Gorad, Ajinkya; Zhao, Zheng; Särkkä, Simo

Parameter estimation in non-linear state-space models by automatic differentiation of non-linear kalman filters

Published in:
Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing, MLSP 2020

DOI:
10.1109/MLSP49062.2020.9231844

Published: 01/09/2020

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Gorad, A., Zhao, Z., & Särkkä, S. (2020). Parameter estimation in non-linear state-space models by automatic differentiation of non-linear kalman filters. In *Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing, MLSP 2020* Article 9231844 (IEEE International Workshop on Machine Learning for Signal Processing). IEEE. https://doi.org/10.1109/MLSP49062.2020.9231844

PARAMETER ESTIMATION IN NON-LINEAR STATE-SPACE MODELS BY AUTOMATIC DIFFERENTIATION OF NON-LINEAR KALMAN FILTERS

Ajinkya Gorad, Zheng Zhao, and Simo Särkkä

Department of Electrical Engineering and Automation
Aalto University, Finland

ABSTRACT

In this article, we propose automatic differentiation based methods for parameter estimation in non-linear state-space models. We use extended Kalman filter and cubature Kalman filters for approximating the negative log-likelihood (i.e., the energy function) of the parameter posterior distribution and compute the gradients and Hessians of this function by using automatic differentiation of the filter recursions. The proposed approach enables computing MAP estimates and forming Laplace approximations for the parameter posterior without a need for implementing complicated derivative recursions or manual computation of Jacobians. The methods are demonstrated in parameter estimation problems on a pendulum model and coordinated turn model.

***Index Terms*—** automatic differentiation, parameter estimation, non-linear state space model, extended Kalman filter, cubature Kalman filter

1. INTRODUCTION

Automatic differentiation [1] is a methodology that has gained a lot of attention lately due to its usefulness in deep learning [2]. In particular, it is useful in computing gradients of models where the parameter appear in complicated manner in expression and hence manual computation of the gradient would be tedious. The aim of this article is to discuss the use of automatic differentiation in the context of parameter estimation in non-linear state-space models [3, 4].

In the context of state-space models the need for computing derivatives first arises from the linearizations used by the extended Kalman filter (EKF) which are based on the use of Jacobians of the non-linear functions in the model. Second, when computing estimates for the parameters in the model by using optimization methods on top of an EKF, the log marginal likelihood computed by the EKF needs to be differentiated with respect to the parameters which corresponds to differentiation of the whole EKF recursions with respect to them. Although so-called sigma-point filters such as unscented Kalman filters (UKFs, e.g., [4]) do not require the computation of Jacobians for the filter evaluation, they need to for the log marginal likelihood computation. Although (complicated) explicit recursions for these derivatives have been derived (see, e.g., [5, 4, 6]), we argue that the derivatives are much easier to compute by automatically differentiating the EKF and sigma-point filtering algorithms.

The idea in background of automatic differentiation is that any function evaluation done by a computer is performed by applying simple primitive operations such as summation, subtraction, multiplication as well as simple functions such as $\sin$, $\cos$, or $\exp$. Provided that we have a rule to compute the derivative of each of these primitive operations, we can automatically compute the derivative of any function "on the side" while evaluating the value of the function. This requires augmentation of the evaluation of the primitive operations with the corresponding derivative computations. This can be implemented, for example, by using operator overloading [7]. Mathematically this procedure can be formulated by replacing the real numbers in the computations with so called dual numbers [8] which contain the value part and the derivative part and the primitive operations are redefined so that this property is preserved in the computations. In this article we use operator overloading to implement the automatic differentiation.

The contributions of this article are methodologies for (1) automatic differentiation for implementation of EKFs and (2) automatic differentiation of for parameter estimation in non-linear state-space models as well as (3) an experimental evaluation of the methods.

2. AUTOMATIC DIFFERENTIATION VIA OPERATOR OVERLOADING

The concept of automatic differentiation can be understood through dual numbers [8]. They involve an hypothetical number ϵ such that $\epsilon^2 = 0$. For a dual number $x + \epsilon y$, the value of function f is given as $f(x + \epsilon y) = f(x) + f'(x)\epsilon y$ which allows us to compute derivatives by simply evaluating the function at the dual number. It is not only useful to compute the

The authors would like to thank Academy of Finland and Aalto ELEC doctoral school for funding.

derivative of a single function, but the output of the function is a dual number which can be acted upon by another function to give the combined derivative result.

