# Toolbox for nonlinear estimation

Miroslav Flídr, Miroslav Šimandl, Jindřich Duník and Ondřej Straka

Research Centre Data, Algorithms and Decision Making
Faculty of Applied Sciences
University of West Bohemia in Pilsen
Czech Republic

ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

## Outline

## Problem formulation

**Consider multivariate nonlinear stochastic system**

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), & k &= 0, 1, 2, \ldots \\ \boldsymbol{z}_k &= \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k), & k &= 0, 1, 2, \ldots \end{aligned}$$

$\boldsymbol{x}_k \in \mathbb{R}^{n_x} \ldots$ non-measurable state
$\boldsymbol{z}_k \in \mathbb{R}^{n_z} \ldots$ measurement
$\boldsymbol{u}_k \in \mathbb{R}^{n_u} \ldots$ control
$\boldsymbol{w}_k \in \mathbb{R}^{n_x} \ldots$ state white noise
$\boldsymbol{v}_k \in \mathbb{R}^{n_z} \ldots$ measurement white noise

- ✓ Both noises are mutually independent and they are also independent of the known initial state $\boldsymbol{x}_0$ pdf $p(\boldsymbol{x}_0)$.
- ✓ The vector mappings $\boldsymbol{f} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, $\boldsymbol{h} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ are known

**The aim**: to the estimate the non-measurable state $x_k$

The posterior pdf $p(\boldsymbol{x}_k | \boldsymbol{z}^\ell, \boldsymbol{u}_{k-1})$ is sought!

$\boldsymbol{z}^\ell \triangleq [\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell] \ldots$ set of measurements

## General Solution

**General solution obtainable by Bayesian approach**

➢ solution of the **filtering problem** ($\ell = k$)

$$p(\mathbf{x}_k|\mathbf{z}^k, \mathbf{u}_{k-1}) = \frac{p(\mathbf{x}_k|\mathbf{z}^{k-1}, \mathbf{u}_{k-1})p(\mathbf{z}_k|\mathbf{x}_k)}{\int p(\mathbf{x}_k|\mathbf{z}^{k-1}, \mathbf{u}_{k-1})p(\mathbf{z}_k|\mathbf{x}_k)d\mathbf{x}_k},$$

$$p(\mathbf{x}_k|\mathbf{z}^{k-1}, \mathbf{u}_{k-1}) = \int p(\mathbf{x}_{k-1}|\mathbf{z}^{k-1}, \mathbf{u}_{k-2})p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})d\mathbf{x}_{k-1}$$

➢ solution of the **multistep prediction problem** ($\ell < k$)

$$p(\mathbf{x}_k|\mathbf{z}^\ell, \mathbf{u}_{k-1}) = \int p(\mathbf{x}_{k-1}|\mathbf{z}^\ell, \mathbf{u}_{k-2})p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})d\mathbf{x}_{k-1}$$

➢ solution of the **multistep smoothing problem** ($\ell > k$)

$$p(\mathbf{x}_k|\mathbf{z}^\ell, \mathbf{u}_{k-1}) = p(\mathbf{x}_k|\mathbf{z}^k, \mathbf{u}_{k-1}) \int \frac{p(\mathbf{x}_{k+1}|\mathbf{z}^\ell, \mathbf{u}_k)}{p(\mathbf{x}_{k+1}|\mathbf{z}^k, \mathbf{u}_k)} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)d\mathbf{x}_{k+1}$$

## Solutions of Bayesian recursive relations for filtering, prediction and smoothing problems

### Exact solutions - valid only for special class of systems

- ➢ Kalman filter
- ➢ Gaussian sum filter
- ➢ Daum filter

### Approximate local methods

- ➢ Extended Kalman filter
- ➢ Divided difference filter
- ➢ Unscented Kalman filter

### Approximate global methods

- ➢ Gaussian sum filter
- ➢ Point-mass method
- ➢ Particle filters

## The objective: To design toolbox facilitating easy estimator design and testing

### What criteria should the toolbox meet?

- ✔ to be highly modular, easily extensible and user friendly
- ✔ to provide multi-step prediction, filtering and multi-step smoothing
- ✔ to be build in MATLAB environment

