Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Nonlinear Estimation Software Framework in Optimal and Adaptive Control Problems

Miroslav Flídr, Ondřej Straka and Miroslav Šimandl

European Centre of Excellence "NTIS - New Technologies for the Information Society", Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic, e-mail: {flidr, straka30, simandl}@ntis.zcu.cz

Abstract: The paper presents the use of a MATLAB based software framework designed for nonlinear state estimation of discrete-time dynamic systems in optimal and adaptive control problems. The main focus of the framework is to facilitate implementation, testing and use of various nonlinear state estimation methods. Nevertheless, due to its versatility, the framework is also suitable for adaptive control purposes. The designer of the controller can utilize any of the large number of offered estimators which provide the necessary state and parameter estimates in the form of conditional probability density function. The paper presents the possibilities of the toolbox usage in optimal and adaptive control problems. It will be demonstrated how easily and naturally an estimation problem can be described by means provided by the framework and how easily it can be fitted into and controller.

Keywords: Nonlinear estimation, optimal control, adaptive control, software framework

1. INTRODUCTION

The problem of control of the stochastic systems is tightly coupled with the problem of state and parameter estimation. In order to be able to properly fulfill the control goal, it is necessary to obtain sufficient characteristics of the unknown state and parameters. Generally, these two problems are inevitably interwoven since the solution of one of the problems may have significant influence on the quality of the solution of the other problem (Feldbaum, 1965). Thus, any controller dealing with uncertainties in state and parameters incorporates a suitable estimator.

Nonlinear estimation of discrete-time stochastic dynamic systems is a rapidly developing field of study. In the last decade it has undergone rapid development of various new estimation methods. All those methods strive to overcome a problem with closed form solvability of the general solution to state estimation problem which can be found by means of Bayesian functional relations (BFRs) (Candy, 2008).

The development has proceeded in both basic categories of estimation methods differentiated by the validity of the provided estimates, i.e. the global and local methods. The global methods such as the Gaussian sum method (Straka and Šimandl, 2005), the point-mass method (Šimandl et al., 2006), and most notably the particle filter (Doucet et al., 2001) provide state estimates, which are valid in almost whole state space, in the form of a probability density function (PDF). However, their high computational costs often hinder their practical use.

The local methods are usually based on an approximation of the nonlinear functions in the state or measurement equation or on an approximate computation of first two moments of a nonlinearly transformed random variable so that the Kalman filter (KF) (Grewal and Andrews, 2001) design technique can be used to obtain the estimate. This rough approximation of the posterior estimates induces local validity of the provided point state estimates. As the concept is based on the adoption of the structure of the KF, it is referred to as Kalman filtering framework (KFF).

The traditionally widely used Taylor series based linearization was gradually replaced by stochastic and polynomial linearizations or by use of quadrature or cubature integration rules (Ito and Xiong, 2000; Arasaratnam and Haykin, 2009). The main advantage of these new techniques over the traditional ones is that they do not require evaluation of the Jacobi matrix of the nonlinear functions in the state and measurement equations of the model (Šimandl and Duník, 2009). The idea of the stochastic linearization is to approximate a random variable by a set of points which are transformed through nonlinear functions (i.e. functions in the model) (Julier and Uhlmann, 2004; Ito and Xiong, 2000; Duník et al., 2005). The unscented transform (UT) used in the unscented Kalman filter (UKF) belongs to stochastic linearization methods (Lefebvre et al., 2002). The idea of the polynomial linearization is to approximate a nonlinear function by a first or second order polynomial interpolation (Ito and Xiong, 2000; Nørgaard et al., 2000). The divided difference filter (DDF) is the main representative of the filters utilizing the polynomial linearization. Recently, several nonlinear filters have been proposed based on numerical calculation of the predictive statistics within the KFF. They use Gauss-Hermite quadrature (Ito and Xiong, 2000) or cubature integration rules (Arasaratnam and Haykin, 2009). Duník et al. (2013) have proposed a new filter called the stochastic integration filter (SIF). Its core is the stochastic integration rule which possesses some favorable properties as opposed to the UT.