Automatic differentiation can be implemented by extending the evaluation of functions to work with dual numbers instead of ordinary numbers. A major advantage of automatic differentiation over numerical differentiation is that it provides the derivative of a composite function in terms of derivatives of simpler composite functions and gives the exact result. Furthermore, automatic differentiation eliminates errors that might occur in manual computation of derivatives, because derivative computation is automatically done during the evaluation of the function.

Automatic differentiation can be implemented through a class which uses operator overloading for required mathematical functions for dual numbers [7]. It allows the existing operators and functions to be redefined for this new class. Using this approach, we obtain so-called forward-mode automatic differentiation methodology. We implement this kind of forward-mode automatic differentiation for parameter estimation in extended and cubature Kalman filtering, but reverse-mode automatic differentiation is more popular in deep learning [2]. The number of evaluations of the function in forward-mode automatic differentiation is proportional to the number of inputs, while for reverse mode automatic differentiation, it is proportional on the outputs [1]. However, the computational overhead, implementation complexity, and memory footprint of the reverse-mode automatic differentiation is much higher than of the forward mode, and therefore we only consider the forward-mode automatic differentiation in this article.

The class for the dual numbers can be defined with properties v and dv which correspond to $f(x)$ and $f'(x)$, respectively. Properties v and dv need not be numbers, but they can be dual numbers themselves which allows for computation of higher order derivatives easily. Example snippet of such an implementation is shown here (in Matlab notation):

```
classdef Dual
    properties
        v
        dv
    end

    methods
        .
        .
        .
end
```

Methods of the class contains all overloaded operators and functions like '+' plus, '-' minus, sin, cos, log, exp and so on. These overloaded functions f when invoked with a dual number objects x, they produce dual number object corresponding to a pair $(f(x), f'(x))$. For example, plus and sin can be implemented as

```
function res = plus(v1,v2)
    res = Dual(v1.v+v2.v,
               v1.dv+v2.dv);
end
```

```
function res = sin(v)
    res = Dual(sin(v.v),
               v.dv .* cos(v.v));
end
```

The additional operations needed for evaluating the second arguments in the dual operations can be easily derived using the elementary differentiation rules of primitive operations together with the chain rule. For log-determinant evaluation and sigma-point filter implementation we also need the derivative of the Cholesky factorization for which algorithms can be found in appendix of [4].

Provided that we allow the methods to be called with Dual objects whose members are of Dual type themselves, this allows for computation of higher order derivatives without without need of explicitly writing them in the code [7]. This can be accomplished by essentially differentiating the automatic differentiation process by recursively calling the automatic differentiator. Therefore, once we have written a code for computing a Jacobian, we can use the same code to compute Hessians and higher order derivatives simply by calling the automatic differentiator recursively.

3. AUTOMATIC DIFFERENTIATION OF GAUSSIAN FILTERS

Consider a non-linear discrete-time state-space model (cf. [4])

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}; \boldsymbol{\theta}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k; \boldsymbol{\theta}) + \mathbf{r}_k, \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^D$ is the state, $\mathbf{y}_k \in \mathbb{R}^Z$ is the measurement, and $\mathbf{f}$ and $\mathbf{h}$ are some regular non-linear functions. The random variables $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1}(\boldsymbol{\theta}))$ and $\mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k(\boldsymbol{\theta}))$ are assumed to be Gaussian endowed with covariances $\mathbf{Q}_{k-1}(\boldsymbol{\theta})$ and $\mathbf{R}_k(\boldsymbol{\theta})$, respectively. In the above model (1), $\boldsymbol{\theta} \in \mathbb{R}^S$ is a vector containing the unknown parameters of the model to be estimated. Moreover, we use Bayesian approach for parameter estimation and postulate a prior on the unknown parameters, such that $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$.

