>
3. <http://web.ing.puc.cl/~asoto/papers/Soto-IJCAI-05.pdf>
4. <http://faculty.chicagobooth.edu/nicholas.polson/teaching/41900/pf-intro.pdf>