This vast number of different state estimation methods can make the choice of a suitable estimation technique for the control purposes quite burdensome. All their theoretical demands,

⋆ This work was supported by the European Regional Development Fund (ERDF), project "NTIS New Technologies for the Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090 and by the Ministry of Education, Youth and Sports project INGO II (LG14039).

the number of design parameters and lack of recommendations, which could help the user who looks for a suitable estimation method for his control task, call for the employment of a suitable software package. A desirable package should ease the burden of choosing and implementing the right estimation technique. The control method designer should be able to chose the estimation technique at will and concentrate on the main task, i.e. control. Such package should act as a toolbox that implements various estimation methods, facilitate their use and makes it easy to setup the estimator for the control purposes.

The Nonlinear Estimation Framework (NEF) (Straka et al., 2009; Straka et al., 2010; Flídr et al., 2013) can be seen as a candidate. It serves as a supporting platform for developing and testing various estimation techniques. It provides implementation of many of the traditional and modern estimation techniques and supports processing of the filtering, prediction, and fixed-lag smoothing tasks. It can also be easily employed for implementation of optimal and adaptive controllers. It was successfully used for numerical simulation of suboptimal adaptive dual controllers for systems with both parameter and functional uncertainties (Flídr and Šimandl, 2011, 2013; Král and Šimandl, 2011, 2013).

The main goal of this paper is thus twofold, to present the Nonlinear Estimation Framework (NEF) software toolbox and to demonstrate its use in two control examples.

The structure of the paper is as follows. Section 2 is devoted to a brief introduction of the structure, the provided components and possibilities of the NEF toolbox. Afterwards, in Section 3, a demonstration of the framework possibilities will be presented in two examples. The paper is concluded by Section 4.

2. INTRODUCTION TO THE NONLINEAR ESTIMATION FRAMEWORK

The Nonlinear Estimation Framework is a freely available [1] MATLAB toolbox. The toolbox is comprised of components representing all the necessary pieces for estimation experiment description, processing of the data and performance evaluation (see Fig. 1). These components provide set of classes which instances are used for the problem description, choice and parametrization of estimator and evaluation of the results. The design of the toolbox facilitates a straightforward and natural experiment setup for casual user, however, at the same time it provides a powerful tool with full control in the hands of advanced users. The toolbox is also easily extensible due to the use of object oriented programming methodology.

The key features of the NEF are

- structural and probabilistic modeling,
- support for time-varying models,
- support for filtering, multi-step prediction and fixed-lag smoothing,
- implementation of both standard and numerically stable estimation algorithms,
- full estimator parametrization by means of the standard MATLAB property-value mechanism,
- evaluation of estimate quality.

The remainder of this section is devoted to the description of the core components.

[1] The toolbox is downloadable upon registration at `http://nft.kky.zcu.cz/nef`.

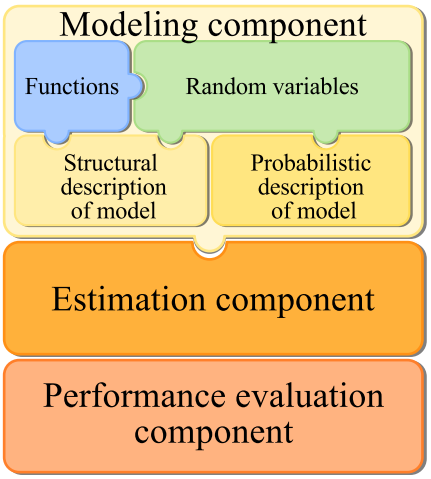

Fig. 1. Nonlinear Estimation Framework components

2.1 Components for Problem Description

In order to describe the estimation experiment at hand, the toolbox has to provide means for proper description of functions and random variables which can than be used for model description. The resulting model description can be afterwards employed in the estimator itself (see the Fig. 2).

The toolbox provides two model classes `nefEqSystemn` and `nefPDFSystem` for structural and probabilistic description of time-varying models, respectively. The structural model description is given as follows

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), \qquad k = 0, 1, \ldots, \tag{1}$$

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \qquad k = 0, 1, \ldots, \tag{2}$$

where $\boldsymbol{x}_k \in \mathcal{R}^{n_x}$, $\boldsymbol{z}_k \in \mathcal{R}^{n_z}$ and $\boldsymbol{u}_k \in \mathcal{R}^{n_u}$ are state, measurement and control of the model, respectively, $\boldsymbol{w}_k \in \mathcal{R}^{n_x}$ and $\boldsymbol{v}_k \in \mathcal{R}^{n_z}$ are state and measurement white noises described by $p(\boldsymbol{w}_k)$ and $p(\boldsymbol{v}_k)$, respectively. Both noises are mutually independent and they are also independent of the known initial state $\boldsymbol{x}_0$. The functions $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ are supposed to be known.

