Nonlinear Estimation Framework in Target Tracking

Ondřej Straka[1], Miroslav Flídr[1], Jindřich Duník[1], Miroslav Šimandl[1] and Erik Blasch[2]

[1] Department of Cybernetics & Research Centre Data - Algorithms - Decision making
Faculty of Applied Sciences, University of West Bohemia, Czech Republic

[2] Defence R&D Canada-Valcartier
2459 Pie-XI Blvd. North
Québec City, QB G3J 1X5

FUSION 2010, July, Edinburgh

Outline

Introduction of the Nonlinear Estimation Framework (NEF)

Goal of the paper

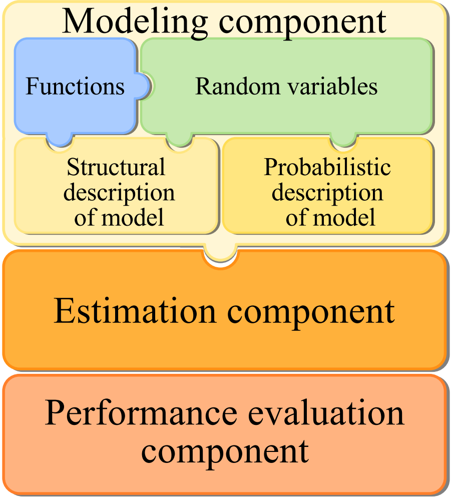
to introduce NEF and illustrate its use in target tracking problems

- a collection of MATLAB classes and functions for
  - modelling system behavior
  - state estimation
  - evaluation of the results
- development driven by the need for a tool that can
  - evaluate the quality of a state estimation method in arbitrary case
  - compare performance of several state estimators
  - provide means for effortless rapid prototyping of new state estimators
- most recent generation of software toolboxes developed by authors (former NFTools and NFToolsCD).

Key features of the NEF

- structural and probabilistic modelling,
- 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,
- complete evaluation of estimate quality.

Components of the NEF



Modelling component

Structural description

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

$p(\boldsymbol{w}_k)$, $p(\boldsymbol{v}_k)$, $p(\boldsymbol{x}_0)$

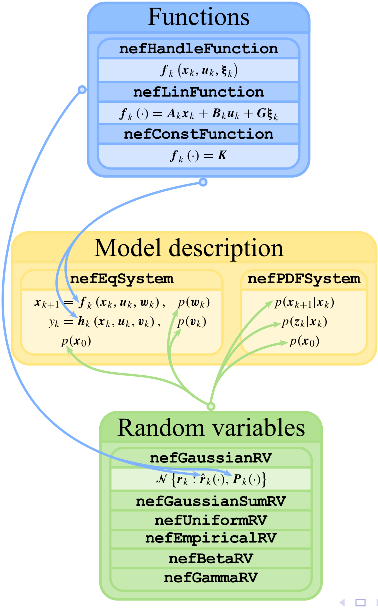
- $\boldsymbol{x}_k \in \mathcal{R}^{n_x}$, $\boldsymbol{z}_k \in \mathcal{R}^{n_z}$, $\boldsymbol{u}_k \in \mathcal{R}^{n_u}$, $\boldsymbol{w}_k \in \mathcal{R}^{n_x}$, $\boldsymbol{v}_k \in \mathcal{R}^{n_z}$

Probabilistic description

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

$p(\boldsymbol{x}_0)$

Scheme of NEF modelling component



Estimation

The estimate of the state $\boldsymbol{x}_k$

given by the posterior pdf $p(\boldsymbol{x}_k|\boldsymbol{z}^\ell, \boldsymbol{u}^\ell)$, where $\boldsymbol{z}^\ell$ is the sequence of measurements up to time instant ℓ, i.e., $\boldsymbol{z}^\ell \triangleq [\boldsymbol{z}_0^T, \boldsymbol{z}_1^T, \ldots, \boldsymbol{z}_\ell^T]^T$.

- If $\ell = k$, the problem is called *filtering*.
- If $\ell < k$, the problem is called *prediction*.
- If $\ell > k$, the problem is called *smoothing*.

Solution to the estimation problem

- provided by the Bayesian functional relations (BFR)
- analytically tractable for a few special models
- mostly an approximate solution is being looked for
- BFR idea is embodied by `nefEstimator`

Estimators implemented in the NEF estimation component

`nefKalman`, `nefSKalman`, `nefUDKalman`	(extended) Kalman filter (standard, square-root and UD versions)
`nefDD1`, `nefSDD1`, `nefDD2`	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
`nefGSM`	Gaussian sum filter
`nefPF`	bootstrap filter, generic particle filter, auxiliary particle filter, unscented particle filter
`nefEnKF`	ensemble Kalman filter.

Estimation tasks supported by individual estimators

estimator	filtering	prediction	smoothing
`nefKalman`	✓	✓	✓
`nefSKalman`	✓	✓	✓
`nefUDKalman`	✓	✓	
`nefItKalman`	✓	✓	✓
`nefDD1`	✓	✓	✓
`nefSDD1`	✓	✓	✓
`nefDD2`	✓	✓	✓
`nefUKF`	✓	✓	✓
`nefSUKF`	✓	✓	✓
`nefGSM`	✓	✓	
`nefPF`	✓	✓	
`nefEnKF`	✓	✓	

Performance evaluation

Aim

- to measure estimation error
- to compare performance of several estimators against the true value of the state

Steps to measure performance

1. collecting data from Monte Carlo simulations,
2. extracting appropriate indicators from the conditional distribution of the state provided by individual estimators
3. evaluating the performance index

