УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Кристина Матовић

КАЛМАНОВ ФИЛТЕР И ЊЕГОВЕ МОДИФИКАЦИЈЕ С ПРИМЕНОМ У ПРАЋЕЊУ ПОКРЕТНОГ ЦИЉА

мастер рад

Београд, 2021.

Ментор:

др Бојана Милошевић, доцент
Универзитет у Београду, Математички факултет

Чланови комисије:

др Анђелка Ковачевић, ванредни професор
Универзитет у Београду, Математички факултет

др Марко Обрадовић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 30. септембар 2021.

Велико хвала менторки Бојани на несебичној подршци и сугестијама током израде мастер рада!

Наслов мастер рада: Калманов филтер и његове модификације с применом у праћењу покретног циља

Резиме: Калманов филтер је оптимални рекурзивни алгоритам за обраду података који даје добре резултате у оцени стања динамичких система из зашумљених и некомплетних мерења. Прва примена Калмановог филтера била је у праћењу свемирских летелица, након чега Калманов филтер налази примену у роботици дигиталним израчунавањима у системима управљања, навигацији и финансијама. Калманов филтер је рачунарски ефикасан и једноставан за имплементацију у реалном времену, јер при новом мерењу не захтева поновну обраду претходних података. У овом раду упознаћемо се са Калмановим филтером и неким од његових модификацијам, које дају добре резултате у случајевима система са нелинеарном динамиком, најпре са теоријског становишта, а потом и кроз њихову примену у праћењу покретног циља.

Рад је подељен у четири целине. У првој целини уведени су основни појмови потребни за разумевање даљег садржаја рада. У другој целини упознаћемо се са традиционалним Калмановим филтером, извести његов предикциони и корекциони корак за дискретан случај и приказати их на конкретном примеру. У трећој целини пажња је посвећена модификованим верзијама Калмановог филтера које су значајно проширила област његове примене, а то су: проширени Калманов филтер и Калманов филтер без мириса. На крају, последња целина је посвећена примени Калмановог филтера у праћењу покретног циља. Кроз примере праћења покретног циља, у овом случају на основу радарског сигнала, назначене су предности и мане употребе модификованих верзија Калмановог филтера у односу на традиционални линеарни Калманов филтер. Имплементација поменутих филтера одрађена је у програмском језику Пајтон (*Python*), а кодови су дати на крају рада.

Кључне речи: Калманов филтер, модификације Калмановог филтера, оцена стања система, праћење покретног циља

Садржај

Глава 1

Увод

У наставку је дат кратак преглед статистичких и математичких појмова, као и основних концепата који су значајни за разумевање даљег садржаја мастер рада.

1.1 Апсолутно непрекидне случајне величине

Дефиниција 1.1.1 Случајна величина X је апсолутно непрекидног типа, ако постоји ненегативна Борелова функција f дефинисана на $\mathbb{R}$ и

$$\mathbb{P}(X \leqslant x) = \int_{-\infty}^{x} f(t)dt, \quad \forall x \in \mathbb{R}. \tag{1.1}$$

Функција f је густина расподеле апсолутно непрекидне случајне величине X и важи

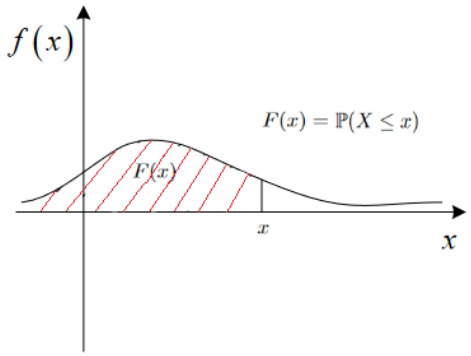
$$\int_{-\infty}^{\infty} f(t)dt = 1.$$

Дефиниција 1.1.2 Нека је X апсолутно непрекидна случајна величина са функцијом густине f. Вредност функције расподеле случајне величине X је дата једнакошћу

$$F(x) = \int_{-\infty}^{x} f(t)dt, \quad \forall x \in \mathbb{R}. \tag{1.2}$$

и важи:

1. $f(x) = F'(x)$
2. $\mathbb{P}(X \in (a,b)) = \mathbb{P}(X \in [a,b]) = \int_a^b f(t)dt, \quad a,b \in \mathbb{R},\ a < b$



Слика 1.1: Апсолутно непрекидна случајна величина X са функцијом густине f

3. $\mathbb{P}(X = x) = 0, \quad \forall x \in \mathbb{R}.$

Дефиниција 1.1.3 Математичко очекивање $E(X)$ апсолутно непрекидне случајне величине X са функцијом густине f израчунава се по формули:

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx, \tag{1.3}$$

под условом да одговарајући интеграл апсолутно конвергира.

Теорема 1.1.1 Ако је X апсолутно непрекидна случајна величина са функцијом густине f и ϕ реална функција реалне променљиве, онда важи

$$E(\phi(X)) = \int_{-\infty}^{\infty} \phi(x) f(x) dx. \tag{1.4}$$

Уместо назива математичко очекивање, често се користи и назив момент првог реда. Применом једнакости (1.4), тако да је $\phi(X) = X^k$, добија се формула за момент k-тог реда, односно

$$E(X^k) = \int_{-\infty}^{\infty} x^k f(x) dx. \tag{1.5}$$

Дефиниција 1.1.4 Ако је X апсолутно непрекидна случајна величина, онда је њена дисперзија или варијанса дефинисана једнакошћу

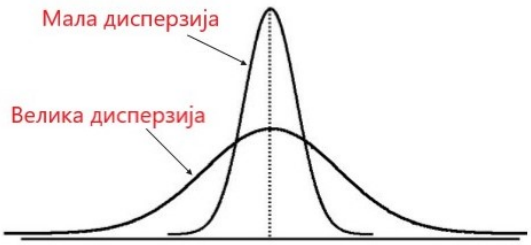
$$D(X) = Var(X) = E((X - E(X))^2). \tag{1.6}$$

Као последица Теореме 1.1.1, често се за налажење дисперзије користи израз:

$$D(X) = Var(X) = E(X^2) - (E(X))^2. \tag{1.7}$$

Дисперзија је нумеричка карактеристика која представља меру варијабилности или расипања вредности случајне величине X од њене средње вредности. Што је расејаност вредности случајне величине X од њене средње вредности већа, дисперзија је већа (Слика 1.2).
Квадратни корен дисперзије назива се стандардна девијација или стандардно одступање.

$$\sigma(X) = \sqrt{D(X)}$$



Слика 1.2: Функције густина случајних величина са истим математичким очекивањем и различитом дисперзијом

Ако је X апсолутно непрекидна случајна величина и a,b реалне константе, важи

$$E(aX + b) = aE(X) + b, \quad D(aX + b) = a^2 D(X). \tag{1.8}$$

Стандардизована случајна величина X^* се добија из случајне величине X трансформацијом

$$X^* = \frac{X - E(X)}{\sqrt{D(X)}} \tag{1.9}$$

Математичко очекивање стандардизоване случајне величине има вредност 0, а дисперзија има вредност 1.

Дефиниција 1.1.5 Апсолутно непрекидне случајне величине X и Y дефинисане на истом простору вероватноћа су независне ако и само ако важи

$$f_{X,Y}(x, y) = f_X(x) f_Y(y), \quad \forall x \in X, \forall y \in Y \tag{1.10}$$

где је $f_{X,Y}$ заједничка густина случајних величина X и Y чије су појединачне густине, редом, f_X и f_Y.

Дефиниција 1.1.6 Нека су X и Y апсолутно непрекидне случајне величине дефинисане на истом простору вероватноћа. Коваријација између случајних величина X и Y је мера заједничког варијабилитета између њих и израчунава се по формули:

$$Cov(X,Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(X - E(X))(Y - E(Y))f_{X,Y}(x,y)dxdy, \tag{1.11}$$

где је $f_{X,Y}$ заједничка густина случајних величина X и Y.

Дефиниција 1.1.7 Нека су X и Y апсолутно непрекидне случајне величине дефинисане на истом простору вероватноћ. Коефицијент корелације случајних величина X и Y се израчунава по формули:

$$\rho = \frac{Cov(X,Y)}{\sqrt{D(X)D(Y)}}, \tag{1.12}$$

и важи $-1 \leq \rho \leq 1$.

Ако је вредност коваријансе 0, не постоји линеарна зависност између случајних величина X и Y, односно, случајне величине X и Y су некорелисане. Независне случајне величине су некорелисане, док некорелисане случајне величине нису нужно независне.

1.2 Нормална расподела

Нормална (Гаусова) расподела припада фамилији апосолутно непрекидних расподела вероватноће. Због своје примене у многим пољима нормална расподела је најчешће коришћена фамилија расподела у теорији вероватноће и статистици. Други назив за нормалну расподелу је Гаусова расподела. Користе се и називи Гаусов закон, Гаус-Лапласова расподела, други Лапласов закон. Нормалну расподелу је први проучавао и користио, у апроксимацији биономне расподеле, француски математичар Муавр (Abraham de Moivre, 1667 - 1754) још 1733. године.
Грешке при мерењу неке величине често имају нормалну расподелу. Нормална расподела добро апроксимира неке друге непрекидне и дискретне расподеле. Под одређеним условима, нормална расподела добро апроксимира и

збир великог броја независних случајних величина са истом расподелом. Постоји више тврђења која се на то односе и свако од њих носи назив централна гранична теорема.

Дефиниција 1.2.1 Нека је X апсолутно непрекидна случајна величина са функцијом густине

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}, \quad x \in \mathbb{R} \tag{1.13}$$

и функцијом расподеле

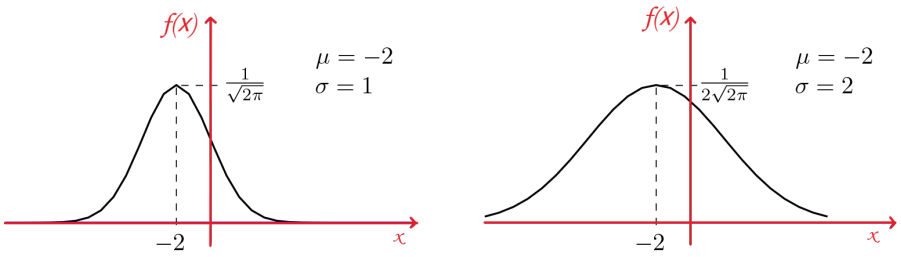
$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}, \quad x \in \mathbb{R} \tag{1.14}$$

Случајна величина X тада има нормалну или Гаусову расподелу и $\mu(\mu \in \mathbb{R})$ је параметар локације, а $\sigma(\sigma > 0)$ параметар скалирања.

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$E(X) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\} = \mu$$

$$D(X) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} (x-\mu)^2 \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\} = \sigma^2$$



Слика 1.3: Графици густине нормалне расподеле у зависности од промене параметра скалирања σ

Нека својства нормалне расподеле

1. Густина расподеле је симентрична у односну на $x = \mu$.
2. Максимум густине нормалне расподеле је $\frac{1}{\sigma\sqrt{2\pi}}$.
3. Ако $X \sim \mathcal{N}(\mu, \sigma^2)$, тада $aX + b \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$.
4. Ако $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ за $i = 1, ..., n$ и ако су $X_1, X_2, ..., X_n$ независне случајне величине,тада

$$X_1 + \cdots + X_n \sim \mathcal{N}(\mu_1 + \cdots + \mu_n, \sigma_1^2 + \cdots + \sigma_n^2)$$

Дефиниција 1.2.2 Нека је $\mathbf{X}$ n-димензиони случајни вектор [1] са функцијом густине

$$f(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left((x-\mu)^\intercal \mathbf{\Sigma}^{-1}(x-\mu)\right)\right\}, \quad x \in \mathbb{R}^n \tag{1.15}$$

где је $\mathbf{\Sigma}$ симетрична и позитивно дефинитна коваријациона матрица, а $|\mathbf{\Sigma}|$ њена детерминанта. Случајни вектор $\mathbf{X}$ тада има n-димензионалну нормалну расподелу.

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$$

Нека својства вишедимензионалне нормалне расподеле

1. Ако $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, тада $\mathbf{AX} + \mathbf{B} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{B}, \mathbf{A\Sigma A}^\intercal)$.
2. Ако за случајни вектор $\mathbf{Z} = (\mathbf{X}^\intercal, \mathbf{Y}^\intercal)^\intercal$ важи

$$\mathbf{Z} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{pmatrix}, \begin{pmatrix}\mathbf{\Sigma}_{xx} & \mathbf{\Sigma}_x y \\ \mathbf{\Sigma}_{xy}{}^\intercal & \mathbf{\Sigma}_{yy}\end{pmatrix}\right),$$

онда за маргиналне расподеле за $\mathbf{X}$ и $\mathbf{Y}$ важи

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_x, \mathbf{\Sigma}_{xx})$$

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma}).$$

[1] За објашњење појма случајни вектор погледати одељак 1.4 о случајном вектору

Такође, за условну расподелу $\mathbf{Y}|\mathbf{X} = x$ важи следеће:

$$\mathbf{Y}|\mathbf{X} = x \sim \mathcal{N}\left(\boldsymbol{\mu}_y + \boldsymbol{\Sigma}_{xy}{}^{\intercal}\boldsymbol{\Sigma}_{xx}^{-1}(x - \mu_x), \boldsymbol{\Sigma}_{yy} - \boldsymbol{\Sigma}_{xy}{}^{\intercal}\boldsymbol{\Sigma}_{xx}^{-1}\boldsymbol{\Sigma}_{xy}\right)$$

3. Случајни вектор $\mathbf{X}$ има вишедимензионалну нормалну расподелу ако и само ако за сваки вектор $\mathbf{u}$, који је различит од нуле, $\mathbf{u}^{\intercal}\mathbf{X}$ има једнодимензиони нормалну расподелу.

1.3 Јакобијева и Хесијан матрица

Дефиниција 1.3.1 ([25]) Нека је $f\colon \mathbb{R}^n \to \mathbb{R}^m$ векторска функција више променљивих, која се може представити у облику

$$f(x_1, x_2, ..., x_n) = (f_1(x_1, x_2, ..., x_n), f_2(x_1, x_2, ..., x_n), ..., f_m(x_1, x_2, ..., x_n)),$$

где су $f_i\colon \mathbb{R}^n \to \mathbb{R}$. Сви парцијални изводи ових функција, ако постоје, могу се представити матрицом димензија $n \times m$, познатом као Јакобијева матрица (Carl Gustav Jacob Jacobi, 1804 - 1851) пресликавања f, на следећи начин:

$$J = J_f(x_1, ...x_n) = \frac{\partial(f_1, ..., f_m)}{\partial(x_1, ...x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{1.16}$$

Јакобијева матрица представља најбољу линеарну апроксимацију диференцијабилне функције у околини дате тачке у следећем смислу

$$f(x) = f(p) + J_f(p)(x - p) + o(||x - p||),$$

где x припада довољно малој околини тачке p.

Дефиниција 1.3.2 Други извод функције $f\colon \mathbb{R}^n \to \mathbb{R}$ у тачки $\mathrm{X} \in \mathbb{R}^n$ дефинише се на следећи начин:

$$D^2 f(X) = \nabla^2 f(X) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2}, \end{bmatrix}. \tag{1.17}$$

Матрица $\nabla^2 f(X)$ је позната као Хесијан матрица (Ludwig Otto Hesse, 1811 - 1874) функције f у тачки X. У даљем раду, биће искоришћена приликом извођења предикционог и корекционог корака једне од модификованих верзија Калмановог филтера која је, у случајевима високе нелинеарности, чест избор.

1.4 Случајни вектор

Појам случајног вектора је уопштење појма случајне величине. Случајни вектор $\mathbf{x}$, односно n-димензионални случајни вектор $\mathbf{x}$, је уређена n-торка случајних величина x_i и означава се са

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Нека је $\{x_1, x_2, ..., x_n\}$ n-димензионални вектор горе наведене форме. Хилбертов простор $\mathcal{H}$ можемо дефинисати тако да садржи све векторе чије су компоненте линеарна комбинација од x_i, $i = \overline{1, n}$.

Дефиниција 1.4.1 Нека су $\mathbf{x}$ и $\mathbf{y}$ елементи Хилбертовог простора $\mathcal{H}$. Њихов унутрашњи производ дефинишемо са

$$\langle \mathbf{x}, \mathbf{y} \rangle = E(\mathbf{x}^\intercal \mathbf{y}) = E\left(\sum_{i=1}^{n} x_i y_i \right). \tag{1.18}$$

Њиме индуковану норму вектора x у простору $\mathcal{H}$ записујемо са

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{E(x_1^2 + x_2^2 + ... + x_n^2)} \tag{1.19}$$

$$= \sqrt{E(x_1^2) + E(x_2^2) + ... + E(x_n^2)} = \left(\mathrm{Tr}\left(E(\mathbf{x}\mathbf{x}^\intercal) \right) \right)^{\frac{1}{2}}, \tag{1.20}$$

где је $E(\mathbf{x}\mathbf{x}^\intercal)$ очекивана вредност случајне матрице $\mathbf{x}\mathbf{x}^\intercal$, односно

$$E(\mathbf{x}\mathbf{x}^\intercal) = \begin{bmatrix} E(x_1x_1) & E(x_1x_2) & \cdots & E(x_1x_n) \\ E(x_2x_1) & E(x_2x_2) & \cdots & E(x_2x_n) \\ \vdots & \vdots & \ddots & \vdots \\ E(x_nx_1) & E(x_nx_2) & \cdots & E(x_nx_n) \end{bmatrix}. \tag{1.21}$$

Унутрашњи производ $\langle \mathbf{x}, \mathbf{y} \rangle$ се тада може записати и на следећи начин:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathrm{Tr}\left(E(\mathbf{x}\mathbf{y}^\intercal)\right). \tag{1.22}$$

Дефиниција 1.4.2 Два вектора су ортогонална, односно $\mathbf{x} \perp \mathbf{y}$, ако важи $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

Теорема 1.4.1 Ако су $\mathbf{x}$ и $\mathbf{y}$ некорелисани и $E(\mathbf{x}) = E(\mathbf{y}) = 0$, онда су $\mathbf{x}$ и $\mathbf{y}$ ортогонални један на други.

Дефиниција 1.4.3 Коваријациона матрица случајног вектора $\mathbf{x}$ је матрица дата са

$$\Sigma = E((\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^\intercal)). \tag{1.23}$$

Ако је $E(\mathbf{x}) = 0$, коваријациона матрица се може записати и као $E(\mathbf{x}\mathbf{x}^\intercal)$.

Глава 2

Калманов филтер

У наставку је дат кратак осврт на историју и значај употребе Калмановог филтер. Описан је традиционални линеарни Калманов филтер за дискретан случај и изведени његов предикциони и корекциони корак. На крају поглавља, у циљу илустрације и разумевања Калмановог филтера, дат је кратак и једноставан пример његове употребе.

2.1 О Калмановом филтеру

Претпоставимо да се налазимо у аутономном возилу. Возило користи различите типове сензора, као што су нпр. ГПС (енгл. Global Positioning System), акцелерометар и камера, који пружају увид у локацију на којој се возило налази и његову интеракцију са другим возилима и пешацима на путу. Ипак, током боравка возила у дугачким тунелима, као последица интерференције, ГПЦ сигнал слаби и пружа мање прецизне информације о тренутној локацији возила. Аутономно возило тада нема тачан увид у локацију на којој се тренутно налази, те се поставља питање на који начин се може решити овај проблем. Како је тренутно убрзање једнако првом изводу брзине по времену, или другом изводу координате по времену, једно од могућих решења је коришћење података акцелерометра у комбинацији са слабим ГПС сигналом. Поменута мерења су зашумљена и не може им се потпуно веровати, те се у циљу решавања овог проблема користи Калманов филтер који без обзира на прецизност, у циљу оцене тренутне вредности променљиве од интереса, процесуира сва доступна мерења. Ово је само један од многобројних примера примене Калмановог филтера.

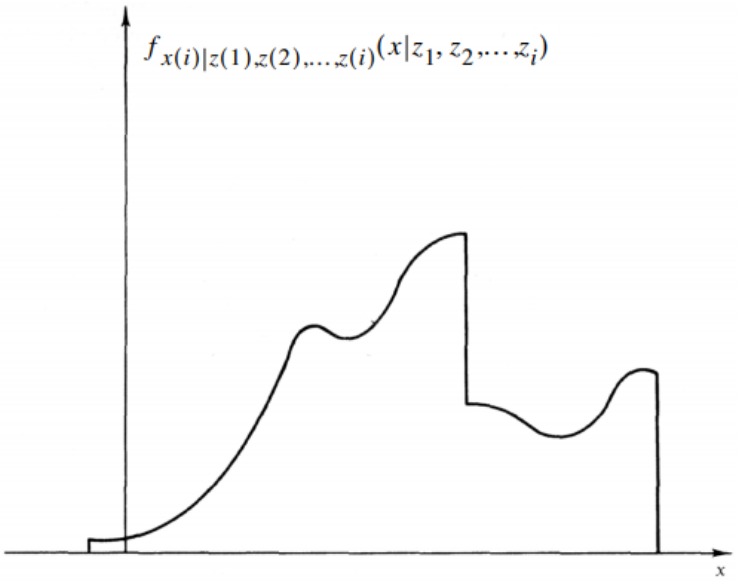
Калманов филтер је добио назив по свом оснивачу Рудолфу Калману (Rudolf E. Kálmán 1930 - 2016) који 1960. године објављује рад, у којем презентује прву верзију

Калмановог филтера. Након што је рад објављен, Национална ваздухопловна и свемирска администрација (енгл. The National Aeronautics and Space Administration) је препознала могућу примену Калмановог филтера у процени трајекторија свемирског брода (пројекат „Appolo") и на тај начин решила проблем распознавања исправних података од неисправних. Данас, након 60 година, Калманов филтер није изгубио на значају већ је, и даље, са својим модификацијама које су проширile област његове примене, најпознатији и најкоришћенији алгоритам за оцену стања система.
Калманов филтер је оптимални рекурзивни алгоритам обраде података. Постоји више начина за дефинисање оптималности, у зависности од одабраног критеријума за анализу перформанси. Испоставаља се да је, под одређеним претпоставкама, Калманов филтер оптималан у односу на било који критеријум који има смисла сагледати. Један од аспеката оптималности Калмановог филтера је свакако обједињавање свих доступних информација. Он процесуира сва доступна мерења, без обзира на њихову прецизност, у циљу оцене тренутне вредности променљиве од интереса. Приликом оцене користи се:

1. знање о систему и динамици мерног инструмента
2. статистички опис шума, грешке мерења и непоузданости динамичког модела
3. било које доступне информације о почетним условима променљиве од интереса

На пример, претпоставимо да за одређивање брзине авиона користимо доплер радар, инерцијални навигациони систем, питоову цев, као и информације о правцу ветра и статичком притиску. Радије него да игнорише неки од претходних података, Калманов филтер комбинује све доступне информације о променљивој од интереса, заједно са примарним знањем о самом моделу и уређајима за мерење, како би одредио статистички најбољу оцену брзине. Другим речима, то би значило да уколико употребимо више различитих филтера више пута за исти проблем, просечан резултат Калмановог филтера ће бити бољи од просечног резултата било којег другог филтера. Једна од значајних предности Калмановог филтера када је реч о његовој практичној имплементацији је рекурзивност, јер, за разлику од извесног броја метода обраде података, не захтева складиштење и поновну обраду свих претходних података сваки пут када долази до нових мерења. Реч филтер користи се због чињенице да се при оцени променљиве од интереса филтрира шум из података.
Оно што било који тип филтера идејно покушава да уради јесте постизање оптималне (у смислу минимизације грешке) оцене променљиве од интереса из зашумљених и некомплетних мерења. Постоји више начина за реализацију овог циља, а један од њих је и Бајесов приступ (Thomas Bayes 1702-1761). Важност овог приступа је у успостављању математичког правила које објашњава начин на који је потребно мењати

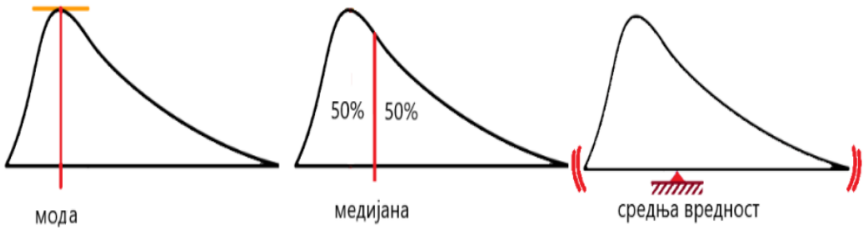
постојећа убеђења у складу са новим подацима, што би значило да овај приступ омогућава комбиновање нових података са већ постојећим знањем. У Бајесовском приступу, при доношењу закључака, битну улогу имају условне вероватноће и с њима повезане условне расподеле. Слика 2.1 (преузета из [22]) представља пример графика функције условне густине расподеле промењљиве од интереса x у i-том временском тренутку $(x(i))$, ако су познате вредности које су добијене са мерних уређаја у временским тренуцима $1, 2..., i$ $(z(1), z(2), ..., z(i))$. Промењљива од интереса $x(i)$ представља једнодимензиони вектор позиције возила у тренутку i, а $z(i)$ дводимензиони вектор који представља измерене вредности позиције возила са два различита сензора у тренутку i. Оваква условна густина расподеле садржи све доступне информације



Слика 2.1: Пример условне густине расподеле

о случајној величини $x(i)$, што би значило да, за вредности прикупљене са сензора закључно са тренутком i, пружа информације о вероватноћи да случајна величина $x(i)$ узме вредност која припада неком опсегу вредности или неку тачно одређену вредност. Термин условна у функцији густине расподеле случајне величине $x(i)$ указује на зависност њеног облика и положаја на x оси од измерених вредности које су добијене са сензора. Облик функције густине тада носи информацију прецизности вредности промењљиве од интереса (у овом случају брзине). Уколико график функције густине има изражен врх, највећи део вероватноће сконцентрисан је на уском опсегу вредности. Са друге стране, уколико график функције нема значајно изражен врх, вероватноћа се простире на ширем опсегу вредности, па је непрецизност вредности промењљиве од интереса већа. За оптималну оцену промењљиве од интереса

($x(k)$) може се одабрати (1.) средња вредност, (2.) мода или (3.) медијана условне расподеле. (Слика 2.2)



Слика 2.2: Геометријски приказ средње вредности, моде и медијане произвољне густине расподеле

Калманов филтер је управо заснован на Бајесовој формули, за проблеме у којима се систем може описати линеарним моделом а шум система и шум мерења Гаусовим белим шумом. Бели шум је сваки случајан процес са независним вредностима, средње вредности $\mu = 0$ и дисперзије σ^2. Није обавезно да буде из нормалне расподеле, али уколико јесте онда се назива Гаусов бели шум. Под овим условима, средња вредност, модус, медијана и практично било који разуман избор за оптималну оцену променљиве од интереса се коинцидирају, па постоји јединствена статистички најбоља оцена вредности променљиве од интереса. Такође, под овим претпоставкама, Калманов филтер даје боље резултате у односу на било који други филтер. Неке од претходних претпоставки се могу ослабити. На пример, уколико се претпоставка о нормалној расподели белог шума уклони, може се показати да је Калманов филтер и даље најбољи (у смислу минималне варијансе грешке) филтер,из класе линеарно непристрасних филтера. Ипак, претходне претпоставке су оправдане за многе потенцијалне примене Калмановог филтера. Линеарни модел система је оправдан из више разлога. Наиме, линеарни системи су погоднији за рад због лакше манипулације над њима, а такође, и сама теорија линеарних система (или диференцијалних једначина) је комплетнија и практичнија од нелинеарне. У случају нелинеарности чест приступ проблему је његова линеаризација у околини одговарајуће тачке. Ипак, уколико се линеарни системи у таквим случајевима покажу неадекватним, концепт Калмановог филтера се може проширити на нелинеарне системе. Неке од модификација Калмановог филтера, као што су проширени Калманов филтер и Калманов филтер без мириса, решавају проблем оцене стања код нелинеарних система. Оправдање за претпоставку о нормалној расподели белог шума може се пронаћи у првој глави,

односно у одељку 1.2 о нормалној (Гаусовој) расподели.
Приликом коришћења Калмановог филтера посматра се одређени математички модел. Математички модели имају више различитих подела и једна од њих је и подела на динамичке и статичке моделе. За разлику од статичких, динамички модели се непрестано мењају у односу на време. Такође, још један од начина класификације математичких модела је и на стохастичке и детерминистичке. За разлику од детерминистичког модела који не садржи случајну компоненту, стохастички модел препознаје случајну природу улазних компоненти.