From the above mentioned description it is clear that the user needs to specify the state and measurement functions accompanied with PDFs of the state and measurement noises and of the prior state. Of all the function classes the class `nefHandleFunction` is the most prominent for description of the $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ functions. This class makes it possible to provide a unified interface within the toolbox for description of arbitrary generally time-varying function. The attributes of the class instance are given either as anonymous functions or function handles representing the function itself accompanied by specification of system variable dimensions and if needed its first and the second derivatives.

For the description of random variables, the toolbox provides many classes which in fact represent PDFs. Their instances are given by a specification of parameters of the corresponding PDF, e.g. in case of the Gaussian PDF by the mean vector and covariance matrix. In order to be able to describe time varying parameters or to describe conditional PDF necessary for probabilistic model description, it is possible to state these PDF parameters using instances of function classes. This feature is naturally exploited in case of the probabilistic model description which is specified by the transient PDF (3), the measurement PDF (4):

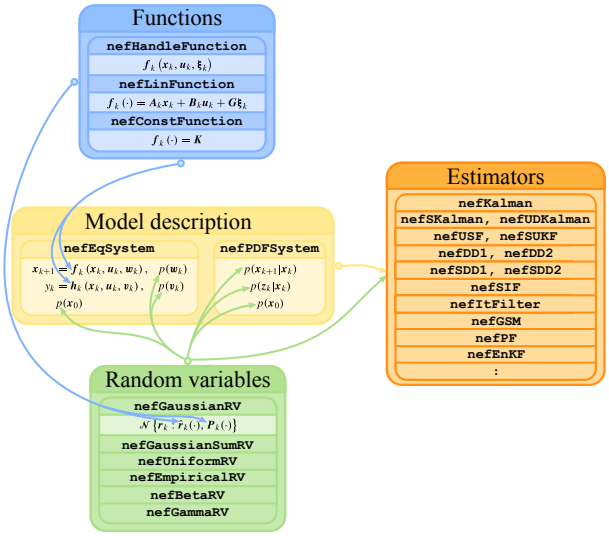

Fig. 2. Overview of toolbox classes used for estimation experiment description

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, 1, \ldots, \tag{3}$$

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, 1, \ldots \tag{4}$$

and PDF $p(\boldsymbol{x}_0)$ of the initial state.

2.2 Estimators and class methods for estimation

The crucial toolbox component is undoubtedly the estimator component. This components offers implementation of many traditional and modern local and global estimation methods as presented in Table 1.

Table 1. Estimators implemented in the NEF estimation component

name in NEF	estimators
`nefKalman`, `nefSKalman` `nefUDKalman`	(extended) Kalman filter (standard, square-root and UD versions)
`nefDD1`, `nefSDD1`, `nefDD2`, `nefSDD2`	central difference Kalman filter, divided difference filter (1^{st} and 2^{nd} order; standard and square-root version)
`nefUKF`, `nefSUKF`	unscented Kalman filter (standard and square-root version), cubature Kalman filter
`nefItKalman`	iterated Kalman filter based on any of the above local filter
`nefGSM`	Gaussian sum filter based on any of the above local filter
`nefPF`	bootstrap filter, generic particle filter, auxiliary particle filter, unscented particle filter
`nefEnKF`	ensemble Kalman filter
`nefSIF`	stochastic integration filter

The provided implementations are often broadly parametrizable (as is the case of the particle filter and UKF implemented by classes `nefPF` and `nefUKF`, respectively). Apart from the canonical implementation, also more advanced algorithma such as the UKF with adaptation of the parameter of the UT, are also offered.