Now suppose that we have a set of measurements $\mathbf{y}_{1:T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, the aim is to learn the posterior

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:T})}. \tag{2}$$

It is also convenient to factorize the likelihood by

$$p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) = p(\mathbf{y}_1 \mid \boldsymbol{\theta}) \prod_{k=2}^{T} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}). \tag{3}$$

Unfortunately, the posterior (2) can not be obtained in closed form, except for linear models or several isolated cases. Maximum a posteriori (MAP) estimation is a practical and efficient method to approximate the parameter posterior by

$$\boldsymbol{\theta}^{\mathrm{MAP}} = \arg\min_{\boldsymbol{\theta}} \varphi(\boldsymbol{\theta}), \tag{4}$$

where the loss (energy) function is defined to be the unnormalized negative logarithm of the posterior distribution:

$$\begin{aligned}\varphi(\boldsymbol{\theta}) &= -\log p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \\ &= -\sum_{k=1}^{T} \log p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}).\end{aligned} \tag{5}$$

Given the MAP estimate (4), we can also form a Gaussian approximation on the posterior distribution, which is called the Laplace approximation [9]

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{\mathrm{MAP}}, \mathbf{B}^{-1}), \tag{6}$$

where $\mathbf{B}$ is the Hessian of the energy function $\varphi(\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}^{\mathrm{MAP}}$.

To proceed with the loss function, we also need to obtain an approximation to $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$. Formally, $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ can be computed as

$$\begin{aligned}&p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) \\ &= \int p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{x}_k.\end{aligned}$$

Note that the density $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_k \mid \mathbf{h}(\mathbf{x}_k; \boldsymbol{\theta}), \mathbf{R}_k(\boldsymbol{\theta}))$ is known by definition in (1). In the Gaussian filtering framework [4] we approximate both the filtering distributions $p(\mathbf{x}_k \mid \mathbf{y}_{1:k}, \boldsymbol{\theta})$ and predicted distributions $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ as Gaussian. Further approximating the distribution $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ as Gaussian the leads to the following approximation for the energy function.

We first compute the posterior means $\mathbf{m}_k(\boldsymbol{\theta})$ and covariances $\mathbf{P}_k(\boldsymbol{\theta})$) for each $k = 1, \ldots, T$ using the following recursions:

$$\begin{aligned}\mathbf{m}_k^-(\boldsymbol{\theta}) &= \mathbb{E}[\mathbf{f}(\mathbf{x}_{k-1}; \boldsymbol{\theta})], \\ \mathbf{P}_k^-(\boldsymbol{\theta}) &= \mathrm{Cov}[\mathbf{f}(\mathbf{x}_{k-1}; \boldsymbol{\theta})] + \mathbf{Q}_{k-1}(\boldsymbol{\theta}), \\ \mathbf{S}_k(\boldsymbol{\theta}) &= \mathrm{Cov}[\mathbf{h}(\mathbf{x}_k; \boldsymbol{\theta})] + \mathbf{R}_k(\boldsymbol{\theta}), \\ \mathbf{C}_k(\boldsymbol{\theta}) &= \mathrm{Cov}[\mathbf{x}_k, \mathbf{h}(\mathbf{x}_k; \boldsymbol{\theta})], \\ \mathbf{K}_k(\boldsymbol{\theta}) &= \mathbf{C}_k(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta}), \\ \mathbf{v}_k(\boldsymbol{\theta}) &= \mathbf{y}_k - \mathbb{E}[\mathbf{h}(\mathbf{x}_k; \boldsymbol{\theta})], \\ \mathbf{m}_k(\boldsymbol{\theta}) &= \mathbf{m}_k^-(\boldsymbol{\theta}) + \mathbf{K}_k(\boldsymbol{\theta})\, \mathbf{v}_k(\boldsymbol{\theta}), \\ \mathbf{P}_k(\boldsymbol{\theta}) &= \mathbf{P}_k^-(\boldsymbol{\theta}) - \mathbf{K}_k(\boldsymbol{\theta})\, \mathbf{S}_k(\boldsymbol{\theta})\, \mathbf{K}_k^\top(\boldsymbol{\theta}).\end{aligned} \tag{7}$$

For the derivation of the above Gaussian filter, we refer to [4] for details. By using the Gaussian filter result, we can now write down the approximate loss (energy) function $\varphi(\boldsymbol{\theta})$ as

$$\begin{aligned}\varphi(\boldsymbol{\theta}) \approx \frac{1}{2} \sum_{k=1}^{T} &\left[\mathbf{v}_k^\top(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \mathbf{v}_k(\boldsymbol{\theta}) + \log \det(2\pi\, \mathbf{S}_k(\boldsymbol{\theta}))\right] \\ &- \log p(\boldsymbol{\theta}).\end{aligned} \tag{8}$$

