

# Stacked iterated posterior linearization filter

Matti Raitoharju  
Patria Aviation  
Tampere University  
Aalto University  
Tampere, Finland

matti.raitoharju@patriagroup.com

Ángel F. García-Fernández  
University of Liverpool  
Liverpool, UK

Angel.Garcia-Fernandez@liverpool.ac.uk

Simo Ali-Löytty  
Tampere University  
Tampere, Finland

simo.ali-loytty@tuni.fi

Simo Särkkä  
Aalto University  
Helsinki, Finland

simo.sarkka@aalto.fi

**Abstract**—The Kalman Filter (KF) is a classical algorithm that was developed for estimating a state that evolves in time based on noisy measurements by assuming linear state transition and measurements models. There exist various KF extensions for non-linear situations, but they are not exact and provide different linearization errors. The Iterated Posterior Linearization Filter (IPLF) does the linearizations iteratively to achieve better linearizations. However, it is possible that some measurements cannot be well linearized using the current knowledge, but their linearization may be better after more measurements are available. Thus, we propose an algorithm that can store the older state elements and measurements when their linearization error is high. The resulting algorithm, the Stacked Iterated Posterior Linearization Filter (S-IPLF), is based on linear dynamic models and uses information from multiple time instances to make the linearization of the measurement function. Results show that the proposed algorithm outperforms traditional KF extensions when some of the measurements cannot be well linearized with the current knowledge, but can be when future information is available.

**Index Terms**—Kalman filtering, Bayesian filtering, posterior linearization.

## I. INTRODUCTION

The target of this work is to develop a method that improves state estimation with non-linear measurements compared to standard non-linear Kalman Filter (KF) extensions [1]. The KF [2] is optimal when it is used in a system that consists of linear state transformations between time steps and linear measurements. For non-linear measurement and state transition models, the KF has been extended in various ways. The Extended Kalman Filter (EKF) [3] does the linearization by differentiating the non-linear functions at the mean. The Unscented Kalman Filter (UKF) [4] on the other hand does not need the explicit differentiation and is based on evaluating functions at multiple points, so-called sigma points, which match the mean and covariance of the prior density. The Cubature Kalman Filter (CKF) [5] is also based on the use of sigma points. There also exist other KF extensions known as statistical linearization filters that make the statistical linearization in different ways [6].

Several algorithms carry out iterated linearizations [7]. Instead of using prior, the Iterated Posterior Linearization Filter (IPLF) uses iterations to make the linearization via statistical linear regression with respect to the posterior [8]. It is shown that it is better to make the statistical linearizations

using the posterior instead of the prior, as this minimizes the mean square error of the measurement linearization given the measurement value. There are also iterative methods for non-linear state transition models [7].

For cases, where the posterior estimate is multimodal and cannot be accurately approximated with a single Gaussian, there exist several algorithms. For example, particle filters [9] estimate the state using Monte Carlo samples and Gaussian Mixture Filters (GMFs) [10] use mixtures of Gaussians. However, in this paper we assume that the posterior can be estimated accurately with a Gaussian when we use measurement information from multiple time instances, but measurements from a single time instance are not enough. In comparison to GMF [10], we do not use multiple Gaussians, but instead we use one Gaussian containing elements from multiple time instances, when necessary. In our approach, the state representation size may grow only by adding state variables used in new measurements to the state, unlike in particle filters and GMFs, which vary the number of particles or Gaussians. We will also present how to limit the size of the state.

Usually in filters, the state estimation only proceeds forward in time when new measurements become available. Smoothers use all measurements, also from future time steps, to compute more accurate state estimates retrospectively. In [11], the smoothing uses iterated statistical linear regression and processes the measurements multiple times, giving rise to the iterated posterior linearization smoother (IPLS). However, this smoother uses all time steps and all measurements, though it is also possible to only apply smoothing in a window of the last  $L$  time steps, giving rise to the  $L$ -scan IPLS, see Algorithm 4 in [11].

In this paper, we are interested only in the current state estimate, but assume some measurements from the past may be useful to be processed again with better linearizations. We do not, however, assume that any future measurements are available, but only that measurements from the past may be re-evaluated when the state estimate and thus linearizations gets better.

In this paper, we propose the Stacked Iterated Posterior Linearization Filter (S-IPLF). It is an algorithm for linear dynamic systems that can use non-linear measurements from multiple time steps to perform the linearization. Measurements that are used in estimation of later states are chosen according

to their non-linearity and using them increases the size of the state. When the non-linearity of a measurement is found to be insignificant or the state has grown to be too big, the measurement and elements referring to past variables will not be used any more.