2.2 Концепција простора стања

Општи модел који покрива целу класу специјалних случајева који су нам од интереса је математички модел система у простору стања (енгл. state space model) или, под другачијим називом, динамички линеарни модел. Увео га је Калман 1960. године, а касније побољшали Ричард Б. (Richard S. Bucy, 1935-2019) и Калман 1961. године. Иако је модел првобитно формиран за праћење свемирских бродова, нашао је примену и у моделирању података из многих других области.
Динамички линеарни модел, у свом основном облику, користи ауторегресиони модел првог реда за **једначину стања**, односно

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k, \qquad k = 0, 1, 2... \tag{2.1}$$

где је $\mathbf{\Phi}_k$ матрица преласка из стања $\mathbf{x}_k$ у $\mathbf{x}_{k+1}$, димензија $n \times n$. Процесни шум у кораку k је n-димензионални вектор $\mathbf{w}_k$, који чини стохастички део једначине стања. Претходна једначина стања заправо дефинише правило по коме се n-димензионални вектор стања $\mathbf{x}_k$, мења.
У динамичком линеарном систему, претпоставља се да почетно стање $\mathbf{x}_0$ није познатно, већ да је дато са познатном средњом вредношћу $\hat{\mathbf{x}}_0$ и коваријационом матрицом $\mathbf{P}_0$, односно важи:

$$\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0)$$

Претпостављајући да не посматрамо вектор стања $\mathbf{x}_k$ директно, већ линеарно трансформисан са додатим шумом, једначина

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \qquad k = 0, 1, 2... \tag{2.2}$$

је **једначина мерења** (једначина опсервација) са матрицом мерења $\mathbf{H}_k$, димензије $m \times n$. Шум мерења је n-димензионални вектор $\mathbf{v}_k$, димензија .

Претпоставља се да су оба шума, тј. процесни шум и шум мерења, Гаусови бели шумови нултих очекиваних вредности и важе следеће претпоставке:

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$$

$$\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$$

$$E(\mathbf{w}_k, \mathbf{w}_j^\intercal) = \begin{cases} Q_k, & j = k \\ 0, & j \neq k \end{cases}$$

$$E(\mathbf{v}_k, \mathbf{v}_j^\intercal) = \begin{cases} \mathbf{R}_k, & j = k \\ 0, & j \neq k \end{cases}$$

$$Cov(\mathbf{w}_k, \mathbf{v}_j^\intercal) = 0 \quad \forall k \geq 0, j \geq 0$$

Оба шума су некорелисана са поченим стањем $\mathbf{x}_0$.

$$Cov(\mathbf{x}_0, \mathbf{w}_k^\intercal) = 0 \quad \forall k \geq 0,$$

$$Cov(\mathbf{x}_0, \mathbf{v}_k^\intercal) = 0 \quad \forall k \geq 0,$$

Једначина стања, као и једначина мерења, може садржати егзогене променљиве (променљиве које су у целости одређене изван система) и фиксне улазе. Нека је r- димензионални вектор $\mathbf{u}_k$ вектор улаза у кораку k са матрицом улаза $\mathbf{B}_k$, димензија $r \times n$. У овом случају, претходни модел се може записати на следећи начин

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k, \quad k = 0, 1, 2... \tag{2.3}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \quad k = 0, 1, 2... \tag{2.4}$$

где вектор $\mathbf{u}_k$ називамо контролним вектором, а матрицу $\mathbf{B}_k$ контролном матрицом. Једначина 2.3 је једначина стања, а једначина 2.4 је једначина мерења.
Напомена: Матрица преласка $\mathbf{\Phi}_k$, матрица мерења $\mathbf{H}_k$ и контролна матрица $\mathbf{B}_k$ су у већини случајева инваријантне у односу на време, тј. $\mathbf{\Phi}_k = \mathbf{\Phi}$, $\mathbf{H}_k = \mathbf{H}$ и $\mathbf{B}_k = \mathbf{B}$, $\forall k \geq 0$.

Пример 2.2.1 (Једначина стања за авион са приближно константним убрзањем)

Посматрајмо авион који се креће са приближно константним убрзањем. Стање авиона описујемо његовом позицијом, брзином и убрзањем у Декартовом координатном систему. Дакле, вектор стања $\mathbf{x}_k$ укључује позицију (a_k, b_k, c_k), брзину $(\dot{a_k}, \dot{b_k}, \dot{c_k})$

и убрзање $(\ddot{a_k}, \ddot{b_k}, \ddot{c_k})$ авиона у Декартовим координатанама:

$$\mathbf{x}_k = \begin{bmatrix} a_k \\ b_k \\ c_k \\ \dot{a_k} \\ \dot{b_k} \\ \dot{c_k} \\ \ddot{a_k} \\ \ddot{b_k} \\ \ddot{c_k} \end{bmatrix}.$$

Контролни вектор и контролна матрица су у овом случају једнаки нули. Даље знамо да, на пример, позиција авиона a у наредном кораку зависи од претходне позиције и пређеног пута, односно његове брзине $\dot{a}$ и убрзања $\ddot{a}$ у претходном кораку, као и времена које је протекло између та два узастопна корака. Дакле, на основу закона о кретању тела, имамо следећи систем једначина:

$$\begin{cases} a_{k+1} = a_k + \dot{a_k}T + \frac{1}{2}\ddot{a_k}T^2 \\ b_{k+1} = b_k + \dot{b_k}T + \frac{1}{2}\ddot{b_k}T^2 \\ c_{k+1} = c_k + \dot{c_k}T + \frac{1}{2}\ddot{c_k}T^2 \\ \dot{a_{k+1}} = \dot{a_{k-1}} + \ddot{a_k}T \\ \dot{b_{k+1}} = \dot{b_{k-1}} + \ddot{b_k}T \\ \dot{c_{k+1}} = \dot{c_{k-1}} + \ddot{c_k}T \\ \ddot{a_{k+1}} = \ddot{a_k} \\ \ddot{b_{k+1}} = \ddot{b_k} \\ \ddot{c_{k+1}} = \ddot{c_k}, \end{cases}$$

где је T време које је протекло између k-тог и $k+1$-ог корака. Једначина стања модела је тада дата са:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k, \qquad k = 0, 1, 2...$$

где је матрица преласка $\mathbf{\Phi}_k$ дата са

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 & 0 & \frac{1}{2}T^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & \frac{1}{2}T^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

а $\mathbf{w}_k$ је процесни шум.

Пример 2.2.2 (Једначина стања за авион са доступним мерењима убрзања)

Овај пример је сличан претходном примеру, с тим што овог пута имамо додатне информације о убрзању авиона на основу команди пилота. Вектор стања $\mathbf{x}_k$ укључује позицију (a_k, b_k, c_k) и брзину $(\dot{a_k}, \dot{b_k}, \dot{c_k})$ авиона у Декартовим координатанама:

$$\mathbf{x}_k = \begin{bmatrix} a_k \\ b_k \\ c_k \\ \dot{a_k} \\ \dot{b_k} \\ \dot{c_k} \end{bmatrix}.$$

Контролни вектор $\mathbf{u}_k$ садржи информације о убрзању авиона у Декартовим координатанама:

$$\mathbf{u}_k = \begin{bmatrix} \ddot{a_k} \\ \ddot{b_k} \\ \ddot{c_k} \end{bmatrix}.$$

Систем једначина, на основу закона о кретању тела, је исти као у претходном примеру.

Једначина стања модела је тада дата са:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k, \qquad k = 0, 1, 2...,$$

где су матрица преласка $\mathbf{F}_k$ и контролна матрица $\mathbf{B}_k$ дате са

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_k = \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 \\ 0 & 0 & \frac{1}{2}T^2 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix},$$

а $\mathbf{w}_k$ је процесни шум.

2.3 Предикциони корак Калмановог филтера

Посматрајмо математички модел из одељка 2.2 о концепцији простора стања. Због једноставности, с обзиром да је вектор улаза $\mathbf{u}_k$ контролисан, претпоставимо да је $\mathbf{u}_k = 0$. У том случају, једначина стања и једначина мерења имају следећи облик:

$$\mathbf{x}_k = \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}, \quad k = 1, 2... \tag{2.5}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \quad k = 0, 1, 2... \tag{2.6}$$

Калманов филтер има за циљ да оцени тренутно стање система и до оцене долази на основу претходно оцењеног стања система, новог мерења и знања о систему који се моделира. Након извршеног мерења Калманов филтер тренутно стање оцењује на основу два корака:

1. (Предикциони корак) Користећи једначину стања, на основу претходног стања система (у $(k-1)$-ом временском кораку), предвиђамо тренутно стање система (у k-том временском кораку).

2. (Корекциони корак) На основу вредности мерења у k-том временском кораку ($\mathbf{z}_k$) предвиђено стање система у k-том временском кораку се коригује.

Означимо са $\mathbf{Z}_k$ сва мерења закључно са k-тим временским кораком ($\mathbf{z}_{1:k}$). Са Бајесове тачке гледишта предикциони корак се може посматрати као $P(\mathbf{x}_k|\mathbf{Z}_{k-1})$. Односно, у предикционом кораку на основу претходних уверења, тј. $P(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$, предвиђамо стање система у k-том временском кораку, тј. $P(\mathbf{x}_k|\mathbf{Z}_{k-1})$.

Даље извођење предикционог корака Калмановог филтера омогућава нам једначина Чепмен-Колмогорова (енгл. Chapman–Kolmogorov). Односно, на основу зависности стања система у k-том временском кораку само од стања система у $(k-1)$-ом временском кораку и независности стања система у k-том временском кораку од $\mathbf{Z}_{k-1}$ када је дато стања система у $(k-1)$-ом временском кораку, имамо следеће:

$$\begin{aligned} P(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \frac{P(\mathbf{x}_k, \mathbf{Z}_{k-1})}{P(\mathbf{Z}_{k-1})} &= \frac{\int p(\mathbf{x}_k, \mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}}{P(\mathbf{Z}_{k-1})} \\ &= \frac{\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) P(\mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}}{P(\mathbf{Z}_{k-1})} \\ &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) d\mathbf{x}_{k-1} \\ &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}. \quad \forall k \geq 1 \end{aligned}$$

Претходни израз представља управо једначину Чепмен-Колмогорова.

Како смо претпоставили да процес почиње нормалном случајном величином $\mathbf{x}_0$, на основу особина нормалне расподеле (одељак 1.2 о нормалној (Гаусовој) расподели) индукцијом се може показати да важи:

$$\mathbf{x}_k|\mathbf{x}_{k-1} \sim \mathcal{N}(\mathbf{\Phi}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}),$$

$$\mathbf{x}_{k-1}|\mathbf{Z}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}),$$

где са $\mathbf{P}_{k-1|k-1}$ означавамо апостериорну коваријансу грешке оцене за $k-1$.

Напомена: Детаљнији поступак претходног извођења може се пронаћи у [4]
Даље имамо:

$$P(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int f(\mathbf{x}_k; \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}) f(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}) d\mathbf{x}_{k-1},$$

где је f густина случајог вектора са нормалном расподелом тј.

$$f(x;\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left((x-\boldsymbol{\mu})^{\intercal}\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)\right\}.$$

па важи

$$\mathbf{x}_k|\mathbf{Z}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}), \tag{2.7}$$

где је

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \boldsymbol{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \\ \\ \mathbf{P}_{k|k-1} = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\boldsymbol{\Phi}_{k-1}{}^{\intercal} + \mathbf{Q}_{k-1} \end{cases} \tag{2.8}$$

Претходне једначине 2.8 представљају први, односно предикциони, корак Калмановог филтера.

2.4 Корекциони корак Калмановог филтера

Као што се, са Бајесове тачке гледишта, предикциони корак посматра као $P(\mathbf{x}_k|\mathbf{Z}_{k-1})$, корекциони корак се може посматрати као $P(\mathbf{x}_k|\mathbf{Z}_k)$. Односно, корекциони корак користи ново мерење $\mathbf{z}_k$, у циљу израчунавања апостериорне вероватноће $P(\mathbf{x}_k|\mathbf{Z}_k)$. На основу Бајесове формуле, имамо следеће:

$$P(\mathbf{x}_k|\mathbf{Z}_k) = P(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) = \frac{P(\mathbf{x}_k, \mathbf{z}_k|\mathbf{Z}_{k-1})}{P(\mathbf{z}_k|\mathbf{Z}_{k-1})}$$

$$= \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1})P(\mathbf{x}_k|\mathbf{Z}_{k-1})}{P(\mathbf{z}_k|\mathbf{Z}_{k-1})}.$$

Како је ново мерење $\mathbf{z}_k$, на основу претпоставки датих у одељку 2.2 о концепцији простора стања, независно од претходних мерења, имамо следеће:

$$P(\mathbf{x}_k|\mathbf{Z}_k) = \frac{P(\mathbf{z}_k|\mathbf{x}_k)P(\mathbf{x}_k|\mathbf{Z}_{k-1})}{P(\mathbf{z}_k|\mathbf{Z}_{k-1})}. \tag{2.9}$$

Даље важи:

$$\begin{aligned} P(\mathbf{z}_k|\mathbf{Z}_{k-1}) &= \int p(\mathbf{z}_k, \mathbf{x}_k|\mathbf{Z}_{k-1})d\mathbf{x}_k \\ &= \int p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1})p(\mathbf{x}_k|\mathbf{Z}_{k-1})d\mathbf{x}_k \\ &= \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_{k-1})d\mathbf{x}_k. \end{aligned} \tag{2.10}$$

Када се једнакост 2.10 уврсти у једнакост 2.9, добија се следеће:

$$P(\mathbf{x}_k|\mathbf{Z}_k) = \frac{P(\mathbf{z}_k|\mathbf{x}_k)P(\mathbf{x}_k|\mathbf{Z}_{k-1})}{\int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_{k-1})d\mathbf{x}_k}. \tag{2.11}$$

На основу једначине мерења и претпоставке о нормалној расподели шума мерења важи:

$$\mathbf{z}_k|\mathbf{x}_k \sim \mathcal{N}(\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k). \tag{2.12}$$

Користећи резултате 2.7 и 2.12 имамо:

$$P(\mathbf{x}_k|\mathbf{Z}_k) = \frac{\mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})}{\int \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})d\mathbf{x}_k} \tag{2.13}$$

Посматрајмо прво бројилац.

$\mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{N}(\mathbf{x}_k; (\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$

$$= \frac{1}{\sqrt{det(2\pi\mathbf{R}_k)}} e^{-\frac{1}{2}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)^\intercal \mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)} \frac{1}{\sqrt{det(2\pi\mathbf{P}_{k|k-1})}} e^{-\frac{1}{2}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^\intercal \mathbf{P}_{k|k-1}^{-1}(\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1})}$$

$$= \frac{1}{2\pi\sqrt{det(\mathbf{R}_k)det(\mathbf{P}_{k|k-1})}} e^{-\frac{1}{2}[(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)^\intercal \mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k) + (\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^\intercal \mathbf{P}_{k|k-1}^{-1}(\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1}]}$$

$$= \frac{1}{\sqrt{det(2\pi(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal))}} e^{-\frac{1}{2}(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})^\intercal (\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})}$$

$$\times \frac{1}{\sqrt{det(2\pi(\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)^{-1})}} e^{-\frac{1}{2}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^\intercal (\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})}$$

Одатле имамо,

$$\mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$$

$$= \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}, \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, (\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)^{-1}). \tag{2.14}$$

Даље, посматрајмо именилац.

$$\int \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})d\mathbf{x}_k$$

$$= \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}, \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal) \underbrace{\int \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, (\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)^{-1})d\mathbf{x}_k}_{1}$$

$$= \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}, \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal) \tag{2.15}$$

Напомена: За детаљнији поступак извођења 2.14 и 2.15 погледати [21]

Даље, на основу претходног, за једнакост 2.13 важи

$$P(\mathbf{x}_k|\mathbf{Z}_k) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}), \tag{2.16}$$

где је

$$\mathbf{P}_{k|k} = (\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)^{-1}.$$

За извођење наредних корака уводимо следећу теорема.

Теорема 2.4.1 (Вудбуријева формула)
Нека су A, T и $A + BTC$ регуларне матрице, тако да $A \in \mathbb{M}_m$, $B \in \mathbb{M}_{m,n}$, $C \in \mathbb{M}_{n,m}$ и $T \in \mathbb{M}_n$. Тада важи

$$(A + BTC)^{-1} = A^{-1} - A^{-1}B(T^{-1} + CA^{-1}B)CA^{-1}.$$

Одатле имамо:

$$\mathbf{P}_{k|k} = (\mathbf{P}_{k|k-1}^{-1} + \mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{H}_k)^{-1}$$

$$= \mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)^{-1}\mathbf{H}_k\mathbf{P}_{k|k-1}$$

$$= (I - \mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)^{-1}\mathbf{H}_k)\mathbf{P}_{k|k-1}$$

$$= (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1},$$

где је

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)^{-1}$$

и назива се Калмановим појачањем (енгл. Kalman Gain).

Даље, на основу претходних корака, можемо одредити $\hat{\mathbf{x}}_{k|k}$, односно

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k} &= (I - \mathbf{K}_k\mathbf{H}_k)\hat{\mathbf{x}}_{k|k-1} + (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{z}_k \\
&= \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{z}_k - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1}\mathbf{z}_k \\
&= \hat{\mathbf{x}}_{k|k-1} + (\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal)^{-1}(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal) \\
&\quad - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1})\mathbf{z}_k - \mathbf{K}_k\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \\
&= \hat{\mathbf{x}}_{k|k-1} + (\mathbf{K}_k(I + \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1}) - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1})\mathbf{z}_k - \mathbf{K}_k\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \\
&= \hat{\mathbf{x}}_{k|k-1} + (\mathbf{K}_k + \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\intercal\mathbf{R}_k^{-1})\mathbf{z}_k - \mathbf{K}_k\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \\
&= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})
\end{aligned}$$

Овим смо извели и једначине које представљају други, односно корекциони, корак Калмановог филтера.

Дакле, да закључимо, Калманов филтер је одређен следећим једначима:

1. *Предикциони корак*

$$\begin{cases}
\hat{\mathbf{x}}_{k|k-1} = \mathbf{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \\
\mathbf{P}_{k|k-1} = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}{\mathbf{\Phi}_{k-1}}^\intercal + \mathbf{Q}_{k-1}
\end{cases}$$

2. *Корекциони корак*

$$\begin{cases}
\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}) \\
\mathbf{P}_{k|k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}
\end{cases}$$

Уколико немамо доступне информације о почетном стању система, могуће је изабрати $\hat{\mathbf{x}}_0 = \mathbf{0}$ и $\mathbf{P}_0 = p_0\mathbf{I}$, али тако да за p_0 изаберемо велику вредност, јер је наша процена почетне вредности, у том случају, врло непрецизна.

2.5 Калманов филтер кроз пример

У претходна два одељка приказани су и изведени предикциони и корекциони корак Калмановог филтера. У циљу илустрације и бољег разумевања претходних корака, у наставку је дат кратак и једноставан пример употребе Калмановог филтера.

Претпоставимо да имамо мерни инструмент помоћу кога меримо температуру воде у акваријуму на сваких 5 секунди. Од произвођача нам је позната прецизност мерног инструмента, односно он оцењује температуру са дисперзијом од око 0.01 °C. Да бисмо могли да применимо Калманов филтер на претходни проблем, потребно га је записати у погодној форми, односно дефинисати једначину стања и једначину мерења. За дефинисање једначине стања узећемо да се темепратура воде у акваријуму не мења с временом, па је једначина стања дата

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_k, \tag{2.17}$$

где је $\mathbf{x}_k$ једнодимензиони вектор стања. Даље, претпоставимо да су наша веровања у константност температуре у акваријуму оправдана, па је самим тим и поузданост у истинитост претходно дефинисаног модел велика. У том случају бирамо веома малу дисперзију процесног шума, односно нека $\mathbf{w}_k$ узима вредност 0.00001 ($\mathbf{Q}_k = 0.00001 \ \forall k$).

Имајући у виду прецизност мерног инструмента, једначина мерења се може записати на следећи начин:

$$\mathbf{z}_k = \mathbf{x}_k + \mathbf{v}_k, \tag{2.18}$$

где је $\mathbf{z}_k$ једнодимезиони вектор мерења, а дисперзија шума мерења $\mathbf{v}_k$ има вредност 0.01 ($\mathbf{R}_k = 0.01 \ \forall k$).

Даље, с обзиром да немамо увид у почетно стање, тј. у почетну вредност температуре акваријума, претпоставимо да је она 18 °C ($\widetilde{x}_0 = 18$). Ипак, наша процена почетне вредности температура акваријума је врло непрецизна, па бирамо да је $\mathbf{P}_0 = 100$.

Сада када смо претходни модел записали у погодној форми и одредили све потребне вредности, можемо применити Калманов филтер. Једначине Камановог филтера ће имати следећи облик:

1. *Предикциони корак*

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} \\ \mathbf{P}_{k|k-1} = \mathbf{P}_{k-1|k-1} + 0.00001 \end{cases}$$

2. *Корекциони корак*

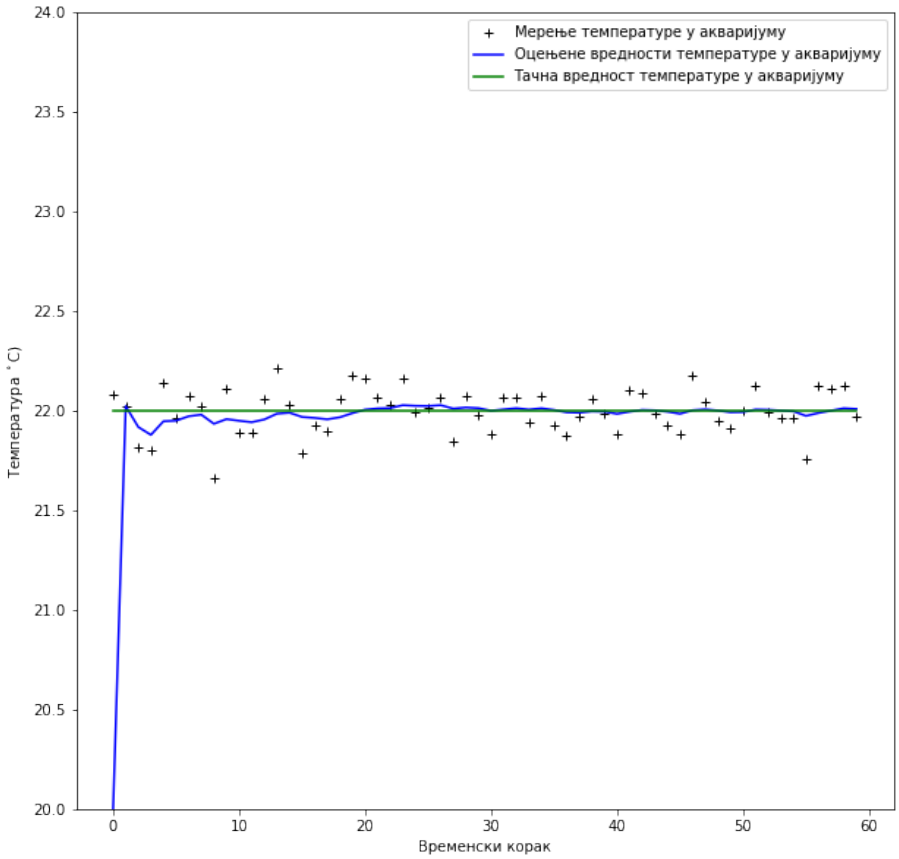
$$\begin{cases} \mathbf{K}_k = \mathbf{P}_{k|k-1}(0.01 + \mathbf{P}_{k|k-1})^{-1} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_{k|k} = (I - \mathbf{K}_k)\mathbf{P}_{k|k-1} \end{cases}$$

За потребе овог примера, симулирали смо мерења температуре у акваријуму у програмском језику Python и након тога користећи израчунате параметре имплементирали Калманов филтер за наведни проблем. Резултати су дати у наставку.