All the power of the estimator classes is inherited from the abstract class `nefEstimator`. This class defines the interface provided by all estimator classes and offers a ready-to-use mechanism for processing the basic estimation tasks, i.e. filtering, multi-step prediction and smoothing (fixed lag, fixed point and fixed interval). The simplest way of executing the estimation experiment is to employ the `estimate` method which embodies the BFR. This methods automatically processes all the data (see Fig. 3) and gives the results in the form of PDFs. However, this way is not suitable for control purposes as it is currently not possible to insert the evaluation of the control law into the process. Nevertheless, this is not a serious obstacle as the user is able to use directly methods `timeUpdate`, `measurementUpdate` (and possibly `smoothUpdate`).

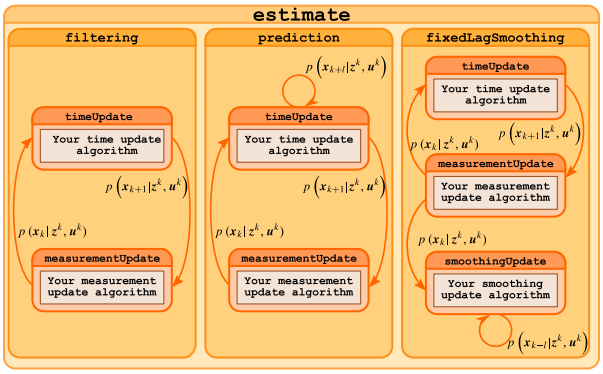

Fig. 3. The scheme of processing schemes of basic estimation tasks supported by the `estimate` method of the `nefEstimator` class.

2.3 Performance evaluation component

The last component supports the task of an experiment result evaluation. It is primarily aimed at measuring the estimation error and compare performance of several estimators against the true value of the state. However, most of the provided measures can also be advantageously used for evaluation of the quality of the controllers. Currently, the performance evaluation component provides performance indices (Li and Zhao, 2006; Blasch et al., 2006) given in Table 2.

Table 2. Performance indices implemented in the NEF performance evaluation component

	ABSOLUTE ERROR MEASURES
MSEM	mean squared error matrix
RMSE	root mean squared error
AEE	average Euclidean error
HAE	harmonic average error
GAE	geometric average error
MEDE	median error
MODE	mode error
	RELATIVE ERROR MEASURES
RMSRE	root mean squared relative error
ARE	average Euclidean relative error
BEEQ	Bayesian estimation error quotient
EMER	estimation error relative to measurement error
	PERFORMANCE MEASURES
NCI	non-credibility index
ANEES	average normalized estimation error squared

The class `nefPerformanceEvaluator` defines methods for initialization, data processing and evaluating the performance index. The instance of this class fulfills the task of i) collecting data from Monte Carlo (MC) simulations, ii) extracting appropriate indicators from the conditional distribution of the state provided by individual estimators and iii) evaluating the performance index.

3. NEF EMPLOYMENT IN CONTROL PROBLEMS

This section presents examples of application of the NEF to the control problems. First, the classical LQG controller, where the state estimates will be provided by UD factorized Kalman filter, will be shown. Second, the example where some parameters of the controlled systems are not known will be presented. In this example the estimation problem is nonlinear in nature and UKF will be used in order to obtain the filtering mean and covariance matrix.

In both examples, the following controlled system is considered

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k, \tag{5}$$

$$z_k = \boldsymbol{C}\boldsymbol{x}_k + v_k \tag{6}$$

with

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ \theta_{1k} & \theta_{2k} \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 \\ \theta_{3k} \end{pmatrix}, \boldsymbol{C} = (0\ 1)\,. \tag{7}$$

The parameter vector $\boldsymbol{\theta}_k = \left(\theta_{1,k}, \theta_{2,k}, \theta_{3,k}\right)^T$ is considered to be constant, i.e. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k$. The actual parameters are given by $\boldsymbol{\theta}_k = (-2.0427, 0.3427, 1)^T$, $\forall k$ (i.e. the controlled system is unstable) and the initial state is fixed to value $\boldsymbol{x}_0 = (1, -0.5)^T$.

The PDFs of the state and measurement noises are specified as

$$\boldsymbol{w}_k \sim \mathcal{N}\,(0, 0.0001)\,, \tag{8}$$

$$v_k \sim \mathcal{N}\,(0, 0.001)\,. \tag{9}$$

where $\mathcal{N}\left(\hat{r}, P_r\right)$ denotes normal distribution with mean $\hat{r}$ and covariance matrix P_r.