This paper is structured as follows. In Section II we go through the background of the algorithms. In Section III we propose the new algorithm, show how to add and remove necessary variables from the state, and show how to do the filtering. Section IV presents two example applications and Section V concludes the article.

## II. BACKGROUND ALGORITHMS

In this section, we present the algorithms that will be used as the background of the proposed algorithm. The state is modelled as a Gaussian and can be defined with mean  $\mu$  and covariance  $P$ . The state transition model is assumed to be linear. The measurement model may be non-linear, but its noise is assumed to be additive and Gaussian.

### A. Prediction

For prediction we assume to have the state mean  $\mu_{k-1}$  and covariance  $P_{k-1}$  at time step  $k-1$ . The state transition is linear and can be written as:

$$x_k = F_k x_{k-1} + \varepsilon_k^Q, \quad (1)$$

where  $x_k$  is the state at time step  $k$ ,  $F_k$  is the state transition matrix and  $\varepsilon_k^Q$  is the process noise. This noise is assumed Gaussian. It has zero mean and covariance  $Q_k$ . In this paper, we assume that the state transition matrix  $F_k$  and the noise covariance  $Q_k$  are known.

The following formulas can be used to get mean and covariance of prior estimates [1]

$$\mu_k^- = F_k \mu_{k-1}, \quad (2)$$

$$P_k^- = F_k P_{k-1} F_k^T + Q_k. \quad (3)$$

Variables with superscripts  $-$  correspond here to the prior and the variables without this superscript refer to the posterior.

### B. Update

A measurement depends on the state as:

$$y_k = h(x_k) + \varepsilon_k^R, \quad (4)$$

where  $y_k$  is the received measurement value,  $h(x_k)$  is a measurement function, and  $\varepsilon_k^R$  is a Gaussian noise term with zero mean and covariance  $R_k$  [12]. Equation (4) can be approximated using a linear model:

$$y_k \approx A_k x_k + b_k + \varepsilon_k^R + \varepsilon_k^\Omega, \quad (5)$$

where  $A_k$  is a matrix,  $b_k$  is a vector, and  $\varepsilon_k^\Omega$  is a Gaussian zero mean noise term with covariance  $\Omega_k$  and it is independent of  $\varepsilon_k^R$ . This error term is caused by the non-linearity. [8]

To update the prior with the above measurement, the following equation can be used based on (5) [8]:

$$\mu_k = \mu_k^- + P_k^- A_k^T (A_k P_k^- A_k^T + \Omega_k + R_k)^{-1} \times (y_k - A_k \mu_k^- - b_k), \quad (6)$$

$$P_k = P_k^- - P_k^- A_k^T (A_k P_k^- A_k^T + \Omega_k + R_k)^{-1} A_k P_k^-. \quad (7)$$

The linearization variables  $A_k$ ,  $b_k$ , and  $\Omega_k$  can be obtained via Statistical Linearization Regression (SLR), if we perform SLR w.r.t.  $\mu_k^-$  and  $P_k^-$  [13]. This results in

$$A_k = \Psi_k^T P_k^-^{-1}, \quad (8)$$

$$b_k = z_k - A_k \mu_k^-, \quad (9)$$

$$\Omega_k = \Phi_k - A_k P_k^- A_k^T, \quad (10)$$

where variables  $z_k$ ,  $\Psi_k$ , and  $\Phi_k$  are defined using integrals. These integrals are the following

$$z_k = \int h(x) p_x(x) dx, \quad (11)$$

$$\Psi_k = \int (x - \mu_k)(h(x) - z_k)^T p_x(x) dx, \quad (12)$$

$$\Phi_k = \int (h(x) - z_k)(h(x) - z_k)^T p_x(x) dx, \quad (13)$$

where  $p_x(x)$  is a probability density function (pdf) of the state, which is assumed here to be Gaussian. In conventional non-linear KF it has mean  $\mu_k^-$  and covariance  $P_k^-$ . The IPLF [8] tries to use posterior distribution instead of prior when computing  $p_x(x)$ . The posterior estimate is obtained iteratively starting from the prior [11].

### C. Sigma point methods

The integrals in (11)-(13) are not usually solvable in closed form. However, in general, approximations called sigma point methods can be used [14].