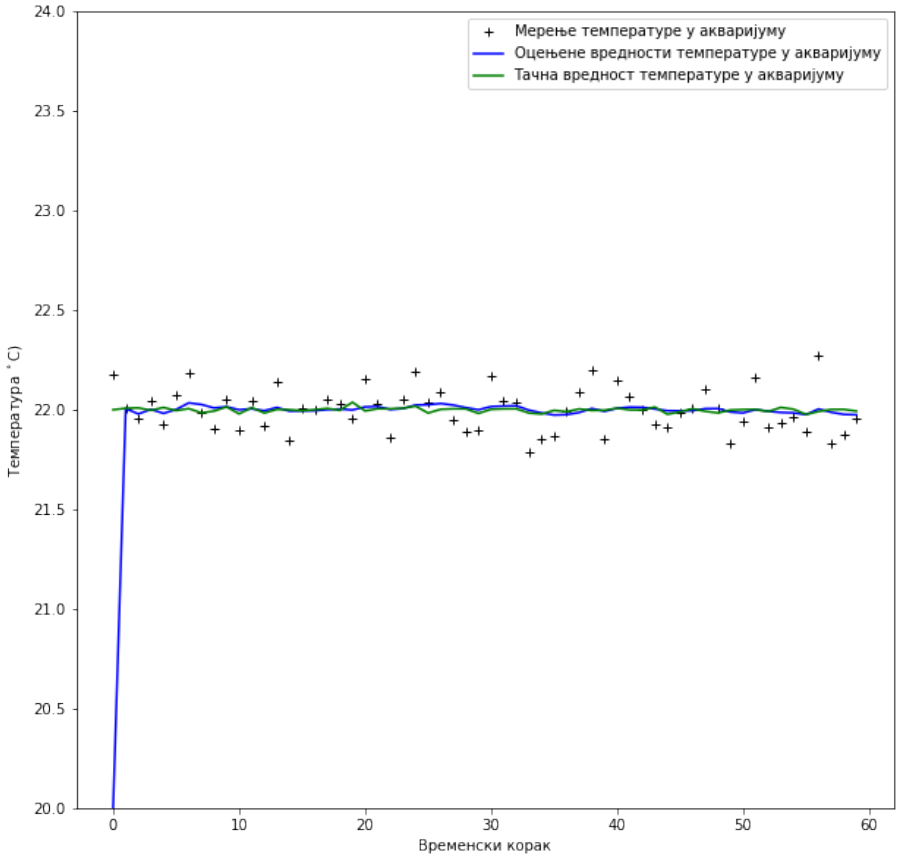
Приметимо да у случају када је стварна вредност температуре у акваријуму константна (слика 2.3) Калманов филтер даје добре резултате за оцену температуре на основу зашумљених мерења. Међутим, уколико температура воде у акваријуму расте за нпр. 0.1 °C на сваких 5 секунди (слика 2.5) Калманов филтер не даје добре резултате за оцену температуре у акваријуму на основу зашумљених мерења. Разлог томе је наша претпоставка о константности у једначини стања (2.17) и велика поузданост у истинитост таквог модел, тј. веома мала дисперзија процесног шума ($\mathbf{Q}_k = 0.00001 \ \forall k$). Претходни проблем се може решити на два начина:

1. Уколико знамо да се температура воде у акваријуму може мењати линеарно, могуће је дефинисати нови модел који ће у једначини стања узети у обзир могућност линеарне промене температуре. (нпр. слика 2.6)

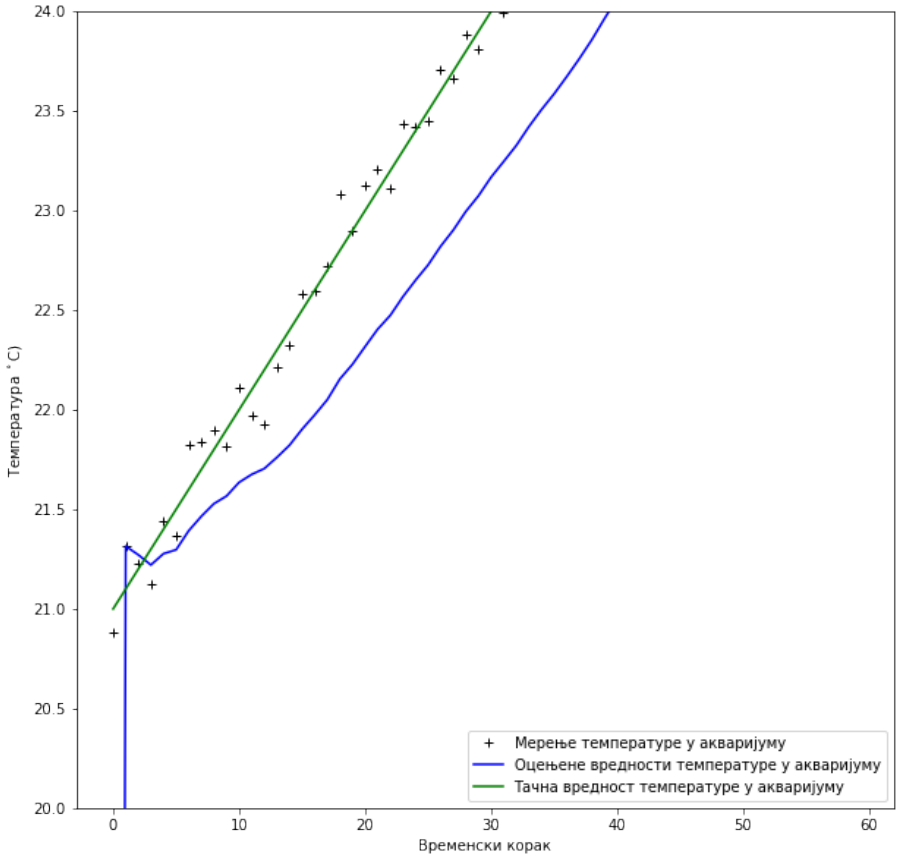
2. Претходни приступ се најчешћи користи за решење наведног проблема. Ипак, уколико се промене температуре не могу моделирати, перформансе Калмановог филтера се на тај начин неће унапредити. У том случају потребно је смањити поузданост у истинитост модела, тј. повећати варијансу процесног шума. (нпр. слика 2.7)



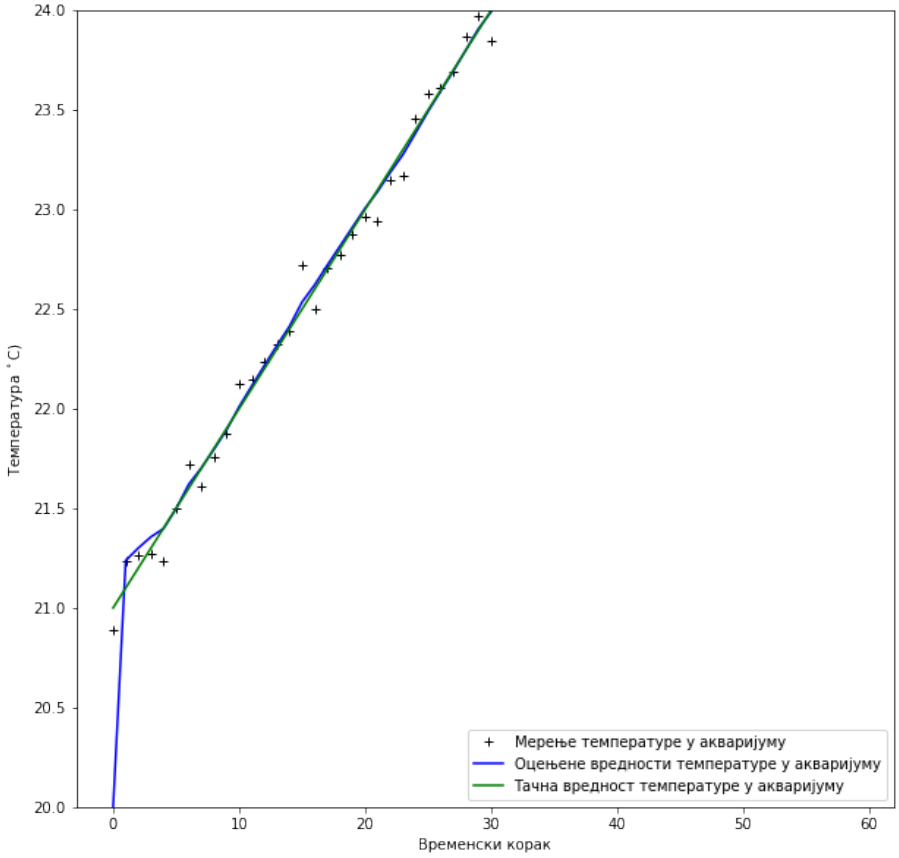
Слика 2.3: График температуре у случају када је тачна вредност температуре константна.



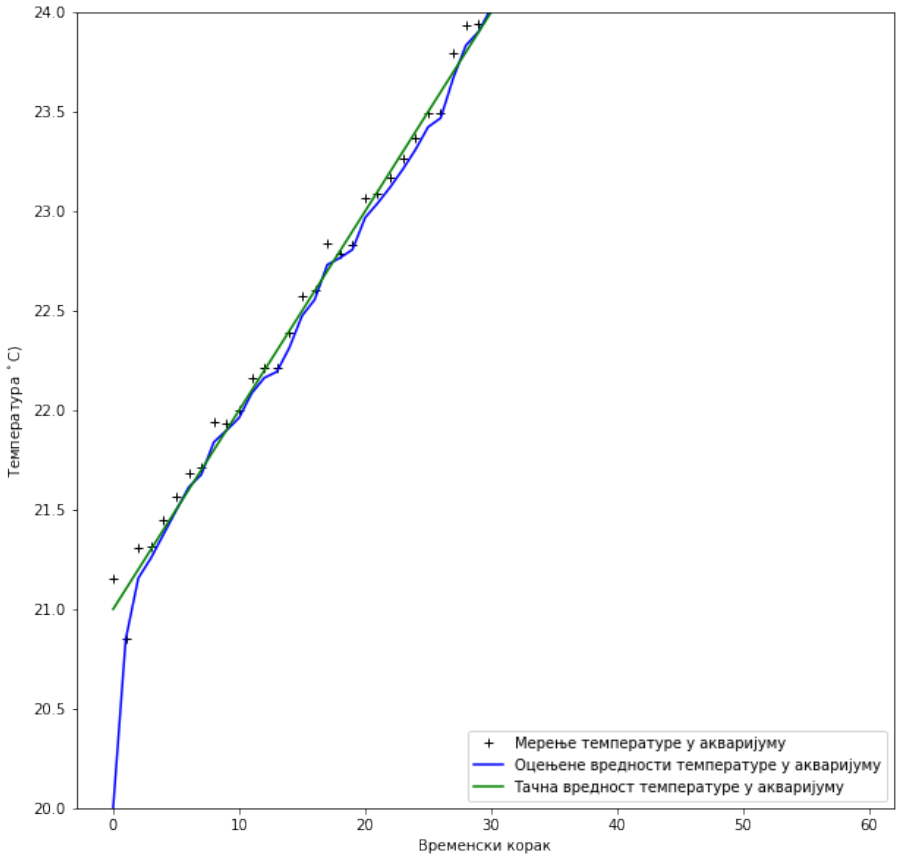
Слика 2.4: График температуре у случају када тачна вредност температуре варира.



Слика 2.5: График температуре у случају када се се тачна вредност температуре мења линеарно.



Слика 2.6: График температуре у случају када се тачна вредност температуре мења линеарно и једначина стања узима у обзир могућност линеарне промене температуре.



Слика 2.7: График температуре у случају када се тачна вредност температуре мења линеарно. Једначина стања не узима у обзир могућност линеарне промене температуре, али је поузданост у истинитост модела смањена тј. $\mathbf{Q}_k = 0.001$

Глава 3

Модификације Калмановог филтера

Традиционални Калманов филтер је ограничен на системе са линеарном динамиком, у којима се једначина стања и једначина мерења могу записати у облику једначина 2.1 и 2.2 из претходне главе. Како су реални системи углавном нелинеарни, а традиционални Калманов филтер у таквим случајевима често не даје добре резултате у оцени стања система, јавила се потреба за модификованим верзијама Калмановог филтера које би успешно решиле проблем оцене стања код нелинеарних система. У наставку су описане неке од модификација Калмановог филтера које су значајно проширile област његове примене, а то су: проширени Калманов филтер и Калманов филтер без мириса.

3.1 Нелинеарни простор стања

Нелинеарни систем у дискретном времену се може описати једначином стања и једначином мерења које имају следећи облик:

1. *Једначина стања*

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, k) + \mathbf{w}_k \quad k = 1, 2... \tag{3.1}$$

2. *Једначина мерења*

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, k) + \mathbf{v}_k \quad k = 1, 2... \tag{3.2}$$

где су $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{Z}^+ \longrightarrow \mathbb{R}^n$ и $\mathbf{h} : \mathbb{R}^n \times \mathbb{Z}^+ \longrightarrow \mathbb{R}^p$ нелинеарне функције које су непрекидно диференцијалбине на посматраном домену. Вектори $\mathbf{x}_k$, $\mathbf{z}_k$ и $\mathbf{u}_k$ су редом n -димензионални вектор стања, p -димензионални вектор мерења и r -димензионални контролни вектор, а вектори $\mathbf{w}_k$ и $\mathbf{v}_k$ су редом процесни шум и шум мерења за које важе исте претпоставке као и у одељку 2.2 о концепцији простора стања. Такође, и за почетно стање $\mathbf{x}_0$ важе исте претпоставке као и у одељку о концепцији простора стања. Претходне једначине ће у наредним одељцима бити поједностављене, ради лакшег објашњења, али је поступак исти и у свим другим случајевима када нпр. имамо контролни вектор и слично.

3.2 Проширени Калманов филтер

Проширени Калманов филтер (енгл. extended Kalman filter) је најкоришћенија модификација Калмановог филтера за проблеме оцене стања код нелинеарних система. Идеја проширеног Калмановог филтера је заснована на концепту линеаризације система пре примене Калмановог филтера, тј. тачније линеаризације једначине стања и једначине мерења око његове последње оцене користећи Тејлоров развој.
Дакле, претпоставимо да у $(k-1)$-ом временском тренутку имамо оцену стања $\hat{\mathbf{x}}_{k-1|k-1}$ са коваријационом матрицом $\mathbf{P}_{k-1|k-1}$. Тада, у предикционом кораку Калмановог филтера, имамо следеће:

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= E(\mathbf{x}_k|\mathbf{Z}_{k-1}) \\ &= E(\mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1}|\mathbf{Z}_{k-1}) \\ &= E(\mathbf{f}(\mathbf{x}_{k-1})|\mathbf{Z}_{k-1})\end{aligned}$$

Развојем нелинеарне векторке функције $\mathbf{f}$ у Тејлоров полином првог реда око последње расположиве оцене (тј. $\hat{\mathbf{x}}_{k-1|k-1}$) добијамо следеће:

$$\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1},$$

где је $J_f(\hat{\mathbf{x}}_{k-1|k-1})$ Јакобијева матрица дефинисана у одељку 1.3, а

$$\mathbf{e}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}.$$

Даље, на основу претходног, имамо:

$$E(\mathbf{f}(\mathbf{x}_{k-1})|\mathbf{Z}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + J_f(\hat{\mathbf{x}}_{k-1|k-1})E(\mathbf{e}_{k-1}|\mathbf{Z}_{k-1})$$

Како је $E(\mathbf{e}_{k-1}|\mathbf{Z}_{k-1}) = 0$, прва једначина предикционог корака проширеног Калмановог филтера гласи:

$$\hat{\mathbf{x}}_{k|k-1} \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \tag{3.3}$$

Такође, приметимо следеће:

$$\begin{aligned}\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} - \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \\ &\approx J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1} + \mathbf{w}_{k-1}.\end{aligned}$$

Даље, на основу претходног, добијамо другу једначину предикционог корака проширеног Калмановог филтера, односно априорну коваријансу грешке оцене:

$$\begin{aligned}\mathbf{P}_{k|k-1} &= E((\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^\intercal) \\ &= J_f(\hat{\mathbf{x}}_{k-1|k-1})E((\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^\intercal)J_f(\hat{\mathbf{x}}_{k-1|k-1})^\intercal + E(\mathbf{w}_{k-1}\mathbf{w}_{k-1}^\intercal) \\ &= J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}J_f(\hat{\mathbf{x}}_{k-1|k-1})^\intercal + \mathbf{Q}_{k-1}.\end{aligned}$$

Сада када имамо $\hat{\mathbf{x}}_{k|k-1}$ и $\mathbf{P}_{k|k-1}$, потребно је да на основу новог мерења у k-том тренутку пронађемо најбољу непристрасну оцену за $\mathbf{x}_k$, односно $\hat{\mathbf{x}}_{k|k}$ у смислу најмањих квадрата. Један од могућих начина је да претпоставимо да је оцена $\hat{\mathbf{x}}_{k|k}$ линеарна комбинација $\hat{\mathbf{x}}_{k|k-1}$ и новог мерења у k-том тренутку ($\mathbf{z}_k$). Детаљније објашњење се може пронаћи у [15].

Односно, имамо следеће:

$$\hat{\mathbf{x}}_{k|k} = \mathbf{c} + \mathbf{K}_k\mathbf{z}_k,$$

где се c може одредити из услова непристрасности оцене:

$$\begin{aligned}0 &= E(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}|\mathbf{Z}_k) \\ &= E(\hat{\mathbf{x}}_{k|k-1} + \mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} - (\mathbf{c} + \mathbf{K}_k\mathbf{h}(\mathbf{x}_k) + \mathbf{K}_k\mathbf{v}_k|\mathbf{Z}_k) \\ &= \hat{\mathbf{x}}_{k|k-1} - (\mathbf{c} + \mathbf{K}_k(\mathbf{h}(\mathbf{x}_k))|\mathbf{Z}_k) \\ &\implies \mathbf{c} = \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{h}(\mathbf{x}_k)|\mathbf{Z}_k)\end{aligned}$$

Дакле,

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - (h(\mathbf{h}_k)|\mathbf{Z}_k))$$

Као и у предикционом кораку, развојем нелинеарне векторке функције $\mathbf{h}$ у Тејлоров полином првог реда око последње расположиве оцене (у овом случају је то $\hat{\mathbf{x}}_{k|k-1}$) добијамо следеће:

$$\mathbf{h}(\mathbf{x}_k) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + J_h(\hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}),$$

где је $J_h(\hat{\mathbf{x}}_{k|k-1})$ Јакобијева матрица.

Даље, на основу претходног, имамо:

$$E(h(\mathbf{h}_k)|\mathbf{Z}_k) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + J_h(\hat{\mathbf{x}}_{k|k-1})E(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}|\mathbf{Z}_k)$$

Како је $E(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}|\mathbf{Z}_k) = 0$ (на основу претходних корака), добијамо једну од једначина корекционог корака проширеног Калмановог филтера, односно:

$$\hat{\mathbf{x}}_{k|k} \approx \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}))$$

Такође, важи:

$$\begin{aligned}
\mathbf{x}_k - \hat{\mathbf{x}}_{k|k} &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} - \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})) \\
&\approx \mathbf{f}(\mathbf{x}_{k-1}) - f(\hat{\mathbf{x}}_{k-1|k-1}) + \mathbf{w}_{k-1} - \mathbf{K}_k(\mathbf{h}(\mathbf{x}_k) - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{v}_k) \\
&\approx J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1} + \mathbf{w}_{k-1} - \mathbf{K}_k(J_h(\hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) + \mathbf{v}_k) \\
&\approx J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1} + \mathbf{w}_{k-1} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1})(J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1} + \mathbf{w}_{k-1}) - \mathbf{K}_k\mathbf{v}_k \\
&\approx (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{e}_{k-1} + (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{w}_{k-1} - \mathbf{K}_k\mathbf{v}_k
\end{aligned}$$

Даље, на основу претходног, имамо;

$$\begin{aligned}
\mathbf{P}_{k|k} &= E((\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^\intercal) \\
&= (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}(J_f(\hat{\mathbf{x}}_{k-1|k-1}))^\intercal(\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal \\
&+ (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{Q}_{k-1}(\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\intercal \\
&= \mathbf{P}_{k|k-1} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1}J_h(\hat{\mathbf{x}}_{k|k-1})^\intercal\mathbf{K}_k^\intercal \\
&+ \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal\mathbf{K}_k^\intercal + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\intercal
\end{aligned}$$

Како претходно важи за свако $\mathbf{K}_k$, Калманово појачање добијамо минимизацијом $Tr(\mathbf{P}_{k|k})$ по $\mathbf{K}_k$, дакле:

$$\begin{aligned}
0 &= \frac{\partial Tr(\mathbf{P}_{k|k})}{\partial \mathbf{K}_k} \\
&= -(J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1})^\intercal - \mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal + 2\mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal + 2\mathbf{K}_k\mathbf{R}_k \\
&\implies \mathbf{K}_k = \mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal(J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal + \mathbf{R}_k)^{-1}
\end{aligned}$$

Даље, на основу претходног, добијамо и последњу једначину корекционог корака проширеног Калмановог филтера, односно апостериорну коваријансу грешке оцене:

$$\begin{aligned}
\mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1} - (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal\mathbf{K}_k^\intercal + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\intercal \\
&= (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1} \\
&\quad - (\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal - \mathbf{K}_k\mathbf{R}_k)\mathbf{K}_k^\intercal \\
&= (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1} - (\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal - \mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal)\mathbf{K}_k^\intercal \\
&= (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1}
\end{aligned}$$

Дакле, да закључимо, проширени Калманов филтер је одређен следећим једначинама:

1. *Предикциони корак*

$$\begin{cases}
\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \\
\mathbf{P}_{k|k-1} = J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}J_f(\hat{\mathbf{x}}_{k-1|k-1})^\intercal + \mathbf{Q}_{k-1}
\end{cases}$$

2. *Корекциони корак*

$$\begin{cases}
\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})) \\
\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k J_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1}
\end{cases}$$

Како су $\mathbf{Q}_k$ и $\mathbf{R}_k$ симетричне позитивно дефинитне матрице, имамо следеће:

$$\mathbf{Q}_k = \mathbf{M}_k\mathbf{M}_k{}^\intercal$$

$$\mathbf{R}_k = \mathbf{N}_k \mathbf{N}_k{}^\intercal$$

Такође,

$$\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\hat{\mathbf{x}}_{k|k}) = J_f(\hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}) + \phi(\mathbf{x}_k, \hat{\mathbf{x}}_{k|k})$$
$$\mathbf{h}(\mathbf{x}_k) - \mathbf{h}(\hat{\mathbf{x}}_{k|k}) = J_h(\hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}) + \gamma(\mathbf{x}_k, \hat{\mathbf{x}}_{k|k})$$

где су са ϕ и γ означени чланови вишег реда.

Теорема 3.1 у [14] показује да је грешка оцене $\mathbf{e}_k$ (тј. $\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$) ограничена ако важи следеће:

- Постоје позитивни реални бројеви a, b, c, d, r, q тако да за свако $k \geqslant 0$ важи:

$$||J_f(\hat{\mathbf{x}}_{k|k})|| \leqslant a$$
$$||J_h(\hat{\mathbf{x}}_{k|k})|| \leqslant b$$
$$c\mathbf{I} \leqslant \mathbf{P}_{k|k} \leqslant d\mathbf{I}$$
$$q\mathbf{I} \leqslant \mathbf{Q}_k$$
$$r\mathbf{I} \leqslant \mathbf{R}_k$$

- J_f је несингуларна за свако $k \geqslant 0$
- Постоје позитивни реални бројеви $\zeta_\phi, \zeta_\gamma, \psi_\phi, \psi_\gamma$ тако да су нелинеарне функције ϕ и γ ограничене са:

  $\phi(\mathbf{x}_k, \hat{\mathbf{x}}_{k|k}) \leqslant \zeta_\phi ||\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}||^2$ за $||\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}|| \leqslant \psi_\phi$

  $\zeta_\gamma(\mathbf{x}_k, \hat{\mathbf{x}}_{k|k}) \leqslant \zeta_\gamma ||\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}||^2$ за $||\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}|| \leqslant \psi_\phi$

Дакле, уколико важи претходно, грешка оцене $\mathbf{e}_k$ је ограничена под условом да је

$$||\mathbf{e}_0|| \leqslant \xi, \qquad \mathbf{M}_k \mathbf{M}_k{}^\intercal \leqslant \delta\mathbf{I}, \qquad \mathbf{N}_k \mathbf{N}_k{}^\intercal \leqslant \delta\mathbf{I}$$

за неко $\delta, \xi > 0$

Приликом извођења претходних једначина за предикциони и корекциони корак, видели смо да се проширени Калманов филтер заснива на концепту линеаризације једначине стања и једначине мерења око последње расположиве оцене користећи Тејлоров развој првог реда. Ипак, није лоше поменути да је коришћење Тејлоровог развоја другог реда, у случајевима високе нелинеарности, често бољи избор и даје боље резултате. Овај тип модификација Калмановог филтера се назива проширени Калманов филтер другог реда (енгл. Second order extended Kalman filter - SOEKF) и одређен је следећим једначинама:

1. *Предикциони корак*

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + \frac{1}{2}\sum_i \mathbf{e}_i Tr(H_f^{(i)}(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}) \\ \mathbf{P}_{k|k-1} = J_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}J_f(\hat{\mathbf{x}}_{k-1|k-1})^\intercal \\ \qquad +\frac{1}{2}\sum_i\sum_j \mathbf{e}_i\mathbf{e}_i{}^\intercal Tr(H_f^{(i)}(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}H_f^{(j)}(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}) \\ \qquad +\mathbf{Q}_{k-1} \end{cases}$$

2. *Корекциони корак*

$$\begin{cases} \mathbf{S}_k = J_h(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal \\ \quad +\frac{1}{2}\sum_i\sum_j \mathbf{e}_i\mathbf{e}_i{}^\intercal Tr((H_h^{(i)}(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}H_h^{(j)}(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}) + \mathbf{R}_k \\ \mathbf{K}_k = \mathbf{P}_{k|k-1}(J_h(\hat{\mathbf{x}}_{k|k-1}))^\intercal {\mathbf{S}_k}^{-1} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) - \frac{1}{2}\sum_i \mathbf{e}_i Tr(H_h^{(i)}(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1})) \\ \mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\intercal \end{cases}$$

где су, као и у претходном случају за проширени Калманов филтер J_h и J_f Јакобијеве матрице, док су $H_f^{(i)}$ и $H_h^{(i)}$ Хесијан матрице функција $\mathbf{f}$ и $\mathbf{h}$, односно

$$H_f^{(i)}(\hat{\mathbf{x}}) = \frac{\partial^2 \mathbf{f}_i}{\partial \mathbf{x}^2}\Big|_{\mathbf{x}=\hat{\mathbf{x}}}$$

$$H_h^{(i)}(\hat{\mathbf{x}}) = \frac{\partial^2 \mathbf{h}_i}{\partial \mathbf{x}^2}\Big|_{\mathbf{x}=\hat{\mathbf{x}}}$$

Такође, $\mathbf{e}_i$ је јединични вектор координатне осе i, $\mathbf{S}_k jj$,а $\mathbf{K}_k$ је Калманово појачање.

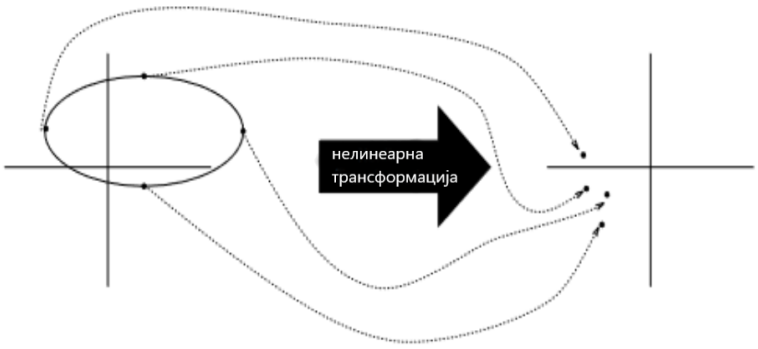
Мана проширеног Калмановог филтера другог реда лежи у његовој рачунарској комплексности, која га чини непогодним за извршавање у реалном времену и примену у вишедимензионалним системима. Ограничење примене проширеног Калмановог филтера, као и проширеног Калмановог филтера другог реда, лежи у томе што линеарна и квадратна апроксимација дају поуздане резултате онда када је пропагација грешке добро апроксимирана линеарном, односно квадратном функцијом. Уколико претходно није задовољено, перформансе ових алгоритама могу бити лоше.

3.3 Калманов филтер без мириса

Калманов филтер без мириса (енгл. Unscented Kalman Filter - UKF) је још једна од модификација Калмановог филтера која се често користи за проблеме оцене стања код нелинеарних система и решава проблем апроксимације који постоји у проширеном Калмановом филтеру и проширеном Калмановом филтеру другог реда. Како би се претходно поменуте проширене верзије Калмановог филтера примениле, потребно је да постоје Јакобијева и Хесијан матрица. Овај услов често није испуњен, а и када јесте, рачунање ових матрица је у многим случајевима комплексно. Калманов филтер без мириса, у односу на метод линеаризације, резултује мањим грешкама и даје боље оцене вектора стања. Ова модификација Калмановог филтера је заснована на UT (енгл. Unscented Transform - UT) трансформацији која представља апроксимативни метод за израчунавање момената расподеле случајног вектора генерисаног пропагацијом кроз нелинеарну функцију. Односно, за разлику од методе линеаризације, UT трансформација не апроксимира нелинеарну функцију, него расподелу случајног вектора детерминисаним избором фиксног броја тзв. сигма тачака у простору тог вектора. Концепт на коме је заснована UT трансформација је приказан на слици 3.1 [2]. Дакле, у овој методи се на детерминисан начин бира фиксан број сигма тачака тако да осликавају прва два момента расподеле случајног вектора $\mathbf{x}$. Затим се сигма тачке пропагирају кроз нелинеарну функцију, у циљу оцене момената вектора $\mathbf{y}$. UT трансформацију као такву је могуће применити на проблем оцене стања код нелинеарних система, и управо таква верзија Калмановог филтера носи назив Калманов филтер без мириса. Једна од главних предности UT трансформације у односу на апроксимацију Тејлоровим полином лежи у бољој оцени момената вишег

[2]Преузето из Julier, Simon J., Jeffrey K. Uhlmann. „*New extension of the Kalman filter to nonlinear systems.*", 1997

рада вектора мерења. Такође, рачунање Јакобијеве и Хесијан матрице у овом случају није потребно, па је и сама процедура у том смислу олакшана.