Taking the derivative of the above $\varphi(\boldsymbol{\theta})$ with respect to the i-th element of parameter $\boldsymbol{\theta}$ yields

$$\begin{aligned}\frac{\partial \varphi(\boldsymbol{\theta})}{\partial \theta_i} = \sum_{k=1}^{T} \Bigg[&-\frac{1}{2} \mathbf{v}_k^\top(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\, \mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \mathbf{v}_k(\boldsymbol{\theta}) \\ &+ \mathbf{v}_k^\top(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \frac{\partial \mathbf{v}_k(\boldsymbol{\theta})}{\partial \theta_i} \\ &+ \frac{1}{2} \operatorname{tr}\left(\mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\right) \Bigg],\end{aligned} \tag{9}$$

and the recursion for the derivatives needed above can be derived by (forward) differentiating the recursions (7) (cf. [4]). However, when approximations for computing the expectations in the filter equations are used, the equations soon become complicated and often need not only Jacobians, but also Hessians of the non-linear functions appearing in the state-space model. For this reason we propose to use automatic differentiation instead.

Given that we have implemented the Dual class as described in Section 2, we can implement the automatic computation of Jacobians and Hessians as follows:

1. Replace the unknown the parameter to be differentiated against with an instance of the Dual class with $dv = 1$ and other variables with $dv = 0$. If we wish to compute higher order derivatives, we can use a suitable hierarchy of Duals.
2. Run recursion (7) to get the means and covariances as Duals. This can be done with the same code as a conventional Gaussian filter due to operator overloading.
3. Compute the energy function with (8), which results in a Dual where the value is given by the v member and derivative by the dv member. In the case of higher order derivatives we need to obtain the results from deeper in the Dual hierarchy, but the principle remains the same.

The Jacobian and Hessian can be evaluated by repeating the above procedure for each parameter/parameter-pair.

4. AUTOMATIC DIFFERENTIATION OF EXTENDED KALMAN FILTERS AND SIGMA POINTS FILTERS

The Gaussian filter approach in the previous section, in principle, gives the gradient and Hessian of the energy function which are all that we need to gradient-based finding of the MAP estimate (4) (using, e.g., quasi-Newton methods [10]) and for forming the Laplace approximation (6). However, it is in practice challenging to perform the general Gaussian filters, as the integrals in (7) are intractable for many non-linear state space models. Thus we need to resort approximate Gaussian filters such as extended Kalman filters (EKF) and sigma points filters [3, 4] instead.

The idea of EKF is to form a linear approximation to the non-linear functions in (1) by using a Taylor expansion, such that

$$\begin{aligned}\mathbf{f}(\mathbf{x};\boldsymbol{\theta}) &\approx \mathbf{f}(\mathbf{m};\boldsymbol{\theta}) + \mathbf{F}_x(\mathbf{m};\boldsymbol{\theta})(\mathbf{x}-\mathbf{m}),\\ \mathbf{h}(\mathbf{x};\boldsymbol{\theta}) &\approx \mathbf{h}(\mathbf{m};\boldsymbol{\theta}) + \mathbf{H}_x(\mathbf{m};\boldsymbol{\theta})(\mathbf{x}-\mathbf{m}),\end{aligned} \tag{10}$$

where $\mathbf{F}_x$ and $\mathbf{H}_x$ are the Jacobians of $\mathbf{f}$ and $\mathbf{h}$ with respect to state $\mathbf{x}$, respectively. The expansion is centered at $\mathbf{m}$ which is usually the mean of the previous state. By substituting the approximation (10) into model (1), the model is then affine with respect to state $\mathbf{x}$, and the equations in (7) reduce to the classical Kalman filter equations (see e.g., [4, Algorithm 5.4]).

However, given that we have our Dual class available for automatic differentiation, we can use it to compute the Jacobians $\mathbf{F}_x$ and $\mathbf{H}_x$ by automatic differentiation. The automatic differentiator function can be itself differentiated by using the approach outlined in the previous section. Hence, we can implement the EKF and parameter estimation without computing any derivatives by hand.