Assume that we have  $m$  sigma points at time  $k$ . The  $i$ th sigma-point has location  $\chi_{k,i}$  and weight  $\omega_{k,i}$ . These are generated by a sigma point method, so that, sum of weights is 1, their mean and covariance match the Gaussian prior distribution [1]

$$\sum_{i=1}^m \omega_{k,i} = 1 \quad (14)$$

$$\sum_{i=1}^m \omega_{k,i} \chi_{k,i} = \mu_k^-, \quad (15)$$

$$\sum_{i=1}^m \omega_{k,i} (\chi_{k,i} - \mu_k^-)(\chi_{k,i} - \mu_k^-)^T = P_k^-. \quad (16)$$

The measurement function now transforms the sigma points as

$$\psi_{k,i} = h(\chi_{k,i}). \quad (17)$$

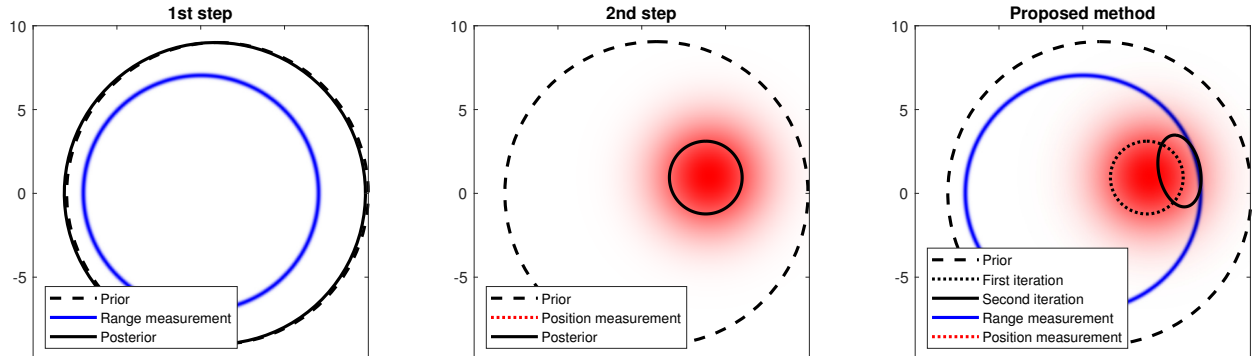


Fig. 1. State estimate after first and second measurement when measurements are processed with the UKF. The third figure shows the output of the proposed method that uses information from both measurement.

Now we can approximate integrals (11)-(13) as

$$z_k \approx \sum_{i=1}^m \omega_{k,i} \psi_{k,i}, \quad (18)$$

$$\Psi_k \approx \sum_{i=1}^m \omega_{k,i} (\chi_{k,i} - \mu_k^-) (\psi_{k,i} - z_k)^T, \quad (19)$$

$$\Phi_k \approx \sum_{i=1}^m \omega_{k,i} (\psi_{k,i} - z_k) (\psi_{k,i} - z_k)^T. \quad (20)$$

#### D. Posterior linearization iterations

In IPLF [8] we start from the prior. By successive SLRs, the estimates get closer to the posterior. The update is done using (6) and (7), in which the prior remains unchanged, but the linearisation parameters are obtained via SLR with respect to the current posterior approximation. In this paper, to avoid confusion with a stacked state that contains old states of variables, we will call Linearization State (LS) the state estimate where we do the linearization. The corresponding elements from LS will be used as posterior estimate, but the prior estimate that will be updated is changed only when measurements are no longer used.

Therefore, integrals (11)-(13), or rather their approximations (17)-(20), are computed using the LS and these are then used to compute variables  $A$ ,  $b$ , and  $\Omega$  (8)-(10) to gain necessary variables for computing a new LS.

#### E. Motivation

The motivation of this work is to improve the state estimation accuracy of non-linear KF extensions. We consider that with information of following measurements the linearization of an older measurement can be better. We study how to use the better linearization of an old measurement to improve the estimation of the current state. The intuition is similar to that in smoothing, the measurements from the future improve the estimates from the past. This allows us to make better linearizations using the past measurements to improve the current state estimate.

Let us consider the following example. The prior has mean  $[1 \ 0]^T$  and covariance  $9^2 I$ . In the state transition model,  $F$

and  $Q$  are equal to  $I$ . We consider the UKF in [15] with weight 0.5 for the sigma-point located at the mean.

The leftmost plot of the Figure 1 illustrates a measurement that gives an approximate range from the origin. In the following prediction step, the variance of state estimate is increased slightly by the state transition model with  $F$  and  $Q$ . In the second time step we get a linear, but inaccurate, position measurement that gives the two dimensional position with a large additive noise.