The goal is be to find a control law minimizing the criterion

$$J = E\left\{\sum_{k=0}^{N-1} \mathcal{L}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)\right\} \tag{10}$$

with respect to the discrete time stochastic system (5)-(6). The cost function is considered to be quadratic and is defined as

$$\mathcal{L}_k(\cdot) = (\boldsymbol{x}_{k+1} - \bar{\boldsymbol{x}}_{k+1})^T \boldsymbol{Q}_{k+1}\,(\boldsymbol{x}_{k+1} - \bar{\boldsymbol{x}}_{k+1}) + \boldsymbol{u}_k^T \boldsymbol{R}_k \boldsymbol{u}_k. \tag{11}$$

The quantity $\bar{\boldsymbol{x}}_{k+1} \in \mathcal{R}^2$ denotes the setpoint vector at time instant $k+1$. The matrices $\boldsymbol{Q}_{k+1} \in \mathcal{R}^{2\times 2}$ and $\boldsymbol{R}_k \in \mathcal{R}^1$ are appropriately chosen positive semidefinite and positive definite matrices, respectively.

In this particular case, the cost function (11) is specified as

$$\mathcal{L}_k(\cdot) = (x_{k+1,2} - \bar{x}_{k+1,2})^2 + 0.001 \cdot u_k^2, \tag{12}$$

i.e. the goal is to drive the second state element to the setpoint values $\bar{x}_{k+1,2}$ and the trajectory of the first state element is not penalized in any way. The cost function (12) corresponds to the general form (11) by choosing the weighting matrices as

$$\boldsymbol{Q}_{k+1} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{R}_k = 0.001. \tag{13}$$

3.1 Implementation of the LQG controller

In case of known parameters of the system the solution to the above described optimization problem leads to the LQG controller. The controller design employs the separation principle. The control law is derived for a deterministic certainty equivalent (CE) system with assumption that the state is completely known and the true state is replaced by its optimal state estimate. The LQ controller represents a deterministic control law and the optimal estimate is provided by KF .

The LQ control law, for a non zero setpoint is given by the relation

$$\boldsymbol{u}_k(\boldsymbol{\mu}_k) = -\boldsymbol{K}_k\boldsymbol{\mu}_k + \boldsymbol{B}^T\left(\boldsymbol{F}_{k+1} - \boldsymbol{Q}_{k+1}\bar{\boldsymbol{x}}_{k+1}\right) \tag{14}$$

with the control gain specified as

$$\boldsymbol{K}_k = \left(\boldsymbol{B}^T\boldsymbol{S}_{k+1}\boldsymbol{B} + \boldsymbol{R}\right)^{-1}\boldsymbol{B}^T\boldsymbol{S}_{k+1}\boldsymbol{A}. \tag{15}$$

The matrix $\boldsymbol{S}_{k+1}$ and the vector $\boldsymbol{F}_{k+1}$ are derived using the Bellman optimization recursion [2]. As the estimation and control tasks do not interfere with each other, it is possible to determine these parameters of the LQ control law beforehand.

The proper implementation of the LQG controller consisting of the LQ controller and KF is crucial. It is necessary to properly propagate the quantities within the controller. The LQ controller needs according to (14) the filtering estimate $\boldsymbol{\mu}_k$ which in turn requires the predictive estimate $\hat{\boldsymbol{x}}_k$. The control $\boldsymbol{u}_k$ is then required to determine the predictive estimate $\hat{\boldsymbol{x}}_{k+1}$. The whole process of evaluation of control law and filtering and predictive estimates is depicted in Fig. 4.