Слика 3.1: Нелинеарна трансформација у Калмановом филтеру без мириса

Скуп сигма тачака $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, ..., \mathbf{x}^{(n_{ut})}\}$ у UT трансформацији се бира на детерминисан начин тако да осликава прва два момента расподеле случајене величине $\mathbf{x}$, односно на начин да буду задовољени следећи услови:

$$\sum_{i=1}^{n_{ut}} \mathbf{w}^{(i)} \mathbf{x}^{(i)} = \bar{\mathbf{x}}$$

$$\sum_{i=1}^{n_{ut}} \mathbf{w}^{(i)} (\mathbf{x}^{(i)} - \bar{\mathbf{x}})(\mathbf{x}^{(i)} - \bar{\mathbf{x}})^{\mathsf{T}} = \mathbf{P}_{xx}$$

где су скаларне величине $\mathbf{w}^{(i)}$ тежински коефицијенти сваке од сигма тачака који могу бити произвољног знака, али морају да задовољавају услов нормирања тј. $\mathbf{w}^{(1)} + \mathbf{w}^{(2)} + ...\mathbf{w}^{(n_{ut})} = 1$

Даље, свака сигма тачка се пресликава у простор вектора $\mathbf{y}$ путем нелинарне функцију f, односно:

$$\mathbf{y}^{(i)} = f(\mathbf{x}^{(i)}) \quad \forall i = 1, ..., n_{ut}$$

Оцене очекивања и коваријационе матрице случајног вектора $\mathbf{y}$ се тада формирају на основу претходно добијеног скупа пресликаних сигма тачака, тј. имамо:

$$\bar{\mathbf{y}} = \sum_{i=1}^{n_{ut}} \mathbf{w}^{(i)} \mathbf{y}^{(i)}$$

$$\mathbf{P}_{yy} = \sum_{i=1}^{n_{ut}} \mathbf{w}^{(i)} (\mathbf{y}^{(i)} - \bar{\mathbf{y}})(\mathbf{y}^{(i)} - \bar{\mathbf{y}})^\intercal$$

Избор сигма тачака, поред тога што мора да задовољи већ претходно наведене услове, у великој мери утиче на перформансе филтера и рачунарска комплексност је пропорционална броју одабраних сигма тачака. Један од избора сигма тачака [3] које задовољавају потребне услове је симетричан скуп од $n_{ut} = 2n_x + 1$, где је n_x димензија вектора $\mathbf{x}$, сигма тачака које су одређене следећим једначинама:

$$\mathbf{x}^{(0)} = \bar{\mathbf{x}}, \mathbf{w}^{(0)} = \frac{\lambda}{n_x + \lambda} \tag{3.4}$$

$$\mathbf{x}^{(i)} = \bar{\mathbf{x}} + [\sqrt{(n_x + \lambda)\mathbf{P}_{xx}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n_x + \lambda)} \quad \forall i = 1, ..., n_x \tag{3.5}$$

$$\mathbf{x}^{(i)} = \bar{\mathbf{x}} + [\sqrt{(n_x + \lambda)\mathbf{P}_{xx}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n_x + \lambda)} \quad \forall i = n_x + 1, ..., 2n_x \tag{3.6}$$

где је $[\sqrt{(n_x + \lambda)\mathbf{P}_{xx}}]_i$ представља i-ту колону матричног корена коваријационе матрице $\mathbf{P}_{xx}$ скалиране са $(n_x + \lambda)$, а $\lambda \in \mathbb{R}$ је фактор скалирања дефинисан улазним параметрима α, k тј.

$$\lambda = \alpha^2(n_x + k) - n_x$$

У случају претпоставке о нормалној расподели случајног вектора $\mathbf{x}$ чест избор за вредност фактора скалирања λ је $n_x - 3$. За израчунавање матричног корена могуће је користити нумерички ефикасану и стабилану методу Чолески декомпозиције (енгл. Cholesky decomposition).

[3] Преглед различитих варијанти сета сигма тачака и предности избора сваког од њих се могу пронаћи у [8])

Калманов филтер без мириса користи претходно описан облик UT трансформације за оцену стања нелинеарног система чија су једначина стања и једначина мерења дате у одељку 3.1.

Дакле, претпоставимо да у $(k-1)$-ом временском тренутку имамо оцену стања $\hat{\mathbf{x}}_{k-1|k-1}$ са коваријационом матрицом $\mathbf{P}_{k-1|k-1}$. На основу расположивих величина сет сигма тачака $\mathcal{A}$ се формира користећи једначине 3.4, 3.5 и 3.6, односно:

$$x^{(0)}_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{w}^{(0)} = \frac{\lambda}{n+\lambda}$$

$$x^{(i)}_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + [\sqrt{(n+\lambda)\mathbf{P}_{k-1|k-1}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n+\lambda)} \quad \forall i = 1, ..., n$$

$$x^{(i)}_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + [\sqrt{(n+\lambda)\mathbf{P}_{k-1|k-1}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n+\lambda)} \quad \forall i = n+1, ..., 2n$$

где је n димензија вектора стања, $[\sqrt{(n+\lambda)\mathbf{P}_{k|k-1}}]_i$ представља i-ту колону матричног корена коваријационе матрице $\mathbf{P}_{k|k-1}$ скалиране са $(n+\lambda)$, а $\lambda \in \mathbb{R}$ је фактор скалирања

Даље, дате сигма тачке се пропагирају кроз нелинарну функцију једначине стања и дају скуп пресликаних сигма тачака $\mathcal{B}$.

$$x^{(i)}_{k|k-1} = \mathbf{f}(x^{(i)}_{k-1|k-1}) \quad \forall i = 1, ..., 2n$$

У складу са претходно описаном UT трансформацијом, једначине предикционог корака Калмановог филтера без мириса се могу добити на следећи начин:

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)} x^{(i)}_{k|k-1}$$

$$\mathbf{P}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)} (x^{(i)}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1})(x^{(i)}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1})^\intercal + \mathbf{Q}_{k-1}$$

Напомена: За $\mathbf{w}^{(0)}$ приликом рачунања $\mathbf{P}_{k|k-1}$, односно у другој једначини предикционог корака Калмановог филтера без мириса, често се уместо вредности $\frac{k}{n+k}$ узима

вредност $\frac{k}{n+k} + (1 - \alpha^2 + \beta)$. Такође, најчешће узимане вредности за параметере α, β и k које дају добре резултате на различитим пољима примене Калмановог филтера без мириса су редом $10^3, 2$ и 0. Смернице у избору, као и разлози избора и сами резултати примене различитих комбинација вредности за претходне параметре се могу пронаћи у [19].

Даље, пропагацијом сигма тачака скупа $\mathcal{B}$ кроз нелинеарну функцију једначине мерења добијамо нови скуп сигма тачака $\mathcal{C}$.

$$z_{k|k-1}^{(i)} = \mathbf{h}(x_{k|k-1}^{(i)}) \quad \forall i = 1, ..., 2n$$

На основу претходно добијених сигма тачака и самог концепта UT трансформације, једначине предикционог корака Калмановог филтера без мириса се могу добити на следећи начин:

$$\mathbf{S}_k = \sum_{i=1}^{2n} \mathbf{w}^{(i)} (z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})(z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})^\intercal + \mathbf{R}_{k-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\intercal$$

Дакле, да закључимо, Калманов филтер без мириса је одређен следећим једначинама:

1. *Предикциони корак*

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)} x_{k|k-1}^{(i)} \\ \mathbf{P}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)} (x_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})(x_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})^\intercal + \mathbf{Q}_{k-1} \end{cases}$$

2. *Корекциони корак*

$$\begin{cases} \mathbf{S}_k = \sum_{i=1}^{2n} \mathbf{w}^{(i)} (z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})(z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})^\intercal + \mathbf{R}_{k-1} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)}) \\ \mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\intercal \end{cases}$$

Данас постоје и различите верзије Калмановог филтера без мириса које су засноване на различитим облицима UT трансформације, као што је нпр. надограђени облик UT трансформације (енгл. augmented UT) код којег се вектор стања проширује са променљивама процесног и мерног шума. Како остале верзије Калмановог филтера без мириса неће бити покривене у овом раду детаљније инфомације је могуће пронаћи у [4]. На крају, на слици 3.2 дат је кратак преглед једначина предикционог и корекционог корака традиционалног Калманов филтера, као и његових модификација тј. проширеног Калмановог филтера и Калмановог филтер без мириса.

Традиционални Калманов филтер

1. *Предикциони корак*

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1|k-1}$$
$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{\Phi}_{k-1}{}^{\intercal} + \mathbf{Q}_{k-1}$$

2. *Корекциони корак*

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})$$
$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}$$

Модификације Калмановог филтера

Проширени Калманов филтер

1. *Предикциони корак*

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1})$$
$$\mathbf{P}_{k|k-1} = \mathbf{J}_f(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}\mathbf{J}_f(\hat{\mathbf{x}}_{k-1|k-1})^{\intercal} + \mathbf{Q}_{k-1}$$

2. *Корекциони корак*

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}))$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{J}_h(\hat{\mathbf{x}}_{k|k-1}))\mathbf{P}_{k|k-1}$$

Калманов филтер без мириса

1. *Предикциони корак*

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)} x_{k|k-1}^{(i)}$$
$$\mathbf{P}_{k|k-1} = \sum_{i=1}^{2n} \mathbf{w}^{(i)}(x_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})(x_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})^{\intercal} + \mathbf{Q}_{k-1}$$

2. *Корекциони корак*

$$\mathbf{S}_k = \sum_{i=1}^{2n} \mathbf{w}^{(i)}(z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})(z_{k|k-1}^{(i)} - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})^{\intercal} + \mathbf{R}_{k-1}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \sum_{i=1}^{2n} \mathbf{w}^{(i)} z_{k|k-1}^{(i)})$$
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^{\intercal}$$

Слика 3.2: Једначине предикционог и корекционог корака Калмановог филтера и његових модификација

Глава 4

Примена Калмановог филтера у праћењу покретног циља

Процес праћења покретних циљева заснива се на оцени наредних стања система, тј. положаја, брзине и убрзања циљева, као и процесу придруживања података, тзв. асоцијација података. Асоцијација података представља процес придруживања измерених вредности новим или неким од већ постојећих циљева. Неке од познатих метода асоцијације података су метода најближих суседа (*Global Nearest Neighbor -GNN*), класа метода за асоцијацију података по вероватноћи (*Probabilistic data association techniques - PDA*), као методе са формирањем хипотеза (*Multiple Hypothesis Tracking - MHT*). Након асоцијације података, користећи алгоритме за оцену стања система, као што је Калманов филтер, на основу претпостављеног модела стања и измерених вредности добијају се оцене положаја, брзине и убрзања циљева. У наставку је анализирана примена Калмановог филтера и његових модификација у праћењу једног покретног циља на основу радараских сигнала. Праћење више покретних циљева, у односу на појединачни циљ, неће се разликовати по питању алгоритама оцене стања система, већ ће само, с обзиром на могућност да се у опсегу посматраног трага нађу два или више циља, захтевати постојање неког од већ горе поменутих алгоритама асоцијације података. У првом делу су приказани резултати праћења добијени на основу симулираног кретања циља, а касније су приказани резултати праћења добијени на основу стварних радарских детекција које су придружене неком циљу. На крају, анализиране су перформансе сваког од имплементираних алгоритама оцене стања система, тј. назначене предности и мане употребе различитних верзија Калмановог филтера, а то су: традиционални Калманов филтер, проширени Калманов филтер и Калманов филтер без мириса. Имплементација поменутих филтера одрађена је у програмском језику Пајтон (*Python*).

4.1 Моделовање кретања циља и примена Калмановог филтера на први случај

Претпоставимо сценарио кретања брода чија путања има облик броја осам. Односно, имамо следеће једначине:

$$x = 2cos(t) \qquad y = sin(2t) \qquad \forall t \in [0, 2\pi] \tag{4.1}$$

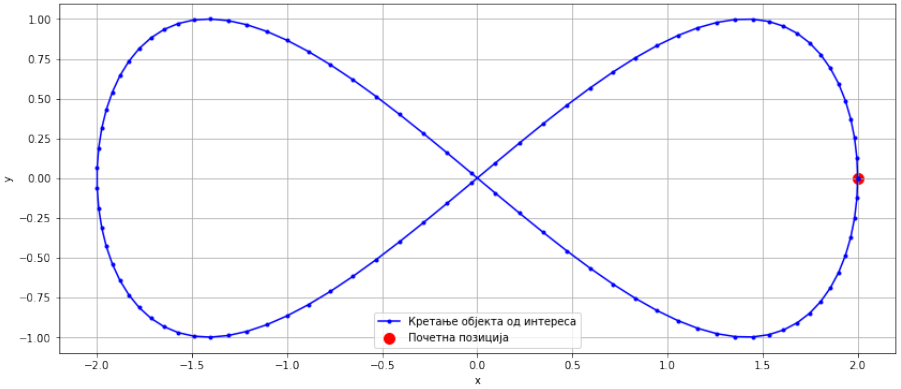
Сходно томе, први, други и трећи извод претходних једначина репрезентују редом брзину, убрзање и трзај брода..

$$v_x = \frac{\partial x}{\partial t} = -2sin(t) \qquad v_y = \frac{\partial y}{\partial t} = 2cos(2t) \tag{4.2}$$

$$a_x = \frac{\partial^2 x}{\partial t^2} = -2cos(t) \qquad a_y = \frac{\partial^2 y}{\partial t^2} = -4sin(2t) \tag{4.3}$$

$$j_x = \frac{\partial^3 x}{\partial t^3} = 2sin(t) \qquad j_y = \frac{\partial^3 y}{\partial t^3} = -8cos(2t) \tag{4.4}$$

На слици 4.1 приказана је путања брода чије је кретање представљено претходним једначинама и његова почетна позиција.



Слика 4.1: Кретање брода чија путања има облик броја осам

Модел кретања брода, односно једначине стања, формирамо на основу закона о кретању тела, слично као и у примерима у 2.2.1 и 2.2.2 у одељку 2.2 о концепцији простора стања.

$$\begin{cases} x_{k+1} = x_k + v_{xk}T + \frac{1}{2}a_{xk}T^2 + +\frac{1}{6}j_{xk}T^3 \\ v_{xk+1} = v_{xk} + a_{xk}T + \frac{1}{2}j_{xk}T^2 \\ a_{xk+1} = a_{xk} + j_{xk}T \\ y_{k+1} = y_k + v_{y_k}T + \frac{1}{2}a_{y_k}T^2 + +\frac{1}{6}j_{y_k}T^3 \\ v_{y_{k+1}} = v_{y_k} + a_{y_k}T + \frac{1}{2}j_{y_k}T^2 \\ a_{y_{k+1}} = a_{y_k} + j_{y_k}T \end{cases} \tag{4.5}$$

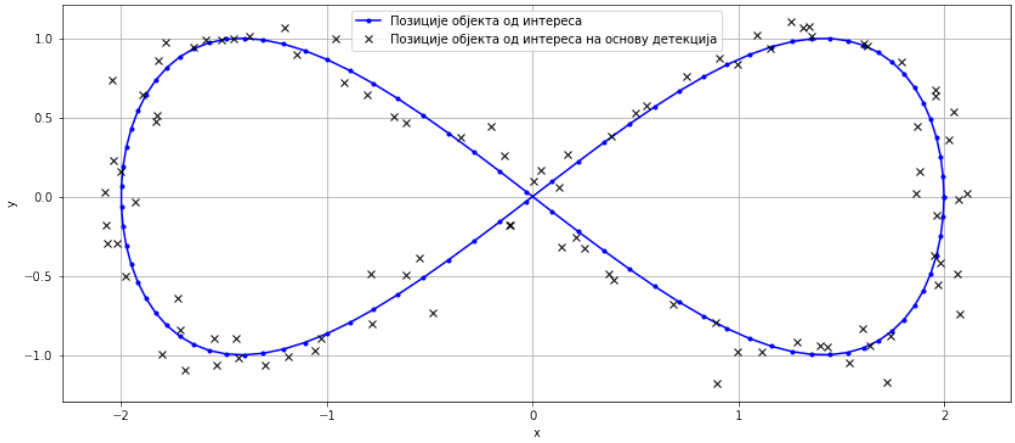
где је T време које је протекло између k-тог и $k+1$-ог корака, и где се за чланове вишег реда (укључујући трзај), односно j_x и j_y, претпоставља да су Гаусови бели шумови нултих очекиваних вредности.

Даље, нека детекције које добијамо на сваких T секунди садрже информацију о позицији брода у Декартовим координатама, угаоној ($angV$) и линеарној ($linV$) брзини брода са додатком шума који резултира у непрецизности претходних мерења. Угаона и линеарна брзина су изражене, редом, у $\frac{rad}{s}$ и $\frac{m}{s}$. Односно, радар оцењује позицију, угаону и линеарну брзину брода са одређеном прецизношћу, тачније дисперзијом од 0.1. (код за симулацију описаних трајекторија брода се налази на страницама бр. 73 и 74). Веза између претходних величина, брзине и убрзања брода је следећа:

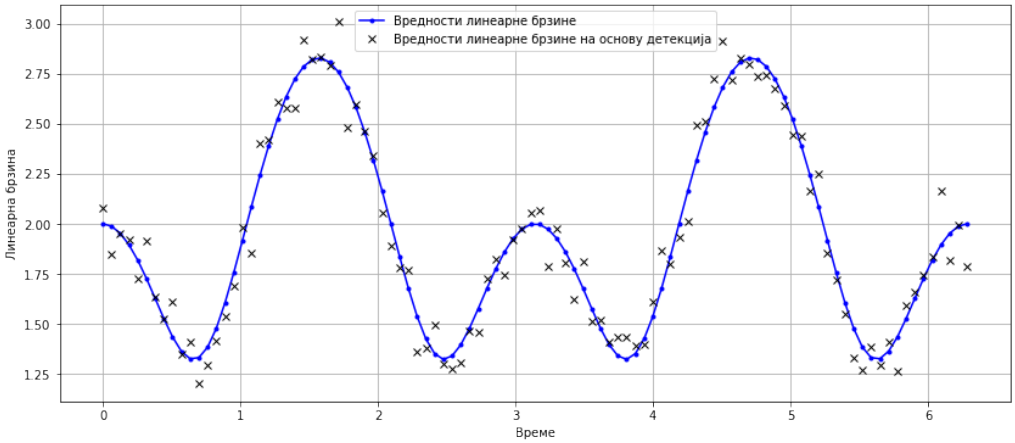
$$linV = \sqrt{v_x^2 + v_y^2} \tag{4.6}$$

$$angV = \frac{v_x a_y - v_y a_x}{\sqrt{v_x^2 + v_y^2}} \tag{4.7}$$

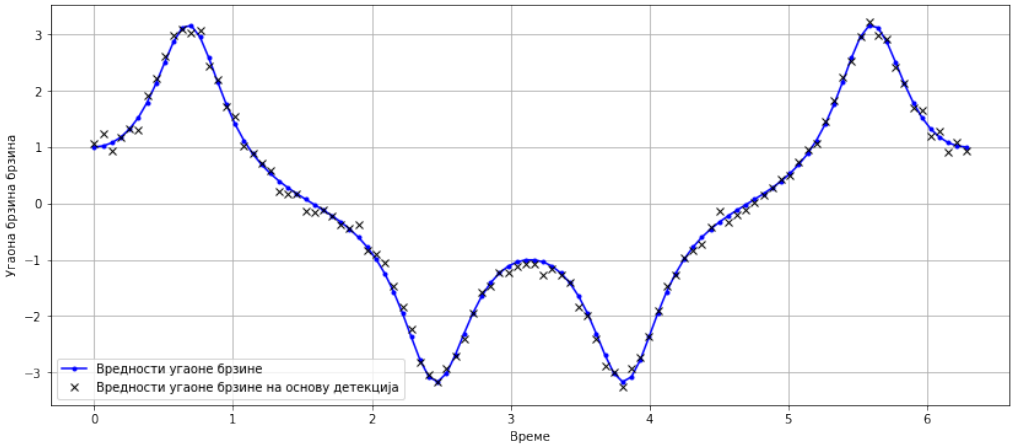
На слици 4.2, 4.3, 4.4 приказане су редом вредности позиција, линеарне и угаоне брзине брода добијене из симулираних детекција за сценарио кретања брода чија путања образује облик броја осам, у односу на тачне вредности. На слици 4.5 представљен је упоредни приказ стварне путање кретања брода и путање кретања брода добијене на основу претходних детекција без примене алгоритама за оцену стања система.



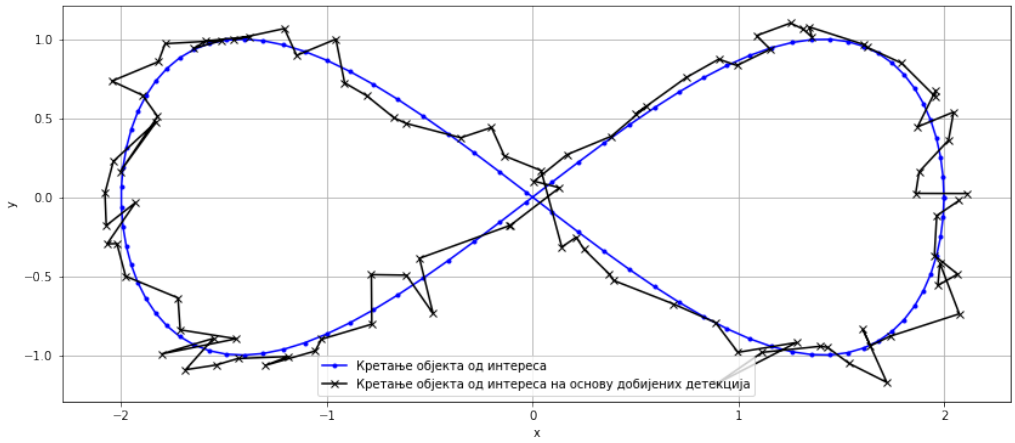
Слика 4.2: Позиција брода



Слика 4.3: Линеарна брзина брода



Слика 4.4: Угаона брзина брода



Слика 4.5: Упоредни приказ стварног кретања брода и кретања брода на основу детекција

Примена Калмановог филтера на први случај

Вектор стања модела из претходног одељка укључује позицију брода (x, y), брзину брода (v_x, v_y) и убрзање брода (a_x, a_y). Дакле, имамо следеће:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ v_{xk} \\ a_{xk} \\ y_k \\ v_{yk} \\ a_{yk} \end{bmatrix}$$

На основу једначина стања (4.5) дефинишемо матрицу преласка Φ_k:

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{1}{2}T^2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

За коваријациону матрица процесног шума Q_k с обзиром на претпоставку да трзаји брода j_x и j_y представљају Гаусов бели шум нултих очекиваних вредности и дисперзија редом σ_{j_x} и σ_{j_y} имамо:

$$\mathbf{Q}_k = Var\left(\begin{bmatrix} \frac{T^3}{6} j_{xk} \\ \frac{T^2}{2} j_{xk} \\ T j_{xk} \\ \frac{T^3}{6} j_{y_k} \\ \frac{T^2}{2} j_{y_k} \\ T j_{y_k} \end{bmatrix}\right) = Var\left(j_{xk} \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \\ 0 \\ 0 \\ 0 \end{bmatrix}\right) + Var\left(j_{y_k} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix}\right)$$

$$= \sigma_{j_x} Var\left(\begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \\ 0 \\ 0 \\ 0 \end{bmatrix}\right) + \sigma_{j_y} Var\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix}\right)$$

За почетак, игноришимо информације о линеарној и угаоној брзини добијене из детекција на сваких T секунди и посматрајмо само информације о позицији. У том случају вектор мерења има следећи облик:

$$\mathbf{z}_k = \begin{bmatrix} z_{xk} \\ z_{y_k} \end{bmatrix}$$

Сходно томе, дефинишемо матрицу мерења $\mathbf{H}_k$:

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

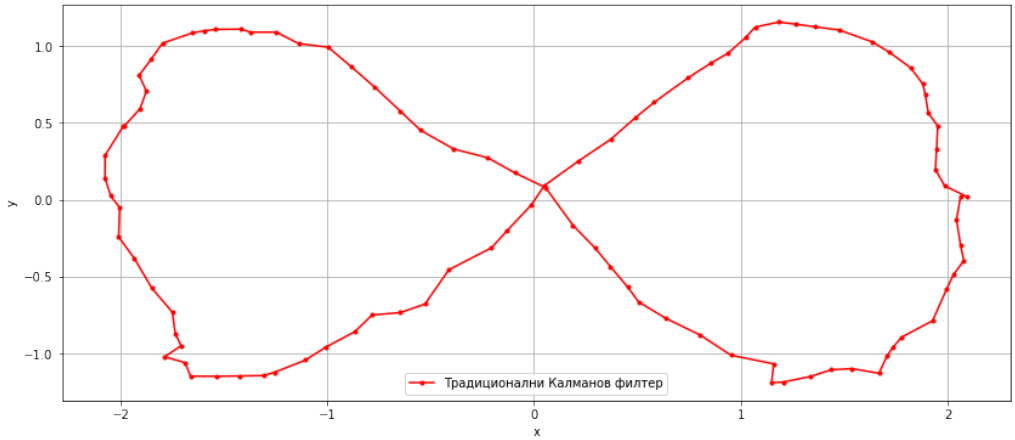
У односу на претходно дефинисану прецизност детекција, формирамо коваријациону матрицу шума мерења:

$$\mathbf{R}_k = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

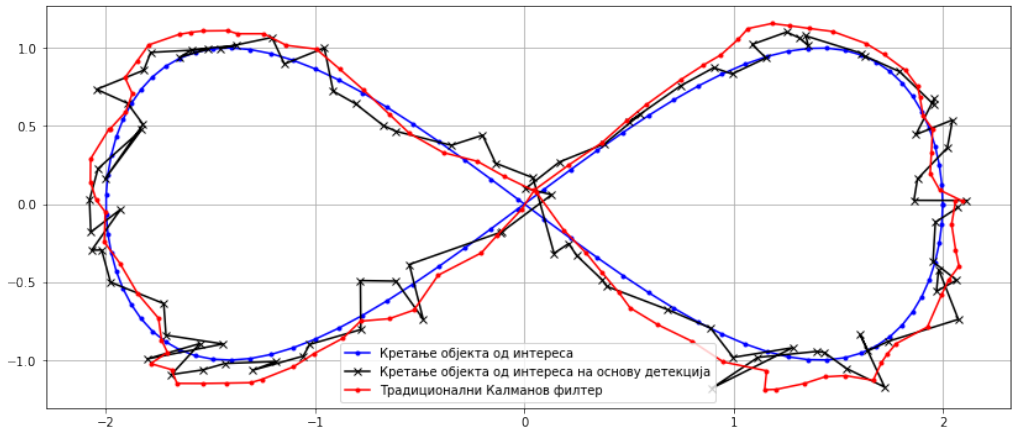
Како имамо увид у почетно стање брода $\mathbf{x_0}$, довољно је узети мање вредности почетне коваријационе матрице $\mathbf{P_0}$.