As for the sigma point filters (SPF), they do not need to compute Jacobian of the non-linear state-space model functions. Instead, they use numerical integration to approximate the Gaussian integrals (expectations) in Equations (7). A typical approximation has the Gauss form

$$\mathbb{E}[\mathbf{f}(\mathbf{x}_k;\boldsymbol{\theta})] \approx \sum_{n=1}^{N} W_n \mathbf{f}(\mathbf{m}_k(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_k(\boldsymbol{\theta})}\,\boldsymbol{\xi}_n;\boldsymbol{\theta}), \tag{11}$$

where N is the number of sigma points, and the matrix square root is defined as $\sqrt{\mathbf{P}_k(\boldsymbol{\theta})}\,\sqrt{\mathbf{P}_k(\boldsymbol{\theta})}^{\top} = \mathbf{P}_k(\boldsymbol{\theta})$. Many methods such as unscented transform, cubature rule, and Gauss–Hermite method give the pairs $(W_n, \boldsymbol{\xi}_n)$ consisting of the sigma point $\boldsymbol{\xi}_n$ and its corresponding weight W_n for $n = 1, \ldots, N$.

It is also challenging to analytically obtain the parameter gradient in the sigma points based filters, though some explicit expression for parameter gradients are given in [4, A. 4] and [6]. As we can see from Equation (11), the parameter $\boldsymbol{\theta}$ appears inside the evaluation of non-linear function. The parameter is also engaged in the Cholesky decomposition. Therefore it is very advantageous to use the automatic differentiation to obtain the gradient and Hessian also in sigma-point filters.

The general procedure for obtaining the energy derivatives of the approximate Gaussian filters is thus the following:

1. Use Duals to compute form the Jacobians of the non-linearities of needed (as in EKF) or use sigma-point rules to approximate the expectations needed in (7).
2. Use the method outlined in Section 3 to compute the energy function derivatives.
3. Compute the MAP estimate by using a gradient based optimization method and compute the Hessian, which can then be used to form the Laplace approximation (6).

5. EXPERIMENTAL RESULTS

In this section, we demonstrate the methodology in two non-linear models: a pendulum model and a coordinated turn model.

5.1. Pendulum model

We first consider a pendulum model

$$\begin{aligned}\begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} &= \begin{bmatrix} x_{1,k-1} + x_{2,k-1}\Delta T \\ x_{2,k-1} - g\,\Delta T\,\sin(x_{1,k-1}) \end{bmatrix} + \mathbf{q}_k,\\ y_k &= \sin(x_{1,k}) + r_k,\end{aligned} \tag{12}$$

where $\mathbf{x}_k = \begin{bmatrix} x_{1,k} & x_{2,k} \end{bmatrix}^{\top}$ is the state and $\mathbf{q}_k$ is a zero mean Gaussian random variable with covariance

$$\mathbf{Q} = 0.01 \begin{bmatrix} \frac{\Delta T^3}{3} & \frac{\Delta T^2}{2} \\ \frac{\Delta T^2}{2} & \Delta T \end{bmatrix}, \tag{13}$$

and $r_k \sim \mathcal{N}(0, R)$. We simulate the model with parameter $g = 9.81$ and time interval $\Delta T = 0.01$ in 500 time steps. The task on this model is to estimate the variance R of the measurement noise r_k, which is set to be $R = 0.1$.

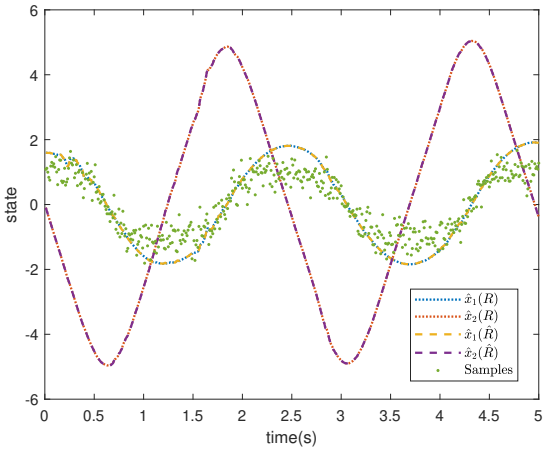

Fig. 1. Simulated data and estimation result for the estimation of measurement noise using EKF in the pendulum model. Model with $R = 0.1$ was used to generate the samples, while MAP estimate was $\hat{R} = 0.089$. In the figure, x_1 and x_2 are states of the model.

We used the EKF and CKF approximations to the Gaussian filter. With EKF, in the BFGS-based quasi-optimization method we used the initial estimate of $R_0 = 0.2$. The loss at the true parameter at $R = 0.1$ was found to be 113.97, while the optimization resulted in $\hat{R} = 0.089$ with loss of 112.59. Variance obtained from the Hessian computed at the MAP estimate was 3.25×10^{-5}. For CKF, with the same initial estimate, loss from true parameters at $R = 0.1$ was found to be 114.1, while the optimization resulted in $\hat{R} = 0.09$ with

loss of 112.8. Variance obtained from the Hessian computed at the MAP estimate was 3.27×10^{-5}. These results are summarized in Table 1.