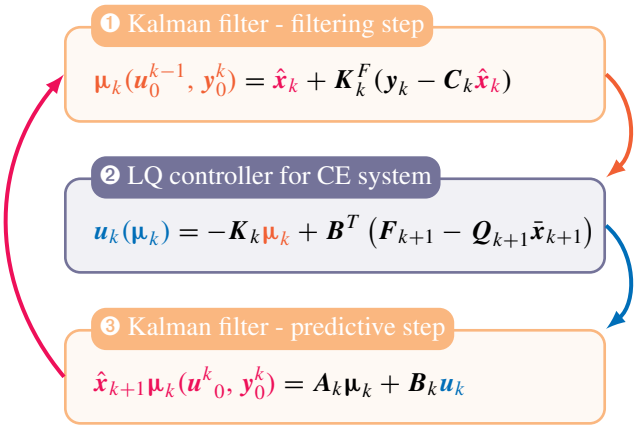

Fig. 4. The processing of the information in the LQG controller

The remaining part of this subsection will present how to implement this procedure employing the means provided by the NEF toolbox. First, it is necessary to describe the problem at hand, i.e. to represent the system (5)-(9) within the toolbox.

Since the functions (5) and (6) are both linear, it would be advantageous to employ the `nefLinFunction` for instantiation of the objects representing these functions

```
f = nefLinFunction(A, B, eye(2));
h = nefLinFunction(C,[],1);
```

where the first, second, and third matrix parameters correspond to matrices multiplying the state, control, and noise, respectively.

The state and measurement noises and the initial state are all Gaussian and thus they can be represented by instances of the `nefGaussianRV` class

```
wmean=[0;0]; Pw=eye(2)*0.0001;
pw=nefGaussianRV(wmean,Pw);

vmean=0; Qv=0.001;
pv=nefGaussianRV(vmean,Pv);

x0=[1;-0.5];
```

[2] The matrices $\boldsymbol{S}_{k+1}$ are given by the Riccati recurrence relation.

```
Px0=diag(0.2*[1 1]);
px0=nefGaussianRV(x0,Px0);
```

The system model is given structurally, hence the `nefEqSystem` class will be use for its definition within the toolbox

```
model=nefEqSystem(f,h,pw,pv,px0);
```

where it is mandatory to provide the state function `f`, the measurement function `h`, the PDFs of the state and parameter noises `pw` and `pv`, respectively, and initial state PDF `px0`.

Subsequently, it is possible to create an object that embodies the chosen estimator. For this particular system, the KF is a natural choice. This example will however use one of the numerically stable implementations of the KF. The chosen filter uses the UD factorization of the covariance matrices and is provided by the class `nefUDKalman`. The instance of this filter is constructed in the following manner

```
UDKalman = nefUDKalman(model);
```

Finally, it is possible to proceed to the implementation of the control loop. It is necessary to implement the process depicted in Fig. 4. For this implementation each of the estimators provided by NEF offers two indispensable methods. These are the `measurementUpdate` and `timeUpdate` methods implementing the filtering and predictive step of the estimator, respectively.

Before the implementation itself it is necessary to mention two notes. First, in the subsequent MATLAB code it will be assumed that all the values of matrix $\boldsymbol{S}_{k+1}$ and vector $\boldsymbol{F}_{k+1}$ are already determined beforehand and stored in cell arrays `S` and `F`. Second, since the UD factorized KF will be used instead of the standard KF, it is necessary to factorize the initial state covariance matrix [3] and to prepare a proper prediction PDF for the filtering step in a structure element `RV`

```
predPDF.RV=nefGaussianRV(x0,...
    nefUDFactorFunction(Px0));
```

Now, everything but measurement of the initial state is prepared for the actual trajectory simulation. This measurement is computed using `simulate` method of the `nefEqSystem` class

```
[z(:,1),x(:,1),model] = ...
    simulate(model,1,[],'initialState',x0)
```

In this case the initial state is equal to variable `x0`. In case when the last two parameters would be omitted, the initial state would be drawn from initial state PDF. The control loop is finally implemented as follows

```
for k = 1:controlHorizon
  % determine current filtering pdf
  filtPDF = measurementUpdate(UDKalman,...
    predPDF,[],z(:,k),k);
  mu(:,k) = evalMean(filtPDF.RV);

  % control law
  u(:,k)=-(B'*S{k+1}*B+R)\...
         (B'*S{k+1}*A*mu(:,k)...
         +B'*F{k+1}-Q*xsetpoint(:,k+1));

  % determine one step predictive pdf
  predPDF = timeUpdate(UDKalman,...
    filtPDF,u(:,k),k);

  % system trajectory simulation
  [z(:,time+1),x(:,time+1),model] = ...
    simulate(model,1,u(:,time));
end
```

[3] Simillar procedure is necessary for all estimators which employ factorizations. For the standard Kalman filter this step is omitted.