Сада, када смо претходни модел записали у погодној форми, можемо применити традиционални Калманов филтер. Резултати добијени традиционалним Калмановим филтером приказани су на слици 4.6. Слика 4.7 представља упоредни приказ

стварне путање кретања брода, путање кретања брода добијене на основу претходних детекција без примене алгоритама за оцену стања система и путање кретања брода добијене на основу претходних детекција применом традиционалног Калмановог филтера.



Слика 4.6: Традиционални Калманов филтер



Слика 4.7: Упоредни приказ кретања

Једначина мерења, у овом случају, имала је следећи облик:

$$\mathbf{z_k} = \mathbf{H}_k\mathbf{x_k} + \mathbf{v_k}, \tag{4.8}$$

где је $\mathbf{z_k}$ претходно дефинисани вектор мерења, $\mathbf{H}_k$ претходно дефинисана матрица мерења, а $\mathbf{v_k}$ шум мерења са претходно дефинисаном коваријационом матрицом.

Уколико бисмо у вектор мерења уврстили и информације о линеарној и угаоној брзини како бисмо добили боље резултате праћења брода, односно:

$$\mathbf{z}_k = \begin{bmatrix} z_{xk} \\ z_{yk} \\ z_{angVk} \\ z_{linVk} \end{bmatrix},$$

традиционални Калманов филтер, с обзиром на линеарност у једначини мерења (4.8), не може да се примени. Разлог лежи у нелинеарној вези линеарне и угаоне брзине са вектором стања датом релацијама (4.6) и (4.7). Као резултат тога, није могуће формирати матрицу мерења $\mathbf{H}_k$, тако да веза између вектора мерења и вектора стања буде одређена једначином (4.8). У том случају, једначину мерења формулишемо на следећи начин:

$$\mathbf{z_k} = \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k}, \tag{4.9}$$

где је $\mathbf{h}$ нелинеарна функција, непрекидно диференцијабилна на посматраном домену. Односно функција $\mathbf{h}$, с обзиром на једначине (4.6) и (4.7), има следећи облик:

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} x \\ y \\ \frac{v_x a_y - v_y a_x}{\sqrt{v_x^2 + v_y^2}} \\ \sqrt{v_x^2 + v_y^2} \end{bmatrix} \approx \begin{bmatrix} z_{xk} \\ z_{yk} \\ z_{angVk} \\ z_{linVk} \end{bmatrix} \tag{4.10}$$

Ради примене проширеног Калмановог филтера, потребно је још израчунати Јакобијеву матрицу функције $\mathbf{h}$, односно:

$$a = \frac{(\hat{\mathbf{v}_{\mathbf{x}}}^2_{k|k} + \hat{\mathbf{v}_{\mathbf{y}}}^2_{k|k})\hat{\mathbf{a}_{\mathbf{y}}}_{k|k} - 2\hat{\mathbf{v}_{\mathbf{x}}}_{k|k}(\hat{\mathbf{v}_{\mathbf{x}}}_{k|k}\hat{\mathbf{a}_{\mathbf{y}}}_{k|k} - \hat{\mathbf{v}_{\mathbf{y}}}_{k|k}\hat{\mathbf{a}_{\mathbf{x}}}_{k|k}))}{(\hat{\mathbf{v}_{\mathbf{x}}}^2_{k|k} + \hat{\mathbf{v}_{\mathbf{y}}}^2_{k|k})^2}$$

$$b = \frac{-\hat{\mathbf{v}_{\mathbf{y}}}_{k|k}}{(\hat{\mathbf{v}_{\mathbf{x}}}^2_{k|k} + \hat{\mathbf{v}_{\mathbf{y}}}^2_{k|k})}$$

$$c = \frac{(\hat{\mathbf{v}_{\mathbf{x}}}^2_{k|k} + \hat{\mathbf{v}_{\mathbf{y}}}^2_{k|k})\hat{\mathbf{a}_{\mathbf{x}}}_{k|k} - 2\hat{\mathbf{v}_{\mathbf{y}}}_{k|k}(\hat{\mathbf{v}_{\mathbf{x}}}_{k|k}\hat{\mathbf{a}_{\mathbf{y}}}_{k|k} - \hat{\mathbf{v}_{\mathbf{y}}}_{k|k}\hat{\mathbf{a}_{\mathbf{x}}}_{k|k}))}{(\hat{\mathbf{v}_{\mathbf{x}}}^2_{k|k} + \hat{\mathbf{v}_{\mathbf{y}}}^2_{k|k})^2}$$

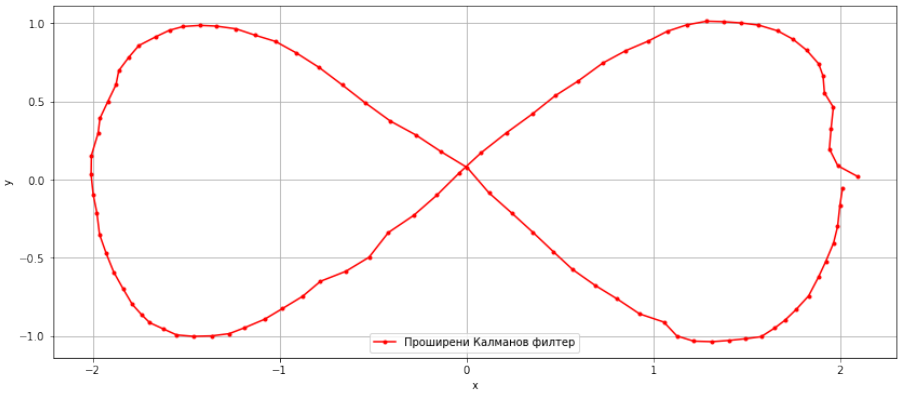
$$d = \frac{\hat{\mathbf{v}}_{\mathbf{x}k|k}}{(\hat{\mathbf{v}}^2_{\mathbf{x}k|k} + \hat{\mathbf{v}}^2_{\mathbf{y}k|k})}$$

$$e = \frac{\hat{\mathbf{v}}_{\mathbf{x}k|k}}{\sqrt{(\hat{\mathbf{v}}^2_{\mathbf{x}k|k} + \hat{\mathbf{v}}^2_{\mathbf{y}k|k})}}$$

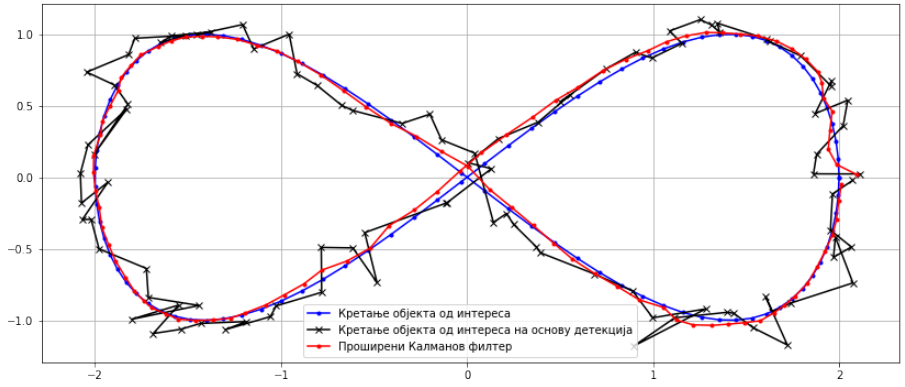
$$f = \frac{\hat{\mathbf{v}}_{\mathbf{y}k|k}}{\sqrt{(\hat{\mathbf{v}}^2_{\mathbf{x}k|k} + \hat{\mathbf{v}}^2_{\mathbf{y}k|k})}}$$

$$J_f(\hat{\mathbf{x}}_{k|k}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & a & b & 0 & c & d \\ 0 & e & 0 & 0 & f & 0 \end{bmatrix}$$

Надаље, осим замене матрице мерења $\mathbf{H}_k$ са Јакобијевом матрицом нелинеарне функције $\mathbf{h}$, корекциони и предикциони корак традиционалног Калмановог филтера и проширеног Калмановог филтера се имплементирају на исти начин. Резултати добијени проширеним Калмановим филтером приказани су на слици 4.8. Слика 4.9 представља упоредни приказ стварне путање кретања брода, путање кретања брода добијене на основу претходних детекција без примене алгоритама за оцену стања система и путање кретања брода добијене на основу претходних детекција применом проширеног Калмановог филтера.

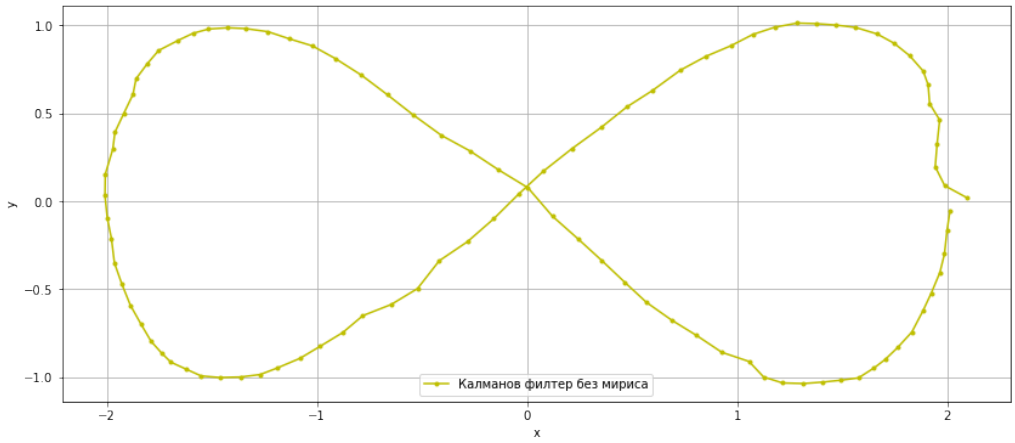


Слика 4.8: Проширени Калманов филтер

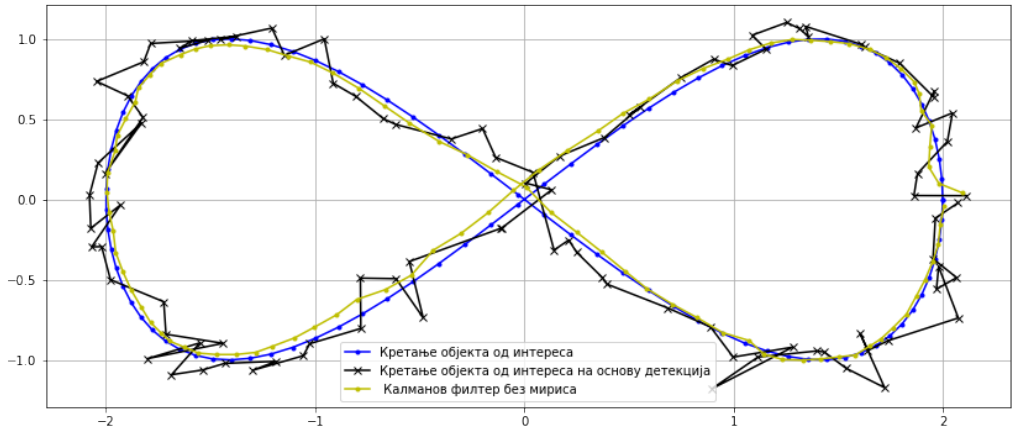


Слика 4.9: Упоредни приказ кретања

На основу сигма тачака добијених из једначина (3.4), (3.5) и (3.6) и њиховом пропагацијом кроз претходно дефинисану нелинеарну функцију $\mathbf{h}$, на претходни модел можемо применити и Калманов филтер без мириса. Вредности параметара α, β и k које утичу на вредност фактора скалирања λ су редом $0.1, 2$ и -1. .Резултати добијени Калмановим филтером без мириса приказани су на слици 4.10. Слика 4.11 представља упоредни приказ стварне путање кретања брода, путање кретања брода добијене на основу претходних детекција без примене алгоритама за оцену стања система и путање кретања брода добијене на основу претходних детекција применом Калмановог филтера без мириса.



Слика 4.10: Калманов филтер без мириса

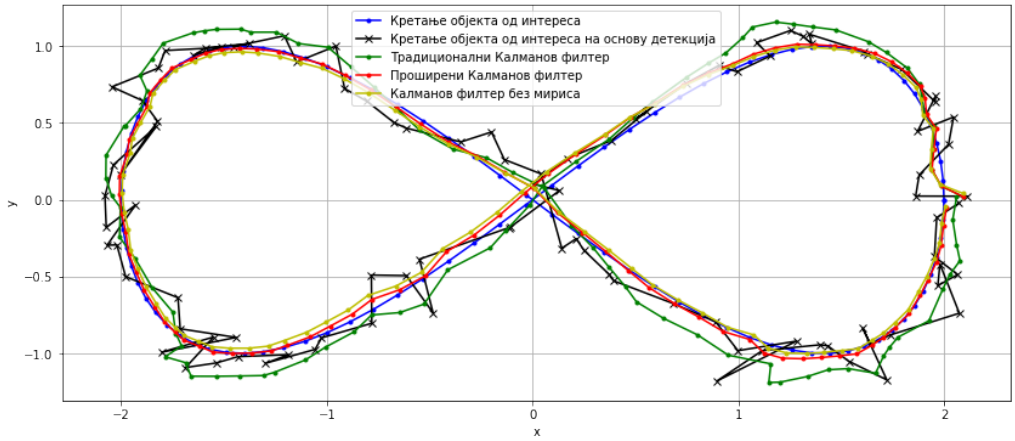


Слика 4.11: Упоредни приказ кретања

Слика 4.12 представља упоредни приказ стварне путање кретања брода, путање кретања брода добијене на основу претходних детекција без примене алгоритама за оцену стања система и путање кретања брода добијене на основу претходних детекција применом традиционалног Калмановог филтера, проширеног Калмановог филтера и Калмановог филтера без мириса. На слици 4.13, иако је акценат на оцени позиције објекта од интереса, информативно су дати и упоредни прикази осталих величине вектора стања применом сваке од претходних верзија Калмановог филтера, односно, редом v_x, v_y, a_x и a_y.

Такође, у табели 4.1 приказане су вредности корена средње квадратне грешке (*RMSE - root mean squared error*) за сваку величину вектора стања за сваку од имплементираних верзија Калмановог филтера.

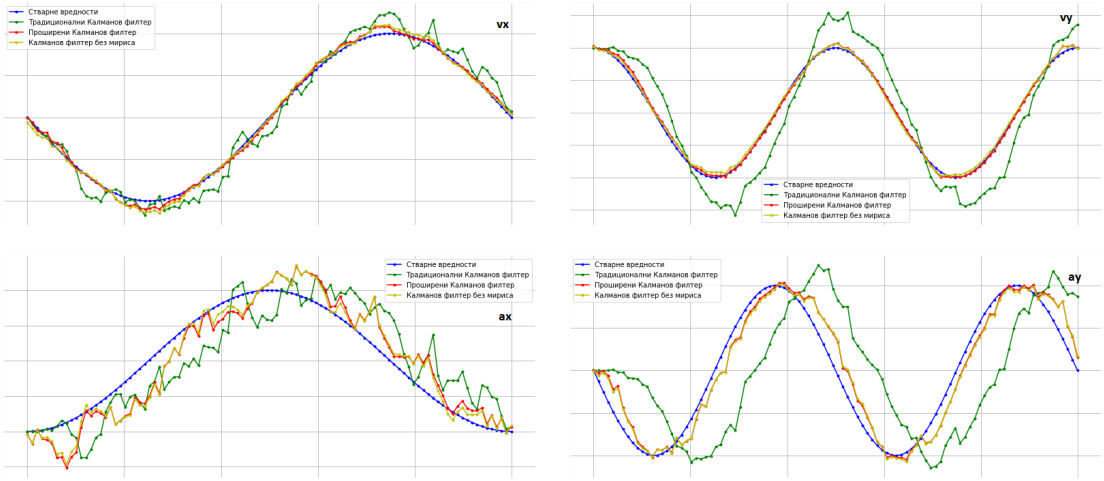
$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_i - f(x_i))^2}$$



Слика 4.12: Упоредни приказ кретања

Табела 4.1: Корен средње квадратне грешке праћења

Тип модела	x (°)	y (°)	vx ($\frac{m}{s}$)	ax ($\frac{m}{s^2}$)	vy ($\frac{m}{s}$)	ay ($\frac{m}{s^2}$)
Калманов филтер	0.05	0.10	0.27	0.81	0.76	2.80
Проширени Калманов филтер	0.03	0.03	0.08	0.58	0.76	0.72
Калманов филтер без мириса	0.02	0.03	0.10	0.55	0.78	0.72



Слика 4.13: Упоредни приказ

На основу претходних упоредних приказа и табеле средње квадратних грешака може се закључити да обе модификоване верзије Калмановог филтера дају значајно боље резултате. Оцењено кретање је изнимно углађеније и у великој мери прати стварну криву. Перформансе проширеног Калмановог филтера и Калмановог филтера без мириса су доста сличне. Погоднијим избором сигма тачака, тј. вредности параметара α, β и k резултати Калмановог филтера без мириса би се вероватно побољшали. Такође, интересантно је посматрати и промене у резултатима при одабиру различитих вредности коваријационих матрица вектора шума и вектора мерења.

4.2 Моделовање кретања циља и примена Калмановог филетера на други случај

Радарске детекције које се добијају на сваких T секунди [1], у овом случају, садрже информацију о радијалној удаљености (ρ), углу (ϕ) и радијалној брзини ($\dot{\rho}$), изражене редом у km,°,rad/s. Представљање позиције у картезинајским координатама пружа несумњиве предности у даљој обради и омогућава потпуну или делимичну линеаризацију једначине кретања. Због тога ће, у оквиру моделовања кретања циља бити извршена конверзија мерења из

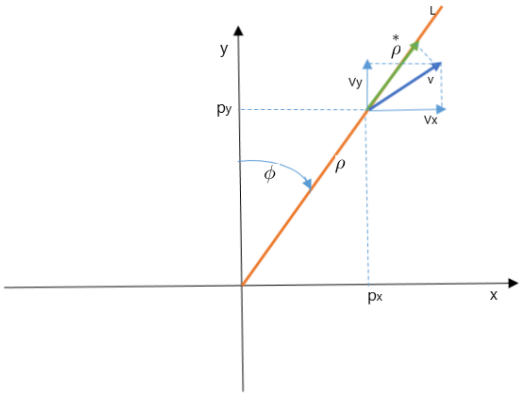
[1] Подаци су доступни на [1]

поларних координата у Декартове координате. Као модел кретања користи се модел са приближно константном брзином, па су једначине стања дате са:

$$\begin{cases} x_{k+1} = x_k + v_{xk}T \\ v_{xk+1} = v_{xk} \\ y_{k+1} = y_k + v_{y_k}T \\ v_{y_{k+1}} = v_{y_k} \end{cases} \tag{4.11}$$

где је T време које је протекло између k-тог и $k+1$-ог корака, а (x_{k+1}, y_{k+1}) и $(v_{xk+1}, v_{y_{k+1}})$, редом, позиција и брзина брода у Декартовим координатама. Веза између претходних величина и величина које имамо на основу радарских детекција, односно радијалне удаљености, угла и радијалне брзине, је следећа:

$$x = \rho\, cos(\phi)$$
$$y = \rho\, sin(\phi)$$
$$v_x = \overset{*}{\rho}\, cos(\phi)$$
$$v_y = \overset{*}{\rho}\, sin(\phi)$$



Слика 4.14: Геометријски приказ везе између претходних величина

Примена Калмановог филтера на други случај

Вектор стања модела из претходног одељка укључује позицију брода (x, y) и брзину брода (v_x, v_y) у Декартовим координатама. Дакле, имамо следеће:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix}$$

На основу једначина стања (4.11) дефинишемо матрицу преласка $\mathbf{\Phi}_k$:

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Како је убрзање брода непознато, можемо га додати компоненти шума, те би се такав случајни шум аналитички могао изразити последњим члановима наредне једначине стања:

$$\begin{cases} x_{k+1} = x_k + v_{xk}T + \frac{1}{2}a_{xk}T^2 \\ y_{k+1} = y_k + v_{yk}T + \frac{1}{2}a_{yk}T^2 \\ v_{xk+1} = v_{xk} + a_{xk}T \\ v_{yk+1} = v_{yk} + a_{yk}T \end{cases} \tag{4.12}$$

Дакле, случајни вектор убрзања $\mathbf{w}_k$ има следећу форму:

$$\mathbf{w}_k = \begin{bmatrix} w_{xk} \\ w_{yk} \\ w_{v_x k} \\ w_{v_y k} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a_{xk}T^2 \\ \frac{1}{2}a_{yk}T^2 \\ a_{xk}T \\ a_{yk}T \end{bmatrix}$$

и описан је нултом очекиваном вредношћу и коваријационом матрицом $\mathbf{Q}_k$.

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$$

Случајни вектор $\mathbf{w}_k$ могуће је разложити на две компоненте, односно матрицу $\mathbf{G}$ димензија 4×2, која не садржи случајну компоненту, и матрицу $\mathbf{a}$ димезија 2×1:

$$\mathbf{w}_k = \begin{bmatrix} \frac{1}{2}a_{xk}T^2 \\ \frac{1}{2}a_{y_k}T^2 \\ a_{xk}T \\ a_{y_k}T \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \end{bmatrix}}_{\mathrm{G}} \underbrace{\begin{bmatrix} a_{xk} \\ a_{y_k} \end{bmatrix}}_{\mathrm{a}},$$

где је T време које је протекло између k-тог и $k+1$-ог корака и убрзање случајни вектор нулте очекиване вредности и стандардних девијација σ_{a_x} и σ_{a_y}.

Даље, за коваријациону матрицу $\mathbf{Q}_k$ имамо следеће:

$$\mathbf{Q}_k = E(\mathbf{w}_k \mathbf{w}_k{}^\intercal) = E(\mathbf{G}\mathbf{a}\mathbf{a}^\intercal\mathbf{G}^\intercal)$$

Како матрица $\mathbf{G}$ не садржи случајну компоненту она се може извући изван очекивања, то јест:

$$\mathbf{Q}_k = \mathbf{G}E(\mathbf{a}\mathbf{a}^\intercal)\mathbf{G}^\intercal = \mathbf{G}\begin{bmatrix} \sigma_{a_x}^2 & \sigma_{a_{xy}} \\ \sigma_{a_{xy}} & \sigma_{a_y}^2 \end{bmatrix}\mathbf{G}^\intercal,$$

где су $\sigma_{a_x}^2$ и $\sigma_{a_y}^2$ редом варијансе за a_{xk} и a_{y_k} за свако k. Такође, претпоставља се да су a_{xk} и a_{y_k} некорелисане случајне величине за свако k, па је коваријација за a_{xk} и a_{y_k}, односно $\sigma_{a_{xy}}$, једнака нули.

Комбинујући све претходно у једну матрицу, дефинишемо коваријациону матрицу процесног шума $\mathbf{Q}_k$:

$$\mathbf{Q_k} = \begin{bmatrix} \frac{T^4}{4}\sigma_{a_x}^2 & 0 & \frac{T^3}{2}\sigma_{a_x}^2 & 0 \\ 0 & \frac{T^4}{4}\sigma_{a_y}^2 & 0 & \frac{T^3}{2}\sigma_{a_y}^2 \\ \frac{T^3}{2}\sigma_{a_x}^2 & 0 & T^2\sigma_{a_x}^2 & 0 \\ 0 & \frac{T^3}{2}\sigma_{a_y}^2 & 0 & T^2\sigma_{a_y}^2 \end{bmatrix} \tag{4.13}$$

Једначину мерења формулишемо на следећи начин:

$$\mathbf{z_k} = \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k}, \tag{4.14}$$

где је $\mathbf{h}$ нелинеарна функција, непрекидно диференцијабилна на посматраном домену и $\mathbf{z_k}$ вектор мерења.

Вектор мерења $\mathbf{z_k}$ има следећи облик:

$$\mathbf{z}_k = \begin{bmatrix} \rho_k \\ \phi_k \\ \overset{*}{\rho}_k \end{bmatrix},$$

а нелинеарна функција $\mathbf{h}$, с обзиром на већ претходно поменуту конверзију у Декартов координатни систем има следећи облик:

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \sqrt{x^2+y^2} \\ arctan(\frac{y}{x}) \\ \frac{xv_x+yv_y}{\sqrt{x^2+y^2}} \end{bmatrix} \tag{4.15}$$

За коваријациону матрицу шума мерења $\mathbf{R}_k$, уз претпоставку да све три компоненте вектора мерења нису међусобно корелисане, узимамо следећу матрицу:

$$\mathbf{R}_k = \begin{bmatrix} 0.09 & 0 & 0 \\ 0 & 0.009 & 0 \\ 0 & 0 & 0.09 \end{bmatrix} \tag{4.16}$$

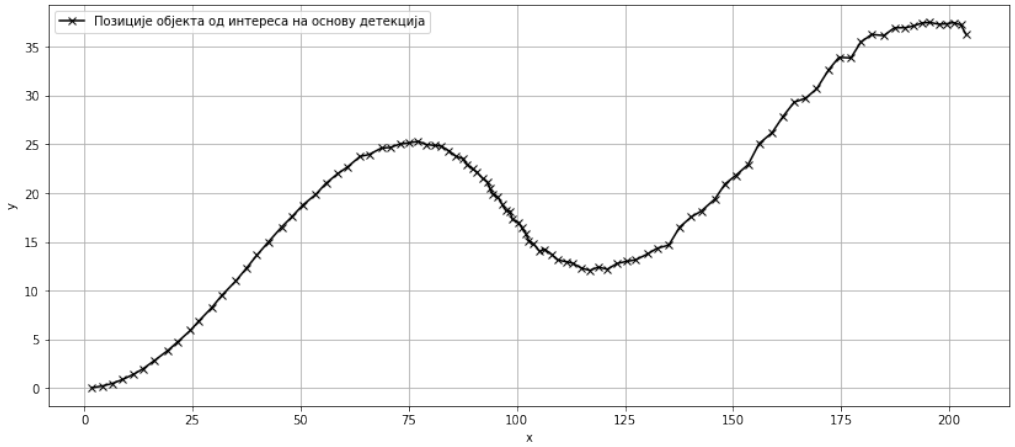
Такође, у случају проширеног Калмановог филтера, потребно је још израчунати Јакобијеву матрицу функције $\mathbf{h}$, односно:

$$d = \sqrt{x^2+y^2}$$
$$j_{11} = \frac{x}{d}$$
$$j_{12} = \frac{y}{d}$$
$$j_{21} = -\frac{y}{d^2}$$
$$j_{22} = \frac{x}{d^2}$$
$$j_{31} = \frac{y(v_x y - v_y x)}{d^{\frac{3}{2}}}$$
$$j_{32} = \frac{x(v_y x - v_x y)}{d^{\frac{3}{2}}}$$

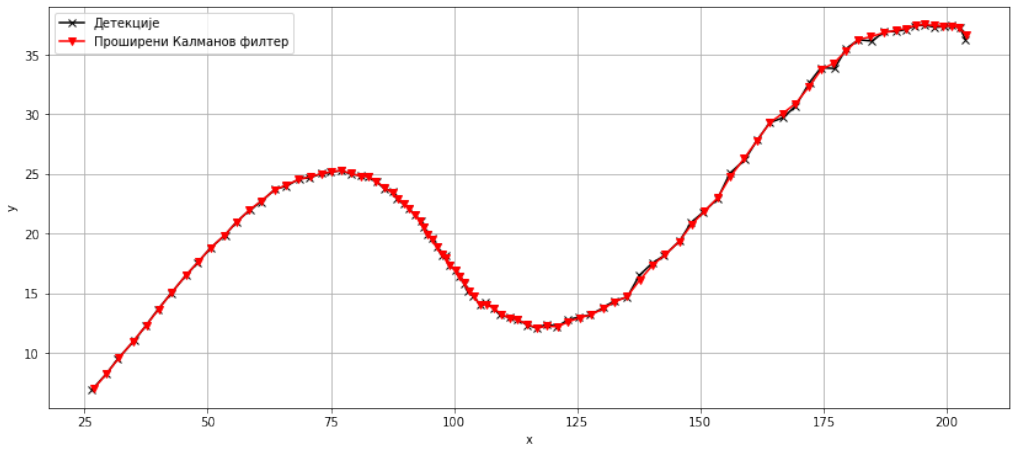
$$J_f(\hat{\mathbf{x}}_{k|k}) = \begin{bmatrix} j_{11} & j_{12} & 0 & 0 \\ j_{21} & j_{22} & 0 & 0 \\ j_{31} & j_{32} & j_{11} & j_{12} \end{bmatrix}$$

На крају, погодним избором вредности стандардних девијација σ_{a_x} и σ_{a_y}, почетног стања $\mathbf{x_0}$, почетне коваријационе матрице $\mathbf{P_0}$, као и погодним одабиром сигма тачака и параметара α, β и k у случају Калмановог филтера без мириса, на претходне податке са радара се може применити проширени Калманов филтер као и Калманов филтер без мириса. Избор претходно поменутих вредности зависи, пре свега, од прецизности наших процена на почетку. На пример, као што је већ речено у глави 2 о Калмановом филтеру, уколико је наша несигурност у изабрану почетну вредност вектора стања велика, онда се за почетну вредност коваријационе матрице $\mathbf{P_0}$ мора узети велика вредност. Детаљније смернице за оптимални избор вредности ових параметара, као и за вредности коваријационе матрице процесног шума, могу се пронаћи у [9].