### Which tasks should be provided by the toolbox?

- ✔ complete description of the system
- ✔ simulation of the system
- ✔ choice and application of the suitable estimator
- ✔ easy extensibility with new estimators

## Aren’t there already Matlab toolboxes for nonlinear estimation?

- ➢ KALMTOOL (HTTP://SERVER.OERSTED.DTU.DK/PERSONAL/OR/KALMTOOL3/)
- ➢ ReBEL (HTTP://CHOOSH.ECE.OGI.EDU/REBEL/)

### Advantages & disadvantages of those toolboxes

- ✔ the computational demands of estimation process are moderate
- ✔ KALMTOOL has Simulink support
- ✘ suitable only for filtering problem
- ✘ not easily reusable code (monolithic design)
- ✘ provide only point estimate

*However, both mentioned toolboxes doesn’t fully meet specified demands!!*

## Features of the Nonlinear Filtering Toolbox (NFT)

### Advantages of presented framework

- ➢ takes advantage of Matlab **object oriented** programming features
- ➢ can handle filtering and multistep prediction and smoothing
- ➢ estimators provide conditional probability density functions
- ➢ provides means for easy control of the whole estimation process
- ➢ easy addition of new estimators

### Structure of NFT

- ➢ probability density function (pdf's) classes
- ➢ system classes
- ➢ estimator classes
- ➢ auxiliary classes

## Probability density function classes

### Pdf's classes features

- ➢ random quantities represented as objects of corresponding pdf class
- ➢ generic class defining mandatory interface of all pdf classes and making them distinguishable as pdf's within toolbox
- ➢ pdf classes provide methods such as:
  - ➪ resetting and reading of pdf parameters,
  - ➪ evaluation of pdf at arbitrary point of state space,
  - ➪ generating of random samples, . . .

### Illustration of creation of Gaussian pdf object

$$p(\boldsymbol{x}) = \mathcal{N}\left\{\begin{pmatrix}-1\\5\end{pmatrix}, \begin{pmatrix}0.1 & 0\\0 & 0.2\end{pmatrix}\right\},$$

```
>> px = gpdf([-1;5],diag([.1,.2]));
>> x = sample(px);
```

## System classes

### Classes provided for system creation and handling

- three classes for definition of multivariate functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$
  - `nfFunction` - general class defining interface for user defined functions
  - `nfSymFunction` - utilizes Symbolic toolbox $\Rightarrow$ slow computations
  - `nfLinFunction` - description of linear multivariate functions
- several classes for various type of system - (**N**on)**L**inear (**N**on)**G**aussian with (**N**on)**A**dditive noises

### Illustration of creation and use of nonlinear system with additive noises

```
>> f = nfSymFunction('[x1*x2+w1;x2+w2]',...
>>                           '','x1,x2','w1,w2');
>> h = nfSymFunction('x1*x2+v','','x1,x2','v');
>> system = nlga(f,h,pw,pv,px0);
>> [z,x,system] = simulate(system);
```

## Estimator classes

### Main task of the estimator classes

The estimator classes essentially implement algorithms necessary to obtain $p(\boldsymbol{x}_k|\boldsymbol{z}^k, \boldsymbol{u}_{k-1})$, $p(\boldsymbol{x}_k|\boldsymbol{z}^\ell, \boldsymbol{u}_{k-1})$ and even possibly $p(\boldsymbol{x}_k|\boldsymbol{z}^\ell, \boldsymbol{u}_{k-1})$, i.e. filtering, predictive and smoothing conditional pdf's, respectively.

### Features of the general class `estimator`

- ➢ its virtual methods sets the interface of actual estimator classes
- ➢ provides method `estimate` that controls the whole estimation process $\Rightarrow$ the designer of the estimator doesn't need to care
- ➢ `estimator` stores the data of multistep operations in dynamical list
- ➢ the lists can hold arbitrary content, however, they are primarily used to store conditional pdf's
- ➢ implements commonly used methods $\Rightarrow$ decreases redundancy and makes possible easy future improvements

## Estimator classes

### Estimators currently implemented in NFT

| | |
|---|---|
| Kalman filter | kalman |
| Extended Kalman filter | extkalman |
| Iterating Kalman filter | itekalman |
| Second order Kalman filter | seckalman |
| Gaussian sums filter | gsm |
| Particle filter | pf |
| Point mass filter | pmf |
| Divided difference filter 1st order | dd1 |
| Divided difference filter 2nd order | dd2 |
| Unscented Kalman filter | ukf |