3.2 Implementation of the cautious controller

Now it will be assumed that the parameter vector $\boldsymbol{\theta}_k$ will be unknown. The problem is not separable and neutral anymore. Also the optimal control law is not attainable in a closed form. It is necessary to resort to some suboptimal solution. For purposes of this example the suboptimal cautious controller will be used. Moreover, to simplify the presentation, only the myopic cautious controller will be considered as this controller can be simply derived and implemented.

The control law is in this case given as

$$\boldsymbol{u}_k = \underset{\boldsymbol{u}_k}{\operatorname{argmin}}\, \mathcal{L}_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right). \tag{16}$$

It should also be noted, that in this case the control is not solely function of the point estimate $\boldsymbol{\mu}_k$ but also of the filtering estimate covariance matrix.

The estimation problem is also slightly more complicated, as the the state has to be augmented with the unknown parameters making the problem nonlinear. The state transient function $\boldsymbol{f}_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right)$ is in this case represented by instance of the class `nefHandleFunction`

```
fFun = @(x,u,w,k) [x(2)+w(1);
       x(3)*x(1)+x(4)*x(2)+x(5)*u+w(2);
       x(3); x(4); x(5)];
f = nefHandleFunction(fFun,[5 1 2 0]);
```

where the `fFun` is MATLAB handle function representing the state transient function augmented with the relations describing constant parameters. The second parameter specifies dimensions of the state, control, state noise and time quantities [4]. The model is again described using the instance of `nefEqSystem` class in the same manner as before.

Next the estimator object should be created. It is possible to use the `nefUDKalman` class constructor again. Then, the estimator will detect the nonlinear state transition function and will automatically use the UD factorized extended KF in the prediction step. However, it should be noted that in this case the user would have to enhance the `nefHandleFunction` contructor with a parameter specifying the Jacobian of the function $\mathbf{f}_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right)$. Of course, it advisable to use a derivative free method instead, such as the UKF

```
UKF = nefUKF(model);
```

The implementation of the control loop is very similar to the previous case. There are three differences in the algorithm. First, it is necessary to extract the covariance matrix from the filtering PDF as it is used in the control law. Second, of the point estimate $\boldsymbol{\mu}_k$ only the first two elements representing the state. Third, it is necessary to create estimated matrices $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ at each time instant using the remaining elements of point estimate $\boldsymbol{\mu}_k$.

[4] The zero dimension for time means that the function is t-invariant.

```
for k = 1:controlHorizon
  % determine current filtering pdf
  filtPDF = measurementUpdate(UKF,...
    predPDF,[],z(:,k),k);

  % extract mean and covariance matrix
  mu(:,k) = evalMean(filtPDF);
  Pf = evalVariance(filtPDF);

  % construct estimated matrices A and B
  estA = [ 0 1; mu(3:4,k)'];
  estB = [ 0; mu(5,k)];

  % cautious control law
  u(:,k)=-(estB'*Q*estB+Pf(5,5)+R)\...
   (estB'*(Q*estA+Pf(3:4,3:4)*mu(1:2,k)...
   -estB'*Q*xsetpoint(:,k+1));

  % determine one step predictive pdf
  predPDF = timeUpdate(UDKalman,...
    filtPDF,u(:,k),k);

  % system trajectory simulation
  [z(:,time+1),x(:,time+1),model] = ...
    simulate(model,1,u(:,time));
end
```

4. CONCLUSION

The paper presented the Nonlinear Estimation Framework that facilitates implementation, testing, use and evaluation of nonlinear state estimation methods. It is aimed at both the casual and the experienced user. It provides easy to use, however, powerful tools for straightforward estimation experiment design.

It was shown that the framework is useful not only for pure estimation purposes. But it can also be advantageously employed in controller design and testing. The toolbox can alleviate the burden of choosing and implementing the estimator and leave more room for controller design.

The usage of the NEF was illustrated in two control examples, more specifically the LQG and cautious control.

REFERENCES

Arasaratnam, I. and Haykin, S. (2009). Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6), 1254–1269.