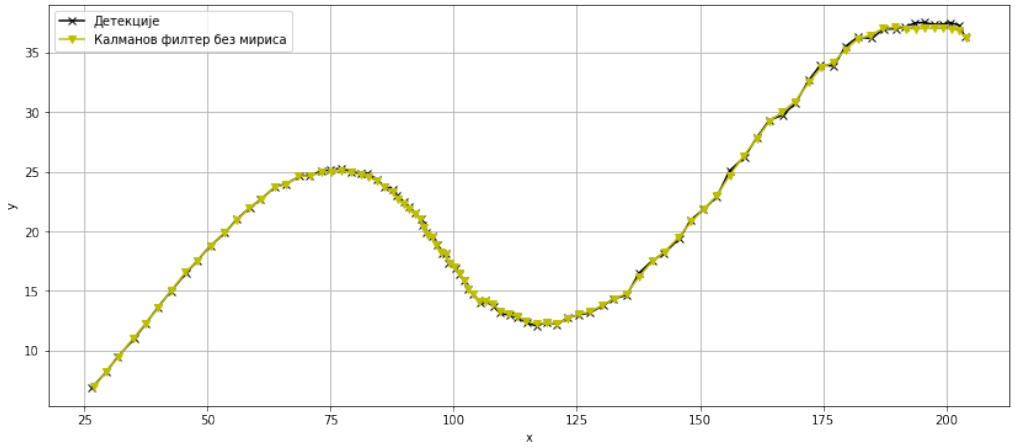
На слици 4.15 приказана је путања кретања добијене на основу радарских детекција без примена алгоритама за оцену стања система. На сликама 4.16 и 4.17 дат је упоредни приказ путање добијене на основу радарских детекција без примене алгоритама за оцену стања система и путање кретања циља добијене на основу претходних радарских детекција применом проширеног Калмановог филтера, односно Калмановог филтера без мириса. На слици 4.18, ради поређења, за одређене временске кораке дате су вредности стања добијене на основу детекција, проширеног Калмановог филтера и Калмановог филтера без мириса.



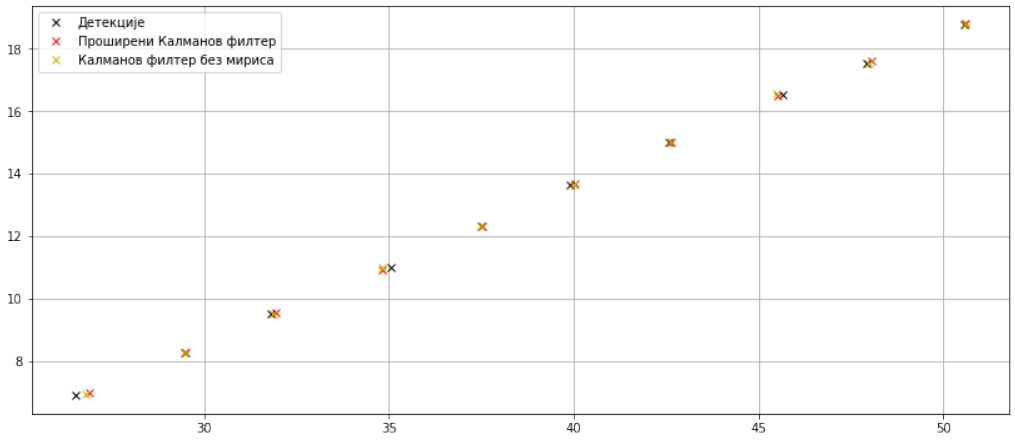
Слика 4.15: Приказ кретања брода на основу детекција



Слика 4.16: Упоредни приказ кретања - Проширени Калманов филтер



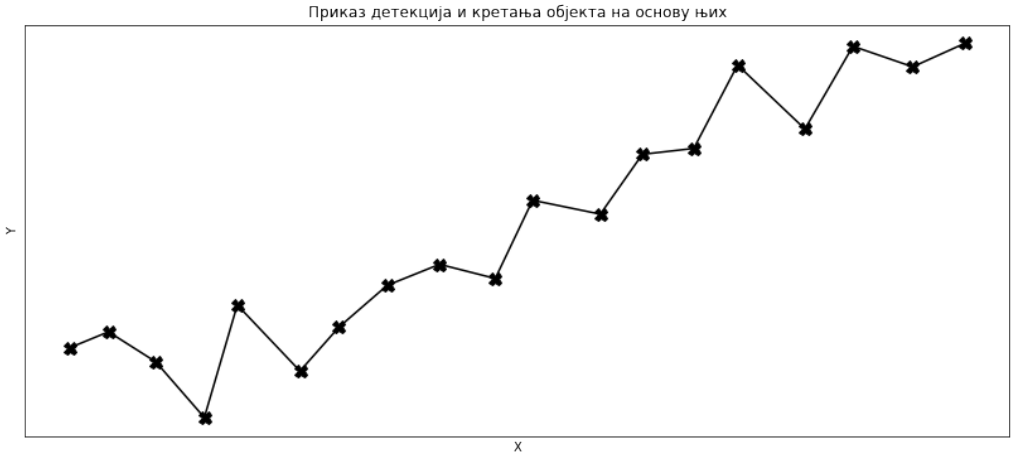
Слика 4.17: Упоредни приказ кретања - Калманов филтер без мириса



Слика 4.18: Упоредни приказ кретања

4.3 Моделовање кретања циља и примена Калмановог филтера на трећи случај

Радар, као и у другом случају, мери удаљеност објекта од интереса, пружајући информације о његовим поларним координатама (ρ, ϕ) на сваких T секунди изражене у редом, km и °. Такође, радар користи Доплеров ефекат у циљу одређивања радијалне компоненете релативне брзине у односу на радар ($\overset{*}{\rho}$) изражене у $\frac{rad}{s}$. Дате детекције, придружене објекту од интереса, се могу конвертовати у Декартов координатни систем применом формула на страни 58 (функција $P2C$ у кодовима). На слици 4.20 приказане су дате детекције и кретање објекта на основу њих.



Слика 4.19: Упоредни приказ кретања

Вектор стања $\mathbf{x}_k$ садржи информације о оцењеној позицији и брзини у k-том временском кораку, изражене у Декартовим координатама.

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix}$$

Вектор мерења $\mathbf{z}_k$ садржи информације о поларним координатама објекта у k-том временском кораку, добијене са радара.

$$\mathbf{z}_k = \begin{bmatrix} \rho_k \\ \phi_k \\ \dot{\rho}_k \end{bmatrix}$$

Нека p_k представља позицију брода у Декартовим координатама у k-том временском кораку. Како је пут који циљ пређе у току једне инстанце (Δd), на основу формула из кинематике, једнак производу просечне брзине кретања тела и времена T за које тело пређе дати пут, имамо:

$$\Delta d = p_k - p_{k-1} = \overline{v}T.$$

Односно, важи:

$$p_k - p_{k-1} = \frac{v_k + v_{k-1}}{2}T,$$

$$p_k = p_{k-1} + \frac{v_k + v_{k-1}}{2}T. \tag{4.17}$$

Са друге стране, такође из формула кинематике, убрзање можемо дефинисати као однос промене брзине и временског интервала у коме се брзина променила.:

$$a = \frac{v_k + v_{k-1}}{T}$$

На основу претходног, важи:

$$v_k = v_{k-1} + aT. \tag{4.18}$$

Дакле, компоненте брзине у две димензије описане су на следећи начин:

$$\begin{cases} v_{xk+1} = v_{xk} + a_{xk}T \\ v_{yk+1} = v_k + a_{xk}T \end{cases} \tag{4.19}$$

Заменом (4.18) у (4.17) имамо:

$$p_k = p_{k-1} + \frac{v_{k-1} + aT + v_{k-1}}{2}T,$$

$$p_k = p_{k-1} + v_{k-1}T + a\frac{T^2}{2}.$$

Узевши дводимензиони случај у обзир, важи следеће:

$$\begin{cases} x_{k+1} = x_k + v_{xk}T + \frac{1}{2}a_{xk}T^2 \\ y_{k+1} = y_k + v_{yk}T + \frac{1}{2}a_{yk}T^2 \end{cases} \tag{4.20}$$

Једначине (4.19) и (4.20) описују једначину стања модела, па матрицу преласка $\mathbf{\Phi}_k$ дефинишемо на следећи начин:

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Коваријациона матрица процесног шума $\mathbf{Q}_k$ добија се на исти начин као у другом случају, додавањем убрзања компоненти шума.

$$\mathbf{Q_k} = \begin{bmatrix} \frac{T^4}{4}\sigma_{a_x}^2 & 0 & \frac{T^3}{2}\sigma_{a_x}^2 & 0 \\ 0 & \frac{T^4}{4}\sigma_{a_y}^2 & 0 & \frac{T^3}{2}\sigma_{a_y}^2 \\ \frac{T^3}{2}\sigma_{a_x}^2 & 0 & T^2\sigma_{a_x}^2 & 0 \\ 0 & \frac{T^3}{2}\sigma_{a_y}^2 & 0 & T^2\sigma_{a_y}^2 \end{bmatrix}$$

Избор вредности дисперзија убрзања мете $\sigma_{a_x}^2$ и $\sigma_{a_y}^2$ утиче на перформансе Калмановог филтера. Неки од начина за њихов избор описани су у [5] и [32]. Најчешће се $\sigma_{a_x}^2$ и $\sigma_{a_y}^2$ бирају на основу претпостављеног модела кретања мета или емпиријски. Један од познатих, ефикасних одабира, ових параметара дефинисан је од стране Калате (*Paul R. Kalata*) у [20]. Ипак, и поред свега претходног, не постоји тачно дефинисан метод за одабир вредности ових параметара.

Напомена: Информације о другим различитим формама коваријационе матрице процесног шума $\mathbf{Q}_k$ за различите моделе кретања циљева могу се наћи у [31].

Претпостављајући да компоненте вектора мерења нису међусобно корелисане коваријациона матрица вектора мерења има следећи облик:

$$\mathbf{R_k} = \begin{bmatrix} \sigma_\rho^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_{\overset{*}{\rho}_k}^2 \end{bmatrix}$$

У овом случају, за радарске детекције, критеријуму за процену грешке се заснивају, пре свега, на амплитуди унутар тачака грида које обухватају контуре критеријумске функције детекционог алгоритма у односу на положај центра масе контуре. Дисперзије мерења по углу и растојању рачунају се према одступањима у односу на центар масе контуре са амплитудама као тежинским факторима. Вредности коваријационе матрица вектора мерења могу се разликовати за сваки временски корак, али у овом примеру, то није случај.

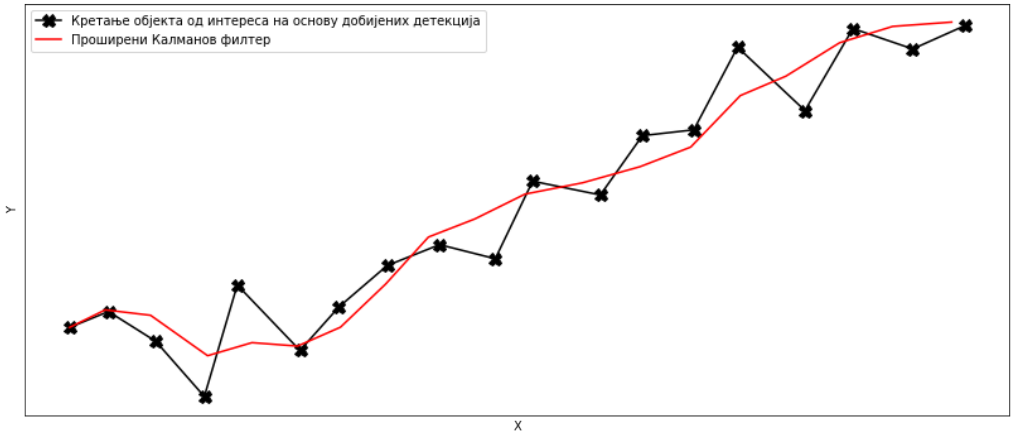
Даље, потребно је још пронаћи нелинеарну функцију $\mathbf{h}$ која пресликава предвиђено стање $\mathbf{x}$ у простор мерења, односно:

$$\begin{pmatrix} x_k \\ y_k \\ v_{xk} \\ v_{y_k} \end{pmatrix} \rightarrow \begin{pmatrix} \rho \\ \phi \\ \overset{*}{\rho} \end{pmatrix}.$$

Нелинеарна функција $\mathbf{x}$, непрекидна на посматраном домену, тада има исти облик као у другом случају.

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ arctan(\frac{y}{x}) \\ \frac{xv_x + yv_y}{\sqrt{x^2+y^2}} \end{bmatrix}$$

Користећи још Јакобијеву матрицу, дефинисану на исти начин као у другом случају, на претходно описане податке могуће је применити проширени Калманов филтер (Код 6.4). Слика 4.20 представља упоредни приказ путање кретања добијене на основу претходних детекција без примене алгоритма за оцену стања система и путање кретања добијене на основу претходних детекција применом проширеног Калмановог филтера.



Слика 4.20: Упоредни приказ кретања

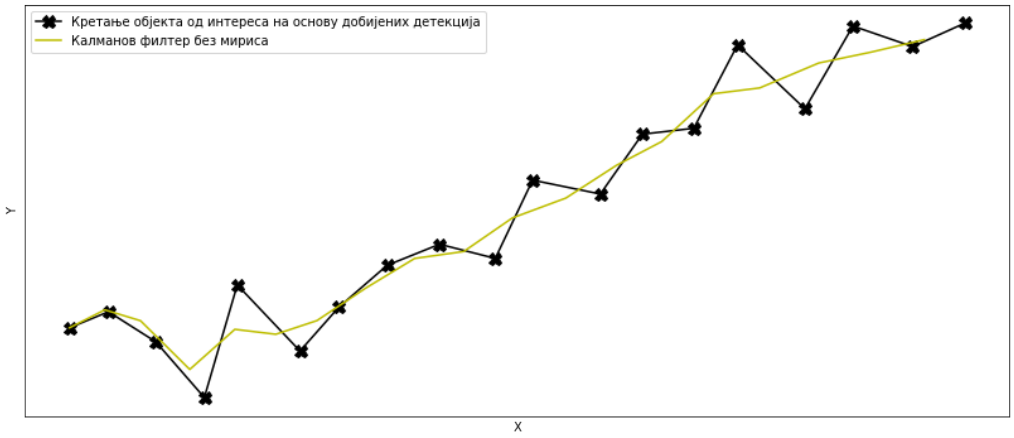
Даље, за примену Калмановог филтера без мириса потребно је још одредити сигма тачке и вредности параметара α, β и k. Симетричан скуп сигма тачака дат је следећим једначинама:

$$\mathbf{x}^{(0)} = \bar{\mathbf{x}}, \mathbf{w}^{(0)} = \frac{\lambda}{n_x + \lambda}$$

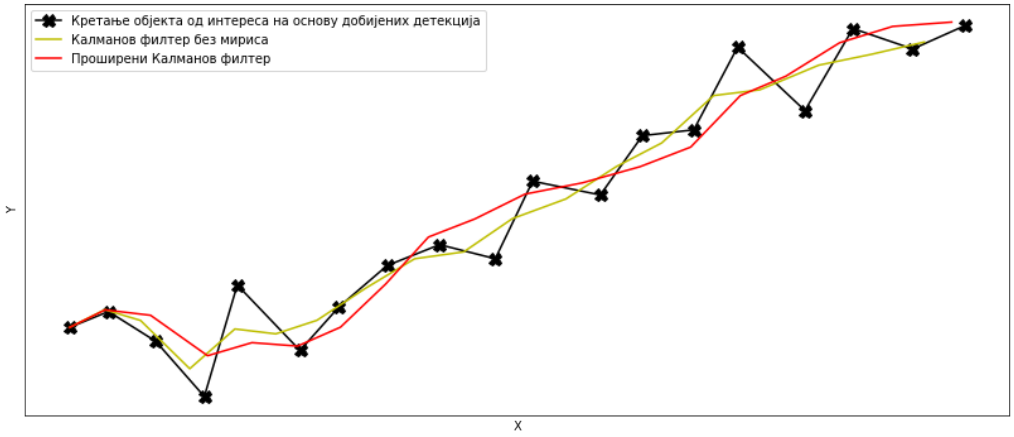
$$\mathbf{x}^{(i)} = \bar{\mathbf{x}} + [\sqrt{(n_x + \lambda)\mathbf{P}_{xx}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n_x + \lambda)} \quad \forall i = 1, ..., n_x$$

$$\mathbf{x}^{(i)} = \bar{\mathbf{x}} + [\sqrt{(n_x + \lambda)\mathbf{P}_{xx}}]_i, \mathbf{w}^{(i)} = \frac{1}{2(n_x + \lambda)} \quad \forall i = n_x + 1, ..., 2n_x$$

За вредности параметра α, β и k, које уједно одређују параметар λ, одабране су најчешће биране вредности, односно 10^3, 2 и 0 (више о томе у одељку 3.3 о Калмановом филтеру без мириса). Слика 4.21 представља упоредни приказ путање кретања добијене на основу претходних детекција без примене алгоритма за оцену стања система и путање кретања добијене на основу претходних детекција применом Калмановог филтера без мириса. Ради прегледности и лакшег поређења, на слици 4.22 дати су упоредни прикази путање кретања добијени на основу детекција без и са применом алгоритама за оцену стања система.



Слика 4.21: Упоредни приказ кретања

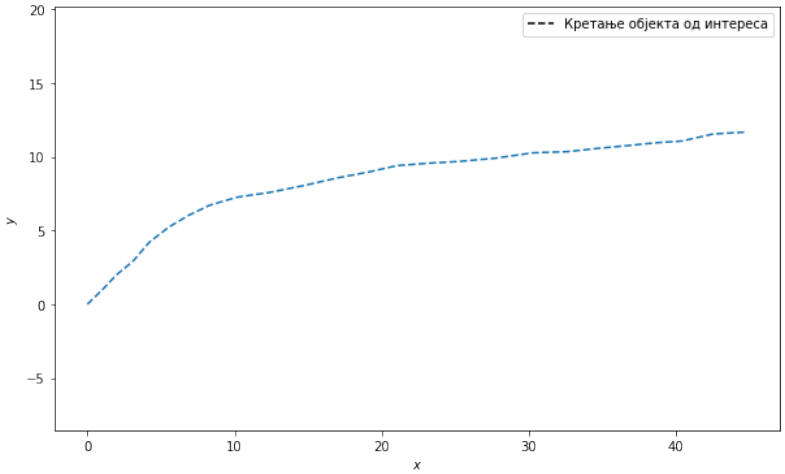


Слика 4.22: Упоредни приказ кретања

4.4 Моделовање кретања циља и примена Калмановог филтера - библиотека „*Stone Soup*”

Методе за праћење и процену стања система примењују се у различитим доменима, као што су на пример: астрономија, надгледање поморских и ваздушних зона, биологија, роботска визија, итд. Алгоритми праћења и процене стања система постају све сложенији, те је истраживачима, као и искусним практичарима, тешко да примене и системски процене ове најсавременије алгоритме. Перформансе алгоритама је потребно објективно проценити, у складу са оперативним захтевима. Како су алати за прикладно извођење оваквих процена недостајали, препознајући проблем, креирана је библиотека отвореног кода, отворена за Пајтон кориснике, под називом „*Stone Soup*”. Библиотека „*Stone Soup*” корисницима пружа могућност развоја, демонстрирања, као и тестирања алгоритама за праћење и процену стања система. На даље, приказаћемо пример употребе ове библиотеке у примени проширеног Калмановог филтера и Калмановог филтера без мириса на нелинеарне моделе.

Претпоставимо једноставан сценарио у коме се објекат од интереса креће приближно консантном брзином. На слици 4.23 приказана је стварна путања кретања циља.



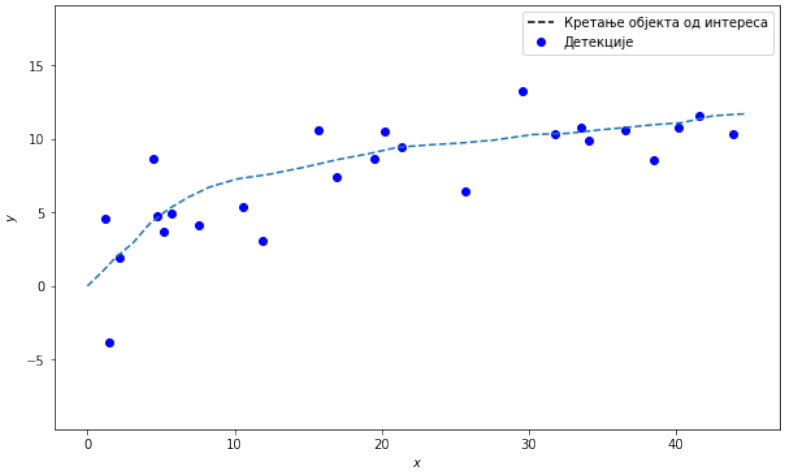
Слика 4.23: Упоредни приказ кретања

Креирани модел представљаће модел радара који мери удаљеност објекта од интереса, пружајући информације о његовим поларним координатама (ρ, ϕ), и важи следеће:

$$\begin{bmatrix} \rho_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} \sqrt{x - {x_r}^2 + y - {y_r}^2} \\ arctan(\frac{y-y_r}{x-x_r}) \end{bmatrix}$$

где су (x, y) координате циља у катерзијанским координатама, а (x_r, y_r) координате радара у Декартовим координатама.

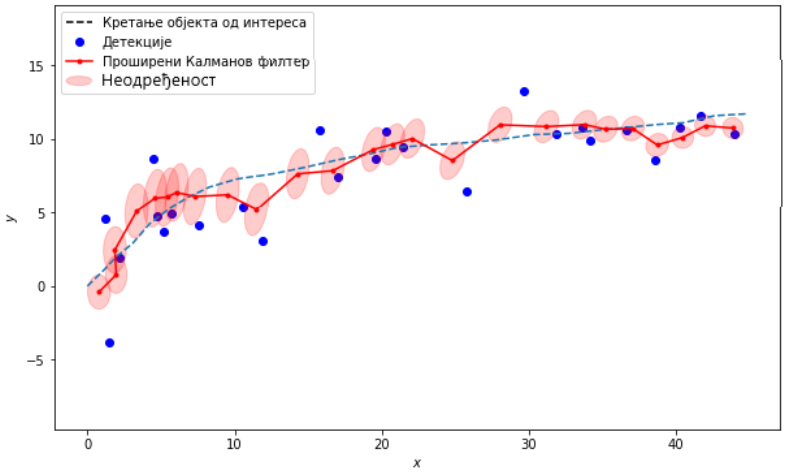
Прецизност мерења радара, тј. дисперзија мерења, по углу је $0.3°$, а по удаљености $1\ m$. Користећи дати модел, уз помоћ библиотеке „*Stone Soup*", креирамо сет детекција за горе наведени објекат од интереса. На слици 4.24 дат је приказ детекција придружених претходно поменутом објетку.



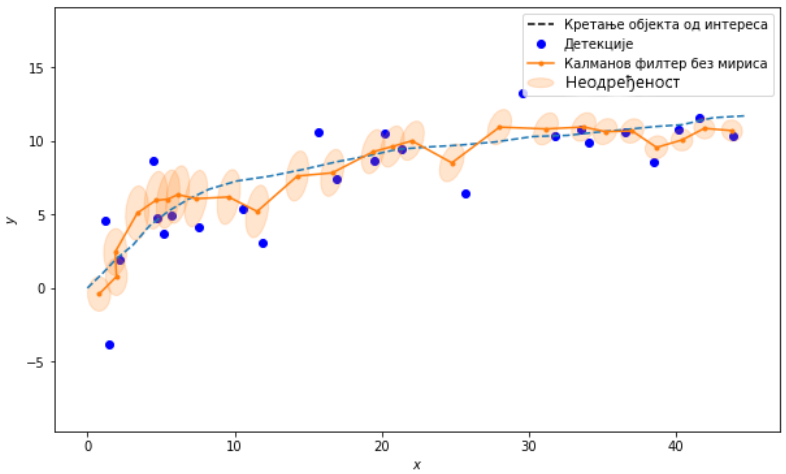
Слика 4.24: Упоредни приказ кретања

Користећи већ уграђене функције библиотеке „*Stone Soup*" на једноставан начин можемо да креирамо и применимо предикциони и корекциони корак проширеног Калмановог филтера, као и Калмановог филтера без мириса, на претходне податке (код 6.5). На крају, на сликама 4.25 и 4.26 дати су упоредни прикази стварног кретања

циља и кретања циља на основу детекција применом проширеног Калмановог филтера, односно Калмановог филтера без мириса. Такође, на сликама су приказане и елипсе грешака за сваку од оцена стања.



Слика 4.25: Упоредни приказ кретања



Слика 4.26: Упоредни приказ кретања

На основу претходних упоредних приказа, можемо закључити да су перформансе проширеног Калмановог филтера и Калмановог филтера без мириса готово исте. Најчешће, као што је случај у овом примеру, уколико су нивои процесног шума и шума мерења значајно велики, проширени Калманов филтер и Калманов филтер без мириса ће давати готово истоветне резултате. Ипак, уколико то није случај, Калманов филтер без мириса може надмашити перформансе проширеног Калмановог филтера, нарочито у случајевима праћење и оцене стања система помоћу фузије више сензора [23].