	$\hat{R}$	$\varphi(R)$	$\varphi(\hat{R})$	$\left(\frac{\partial^2 \varphi}{\partial R^2}\right)^{-1}$
EKF	0.09	113.97	112.59	3.25×10^{-5}
CKF	0.09	114.10	112.80	3.27×10^{-5}

Table 1. Summary of results of the pendulum example.

The resulting Laplace approximations along with the "true" (EKF or CKF approximated) posterior distributions are shown in Figure 2). For visualization purposes we have normalized the densities to have maximum of 1. The Laplace approximations closely match the (approximate) true posteriors and the true parameter value is well withing the posterior probability mass.

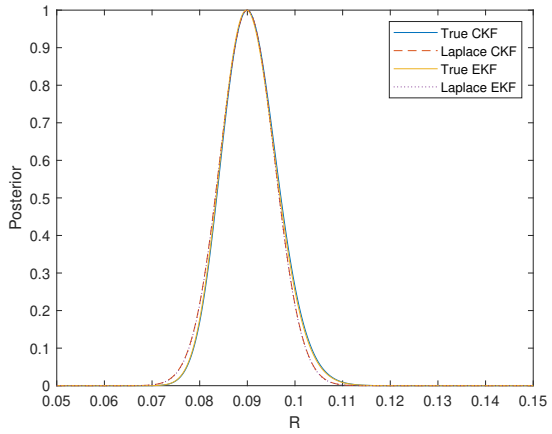

Fig. 2. Estimated posterior for the pendulum example obtained through EKF and CKF as well as using the Laplace approximation.

5.2. Coordinated turn model with unknown angular velocity

Next model we look at is the coordinated turn model with unknown angular velocity [3, 11]. This model has dynamic model with transition function $\mathbf{f}$ given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega\Delta T)}{\omega} & -\frac{1-\cos(\omega\Delta T)}{\omega} & 0 \\ 0 & 1 & \frac{1-\cos(\omega\Delta T)}{\omega} & \frac{\sin(\omega\Delta T)}{\omega} & 0 \\ 0 & 0 & \cos(\omega\Delta T) & -\sin(\omega\Delta T) & 0 \\ 0 & 0 & \sin(\omega\Delta T) & \cos(\omega\Delta T) & 0 \\ 0 & 0 & 0 & 0 & e^{-\lambda\Delta T} \end{bmatrix} \mathbf{x}, \tag{14}$$

where $\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y & \omega \end{bmatrix}^\top$ is the state of the system with coordinates (x,y), velocity (v_x,v_y), and angular velocity ω. Decay of angular rate ω is controlled by the parameter λ.

The process noise covariance is given by

$$\mathbf{Q} = \mathbf{B} \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & q_w \end{bmatrix} \mathbf{B}^\top \tag{15}$$

where

$$\mathbf{B} = \begin{bmatrix} \Delta T^2 & 0 & 0 \\ 0 & \Delta T^2 & 0 \\ \Delta T & 0 & 0 \\ 0 & \Delta T & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

Thus the speed in this model is driven by noise defined by q, and the angular velocity is driven by noise defined by parameter q_w.

The measurement model is provided by

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1} \frac{y}{x} \end{bmatrix}, \tag{17}$$

where the range and bearing are measured while the measurement noise covariance is

$$\mathbf{R} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}. \tag{18}$$

We estimate the parameters $\theta = (\log \lambda, \log q_w)$ with energy optimization using a BFGS-based quasi-Newton algorithm. The gradients are specified from the Jacobians of the energy functions of EKF and CKF calculated using automatic differentiation. Simulated measurements were generated for $\lambda_0 = 5$ and $q_w = 0.2$ with time step $\Delta T = 0.005$ for 250 sample points. Measurements are shown in Fig.3 for EKF and Fig. 4 for CKF filtering on original parameters. Values of other parameters are taken $q = 200$, $r_1 = 0.01$, $r_2 = 0.004$, with initial state $\mathbf{x}_0 = \begin{bmatrix} 2 & 2 & 10 & 0 & 4 \end{bmatrix}^\top$.