Blasch, E., Rice, A., and Yang, C. (2006). Nonlinear track evaluation using absolute and relative metrics. In *Proceedings of SPIE, the International Society for Optical Engineering*, 62360–62360. SPIE.

Candy, J. (2008). *Bayesian signal processing*. Wiley.

Doucet, A., de Freitas, N., and Gordon, N. (eds.) (2001). *Sequential Monte Carlo Methods in Practice*. Springer. (Ed. Doucet A., de Freitas N., and Gordon N.).

Duník, J., Šimandl, M., Straka, O., and Král, L. (2005). Performance Analysis of Derivative-Free Filters. In *Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference ECC'05*, ISBN: 0-7803-9568-9, ISSN: 0191-2216, 1941–1946. Seville, Spain.

Duník, J., Straka, O., and Šimandl (2013). Stochastic integration filter. *IEEE Transactions on Automatic Control*, 58(6), 1561 – 1566.

Feldbaum, A. (1965). *Optimal Control Systems*. Academic Press, New York.

Flídr, M., Straka, O., Havlík, J., and Šimandl, M. (2013). Nonlinear Estimation Framework: a Versatile Tool for State Estimation. In *Proceedings of the 18th International Conference on Methods & Models in Automation & Robotics (MMAR 2013)*, 490–495. Miedzyzdrojie, Poland.

Flídr, M. and Šimandl, M. (2011). Dual Adaptive Controllers Based on Partial Certainty Equivalence. In *Proceedings of 18th IFAC World Congress*, 3457–3462. Elsevier, Milano, Italy.

Flídr, M. and Šimandl, M. (2013). Implicit Dual Controller based on Stochastic Integration Rule. In *Proceedings of the 12th European Control Conference 2013*, 896–901. Zurich, Switzerland.

Grewal, S.G. and Andrews, A.P. (2001). *Kalman Filtering: Theory and Practise Using MATLAB (Second Edition)*. John Wiley & Sons, New York. ISBN 0-471-39254-5.

Ito, K. and Xiong, K. (2000). Gaussian Filters for Nonlinear Filtering Problems. *IEEE Trans. on Automatic Control*, 45(5), 910–927.

Julier, S.J. and Uhlmann, J.K. (2004). Unscented filtering and nonlinear estimation. *IEEE review*, 92(3), 401–421.

Král, L. and Šimandl, M. (2011). Functional adaptive controller for multivariable stochastic systems with dynamic structure of neural network. *International Journal of Adaptive Control and Signal Processing*, 25(11), 949–964.

Král, L. and Šimandl, M. (2013). Dual Adaptive Control for Non-Minimum Phase Systems with Functional Uncertainties. In T. Sophie (ed.), *Proceedings of the 9th IFAC Symposium on Nonlinear Control Systems, 2013*, 815–820. Toulouse, France.

Lefebvre, T., Bruyninckx, H., and de Schutter, J. (2002). Comment on "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators". 47(8), 1406–1409.

Li, X. and Zhao, Z. (2006). Evaluation of estimation algorithms part I: incomprehensive measures of performance. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4), 1340–1358.

Nørgaard, M., Poulsen, N., and Ravn, O. (2000). New developments in state estimation for nonlinear systems. *Automatica*, 36(11), 1627–1638.

Šimandl, M. and Duník, J. (2009). Derivative-free estimation methods: New results and performance analysis. *Automatica*, 45(7), 1749–1757.

Šimandl, M., Královec, J., and Söderström, T. (2006). Advanced point – mass method for nonlinear state estimation. *Automatica*, 42(7), 1133–1145.

Straka, O., Flídr, M., Duník, J., and Šimandl, M. (2009). A Software Framework and Tool for Nonlinear State Estimation. In *Proceedings of the 15th IFAC Symposium on System Identification*, 510–515. Saint-Malo, France.

Straka, O., Flídr, M., Duník, J., Šimandl, M., and Blasch, E. (2010). Nonlinear Estimation Framework in Target Tracking. In *Proceedings of the 13th International Conference on Information Fusion*. Edinburgh, UK.

Straka, O. and Šimandl, M. (2005). Distance-based pruning for Gaussian sum method in non-Gaussian system state estimation. In *Proceedings of the Eighth IASTED International Conference on Intelligent Systems and Control*, 96–101.