Levenberg-Marquardt and Line-Search Extended Kalman Smoothers

LEVENBERG–MARQUARDT AND LINE-SEARCH EXTENDED KALMAN SMOOTHERS

Simo Särkkä**, Lennart Svensson*†

* Department of Electrical Engineering and Automation, Aalto University, Finland
† Dept. of Electrical Engineering, Chalmers University of Technology, Sweden
Emails: simo.sarkka@aalto.fi, lennart.svensson@chalmers.se

ABSTRACT

The aim of this article is to present Levenberg–Marquardt and line-search extensions of the classical iterated extended Kalman smoother (IEKS) which has previously been shown to be equivalent to the Gauss–Newton method. The algorithms are derived by rewriting the algorithm's steps in forms that can be efficiently implemented using modified EKS iterations. The resulting algorithms are experimentally shown to have superior convergence properties over the classical IEKS.

Index Terms— nonlinear estimation, Extended Kalman smoother, Levenberg–Marquardt algorithm, line search.

1. INTRODUCTION

This article is concerned with computation of maximum a posteriori (MAP) smoothing solutions to non-linear state-space models of the following form (see, e.g, [1–3]):

$$x_{k+1} = f_k(x_k) + q_k, \quad z_k = h_k(x_k) + r_k, \tag{1}$$

where $x_k \in \mathbb{R}^n$ are the states, and $z_k \in \mathbb{R}^{m_k}$ are the measurements, for $k = 1, \ldots, N$. The noises are assumed to be zero mean Gaussian with known covariance matrices, $q_k \sim \mathrm{N}(0, Q_k)$, $r_k \sim \mathrm{N}(0, R_k)$, and the initial prior is $x_1 \sim \mathrm{N}(\hat{x}_1, P_1)$ with given $\hat{x}_1$ and P_1. This kind of state-space models and their smoothing solutions have numerous applications, for example, in navigation, tracking, and audio and biomedical signal processing [1–3].

The aim of MAP smoothing is to compute the maximum of the posterior density, which is equivalent to minimization of the following cost function (see, e.g., [4]):

$$\begin{aligned} L(x_{1:N}) &= (x_1 - \hat{x}_1)^\top P_1^{-1} (x_1 - \hat{x}_1) \\ &\quad + \sum_{k=1}^{N} (z_k - h_k(x_k))^\top R_k^{-1} (z_k - h_k(x_k)) \\ &\quad + \sum_{k=1}^{N-1} (x_{k+1} - f_k(x_k))^\top Q_k^{-1} (x_{k+1} - f_k(x_k)). \end{aligned} \tag{2}$$

When the functions $h_k(x_k)$ and $f_k(x_k)$ are affine, the minimization can be done exactly by the Rauch–Tung–Striebel smoother [5].

In the nonlinear case, it is possible to use various optimization methods available in the literature [6] to minimize (2). However, when N is large, the minimization problem is very high-dimensional and using off-the-shelf optimization routines is challenging due to computational and memory demands. It is therefore beneficial to consider optimization methods that explicitly use the structure of the problem to minimize the computational requirements and memory use.

One classic method to solve nonlinear smoothing problems of the form (1) is the iterated extended Kalman smoother (IEKS). It was shown by Bell [7] already in 1994 that IEKS is equivalent to applying the Gauss–Newton method to minimizing the cost function (2). The IEKS makes use of the sparseness generated by the Markov structure of the problem to produce an algorithm with low computational and memory demands.

Compared to other Gaussian smoothing algorithms, such as the unscented Rauch–Tung–Striebel smoother [8] and the cubature Kalman smoother [9], the IEKS often yields better performance. Intuitively speaking, this is because the IEKS makes use of measurements from all times to iteratively improve the linearizations. A family of algorithms that explicitly seeks the best possible linearizations, are the iterated posterior linearizations [10, 11]. However, even though the iterated posterior linearization smoother [12, 13] tends to yield great performance the IEKS is a slightly faster algorithm that often has similar performance.