The initial estimates of 7.5 for λ and 0.3 for q_w were given. EKF energy based optimization gave optimized parameter $\hat{\lambda} = 3.08$ and $\hat{q}_w = 0.15$, while CKF energy based optimization resulted in $\hat{\lambda} = 4.41$ and $\hat{q}_w = 0.17$, which are not far away from the true values $\lambda = 4$ and $q_w = 0.2$. The results are summarized in Table 2.

	$\varphi(\theta)$	$\varphi(\hat{\theta})$	$\hat{\lambda}$	$\hat{q}_w$
EKF	-566.6485	-566.8693	3.08	0.15
CKF	-566.3731	-566.3865	4.42	0.17

Table 2. Summary of results of the coordinated turn example.

The optimal points are shown on the posterior in Figure 5 (whose maxima are normalized to 1) as calculated via automatically differentiated EKF and CKF, respectively. It can be seen that the optimization method has indeed found the correct optima.

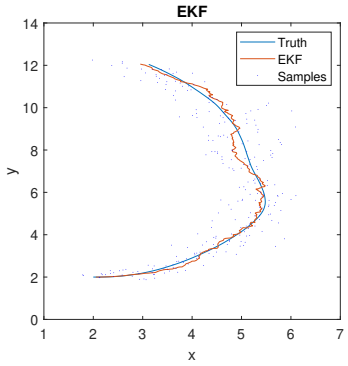

Fig. 3. Trajectory and measurement samples with EKF using original parameters.

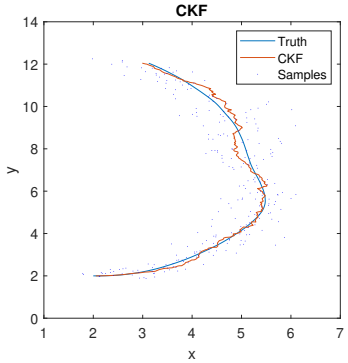

Fig. 4. Trajectory and measurement samples with CKF filtering using original parameters.

6. CONCLUSION

In this article, we have proposed the use of automatic differentiation for parameter estimation in non-linear state-space models by using automatic differentiation on top of extended and sigma-point filters. Using automatic differentiation gradients, higher order derivatives, and mixed derivatives can be computed exactly, and we have demonstrated how they can be used to form Laplace approximations for the parameter posterior distributions.

7. REFERENCES

[1] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2nd edition, 2008.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[3] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, Wiley, 2001.

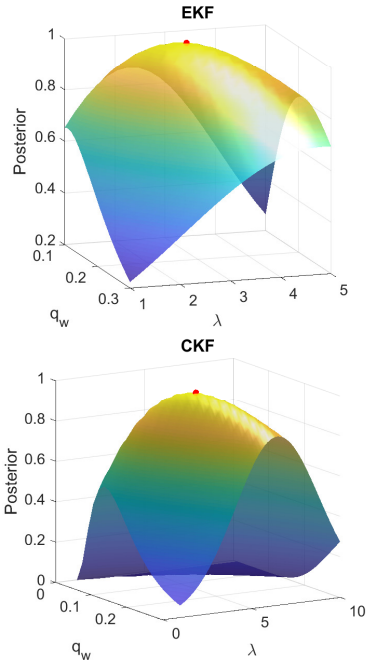

Fig. 5. Posterior for EKF (top) and CKF (bottom) showing optima at $\hat{\lambda} = 3.08$ and $\hat{q}_w = 0.15$ for EKF and $\hat{\lambda} = 4.42$ and $\hat{q}_w = 0.17$ for CKF.

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

[5] I. S Mbalawata, S. Särkkä, and H. Haario, "Parameter estimation in stochastic differential equations with Markov chain Monte Carlo and non-linear Kalman filtering," *CompStat*, vol. 28, no. 3, pp. 1195–1223, 2013.

[6] J. Kokkala, A. Solin, and S. Särkkä, "Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems," *JAIF*, vol. 11, no. 1, pp. 15–30, 2015.

[7] R. D. Neidinger, "Introduction to automatic differentiation and MATLAB object-oriented programming," *SIAM Review*, vol. 52, no. 3, pp. 545–563, 2010.

[8] I. Fischer, *Dual-Number Methods in Kinematics, Statics and Dynamics*, CRC Press, 1998.

[9] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, Chapman and Hall/CRC, 3rd edition, 2013.

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

[11] M. Roth, G. Hendeby, and F. Gustafsson, "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity," in *Proc. FUSION*, 2014, pp. 1–8.