Глава 5

Закључак

У овом раду имплементиране су различите верзије Калмановог филтера, а то су: традиционални Калманов филтер, проширени Калманов филтер и Калманов филтер без мириса. Анализиране су перформансе сваког од њих на примерима праћења покретног циља без маневара, на основу радарских детекција. На темељу упоредних приказа и табеле средње квадратних грешака позиције брода добијених из детекција без примене или са применом различитих верзија Калмановог филтера у првом примеру, могу се уочити знатно бољи резултати у случају коришћења модификованих верзија Калмановог филтера, тј. проширеног Калмановог филтера и Калмановог филтера без мириса. Оцењено кретање је, у том случају, изимно углађеније и у великој мери прати стварну криву (Слика 4.12). Проширени Калманов филтер поседује ограничења о којима је већ било речи, а која се тичу постојања Јакобијеве или Хесијан матрице, као и тачности оцене. Значајан број радова, наведених у литератури, истиче предности Калмановог филтера без мириса у односу на проширени Калманов филтер. Ипак, у случају Калмановог филтера без мириса, приметно је да се не може рећи да постоји предност у рачунарским операцијама, али, с обзиром на велику брзину данашњих рачунара, ово не може бити пресудни критеријум приликом избора алгоритма за оцену стања система.

Глава 6

Кодови

Код 6.1: 2.5 Калманов филтер кроз пример

```
# Temperatura u akvarijumu - Prvi slucaj

import numpy as np
import matplotlib.pyplot as  plt


n_iter=60
size_=(n_iter,)

# stvarna temperatura je konstantna
x=np.full(size_,22)

z=np.random.normal(x,0.1,size=size_)
x_hat=np.zeros(size_)
P=np.zeros(size_)
x_hatMinus=np.zeros(size_)
P_Minus=np.zeros(size_)
K=np.zeros(size_)
Q=np.full(size_,0.00001)
R=np.full(size_,0.001)

# pocetno stanje
x_hat[0]=20
```

```
P[0]=100

for i in range(1,n_iter):
    # Predikcioni korak Kalmanovog filtera
    x_hatMinus[i]=x_hat[i-1]
    P_Minus[i]=P[i-1]+Q[i-1]

    # Korekcioni korak Kalmanovog filtera
    K[i]=P_Minus[i]/(R[i] + P_Minus[i]) # Kalmanovo pojacanje
    x_hat[i]=x_hatMinus[i]+K[i]*(z[i]-x_hatMinus[i])
    P[i]=(1- K[i])*P_Minus[i]



plt.figure()
plt.figure(figsize=(10,10))
plt.plot(z,'k+',label='Merenje temperature u akvarijumu')
plt.plot(x_hat,'b-',label='Ocenjene vrednosti temperature u akvarijumu')
plt.plot(x,color='g',label='Tacna vrednost temperature u akvarijumu')
plt.xlabel('Vremenski korak')
plt.ylabel('Temperatura [$^\circ$C)')
plt.legend()
plt.ylim(20, 24)

# Temperatura u akvarijumu - Drugi slucaj

import numpy as np
import matplotlib.pyplot as  plt

n_iter=60
size_=(n_iter,)
# stvarna temperatura je konstantna sa malim varijacijama
x=np.zeros(size_)
x[0]=22
for i in range(1,n_iter):
    x[i]=22+np.random.normal(0,0.01)
```

```
z=np.random.normal(x,0.1,size=size_) # Simulacija merenja
x_hat=np.zeros(size_)
P=np.zeros(size_)
x_hatMinus=np.zeros(size_)
P_Minus=np.zeros(size_)
K=np.zeros(size_)
Q=np.full(size_,0.00001)
R=np.full(size_,0.001)

# pocetno stanje
x_hat[0]=20
P[0]=100



for i in range(1,n_iter):
    # Predikcioni korak Kalmanovog filtera
    x_hatMinus[i]=x_hat[i-1]
    P_Minus[i]=P[i-1]+Q[i-1]

    # Korekcioni korak Kalmanovog filtera
    K[i]=P_Minus[i]/(R[i] + P_Minus[i]) # Kalmanovo pojacanje
    x_hat[i]=x_hatMinus[i]+K[i]*(z[i]-x_hatMinus[i])
    P[i]=(1- K[i])*P_Minus[i]



plt.figure()
plt.figure(figsize=(10,10))
plt.plot(z,'k+',label='Merenje temperature u akvarijumu')
plt.plot(x_hat,'b-',label='Ocenjene vrednosti temperature u akvarijumu')
plt.plot(x,color='g',label='Tacna vrednost temerature u akvarijumu')
plt.legend()
plt.xlabel('Vremenski korak')
plt.ylabel('Temperatura $^\circ$C)')
plt.ylim(20, 24)
```

```
# Temperatura u akvarijumu - Treci slucaj

import numpy as np
import matplotlib.pyplot as  plt

n_iter=60
size_=(n_iter,)
# stvarna temperatura se povecava za 0.1 svake sekunde
x=np.zeros(size_)
x[0]=21
for i in range(1,n_iter):
    x[i]=x[i-1]+0.1

z=np.random.normal(x,0.1,size=size_) # Simulacija merenja
x_hat=np.zeros(size_)
P=np.zeros(size_)
x_hatMinus=np.zeros(size_)
P_Minus=np.zeros(size_)
K=np.zeros(size_)
Q=np.full(size_,0.00001)
R=np.full(size_,0.001)

# pocetno stanje
x_hat[0]=21
P[0]=100



for i in range(1,n_iter):
    # Predikcioni korak Kalmanovog filtera
    x_hatMinus[i]=x_hat[i-1]
    P_Minus[i]=P[i-1]+Q[i-1]

    # Korekcioni korak Kalmanovog filtera
    K[i]=P_Minus[i]/(R[i] + P_Minus[i]) # Kalmanovo pojacanje
    x_hat[i]=x_hatMinus[i]+K[i]*(z[i]-x_hatMinus[i])
    P[i]=(1- K[i])*P_Minus[i]
```

```
plt.figure()
plt.figure(figsize=(10,10))
plt.plot(z,'k+',label='Merenje temperature u akvarijumu')
plt.plot(x_hat,'b-',label='Ocenjene vrednosti temperature u akvarijumu')
plt.plot(x,color='g',label='Tacna vrednost temperature u akvarijumu')
plt.legend()
plt.xlabel('Vremenski korak')
plt.ylabel('Temperatura $^\circ$C)')
plt.ylim(20, 24)


# Temperatura u akvarijumu - Cetvrti slucaj

import numpy as np
import matplotlib.pyplot as  plt

n_iter=60
size_=(n_iter,)
# temperatura se povecava za 0.1 svake sekunde
x=np.zeros(size_)
x[0]=21

for i in range(1,n_iter):
    x[i]=x[i-1]+0.1
z=np.random.normal(x,0.1,size=size_) # Simulacija merenja
x_hat=np.zeros(size_)
P=np.zeros(size_)
x_hatMinus=np.zeros(size_)
P_Minus=np.zeros(size_)
K=np.zeros(size_)
Q=np.full(size_,0.00001)
R=np.full(size_,0.001)

# pocetno stanje
```

```
x_hat[0]=20
P[0]=100



for i in range(1,n_iter):
    # Predikcioni korak Kalmanovog filtera
    x_hatMinus[i]=x_hat[i-1]+0.1
    P_Minus[i]=P[i-1]+Q[i-1]

    # Korekcioni korak Kalmanovog filtera
    K[i]=P_Minus[i]/(R[i] + P_Minus[i]) # Kalmanovo pojacanje
    x_hat[i]=x_hatMinus[i]+K[i]*(z[i]-x_hatMinus[i])
    P[i]=(1- K[i])*P_Minus[i]



plt.figure()
plt.figure(figsize=(10,10))
plt.plot(z,'k+',label='Merenje temperature u akvarijumu')
plt.plot(x_hat,'b-',label='Ocenjene vrednosti temeprature u akvarijumu')
plt.plot(x,color='g',label='Tacna vrednost temperature u akvarijumu')
plt.legend()
plt.xlabel('Vremenski korak')
plt.ylabel('Temperatura $^\circ$C)')
plt.ylim(20, 24)

# Temperatura u akvarijumu - Peti slucaj

import numpy as np
import matplotlib.pyplot as  plt

n_iter=60
size_=(n_iter,)
# temperatura se povecava za 0.1 svake sekunde
x=np.zeros(size_)
x[0]=21
```

```
for i in range(1,n_iter):
    x[i]=x[i-1]+0.1
z=np.random.normal(x,0.1,size=size_) # Simulacija merenja
x_hat=np.zeros(size_)
P=np.zeros(size_)
x_hatMinus=np.zeros(size_)
P_Minus=np.zeros(size_)
K=np.zeros(size_)
Q=np.full(size_,0.001)
R=np.full(size_,0.001)

# pocetno stanje
x_hat[0]=20
P[0]=100



for i in range(1,n_iter):
    # Predikcioni korak Kalmanovog filtera
    x_hatMinus[i]=x_hat[i-1]
    P_Minus[i]=P[i-1]+Q[i-1]

    # Korekcioni korak Kalmanovog filtera
    K[i]=P_Minus[i]/(R[i] + P_Minus[i]) # Kalmanovo pojacanje
    x_hat[i]=x_hatMinus[i]+K[i]*(z[i]-x_hatMinus[i])
    P[i]=(1- K[i])*P_Minus[i]



plt.figure()
plt.figure(figsize=(10,10))
plt.plot(z,'k+',label='Merenje temperature u akvarijumu')
plt.plot(x_hat,'b-',label='Ocenjene vrednosti temperature u akvarijumu')
plt.plot(x,color='g',label='Tacna vrednost temperature u akvarijumu')
plt.legend()
plt.xlabel('Vremenski korak')
plt.ylabel('Temperatura [$^\circ$C)')
```

```
plt.ylim(20, 24)
```

Код 6.2: 4.2 Примена Калмановог филтера на први случај

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 2*np.pi, 100)
T = t[1] - t[0]

# Pozicija
x = 2*np.cos(t)
y = np.sin(2*t)

# Brzina
vx = -2*np.sin(t)
vy = 2*np.cos(2*t)

# Ubrzanje
ax = -2*np.cos(t)
ay = -4*np.sin(2*t)

# Trzaj
jx = 2*np.sin(t)
jy = -8*np.cos(2*t)


# Kretanje objekta od interesa

plt.figure(figsize=(14,6))
plt.scatter(x[0], y[0], s=100,c='red',label="Pocenta pozicija")
plt.plot(x, y,'.b-',label="Kretanje objekta od interesa")
plt.grid()
plt.legend()
plt.show()


# Detekcije priduzene objektu od interesa
```

```
# brzina

lin_vel= np.sqrt(vx**2 + vy**2)

# ugaona brzina

ang_vel = (vx*ay - vy*ax) / (vx**2 + vy**2)

# greske merenja

pos_sig = 0.1 # stavi sa ovim i stavi sa 0.1 i 0.2 obavezno
ang_sig = 0.1
lin_sig = 0.1 # stavi sa ovim i stavi sa 0.1 i 0.2 obavezno
# kad su ovi sa 0.2 videces vecu nesigurnost

# detekcije
x_det = x + pos_sig * np.random.randn(*x.shape)
y_det = y + pos_sig * np.random.randn(*y.shape)
lin_det = lin_vel +  lin_sig * np.random.randn(*lin_vel.shape)
ang_det = ang_vel +  ang_sig * np.random.randn(*ang_vel.shape)


# Uporedni prikaz stvarnog kretanja objekta
#i kretanja objekta na osnovu dobijenih detekcija

plt.figure(figsize=(14,6))
plt.plot(t, lin_vel,'.b-',label="Vrednosti linearne brzine")
plt.plot(t, lin_det,'xk',
label="Vrednosti linearne brzine na osnovu detekcija")
plt.xlabel("Vreme")
plt.ylabel("Linearna brzina")
plt.grid()
plt.legend()
plt.show()

# Uporedni prikaz stvarnog kretanja objekta
```

```
#i kretanja objekta na osnovu dobijenih detekcija

plt.figure(figsize=(14,6))
plt.plot(x, y, '.b-',label="Pozicija objekta od interesa")
plt.plot(x_det, y_det,'xk',label="
Pozicija objekta od interesa na osnovu detekcija")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

# Uporedni prikaz stvarnog kretanja objekta
#i kretanja objekta na osnovu dobijenih detekcija

plt.figure(figsize=(14,6))
plt.plot(x, y, '.b-',label=
"Kretanje objekta od interesa")
plt.plot(x_det, y_det,'xk-',label=
"Kretanje objekta od interesa na osnovu dobijenih detekcija")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

# Uporedni prikaz stvarnog kretanja objekta
# i kretanja objekta na osnovu dobijenih detekcija

plt.figure(figsize=(14,6))
plt.plot(t, ang_vel,'.b-',label="Vrednosti ugaone brzine")
plt.plot(t, ang_det,'xk',
label="Vrednosti ugaone brzine na osnovu detekcija")
plt.xlabel("Vreme")
plt.ylabel("Ugaona brzina")
plt.grid()
plt.legend()
```

```
plt.show()

# Tradicionalni Kalmanov filter

# Matrica prelaska

F = np.array([
    [1, T, (T**2)/2, 0, 0, 0],
    [0, 1, T, 0, 0, 0],
    [0, 0, 1, 0, 0, 0],
    [0, 0, 0, 1, T, (T**2)/2],
    [0, 0, 0, 0, 1, T],
    [0, 0, 0, 0, 0, 1],
    ])

# Kovarijaciona matrica procesnog suma
# Napomena: Ista je za svaki vremenski trenutak

Q1 = np.array([(T**3)/6, (T**2)/2, T, 0, 0, 0])
Q1 = np.expand_dims(Q1, 1)
Q2 = np.array([0, 0, 0, (T**3)/6, (T**2)/2, T])
Q2 = np.expand_dims(Q2, 1)

j_var = max(np.var(jx), np.var(jy))

Q = j_var * (np.dot(Q1,Q1.T) + np.dot(Q2,Q2.T))

# Matrica merenja
# Prvi slucaj: dobijamo samo infomaciju o poziciji!

H = np.array([
    [1, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0],
    ])

# Kovarijaciona matrica suma merenja
```

```
R = np.diag(np.array([pos_sig**2, pos_sig**2]))

# Pocetno stanje
# Napomena: Ista je za svaki vremenski trenutak

x_initial = np.array([x[0], vx[0], ax[0], y[0], vy[0], ay[0]])
P_initial = 0.05 * np.eye(x_initial.shape[0])

#Tradicionalni Kalmanov filter


# vektor merenja

detections = np.array([x_det, y_det]).T

# dimenzije matrica
nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija


I = np.eye(nx) # jedinicna matrica
x_pred = np.zeros((n, nx))       # predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))  # apriorna kovarijansa greske ocene
x_corKF = np.zeros((n, nx))        # korekcija vektora stanja
P_corKF = np.zeros((n, nx, nx))   # aposteriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))        # Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial

# Za svaki vremenski trenutak za koji imamo
#merenja primenjujemo jednacine predikcionog
# i korekcionog koraka Kalmanovog filtera

for i in range(n):
```

```
    # Predikcioni korak
    if i > 0:
        x_pred[i] = np.dot(F,x_corKF[i-1])
        P_pred[i] = np.dot(np.dot(F,P_corKF[i-1]),F.T) + Q
    # Korekcioni korak
    z=detections[i]
    K[i] = np.dot(np.dot(P_pred[i],H.T),
    np.linalg.inv(np.dot(H,np.dot(P_pred[i],H.T)) + R))
    x_corKF[i] = x_pred[i] + np.dot(K[i],(z - np.dot(H,x_pred[i])))
    P_corKF[i] = np.dot(I - np.dot(K[i],H),P_pred[i])


plt.figure(figsize=(14,6))
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.r-',
label="Tradicionalni Kalmanov filter")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'kx-',label="Detekcije")
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.r-',
label="Tradicionalni Kalmanov filter")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-'
,label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'xk-',
```

```
label="Kretanje objekta od interesa na osnovu detekcija")
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.r-',
label="Tradicionalni Kalmanov filter")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',label="Kretanje objekta od interesa")
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.r-',
label="Tradicionalni Kalmanov filter")
plt.grid()
plt.legend()
plt.show()

# Novi slucaj, kada su nam date i brzina i ugaona brzina

# nelinarna funkcija h

def func_h(x_pred):


    x, vx, ax, y, vy, ay = x_pred

    # ugaona brzina
    ang = (vx*ay - vy*ax) / (vx**2 + vy**2)

    # linearna brzina
    lin = np.sqrt(vx**2 + vy**2)

    return np.array([x, y, ang, lin])

def eval_Jacobian(x_pred):


    x, vx, ax, y, vy, ay = x_pred
```

```
    # parcijalni derivati ugaone brzine
    angdvx = ((vx**2 + vy**2)*ay - 2*vx*(vx*ay-vy*ax)) /
    ((vx**2 + vy**2)**2)
    angdax = -vy / (vx**2 + vy**2)
    angdvy = (-(vx**2 + vy**2)*ax - 2*vy*(vx*ay-vy*ax)) /
   ((vx**2 + vy**2)**2)
    angday = vx / (vx**2 + vy**2)

    # parcijalni derivati linearne brzine
    lindvx = vx / np.sqrt(vx**2 + vy**2)
    lindvy = vy / np.sqrt(vx**2 + vy**2)

    Jacobian=np.array([
        [1, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0],
        [0, angdvx, angdax, 0, angdvy, angday],
        [0, lindvx, 0, 0, lindvy, 0],
        ])

    return Jacobian

# Kovarijaciona matrica suma merenja

R = np.diag(np.array([pos_sig**2, pos_sig**2, ang_sig**2, lin_sig**2]))

# Prosireni Kalmanov filter

# vektor merenja
detections = np.array([x_det, y_det, ang_det, lin_det]).T

# dimenzije matrica
nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija


I = np.eye(nx) # jedinicna matrica
```

```
x_pred = np.zeros((n, nx))       # predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))   # apriorna kovarijansa greske ocene
x_corEKF = np.zeros((n, nx))       # korekcija vektora stanja
P_corEKF = np.zeros((n, nx, nx))   # aposteriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))         # Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial

# for each time-step...
for i in range(n):
    # predikcioni korak
    if i > 0:
        x_pred[i] = np.dot(F,x_corEKF[i-1])
        P_pred[i] = np.dot(np.dot(F,P_corEKF[i-1]),F.T) + Q

    # Korekcioni korak
    z_det = detections[i]
    H_Jac = eval_Jacobian(x_pred[i])
    K[i] = np.dot(np.dot(P_pred[i],H_Jac.T),
     np.linalg.inv(np.dot(H_Jac,np.dot(P_pred[i],H_Jac.T)) + R))
    x_corEKF[i] = x_pred[i] + np.dot(K[i],(z_det - func_h(x_pred[i])))
    P_corEKF[i] = np.dot(I - np.dot(K[i],H_Jac),P_pred[i])

plt.figure(figsize=(14,6))
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',
label="Prosireni Kalmanov filter")
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()


plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-'
```

```
,label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'kx-',label="Detekcije")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',label="EKF")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',
label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'kx',label="Detekcije")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',label="EKF")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',
label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'xk-',
label="Kretanje objekta od interesa na osnovu detekcija")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',label="EKF")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',
label="Kretanje objekta od interesa")
plt.plot(x_corEKF[:,0],x_corEKF[:, 3],'.r-',label="EKF")
plt.grid()
plt.legend()
plt.show()
```

```
plt.figure(figsize=(14,6))
plt.plot(x, y, '.b-',l
abel="Kretanje objekta od interesa")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',label="EKF")
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.g-',
label="Tradicionalni Kalmanov filter")
plt.grid()
plt.legend()
plt.show()

pip install filterpy;

import filterpy ;

def func_f(x_pred, T):

    F = np.array([
    [1, T, (T**2)/2, 0, 0, 0],
    [0, 1, T, 0, 0, 0],
    [0, 0, 1, 0, 0, 0],
    [0, 0, 0, 1, T, (T**2)/2],
    [0, 0, 0, 0, 1, T],
    [0, 0, 0, 0, 0, 1],
    ])

    return np.dot(F, x_pred)

def func_h(x_pred):


    x, vx, ax, y, vy, ay = x_pred

    # ugaona brzina
    ang = (vx*ay - vy*ax) / (vx**2 + vy**2)

    # linearna brzina
```

```
    lin = np.sqrt(vx**2 + vy**2)

    return np.array([x, y, ang, lin])

detections = np.array([x_det, y_det, ang_det, lin_det]).T

# dimenzije matrica
nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija

points = kalman.MerweScaledSigmaPoints(nx, alpha=0.001, beta=2., kappa=1)

kf = kalman.UnscentedKalmanFilter(dim_x=nx, dim_z=ny, dt=T,
 fx=func_f, hx=func_h, points=points)
kf.x=x_initial
kf.P=P_initial
kf.R=R
kf.Q=Q
x_corUKF = np.zeros((n, nx))

for i in range(n):
    kf.predict();
    kf.update(detections[i])
    x_corUKF[i,:]=kf.x.copy()

plt.figure(figsize=(14,6))
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',label="UKF")
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'xk-',
```

```
label="Kretanje objekta od interesa na osnovu detekcija")
plt.plot(x_corUKF[:, 0],x_corUKF[:, 3],'.y-',label="UKF")
plt.grid()
plt.legend()
plt.show()

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,3][i]-y[i])*(x_corKF[:,3][i]-y[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,0][i]-x[i])*(x_corKF[:,0][i]-x[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corEKF[:,0][i]-x[i])*(x_corEKF[:,0][i]-x[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,0][i]-x[i])*(x_corUKF[:,0][i]-x[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corEKF[:,3][i]-y[i])*(x_corEKF[:,3][i]-y[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
```

```
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,3][i]-y[i])*(x_corUKF[:,3][i]-y[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(detections[:,1][i]-y[i])*(detections[:,1][i]-y[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,1][i]-vx[i])*(x_corKF[:,1][i]-vx[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corEKF[:,1][i]-vx[i])*(x_corEKF[:,1][i]-vx[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,1][i]-vx[i])*(x_corUKF[:,1][i]-vx[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,2][i]-ax[i])*(x_corKF[:,2][i]-ax[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
```

```
    rmse=rmse+(x_corEKF[:,2][i]-ax[i])*(x_corEKF[:,2][i]-ax[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,2][i]-ax[i])*(x_corUKF[:,2][i]-ax[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,4][i]-vy[i])*(x_corKF[:,4][i]-vy[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corEKF[:,4][i]-vy[i])*(x_corEKF[:,4][i]-vy[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,4][i]-vy[i])*(x_corUKF[:,4][i]-vy[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corKF[:,5][i]-ay[i])*(x_corKF[:,5][i]-ay[i])
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corEKF[:,5][i]-ay[i])*(x_corEKF[:,5][i]-ay[i])
```

```
rmse=np.sqrt(rmse/100)
rmse

rmse=0
for i in range(0,100):
    rmse=rmse+(x_corUKF[:,5][i]-ay[i])*(x_corUKF[:,5][i]-ay[i])
rmse=np.sqrt(rmse/100)
rmse

plt.figure(figsize=(14,6))
plt.plot(x, y,'.b-',
label="Kretanje objekta od interesa")
plt.plot(x_det, y_det,'xk-',
label="Kretanje objekta od interesa na osnovu detekcija")
plt.plot(x_corKF[:, 0],x_corKF[:, 3],'.g-',
label="Tradicionalni Kalmanov filter")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 3],'.r-',
label="Prosireni Kalmanov filter")
plt.plot(x_corUKF[:, 0],x_corUKF[:, 3],'.y-',
label="Kalmanov filter bez mirisa")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

c=np.linspace(0,100,100)
plt.figure(figsize=(14,6))
plt.plot(c, vx,'.b-',label="
Stvarne vrednosti")
plt.plot(c,x_corKF[:, 1],'.g-',
label="Tradicionalni Kalmanov filter")
plt.plot(c,x_corEKF[:, 1],'.r-',
label="Prosireni Kalmanov filter")
plt.plot(c,x_corUKF[:, 1],'.y-',
abel="Kalmanov filter bez mirisa")
plt.xlabel("Vreme")
```

```
plt.ylabel("vx")
plt.grid()
plt.legend()

 c=np.linspace(0,100,100)
plt.figure(figsize=(14,6))
plt.plot(c, ax,'.b-',
label="Stvarne vrednosti")
plt.plot(c,x_corKF[:, 2],'.g-'
,label="Tradicionalni Kalmanov filter")
plt.plot(c,x_corEKF[:, 2],'.r-',
label="Prosireni Kalmanov filter")
plt.plot(c,x_corUKF[:, 2],'.y-',
label="Kalmanov filter bez mirisa")
plt.xlabel("Vreme")
plt.ylabel("ax")
plt.grid()
plt.legend()
plt.show()

c=np.linspace(0,100,100)
plt.figure(figsize=(14,6))
plt.plot(c, vy,'.b-',
label="Stvarne vrednosti")
plt.plot(c,x_corKF[:, 4],'.g-'
,label="Tradicionalni Kalmanov filter")
plt.plot(c,x_corEKF[:, 4],'.r-',label=
"Prosireni Kalmanov filter")
plt.plot(c,x_corUKF[:, 4],'.y-',
label="Kalmanov filter bez mirisa")
plt.xlabel("Vreme")
plt.ylabel("vy")
plt.grid()
plt.legend()
plt.show()

c=np.linspace(0,100,100)
```

```
plt.figure(figsize=(14,6))
plt.plot(c, ay,'.b-',
label="Stvarne vrednosti")
plt.plot(c,x_corKF[:, 5],'.g-',
label="Tradicionalni Kalmanov filter")
plt.plot(c,x_corEKF[:, 5],'.r-',
label="Prosireni Kalmanov filter")
plt.plot(c,x_corUKF[:, 5],'.y-'
,label="Kalmanov filter bez mirisa")
plt.xlabel("Vreme")
plt.ylabel("ay")
plt.grid()
plt.legend()
plt.show()
```

Код 6.3: 4.4 Примена Калмановог филтера на други случај

```
import matplotlib.pyplot as plt
import numpy as np
from math import sin, cos, sqrt

# Kreiranje potrebnih funkcija

def C2P(x, y, vx, vy, min_value = 0.0001):
 # Konverzija iz kartezijanskih koordinata u polarne

  # min_value <- minimalna vrednost koju r mora da uzme ako nije 0

  r = sqrt(x * x + y * y)
  theta = np.arctan2(y, x)



  if r < min_value:
    r, theta, v = 0, 0, 0
  else:
    v = (x * vx + y * vy) / r

  return r, theta, v
```

```
def P2C(r, theta, v):
# Konverzija iz polarnih koordinata u kartezijanske

  x, y = r * cos(theta), r * sin(theta)
  vx, vy = v * cos(theta) , v * sin(theta)

  return x, y, vx, vy

def T(t1, t2):
# Razlika izmedju dva vremenska trenutka u milisekundama

  if(t2-t1>=0) :
    return(t2-t1) / 1000000.0
  else :
    return 0


def RMSE(predictions, ground_truths): # Srednje kvadratna greska

  xrms, yrms, vxrms, vyrms = [], [], [], []

  for p, t in zip(predictions, truths):

    px, py, pvx, pvy = p.get()
    gtx, gty, gtvx, gtvy = t.get()

    xrms += [(px - gtx) * (px - gtx)]
    yrms += [(py - gty) * (py - gty)]
    vxrms += [(pvx - gtvx) * (pvx - gtvx)]
    vyrms += [(pvy - gtvy) * (pvy - gtvy)]

  X_rms,Y_rms  = sqrt(np.mean(xrms)), sqrt(np.mean(yrms))
  Vx_rms,Vy_rms = sqrt(np.mean(vxrms)), sqrt(np.mean(vyrms))

  return px, py, vx, vy
```

```
def Jacobian(x, y, vx, vy, min_value = 0.0001): # Jakobijeva matrica


  d_2 = x *x + y *y
  d = np.sqrt(d_2)

  if d_2 < min_value:

        Jac=np.matrix(np.zeros([3, 4]))

  else:

    j11 = x / d
    j12 = y / d
    j21 = -y / d_2
    j22 = x / d_2
    j31 = y * (vx * y - vy * x) / (d_2 * d)
    j32 = x * (vy * x - vx * y) / (d_2 * d)

    Jac = np.matrix([[j11, j12, 0, 0],
                     [j21, j22, 0, 0],
                     [j31, j32, j11, j12]])

  return Jac

# Klasa detekcije

import copy

class DetectionorGroundTruth:

  def __init__(self, d):
    self.timestamp = d['timestamp']
    self.name = d['name']
    self.all = d
    self.raw = []
    self.detection = []
```

```
    if self.name == 'Ground_truth':

      self.detection = [d['x'], d['y'], d['vx'], d['vy']]
      self.raw = copy.deepcopy(self.detection)

    elif self.name == 'Radar':

       x, y, vx, vy = P2C(d['r'], d['theta'], d['v'])
       self.detection = [x, y, vx, vy]
       self.raw = [d['r'], d['theta'], d['v']]

    self.all['processed_data'] = self.detection
    self.all['raw'] = self.raw

  def get_raw(self):
    return self.raw

  def get(self):
    return self.detection

  def get_timestamp(self):
    return self.timestamp

  def get_name(self):
    return self.name

# Ucitavanje podataka

def Read_data(path):

  radar_data = []
  ground_truths = []

  with open(path) as f:

    for line in f:
```

```
        data = line.split()
        if(data[0]=='R'):
                radar = DetectionorGroundTruth({
                'timestamp': int(data[4]),
                 'name': 'Radar',
                 'r': float(data[1]),
                 'theta': float(data[2]),
                   'v': float(data[3])
                   })

                truth = DetectionorGroundTruth({
                 'timestamp': int(data[4]),
                  'name': 'Ground_truth',
                  'x': float(data[5]),
                  'y': float(data[6]),
                  'vx': float(data[7]),
                  'vy': float(data[8])
                  })
                radar_data.append(radar)
                ground_truths.append(truth)

    return radar_data, ground_truths

def updateQ(T,a):

    T2 = T * T
    T3 = T * T2
    T4 = T * T3

    x, y = a

    q11 = T4/4 * x
    q13 = T3/2 * x
    q22 = T4/4 * y
    q24 = T3/2 * y
    q31 = T3/2 * x
    q33 = T2 * x
```

```
    q42 = T3/2 * y
    q44 = T2 * y


    Q = np.matrix([[q11, 0, q13, 0],
                   [0, q22, 0, q24],
                   [q31, 0, q33, 0],
                   [0, q42, 0, q44]])
    return Q

  def updateF(T):
    F[0, 2], F[1, 3] = T, T
    return F

# Ucitajmo podatke
_,ground_truth=Read_data
("C:/Users/Lenovo/Desktop/Master_rad_kodovi/.ipynb_checkpoints\Data.txt")
R = np.matrix([[0.01, 0, 0],
               [0, 1.0e-6, 0],
               [0, 0, 0.01]])
R = np.matrix([[0.09, 0, 0],
               [0, 0.0009, 0],
               [0, 0, 0.09]])
F = np.matrix(np.eye(4))
Q = np.matrix(np.zeros([4, 4]))
a=(9,9)

def func_h(x_pred):

    px, py, vx, vy = x_pred
    r, theta, v = C2P(px, py, vx, vy)
    return np.array([r,theta,v]).T

# vektor merenja

radar_detections=[]
for data in radar_data:
```

```
    radar_detections.append(data.get())
detections=np.array(radar_detections)

x_initial = detections[0]
P_initial =  np.matrix([[1, 0, 0, 0],
                [0, 1, 0, 0],
                [0, 0, 1000, 0],
                [0, 0, 0, 1000]])

nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija



I = np.eye(nx) # jedinicna matrica
x_pred = np.zeros((n, nx))       # predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))  # apriorna kovarijansa greske ocene
x_corEKF = np.zeros((n, nx))        # korekcija vektora stanja
P_corEKF = np.zeros((n, nx, nx))    # aposteriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))         # Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial
timestamp=radar_data[0].get_timestamp();

# for each time-step...
for i in range(n):
    # predikcioni korak
    if i > 0:
        x_pred[i] = np.dot(F,x_corEKF[i-1])
        P_pred[i] = np.dot(np.dot(F,P_corEKF[i-1]),F.T) + Q

    # Korekcioni korak
    z_det = radar_data[i].get_raw()
    H_Jac = Jacobian(x_pred[i][0],x_pred[i][1],x_pred[i][2],x_pred[i][3])
    K[i] = np.dot(np.dot(P_pred[i],H_Jac.T),
```

```
np.linalg.inv(np.dot(H_Jac,np.dot(P_pred[i],H_Jac.T)) + R))
    x_corEKF[i] = x_pred[i] + np.dot(K[i],(z_det - func_h(x_pred[i])))
    P_corEKF[i] = np.dot(I - np.dot(K[i],H_Jac),P_pred[i])
    if i<(n-1):
            dt=T(timestamp, radar_data[i+1].get_timestamp())
            timestamp=radar_data[i+1].get_timestamp();
            Q=updateQ(dt,a)
            F=updateF(dt)

plt.figure(figsize=(14,6))

plt.plot(detections[20:,0], detections[20:,1],'xk-',
label="Kretanje objekta na osnovu detekcija")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(detections[20:,0], detections[20:,1],'kx-',
label="Detekcije")
plt.plot(x_corEKF[20:, 0],x_corEKF[20:, 1],'.r-'
,label="EKF")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

def RMSE(predictions, truths): # Srednje kvadratna greska

  xrms, yrms, vxrms, vyrms = [], [], [], []

  for p, t in zip(predictions, truths):

    px, py, pvx, pvy = p
```

```
        gtx, gty, gtvx, gtvy = t

        xrms += [(px - gtx) * (px - gtx)]
        yrms += [(py - gty) * (py - gty)]
        vxrms += [(pvx - gtvx) * (pvx - gtvx)]
        vyrms += [(pvy - gtvy) * (pvy - gtvy)]

    X_rms,Y_rms  = sqrt(np.mean(xrms)), sqrt(np.mean(yrms))
    Vx_rms,Vy_rms = sqrt(np.mean(vxrms)), sqrt(np.mean(vyrms))

    return X_rms, Y_rms,Vx_rms, Vy_rms

RMSE(x_corEKF,real_data)[0],RMSE(x_corEKF,real_data)[1]

RMSE(detections,real_data)[0],RMSE(detections,real_data)[1]

import filterpy ;
from filterpy import kalman;

def func_f(x_pred, T):

    F = np.matrix(np.eye(4))
    F[0, 2], F[1, 3]  = T, T

    return np.dot(F, x_pred)

def func_h(x_pred):

    px, py, vx, vy = x_pred
    r, theta, v = C2P(px, py, vx, vy)
    return np.array([r,theta,v])

R = np.matrix([[0.09,0,0],
               [0, 0.00009, 0],
               [0, 0, 0.09]])
Q = np.matrix(np.zeros([4, 4]))
a=(0.2/9,4/9)
```

```
Q=updateQ(1,a)

radar_detections=[]
for data in radar_data:
    radar_detections.append(data.get())
detections=np.array(radar_detections)

x_initial = detections[0]
P_initial = 0.5*0.5 * np.eye(x_initial.shape[0])

nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija



I = np.eye(nx) # jedinicna matrica
x_pred = np.zeros((n, nx))       # predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))  # apriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))         # Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial

points = kalman.MerweScaledSigmaPoints(nx, alpha=0.0001, beta=2., kappa=0)

kf = kalman.UnscentedKalmanFilter(dim_x=nx, dim_z=ny,dt=1,
 fx=func_f, hx=func_h, points=points)

kf.x=x_initial
kf.P=P_initial
kf.R=R
kf.Q=Q
x_corUKF = np.zeros((n, nx))
P_corUKF=np.zeros((n, nx,4))
timestamp=radar_data[0].get_timestamp();
kf.update(radar_data[0].get_raw())
```

```
x_corUKF[0,:]=kf.x.copy()

for i in range(1,n):
    kf.predict();
    kf.update(radar_data[i].get_raw())
    P_corUKF[i,:]=kf.P_post
    x_corUKF[i,:]=kf.x.copy()

RMSE(x_corUKF,real_data)[0],RMSE(x_corUKF,real_data)[1]

plt.figure(figsize=(14,6))

plt.plot(detections[20:,0], detections[20:,1],'kx-',
label="Detekcije")
plt.plot(x_corUKF[20:, 0],x_corUKF[20:, 1],'.y-',
label="Kalmanov filter bez mirisa")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.legend()
plt.show()

import matplotlib.pyplot as plt
plt.figure(figsize=(14,6))
plt.plot(detections[15:25,0], detections[15:25,1],'xk'
,label="Detekcije")
plt.plot(x_corEKF[15:25, 0],x_corEKF[15:25, 1],'xr',
label="EKF")
plt.plot(x_corUKF[15:25, 0],x_corUKF[15:25, 1],'xy',
label="UKF")
plt.grid()
plt.legend()
plt.show()
```

Код 6.4: 4.5 Примена Калмановог филтера на трећи случај

```
import numpy as np
from math import sin, cos, sqrt
```

```
# Kreiranje potrebnih funkcija

def C2P(x, y, vx, vy, min_value = 0.0001):
# Konverzija iz kartezijanskih koordinata u polarne

  # min_value

  r = sqrt(x * x + y * y)
  theta = np.arctan2(y, x)


  if r < min_value:
    r, theta, v = 0, 0, 0
  else:
    v = (x * vx + y * vy) / r

  return r, theta, v

def P2C(r, theta, v):
# Konverzija iz polarnih koordinata u kartezijanske

  x, y = r * cos(theta), r * sin(theta)
  vx, vy = v * cos(theta) , v * sin(theta)

  return x, y, vx, vy

def T(t1, t2):
# Razlika izmedju dva vremenska trenutka
# u milisekundama

  if(t2-t1>=0) :
    return(t2-t1) / 1000000.0
  else :
    return 0

def Jacobian(x, y, vx, vy, min_value = 0.0001):
```

```
# Jakobijeva matrica


  d_2 = x *x + y *y
  d = np.sqrt(d_2)

  if d_2 < min_value:

        Jac=np.matrix(np.zeros([3, 4]))

  else:

    j11 = x / d
    j12 = y / d
    j21 = -y / d_2
    j22 = x / d_2
    j31 = y * (vx * y - vy * x) / (d_2 * d)
    j32 = x * (vy * x - vx * y) / (d_2 * d)

    Jac = np.matrix([[j11, j12, 0, 0],
                     [j21, j22, 0, 0],
                     [j31, j32, j11, j12]])

  return Jac

 # Klasa detekcije

import copy

class Detection:

  def __init__(self, d):
    self.name = d['name']
    self.all = d
    self.raw = []
    self.detection = []
```

```
      x, y, vx, vy = P2C(d['r'], d['theta'], d['v'])
      self.detection = [x, y, vx, vy]
      self.raw = [d['r'], d['theta'], d['v']]

      self.all['processed_data'] = self.detection
      self.all['raw'] = self.raw

  def get_raw(self):
    return self.raw

  def get(self):
    return self.detection

  def get_timestamp(self):
    return self.timestamp

  def get_name(self):
    return self.name

# Ucitavanje podataka

def Read_data(path):

  radar_data = []

  with open(path) as f:

    for line in f:
      data = line.split()
      if(data[0]=='R'):
            radar = Detection({
              'name': 'Radar',
              'r': (float(data[1])),
              'theta': (float(data[2])),
              'v': (float(data[3]))
                })
```

```
                radar_data.append(radar)

    return radar_data

def updateQ(T,a):

    T2 = T * T
    T3 = T * T2
    T4 = T * T3

    x, y = a

    q11 = T4/4 * x
    q13 = T3/2 * x
    q22 = T4/4 * y
    q24 = T3/2 * y
    q31 = T3/2 * x
    q33 = T2 * x
    q42 = T3/2 * y
    q44 = T2 * y


    Q = np.matrix([[q11, 0, q13, 0],
                   [0, q22, 0, q24],
                   [q31, 0, q33, 0],
                   [0, q42, 0, q44]])
    return Q

def updateF(T):
    F[0, 2], F[1, 3] = T, T
    return F

# Ucitajmo podatke
radar_data=Read_data("C:/Users/Lenovo/Desktop/Master␣rad
kodovi/.ipynb_checkpoints/BPBP22V.txt")

R = np.matrix([[0.9, 0, 0],
```

```
                    [0, 0.09, 0],
                    [0, 0, 0.9]])
F = np.matrix(np.eye(4))
Q = np.matrix(np.zeros([4, 4]))
a=(5,5)

def func_h(x_pred):

    px, py, vx, vy = x_pred
    r, theta, v = C2P(px, py, vx, vy)
    return np.array([r,theta,v]).T

# vektor merenja

radar_detections=[]
for data in radar_data:
    radar_detections.append(data.get())
detections=np.array(radar_detections)

x_initial = detections[0]
P_initial =  0.5*0.5 * np.eye(x_initial.shape[0])

nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija



I = np.eye(nx) # jedinicna matrica
x_pred = np.zeros((n, nx))
# predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))
# apriorna kovarijansa greske ocene
x_corEKF = np.zeros((n, nx))
# korekcija vektora stanja
P_corEKF = np.zeros((n, nx, nx))
# aposteriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))
```

```
#Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial
dt=1

# for each time-step...
for i in range(n):
    # predikcioni korak
    if i > 0:
        x_pred[i] = np.dot(F,x_corEKF[i-1])
        P_pred[i] = np.dot(np.dot(F,P_corEKF[i-1]),F.T) + Q

    # Korekcioni korak
    z_det = radar_data[i].get_raw()
    H_Jac = Jacobian(x_pred[i][0],x_pred[i][1],x_pred[i][2],x_pred[i][3])
    K[i] = np.dot(np.dot(P_pred[i],H_Jac.T),np.linalg.inv
    (np.dot(H_Jac,np.dot(P_pred[i],H_Jac.T)) + R))
    x_corEKF[i] = x_pred[i] + np.dot(K[i],(z_det - func_h(x_pred[i])))
    P_corEKF[i] = np.dot(I - np.dot(K[i],H_Jac),P_pred[i])
    if i<(n-1):
            dt=1
            Q=updateQ(dt,a)
            F=updateF(dt)

# Uporedni prikaz stvarnog kretanja objekta i
# kretanja objekta na osnovu dobijenih detekcija

import matplotlib.pyplot as plt
plt.figure(figsize=(14,6))
plt.plot(detections[:,0], detections[:,1],'Xk-',markersize=10)
plt.title("Prikaz detekcija
i kretanja objekata na osnovu njih")
plt.xlabel("X")
plt.ylabel("Y")
plt.xticks([])
```

```
plt.yticks([])
plt.grid()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(detections[:,0], detections[:,1], 'Xk-',markersize=10,
label="Kretanje objekata od interesa na osnovu detekcija")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 1], 'r-'
,label="Prosireni Kalmanov filter")
plt.xlabel("X")
plt.ylabel("Y")
plt.xticks([])
plt.yticks([])
plt.grid()
plt.legend()
plt.show()

import filterpy
from filterpy import kalman

def func_f(x_pred, T):

    F = np.matrix(np.eye(4))
    F[0, 2], F[1, 3]  = T, T

    return np.dot(F, x_pred)

def func_h(x_pred):

    px, py, vx, vy = x_pred
    r, theta, v = C2P(px, py, vx, vy)
    return np.array([r,theta,v])

R = np.matrix([[0.9,0,0],
               [0, 0.09, 0],
               [0, 0, 0.9]])
Q = np.matrix(np.zeros([4, 4]))
```

```
a=(5,5)
Q=updateQ(1,a)

radar_detections=[]
for data in radar_data:
    radar_detections.append(data.get())
detections=np.array(radar_detections)

x_initial = detections[0]
P_initial = 0.5*0.5 * np.eye(x_initial.shape[0])

nx = Q.shape[0]
ny = R.shape[0]
n = detections.shape[0] # broj iteracija



I = np.eye(nx) # jedinicna matrica
x_pred = np.zeros((n, nx))
# predikcija vektora stanja
P_pred = np.zeros((n, nx, nx))
# apriorna kovarijansa greske ocene
K = np.zeros((n, nx, ny))
# Kalmanovo pojacanje

# Pocetno stanje
x_pred[0] = x_initial
P_pred[0] = P_initial

points = kalman.MerweScaledSigmaPoints(nx, alpha=0.0001, beta=2, kappa=0)

kf = kalman.UnscentedKalmanFilter(dim_x=nx,
dim_z=ny,dt=1, fx=func_f, hx=func_h, points=points)
kf.x=x_initial
kf.P=P_initial
kf.R=R
kf.Q=Q
x_corUKF = np.zeros((n, nx))
```

```
kf.update(radar_data[0].get_raw())
x_corUKF[0,:]=kf.x.copy()
for i in range(1,n):
    kf.predict();
    kf.update(radar_data[i].get_raw())
    x_corUKF[i,:]=kf.x.copy()

plt.figure(figsize=(14,6))
plt.plot(detections[:,0], detections[:,1],'Xk-',markersize=10,
label="Kretanje objekta od interesa
na osnovu dobijenih detekcija")
plt.plot(x_corUKF[:, 0],x_corUKF[:, 1],'y-',
label="Kalmanov filter bez mirisa")
plt.plot(x_corEKF[:, 0],x_corEKF[:, 1],'r-',
label="Prosireni Kalmanov filter")
plt.xlabel("X")
plt.ylabel("Y")
plt.xticks([])
plt.yticks([])
plt.grid()
plt.legend()
plt.show()

plt.figure(figsize=(14,6))
plt.plot(detections[:,0], detections[:,1],'Xk-',markersize=10,
label="Kretanje objekata od interesa na osnovu detekcija")
plt.plot(x_corUKF[:, 0],x_corUKF[:, 1],'r-',
label="Kalmanov filter bez mirisa")
plt.xlabel("X")
plt.ylabel("Y")
plt.xticks([])
plt.yticks([])
plt.grid()
plt.legend()
plt.show()
```

Код 6.5: 4.6 Моделовање кретања циља и примена Калмановог филтера - библиотека „*Stone Soup*”

```
import numpy as np
import datetime
start_time = datetime.datetime.now()
np.random.seed(7)
import stonesoup
import datetime
from stonesoup.plotter import Plotter
from stonesoup.models.measurement.nonlinear
import CartesianToBearingRange
from stonesoup.types.detection import Detection
from stonesoup.models.transition.linear import
CombinedLinearGaussianTransitionModel,
ConstantVelocity
from stonesoup.types.groundtruth import
GroundTruthPath, GroundTruthState
# Radimo po ugledu na
https://stonesoup.readthedocs.io/en/v0.1b6/auto
_tutorials/02_ExtendedKalmanFilterTutorial.html
!
model = CombinedLinearGaussianTransitionModel([
ConstantVelocity(0.08),
ConstantVelocity(0.08)])
groundTruth =
GroundTruthPath([GroundTruthState([0, 1, 0, 1],
timestamp=start_time)])

for k in range(1, 25):
    groundTruth.append(GroundTruthState(
        model.function(groundTruth[k-1],
        noise=True,
        time_interval=datetime.timedelta(second
        s=1)),
        timestamp=start_time+datetime.timedelta
        (seconds=k)))
```

```
plotter = Plotter()
plotter.plot_ground_truths(groundTruth, [0,
2],truths_label='Kretanje objekta od interesa')
np.random.seed(1)
r_x = 50
r_y = 0
measurement_model = CartesianToBearingRange(
    ndim_state=4,
    mapping=(0, 2),
    noise_covar=np.diag([np.radians(0.3), 1]),
    translation_offset=np.array([[r_x], [r_y]])
)
np.random.seed(1)
detections = []
for state in groundTruth:
    detection
    measurement_model.function(state,
    noise=True)
    detections.append(Detection(detection,
    timestamp=state.timestamp,
    measurement_model=measurement_model))

plotter.plot_measurements(detections,
[0,2],measurements_label='Detekcije')
plotter.fig

from stonesoup.predictor.kalman import
ExtendedKalmanPredictor
from stonesoup.updater.kalman import
ExtendedKalmanUpdater
from stonesoup.types.state import GaussianState
from stonesoup.types.hypothesis import
SingleHypothesis
from stonesoup.types.track import Track
predictor = ExtendedKalmanPredictor(model)
updater = ExtendedKalmanUpdater(measurement_mo
del)
```

```
prior = GaussianState([[0], [1], [0], [1]],
np.diag([1.5, 0.5, 1.5, 0.5]),
timestamp=start_time)


from stonesoup.types.hypothesis import SingleHypothesis
from stonesoup.types.track import Track

track = Track()
for detection in detections:
    prediction = predictor.predict(prior,
    timestamp=detection.timestamp)
    hypothesis = SingleHypothesis(prediction,
    detection)  # Group a prediction and
    measurement
    post = updater.update(hypothesis)
    track.append(post)
    prior = track[-1]

plotter.plot_tracks(track, [0, 2],
uncertainty=True,track_label='Prosireni
Kalmanov filter',color='r')
plotter.fig

from stonesoup.updater.kalman import
UnscentedKalmanUpdater
from stonesoup.predictor.kalman import
UnscentedKalmanPredictor

predictor = UnscentedKalmanPredictor(model)
unscented_updater =
UnscentedKalmanUpdater(measurement_model)

prior = GaussianState([[0], [1], [0], [1]],
np.diag([1.5, 0.5, 1.5, 0.5]),
timestamp=start_time)
```

```
track = Track()
for detection in detections:
    prediction = predictor.predict(prior,
    timestamp=detection.timestamp)
    hypothesis = SingleHypothesis(prediction,
    detection)
    post = unscented_updater.update(hypothesis)
    track.append(post)
    prior = track[-1]

plotter = Plotter()
plotter.plot_ground_truths(groundTruth, [0,
2],truths_label='Kretanje objekta od interesa')

plotter.plot_measurements(detections, [0,
2],measurements_label='Detekcije')
plotter.fig

plotter.plot_tracks(track, [0, 2],
uncertainty=True,track_label='Kalmanov filter
bez mirisa')
plotter.fig
```

Библиографија

[1] Подаци за други случај. `https://github.com/PercyJaiswal/Kalman_Filter`, Jun 2021.

[2] Kalman filter. `https://www.kalmanfilter.net/background.html`, Maj 2021.

[3] S. Blackman. *Multiple-target tracking with radar applications*. Dedham, 1986.

[4] H. Masnadi-Shirazi A. Masnadi-Shirazi M. Dastgheib. *A Step by Step Mathematical Derivation and Tutorial on Kalman Filters*. arXiv, 2019.

[5] B. Ekstrand. Some aspects on filter design for target tracking. *Journal of Control Science and Engineering*, 2012.

[6] R. Merwe A. Doucet N. Fretias E.Wan. The unscented particle filter. *Advances in neural information processing systems*, 13:584–590, 2000.

[7] G.Bishop G. Welch. *An Introduction to the Kalman Filter*. University of North Carolina at Chapel Hill, 2001.

[8] J. Ishihara i G. Borges i A. Vargas H. Menegaz. A systematization of the unscented kalman filter theory. *IEEE*, 2015.

[9] P. Abbeel i A. Coates i M. Montemerlo i A. Ng. Discriminative training of kalman filters. *Robotics: Science and Systems I, June 8-11, 2005, Massachusetts Institute of Technology, Cambridge, Massachusetts*, 2005.

[10] I. Jovandić i B. Kovacević. Primena unscented transformacije u estimaciji stanja nelinearnih stohastickih sistema. *50th ETRAN Conference*, 2006.

[11] Y. Wu i D. Hu i M. Wu i X. Hu. Unscented kalman filtering for additive noise case: Augmented vs. non-augmented. *Proceedings of the American Control Conference*, 2005.

[12] J. Mališić i D. Đorić i E. Nikolić-Đorić i V. Jevremović. *Atlas raspodela*. Gradjevinski fakultet, Beograd, 2007.

[13] E. Wan i R. Merwe. The unscented kalman filter for nonlinear estimation,. *Oregon Graduate Institute of Science and Technology*, 2000.

[14] K. Reif i S. Gunther i E. Yaz i R. Unbehauen. Stochastic stability of the discrete-time extended kalman filter. *IEEE Transactions on Automatic Control*, 1999.

[15] S. Lakshmivarahan J. Lewis. Dynamic data assimilation, a least squares approach. 2006.

[16] Towards Data Science Percy Jaiswal. Sensor fusion — part 1: Kalman filter code. `https://towardsdatascience.com/sensor-fusion-part-1-kalman-filter-basics-4692a653a74c`, Jun 2021.

[17] Towards Data Science Percy Jaiswal. Sensor fusion — part 2: Kalman filter code. `https://towardsdatascience.com/sensor-fusion-part-2-kalman-filter-code-78b82c63dcd`, Jun 2021.

[18] H. Jane. Comparasion of ekf, ekf2 and ukf in a loosely coupled ins/gps integration. 2018.

[19] S. Julier. The scaled unscented transformation. *In Proc. 2002 American Control Conf.*, 2002.

[20] P. Kalata. The tracking index: A generalized parameter for alpha-beta and alpha-beta-gamma target trackers. *IEEE Transactions on Aerospace and Electronic Systems*, 1984.

[21] R. Mahler. *Statistical multisource-multitarget information fusion.* Artech House, 2007.

[22] P. Maybeck. *Stochastic models, estimation, and control.* Academic press, 1982.

[23] K. Branco N. Silva, D. Wilson. Performance evaluation of the extended kalman filter and unscented kalman filter. *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

[24] D.Stoffer R. Shumway. *Time series analysis and its applications. Vol. 3. Chapter 6.* Springer, 2000.

[25] Z. Rakić. *Geometrija 3.* Matematicki fakultet, Beograd, 2016.

[26] J. Uhlmann S. Julier. A new extension of the kalman filter to nonlinear systems. *The Robotics Research Group, Department of Engineering Science, The University of Oxford*, 1997.

[27] M. Baum S. Yang. Second-order extended kalman filter for extended object and group tracking. 2016.

[28] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* John Wiley and Sons, 2006.

[29] G. Terejanu. *Extended Kalman Filter Tutorial.* University at Buffalo, Buffalo, NY 14260.

[30] Udacity. Self-driving car nd - sensor fusion - extended kalman filters. `https://d17h27t6h515a5.cloudfront.net/topher/2017/February/58b461d5_sensor-fusion-ekf-reference/sensor-fusion-ekf-reference.pdf`, Maj 2021.

[31] V. Jilkov X. Li. Survey of maneuvering target tracking. part i: Dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 2003.

[32] H. Wu Z. Li. A survey of maneuvering target tracking using kalman filter. *4th Int. Conf. Mechatronics Mater. Chem. Comput. Eng*, 2015.

Биографија аутора

Кристина Матовић је завршила основну школу „Доситеј Обрадовић" у Београду, након чега је наставила школовање у Тринаестој београдској гимназији. Гимназију је завршила у јуну 2014. године. Математички факултет у Београду, смер Математика и модул Статистика, актуарска и финансијска математика, завршила је у фебруару 2020. године. Након тога, уписала је мастер студије на Математичком факултету, смер Математика и модул Статистика, актуарска и финансијска математика. Тренутно је запослена у институту високих технологија Влатацом.