A disadvantage with the Gauss–Newton algorithm, and thus the IEKS algorithm, is that it is unstable and sometimes diverges. The iterated extended Kalman filters (IEKFs) suffer from the same problems, and a number of modifications have been proposed where the Gauss–Newton algorithm is replaced by a more robust alternative, for example, step-size adaptation, Levenberg–Marquardt, or other extensions [14–18]. A Levenberg–Marquardt extension of the ensemble Kalman filter/smoother was also developed in [19, 20].

The contribution of this article is to present two robustifying modifications of the IEKS algorithm: (1) an IEKS version of the Levenberg–Marquardt algorithm, and (2) IEKS based on the Gauss–Newton method with a line search. In the same sense as classic IEKS is equivalent to Gauss–Newton [7], these methods are equivalent to their batch counterparts.

2. LEVENBERG–MARQUARDT IEKS

Similarly to [7], let us now rewrite the cost function (2) in a convenient batch form by defining a block diagonal matrix

$$A = \operatorname{diag}(P_1^{-1}, R_1^{-1}, Q_1^{-1}, \ldots, Q_{N-1}^{-1}, R_N^{-1}) \tag{3}$$

and vector

$$v(x_{1:N}) = \begin{pmatrix} x_1 - \hat{x}_1 \\ z_1 - h_1(x_1) \\ x_2 - f_1(x_1) \\ z_2 - h_2(x_2) \\ \vdots \\ x_N - f_{N-1}(x_{N-1}) \\ z_N - h_N(x_N) \end{pmatrix}. \tag{4}$$

These definitions enable us to rewrite the cost function (2) in the following quadratic form:

$$L(x_{1:N}) = v^\top(x_{1:N}) A v(x_{1:N}). \tag{5}$$

This is a canonical nonlinear least squares optimization problem and hence methods such as Gauss–Newton and Levenberg–Marquardt are applicable. The Jacobian V of v can also be easily computed and the result is a sparse block matrix (left out for space reasons).

Both the Gauss–Newton and Levenberg–Marquardt algorithms [6,21,22] for the cost function (5) are based on the use of linearization

$$v(x_{1:N}) \approx v(x_{1:N}^{(i)}) + V(\hat{x}_{1:N}^{(i)})(x_{1:N} - \hat{x}_{1:N}^{(i)}). \tag{6}$$

Plugging this into the cost function gives the following approximate cost function:

$$\begin{aligned} L_{\mathrm{GN}}^{(i)}(x_{1:N}) &= [v(\hat{x}_{1:N}^{(i)}) + V(\hat{x}_{1:N}^{(i)})(x_{1:N} - \hat{x}_{1:N}^{(i)})]^\top A \\ &\quad \times [v(\hat{x}_{1:N}^{(i)}) + V(\hat{x}_{1:N}^{(i)})(x_{1:N} - \hat{x}_{1:N}^{(i)})]. \end{aligned} \tag{7}$$

The Gauss–Newton algorithm [6] now proceeds to minimize this approximate cost function (which can be done in closed form) and uses its solution as the next iterate $\hat{x}_{1:N}^{(i+1)}$.

To connect the above Gauss-Newton algorithm to IEKS, we note that minimizing (7) is equivalent to minimizing (2) for the affine model

$$\begin{aligned} x_{k+1} &= f_k(\hat{x}_k^{(i)}) + F_k(\hat{x}_k^{(i)})(x_k - \hat{x}_k^{(i)}) + q_k, \\ z_k &= h_k(\hat{x}_k^{(i)}) + H_k(\hat{x}_k^{(i)})(x_k - \hat{x}_k^{(i)}) + r_k, \end{aligned} \tag{8}$$

where F_k and H_k denote the Jacobians of f_k and h_k, respectively. The IEKS uses the Rauch–Tung–Striebel smoother to minimize (2) for the model in (8), in closed form. Since the Gauss-Newton algorithm finds an exact solution to the equivalent loss function in (7), the algorithms must be equivalent.