Performance indices implemented in the NEF

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

NEF in target tracking

- Tracking a ship with unknown control
- $\boldsymbol{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]$
- $$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}_k + \boldsymbol{w}_k$$
- $T = 0.02$, $\boldsymbol{w}_k$ is a Gaussian zero mean white noise with $Q = 10^{-6} \cdot \boldsymbol{I}$
- the control $\boldsymbol{u}_k$ is unknown $\implies$ will be appended to the state variable

NEF in target tracking – modelling dynamics

dynamics function:

```
F = [1 0 T 0;
     0 1 0 T;
     0 0 1 0;
     0 0 0 1];
G = [0 0 1 0;
     0 0 0 1]';
Fm = [F G;zeros(2,4) eye(2)];
fm = nefLinFunction(Fm,[],eye(6));
```

state noise:

```
Q = 1e-6*eye(4);
Qm = [Q zeros(4,2); zeros(2,4) 1e-3*eye(2)];
wm = nefGaussianRV(zeros(6,1),Qm);
```

NEF in target tracking – measurement

position measured in polar coordinates

$$\boldsymbol{z}_k = \begin{bmatrix} \arctan \frac{y_k}{x_k} \\ \sqrt{y_k^2 + x_k^2} \end{bmatrix} + \boldsymbol{v}_k,$$

$\boldsymbol{v}_k$ Gaussian zero mean white noise with covariance matrix
$R = \mathrm{diag}[4 \cdot 10^{-4}(\pi/180); 1 \cdot 10^{-4}]$
measurement function:

```
hm = nefHandleFunction(@(x,u,v,t) ...
[atan(x(2)/x(1));sqrt(x(1)^2+x(2)^2)] + v,[6 0 2 0]);
```

measurement noise:

```
R = [4e-4*(pi/180)^2 0; 0 1e-4];
v = nefGaussianRV(zeros(2,1),R);
```

NEF in target tracking – measurement

intial condition:

```
m0 = [20 50 0 -12 0 0]';
P0 = diag([1e1 1e1 1e1 1e1 1e-1 1e-1]);
x0m = nefGaussianRV(m0,P0);
```

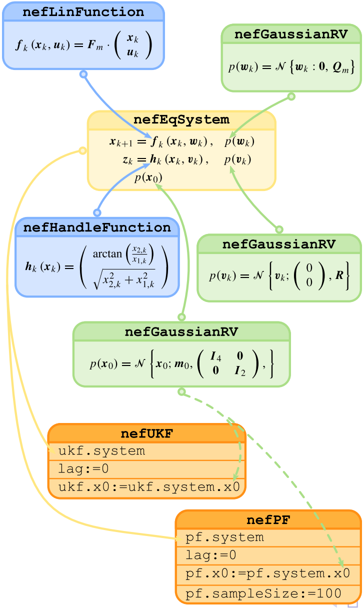
model:

```
model = nefEqSystem(fm,hm,wm,v,x0m,'logLikelihood',LLH);
```

estimators (PF and UKF)

```
PF = nefPF(model,'sampleSize',1000);
UKF = nefUKF(model);
```

NEF in target tracking – experiment setup



NEF in target tracking – performance evaluation

performance evaluators:

```
filters = 2;,  mcrun = 10;
RMSE_PE = nefPerformanceEvaluator(...
model,K,mcrun,filters,'method','RMSE','idxState',[1:4]);
NCI_PE = nefPerformanceEvaluator(...
model,K,mcrun,filters,'method','NCI','idxState',[1:4]);
```

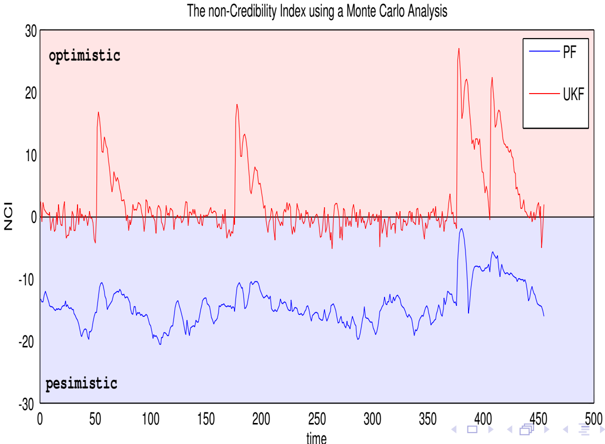
performing 10 Monte Carlo simulations:

```
for i = 1:mcrun
  [val_PF] = estimate(PF,z,u);
  [val_UKF] = estimate(UKF,z,u);
  for k = 1:K
    Data.state{1,k} = [x(:,k); u(:,k)];
    estData{1,1,k} = val_PF{k};
    estData{2,1,k} = val_UKF{k};
  end
  processData(RMSE_PE,Data,estData);
  processData(NCI_PE,Data,estData);
end
```

NEF in target tracking – performance evaluation (cont.)

performance evaluators: obtaining the performance indices:

```
rmse = performanceValue(RMSE_PE);
nci = performanceValue(NCI_PE);
```



Concluding remarks

a suitable tool for testing estimator performance in target-tracking problems

- it provides:
  - versatile description of discrete-time dynamic stochastic systems with continuous state,
  - a number of estimators,
  - a number of performance indexes.

Concluding remarks (cont.)

- Benefits for:

  beginners: problem specification which is reduced to its absolute minimum

  advanced users: the property-value mechanism to fully customize the experiment parameters

- In addition (not shown):
  - rapid prototyping of user-defined estimators
  - complex models or performance evaluators

http://nft.kky.zcu.cz/