### Illustration of creation and use of DD1 estimator object

```
>> filter = dd1(system,0);
>> [est,filter] = estimate(filter,z,[]);
```

## Example of NFT usage

### Considered nonlinear Gaussian system

$$\begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \end{pmatrix} = \begin{pmatrix} x_{1,k}^2 \cdot x_{2,k} \\ x_{2,k} \end{pmatrix} + \boldsymbol{w}_k$$

$$z_k = (4\ 1) \begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix} + v_k$$

### The description of stochastic quantities

$$p(\boldsymbol{w}_k) = \mathcal{N}\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} \right)$$

$$p(v_k) = \mathcal{N}\,(0,\ 0.01)$$

$$p(\boldsymbol{x}_0) = \mathcal{N}\left( \begin{bmatrix} 0.9 \\ -0.85 \end{bmatrix}, \begin{bmatrix} 0.9 & 0 \\ 0 & 0.1 \end{bmatrix} \right)$$

$$p(\boldsymbol{x}_{0|-1}) = \mathcal{N}\left( \begin{bmatrix} 0.2 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

## Example of NFT usage (continuation)

### The task is to obtain two-step prediction $p\left(x_k|z^{k-2}\right)$

- definition of the random variables

```
pw = gpdf([0;0],0.05*eye(2));
pv = gpdf(0,0.5);
px0 = gpdf([0.9;-0.85],diag([0.09,0.1]));
p_apr = gpdf([0.2;-0.5],diag([1,1]));
```

- definition and simulation of the system

```
f = nfSymfunction('[x1^2*x2+w1;x2+w2]',...
                        '','x1,x2','w1,w2');
h = nfLinFunction([4 1],[],1,...
                        '','x1,x2','w1,w2')
system = nlga(f,h,pw,pv,px0);
[z,x,system]=simulate(system,zeros(1,40));
```

- choise of the estimator and the estimation process itself

```
filter = dd1(system,2,p_apr);
[est,filter] = estimate(filter,z,[]);
```

## Future version features

### Plans

- ➢ system description using transition and measurement pdf's
- ➢ support for pdf's parameterized by state, measurement and time
- ➢ possibility to define and use time variant systems
- ➢ fast nonlinear functions prototyping using anonymous function handles
- ➢ support for all estimators implemented in current stable version
- ➢ possibility to automatically approximate pdf's
- ➢ employing new MATLAB class system

## Future version features

### Already implemented features

- ✔ system description using transition and measurement pdf’s
- ✔ support for pdf’s parameterized by state, measurement and time
- ✔ possibility to define and use time variant systems
- ✔ fast nonlinear functions prototyping using anonymous function handles
- ✒ support for all estimators implemented in current stable version
- ✘ possibility to automatically approximate pdf’s
- ✔ employing new MATLAB class system

## Example of trajectory simulation using NFT v3

$$x_k \sim \mathcal{N}(x_k + u_k, 1), \quad z_k \sim \mathcal{N}(x_k^2, 0.7 \cdot k + 0.1), \quad x_0 \sim \mathcal{N}(0, 1)$$

- preparation of pdf's parameters

```
transMean = nfHandleFunction(...
    @(x,u,w,time) x + u, [1 1 0 0]);
measMean = nfHandleFunction(...
    @(x,u,w,time) x^2, [1 0 0 0]);
measVar = nfHandleFunction(...
    @(x,u,w,time) 0.7*time+0.1, [0 0 0 1]);
```

- definition of pdf's

```
initalPdf = nfGaussianRV(0,1)
transPdf = nfGaussianRV(transMean,1);
measPdf = nfGaussianRV(measMean,measVar);
```

- definition and simulation of the system

```
system=nfPDFSystem(transPdf,measPdf,initalPdf);
[z,x]=simulate(system,3,{0 0 0});
```

## Concluding remarks

### Current contribution of the NFT

- provides all necessary tools for estimator design, testing and employment
- the toolbox is easily extensible thanks to object oriented approach
- includes all the basic estimator implementing filtering, prediction and smoothing methods