Let us now see how the Levenberg–Marquardt algorithm [6, 21, 22] can be expressed in this kind of form. In the algorithm we minimize the following regularized version of the approximate cost function

$$\begin{aligned} L_{\mathrm{LM}}^{(i)}(x_{1:N}) &= [v(\hat{x}_{1:N}^{(i)}) + V(\hat{x}_{1:N}^{(i)})(x_{1:N} - \hat{x}_{1:N}^{(i)})]^\top A \\ &\quad \times [v(\hat{x}_{1:N}^{(i)}) + V(\hat{x}_{1:N}^{(i)})(x_{1:N} - \hat{x}_{1:N}^{(i)})] \\ &\quad + \lambda (x_{1:N} - x_{1:N}^{(i)})^\top [S^{(i)}]^{-1} (x_{1:N} - \hat{x}_{1:N}^{(i)}) \end{aligned} \tag{9}$$

and use the result as the next iterate $\hat{x}_{1:N}^{(i+1)}$. Above $S^{(i)}$ is a sequence of given positive definite regularization matrices and $\lambda > 0$ is a reqularization parameter that is adapted as part of the Levenberg–Marquardt algorithm.

Let us now assume that the regularization matrix has a block-diagonal form $S^{(i)} = \operatorname{diag}(S_1^{(i)}, S_2^{(i)}, \ldots, S_N^{(i)})$, where each of the matrices on the right have the size of the state squared $S_k^{(i)} \in \mathbb{R}^{n \times n}$; a common choice would be to select $S_k^{(i)}$ to be the identity matrix. Then the last term of the approximate cost function (9) can be written as

$$\begin{aligned} &\lambda (x_{1:N} - x_{1:N}^{(i)})^\top [S^{(i)}]^{-1} (x_{1:N} - \hat{x}_{1:N}^{(i)}) \\ &= \sum_{k=1}^{N} (\hat{x}_k^{(i)} - x_k)^\top [\lambda^{-1} S_k^{(i)}]^{-1} (\hat{x}_k^{(i)} - x_k), \end{aligned} \tag{10}$$

which has the same form as the measurement term in (2). In this interpretation $\hat{x}_k^{(i)}$ acts as a measurement of x_k with noise covariance $\lambda^{-1} S_k^{(i)}$, thus incentivizing $\hat{x}_k^{(i+1)}$ to be close to $\hat{x}_k^{(i)}$. As the first terms of (9) still correspond to the affine model, the Levenberg–Marquardt step can be seen to be equivalent to running an affine smoother on the following model:

$$\begin{aligned} x_{k+1} &= f_k(\hat{x}_k^{(i)}) + F_k(\hat{x}_k^{(i)})(x_k - \hat{x}_k^{(i)}) + q_k, \\ z_k &= h_k(\hat{x}_k^{(i)}) + H_k(\hat{x}_k^{(i)})(x_k - \hat{x}_k^{(i)}) + r_k, \\ \hat{x}_k^{(i)} &= x_k + e_k, \end{aligned} \tag{11}$$

where $\hat{x}_k^{(i)}$ are treated as measurements in the last equation and $e_k \sim \mathrm{N}(0, \lambda^{-1} S_k^{(i)})$. The resulting regularized affine smoother is shown in Algorithm 1. It is also worth noting that the affine smoother used in the conventional IEKS can be recovered by setting $\lambda = 0$ in the algorithm.

The Levenberg–Marquardt algorithm can be constructed by iterating the Algorithm 1. However, the full algorithm requires the selection of the parameter λ. This can be implemented, for example, using a simple procedure [22,23], where we increase and decrease the parameter depending if the current value leads to increase of decrease in the cost function. The resulting method is shown as Algorithm 2.

3. LINE-SEARCH IEKS

Another way to improve the Gauss–Newton algorithm is to introduce a line-search procedure [6] to the algorithm. How-

Algorithm 1 Regularized IEKS Step

Input: The current trajectory $\hat{x}^c_{1:N}$, regularization parameter λ, regularization matrices $S_{1:N}$, and implicitly the measurements, model functions, Jacobians, and parameters: $z_{1:N}$, $f_{1:N}$, $h_{1:N}$, $F_{1:N}$, $H_{1:N}$, $Q_{1:N}$, $R_{1:N}$, $\hat{x}_1$, and P_1.

Output: The new smoothed trajectory $\hat{x}^s_{1:N}$.

```
1: procedure REG-IEKS-STEP($\hat{x}^c_{1:N}, \lambda, S_{1:N}$)
2:     for $k = 1, \ldots, N$ do
3:         if k = 1 then
4:             // Initial step:
5:             Set $\hat{x}^p_1 \leftarrow \hat{x}_1$ and $\hat{P}^p_1 \leftarrow P_1$
6:         else
7:             // Prediction step:
8:             $\hat{x}^p_k \leftarrow f_{k-1}(\hat{x}^c_{k-1}) + F_{k-1}(\hat{x}^c_{k-1})[\hat{x}^f_{k-1} - \hat{x}^c_{k-1}]$
9:             $\hat{P}^p_k \leftarrow F_{k-1}(\hat{x}^c_{k-1})\hat{P}^f_{k-1}F^\top_{k-1}(\hat{x}^c_{k-1}) + Q_{k-1}$
10:        end if
11:        // Update step:
12:        $\mu_k \leftarrow h_k(\hat{x}^c_k) + H_k(\hat{x}^c_k)[\hat{x}^p_k - \hat{x}^c_k]$
13:        $\Sigma_k \leftarrow H_k(\hat{x}^c_k)\hat{P}^p_k H^\top_k(\hat{x}^c_k) + R_k$
14:        $K_k \leftarrow \hat{P}^p_k H^\top_k(\hat{x}^c_k)\Sigma^{-1}_k$
15:        $\hat{x}^f_k \leftarrow \hat{x}^p_k + K_k[z_k - \mu_k]$
16:        $\hat{P}^f_k \leftarrow \hat{P}^p_k - K_k\Sigma_k[K_k]^\top$
17:        if $\lambda > 0$ then
18:            // Regularization update step:
19:            $\Sigma_k \leftarrow \hat{P}^f_k + \lambda^{-1}S_k$
20:            $K_k \leftarrow \hat{P}^f_k\Sigma^{-1}_k$
21:            $\hat{x}^f_k \leftarrow \hat{x}^f_k + K_k[\hat{x}^c_k - \hat{x}^f_k]$
22:            $\hat{P}^f_k \leftarrow \hat{P}^f_k - K_k\Sigma_k[K_k]^\top$
23:        end if
24:    end for
25:    Set $\hat{x}^s_N \leftarrow \hat{x}^f_N$ and $\hat{P}^s_N \leftarrow \hat{P}^f_N$
26:    for $k = N-1, \ldots, 1$ do
27:        // Smoothing step:
28:        $G_k \leftarrow \hat{P}^f_k F^\top_k(\hat{x}^c_k)[\hat{P}^p_{k+1}]^{-1}$
29:        $\hat{x}^s_k \leftarrow \hat{x}^f_k + G_k[\hat{x}^s_{k+1} - \hat{x}^p_{k+1}]$
30:        $\hat{P}^s_k \leftarrow \hat{P}^f_k + G_k[\hat{P}^s_{k+1} - \hat{P}^p_{k+1}]\,G^\top_k$
31:    end for
32:    return $\hat{x}^s_{1:N}$
33: end procedure
```

ever, when we use the IEKS implementation of the Gauss–Newton (Alg. 1 with $\lambda = 0$), there is no increment computed in the same sense as in the classical formulation of the Gauss–Newton. Fortunately, given the previous iterate $\hat{x}^{(i)}_{1:N}$ and the new iterate $\hat{x}^{(i+1)}_{1:N}$, we can compute the corresponding increment via $\Delta\hat{x}^{(i+1)}_{1:N} = \hat{x}^{(i+1)}_{1:N} - \hat{x}^{(i)}_{1:N}$. Then a line search version of the IEKS can be implemented by introducing a parameter $\alpha > 0$ and putting

$$\hat{x}^{(i+1)}_{1:N}(\alpha) = \hat{x}^{(i)}_{1:N} + \alpha\Delta\hat{x}^{(i+1)}_{1:N}. \qquad (12)$$

An exact line-search procedure can then be implemented using a grid search as shown in Algorithm 3.

Alternatively, we can use inexact line search and only require a sufficient decrease of the cost function by using the Armijo or Wolfe conditions [6]. These conditions can also be easily written in terms of the state-space model. The resulting method for Armijo conditions is shown as Algorithm 4. The Wolfe conditions could be implemented in an analogous manner.

Algorithm 2 Levenberg–Marquardt IEKS

Input: The initial trajectory $\hat{x}^0_{1:N}$, increase/decrease parameter $\nu > 1$ (e.g. $\nu = 10$), initial regularization parameter λ^0, regularization matrices $S_{1:N}$, and implicitly the measurements, model functions, Jacobians, and parameters: $z_{1:N}$, $f_{1:N}$, $h_{1:N}$, $F_{1:N}$, $H_{1:N}$, $Q_{1:N}$, $R_{1:N}$, $\hat{x}_1$, and P_1.

Output: The MAP trajectory $\hat{x}^*_{1:N}$.

```
1: procedure LM-IEKS($\hat{x}^0_{1:N}, \lambda^0, S_{1:N}$)
2:     Set $i \leftarrow 0$ and $\lambda \leftarrow \lambda^0$
3:     repeat
4:         $\hat{x}^{(i+1)}_{1:N} \leftarrow$ REG-IEKS-STEP($\hat{x}^{(i)}_{1:N}, \lambda, S_{1:N}$)
5:         if $L(\hat{x}^{(i+1)}_{1:N}) < L(\hat{x}^{(i)}_{1:N})$ then
6:             // Decrease damping and accept the iterate
7:             $\lambda \leftarrow \lambda/\nu$
8:             Set $i \leftarrow i + 1$
9:         else
10:            // Increase damping and reject the iterate
11:            $\lambda \leftarrow \nu\lambda$
12:        end if
13:    until Converged
14:    return $\hat{x}^{(i)}_{1:N}$
15: end procedure
```

4. EXPERIMENTAL RESULTS

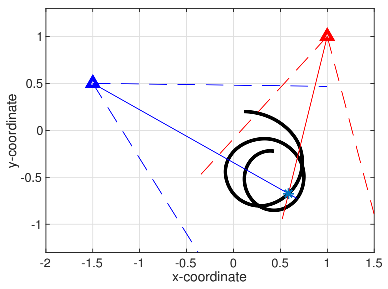


Fig. 1. Simulated coordinated turn bearings only tracking scenario. The blue and red triangles are the sensors and the 1σ quantiles are illustrated with dashed lines.

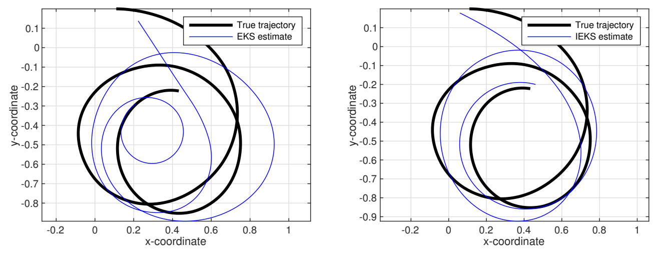
We illustrate the proposed methods in a coordinated turn model with a bearings (direction) only tracking problem [2]. The simulated tracking scenario is illustrated in Figure 1. The target is performing a steep coordinated turn and it is tracked with two directional sensors with measurement variances $\sigma^2 = 1/2^2$ rad^2. The measurement noise is relatively high as can be seen from the illustrated 1σ quantiles. The sensors are also relatively far from the target and hence the tracking problem is very non-linear.

Algorithm 3 Exact Line Search IEKS

```
Input: The initial trajectory $\hat{x}^0_{1:N}$ and implicitly the measurements, model functions, Jacobians, and parameters: $z_{1:N}$, $f_{1:N}$, $h_{1:N}$, $F_{1:N}$, $H_{1:N}$, $Q_{1:N}$, $R_{1:N}$, $\hat{x}_1$, and $P_1$.
Output: The MAP trajectory $\hat{x}^*_{1:N}$.
 1: procedure GN-IEKS-ELS($\hat{x}^0_{1:N}$)
 2:     Set $i \leftarrow 0$
 3:     repeat
 4:         $\hat{x}^{(i+1)}_{1:N} \leftarrow$ REG-IEKS-STEP($\hat{x}^{(i)}_{1:N}, 0, \emptyset$)
 5:         $\Delta\hat{x}_{1:N} \leftarrow \hat{x}^{(i+1)}_{1:N} - \hat{x}^{(i)}_{1:N}$
 6:         // Use grid search to find minimizing $\alpha$:
 7:         $\alpha \leftarrow \arg\min_\alpha L(\hat{x}^{(i)}_{1:N} + \alpha\Delta\hat{x}_{1:N})$
 8:         $\hat{x}^{(i+1)}_{1:N} \leftarrow \hat{x}^{(i)}_{1:N} + \alpha\Delta\hat{x}_{1:N}$
 9:         Set $i \leftarrow i + 1$
10:     until Converged
11:     return $\hat{x}^{(i)}_{1:N}$
12: end procedure
```

Algorithm 4 Inexact Line Search IEKS

```
Input: The initial trajectory $\hat{x}^0_{1:N}$, backtracking parameters $c_1, \tau$, and implicitly the measurements, model functions, Jacobians, and parameters: $z_{1:N}$, $f_{1:N}$, $h_{1:N}$, $F_{1:N}$, $H_{1:N}$, $Q_{1:N}$, $R_{1:N}$, $\hat{x}_1$, and $P_1$.
Output: The MAP trajectory $\hat{x}^*_{1:N}$.
 1: procedure GN-IEKS-ILS($\hat{x}^0_{1:N}, \tau$)
 2:     Set $i \leftarrow 0$
 3:     repeat
 4:         $\hat{x}^{(i+1)}_{1:N} \leftarrow$ REG-IEKS-STEP($\hat{x}^{(i)}_{1:N}, 0, \emptyset$)
 5:         $\Delta\hat{x}_{1:N} \leftarrow \hat{x}^{(i+1)}_{1:N} - \hat{x}^{(i)}_{1:N}$
 6:         // Use Armijo backtracking to find admissible $\alpha$:
 7:         $\alpha \leftarrow 1$
 8:         // Compute the directional derivative:
            $$\begin{aligned} d \leftarrow{}& 2\Delta\hat{x}_1^\top P_1^{-1}(\hat{x}^{(i)}_1 - \hat{x}_1) \\ &+ 2\sum_{k=2}^{N}(\Delta\hat{x}_k - F_{k-1}(\hat{x}^{(i)}_{k-1})\Delta\hat{x}_{k-1})^\top \\ &\quad\times Q_{k-1}^{-1}(\hat{x}^{(i)}_k - f_{k-1}(\hat{x}^{(i)}_{k-1})) \\ &- 2\sum_{k=1}^{N}(H_k(\hat{x}^{(i)}_k)\Delta\hat{x}_k)^\top R_k^{-1}(z_k - h_k(\hat{x}^{(i)}_k)) \end{aligned}$$
 9:         while $L(\hat{x}^{(i)}_{1:N} + \alpha\Delta\hat{x}_{1:N}) \geq L(\hat{x}^{(i)}_{1:N}) + \alpha c_1 d$ do
10:             $\alpha \leftarrow \tau\alpha$
11:         end while
12:         $\hat{x}^{(i+1)}_{1:N} \leftarrow \hat{x}^{(i)}_{1:N} + \alpha\Delta\hat{x}_{1:N}$
13:         Set $i \leftarrow i + 1$
14:     until Converged
15:     return $\hat{x}^{(i)}_{1:N}$
16: end procedure
```

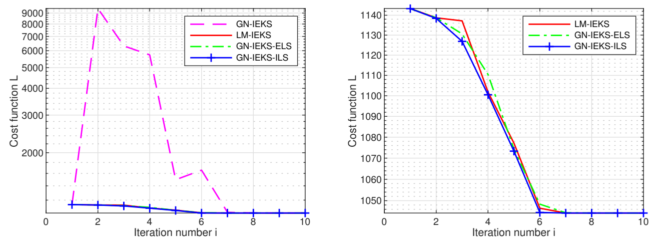


(a) EKS (RMSE 2.6) (b) IEKS (RMSE 0.6)

Fig. 2. EKS and IEKS trajectory estimates.

Due to high nonlinearity and high measurement noise, a single step of standard extended Kalman filter and smoother (Algs. 5.4 and 9.1 in [3]) produces a quite bad result shown in Figure 2(a) with root mean square error (RMSE) of 2.6 units. However, the iteration helps a lot as can be seen from the LM-IEKS result in Figure 2(b) where the RMSE is 0.6 units. Please note that all the other IEKS variations also produce the same estimate as LM-IEKS once they have converged, but that the original Gauss-Newton (IEKS) algorithm sometimes diverges when Levenberg–Marquardt does not [15].

The benefit of the proposed IEKS methods as compared to the standard IEKS can be seen in Figure 3(a) which shows the evolution of the cost functions of both the standard IEKS and the proposed improvements. The classic IEKS takes a long route through higher cost function values before converging to the minimum. Figure 3(b) shows the cost function evolutions of the proposed methods only where it can be seen that all the methods converge monotonically and almost equally fast to the optimum.



(a) All cost functions (b) Costs without classic IEKS

Fig. 3. Cost function convergence of the different methods.

5. CONCLUSION

In the article we have presented Levenberg–Marquardt and line-search extensions of iterated extended Kalman smoother (IEKS). The algorithms are equivalent to their batch versions when applied to a MAP state estimation problem, but they enjoy favourable computational and memory consumption properties. The algorithms were experimentally shown to converge better than the classical IEKS.

6. REFERENCES

[1] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, NY, 1970.

[2] Y. Bar-Shalom and X.-R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS, 1995.

[3] S. Särkkä, *Bayesian Filtering and Smoothing*, Cambridge University Press, 2013.

[4] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems," *IEEE Transactions on Automatic Control*, vol. 9, no. 1, pp. 5–12, January 1964.

[5] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[6] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2nd edition, 2006.

[7] B. M. Bell, "The iterated Kalman smoother as a Gauss–Newton method," *SIAM Journal on Optimization*, vol. 4, no. 3, pp. 626–636, 1994.

[8] S. Särkkä, "Unscented Rauch–Tung–Striebel smoother," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, pp. 845–849, 2008.

[9] I. Arasaratnam and S. Haykin, "Cubature Kalman smoothers," *Automatica*, vol. 47, no. 10, pp. 2245–2250, 2011.

[10] Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä, "Posterior linearization filter: principles and implementation using sigma points," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5561–5573, 2015.

[11] M. Raitoharju, L. Svensson, Á. F. García-Fernández, and R. Piché, "Damped posterior linearization filter," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 536–540, April 2018.

[12] Á. F. García-Fernández, L. Svensson, and S. Särkkä, "Iterated posterior linearization smoother," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 2056–2063, 2017.

[13] F. Tronarp, Á. F. García-Fernández, and S. Särkkä, "Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 408–412, 2018.

[14] H. Moriyama, N. Yamashita, and M. Fukushima, "The incremental Gauss-Newton algorithm with adaptive stepsize rule," *Computational Optimization and Applications*, vol. 26, no. 2, pp. 107–141, 2003.

[15] R. L. Bellaire, E. W. Kamen, and S. M. Zabin, "New nonlinear iterated filter with applications to target tracking," in *Signal and Data Processing of Small Targets*, 1995, vol. 2561, pp. 240–251.

[16] M. Fatemi, L. Svensson, L. Hammarstrand, and M. Morelande, "A study of MAP estimation techniques for nonlinear filtering," in *15th International Conference on Information Fusion (FUSION)*, 2012, pp. 1058–1065.

[17] Á. F. García-Fernández and L. Svensson, "Gaussian MAP filtering using Kalman optimization," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1336–1349, 2014.

[18] M. Skoglund, G. Hendeby, and D. Axehill, "Extended Kalman filter modifications based on an optimization view point," in *18th International Conference on Information Fusion (FUSION)*, 2015, pp. 1856–1861.

[19] Y. Chen and D. S. Oliver, "Levenberg–Marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification," *Computational Geosciences*, vol. 17, no. 4, pp. 689–703, 2013.

[20] J. Mandel, E. Bergou, S. Gürol, S. Gratton, and I. Kasanický, "Hybrid Levenberg–Marquardt and weak-constraint ensemble Kalman smoother method," *Nonlinear Processes in Geophysics*, vol. 23, no. 2, pp. 59–73, 2016.

[21] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[22] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[23] J. Pujol, "The solution of nonlinear inverse problems and the Levenberg-Marquardt method," *Geophysics*, vol. 72, no. 4, pp. W1–W16, 2007.