The first range measurement has value 7 and its measurement function is  $h(x) = \|x\|$  and an additive noise  $\varepsilon^R$  has variance  $0.1^2$ . The measurement does not bring much information to the state estimate that is represented as Gaussian and filtered with the UKF.

The second measurement is not very accurate. It is a two dimensional linear measurement with value  $[4 \ 1]^T$  and the measurement noise covariance is  $R = 5I$ . The posterior, computed with UKF, after these two measurements is close to distribution of the second measurement.

In the third plot in Figure 1, we look at the case where we use augmented state with both time step positions and their correlation in the state. We use IPLF for the augmented state with these two measurements. In this case, the non-linearity is better handled so that the first measurement used with the second one gives more information and the estimate becomes more accurate.

State estimation that uses multiple time steps consumes more memory and gets more resource consuming when time evolves. Other options are the particle filter [16] or GMF [10]. These methods may be computationally more demanding depending on the number of components.

The motivation of this paper is to develop an algorithm that automatically evaluates the non-linearity of a measurement. If it is determined to be too non-linear, then the algorithm will use the measurement in later time steps. The IPLF updates are used and the state estimate size increases when measurements are used from multiple time steps. However, the state size may increase only proportionally to the time steps and the maximum size of the state vector can be limited.

### III. STACKED ITERATED POSTERIOR LINEARIZATION FILTER (S-IPLF)

In this section, we will present the proposed algorithm, the S-IPLF. This section is divided into the following subsections. First in Section III-A we show how to compute the non-linearity of a measurement. The prediction and update steps are presented in Sections III-B and III-C. Then, how to remove a measurement from the measurement stack is presented in Section III-D. The final combination is presented in Section III-E. The common variables used in various algorithms are given with descriptions in Table I.

TABLE I  
COMMON VARIABLE NOTATIONS IN ALGORITHMS

Prior	$\hat{\mu}^-$	Prior mean
	$\hat{P}^-$	Prior covariance
Estimate (LS)	$\hat{\mu}_{LS}$	Estimated mean
	$\hat{P}_{LS}$	Estimated covariance
	$n_{LS}$	Number of original state variables
	$m_{LS}$	Number of all state variables
Measurements:		
	$mStack$	Measurement stack
	$n_M$	Number of measurements
Measurement:		
	$y$	Measurement value
	$h(\cdot)$	Measurement function
	$v$	Indices of state elements used in function
	$R$	Measurement noise covariance

#### A. Measuring the degree of non-linearity of a measurement

Several different non-linearity measures are discussed in [17]–[21]. None of those is directly applicable in this paper. We need to evaluate multiple measurements, maybe from different time instances. Because we are doing filtering and not smoothing, we are interested only on the effect to the current time instance.

In this paper, the measure for the effect of the amount of non-linearity is based on the covariance matrix  $\Omega_i$  (10). We will compute the values  $\Omega_i$  for different measurements. We compute the covariance estimate of the updates after getting the most recent LS. In principle,  $\Omega_i$  variable would be 0 for a linear measurement and larger for more non-linear measurement. In non-linear situation  $A_i$  differs depending on the state mean and covariance when the linearization is done. We assume  $A_i$  matrix to be constant enough to be used in the non-linearity measure.

The idea is to compute how much the determinant of the covariance of the state variables of current state would decrease if a measurement was linear. We consider measurements separately. When we add a measurement to be stored after the current time we store information of which new state variables it affects to. In this information we store separately measurements so that none of them are affected by different measurements and we do not save state variables from older time steps that are not used. For example, if we have speed

measurement of an object and the non-linearity affects that we do not need to keep the old position variables.

For the  $i$ th measurement, we can compute the current state covariance with the measurement

$$P_{ss}^{+,i} = P_{ss} - P_{sv_i} A_i^T (A_i P_{v_i v_i} A_i^T + R_i + \Omega_i)^{-1} A_i P_{v_i s}. \quad (21)$$

Here subscript  $v_i$  refers to variables of the  $i$ th measurement and  $s$  to the variables of the current state.

For computing  $A_i$  we can note that it is independent of other measurements and it is only dependent on variables in  $v_i$ . So it is enough to compute the sigma-points only for  $P_{v_i v_i}$  to get  $A_i$  and  $\Omega_i$ .

We will make comparison of that covariance to one obtained as if there were no non-linearity caused covariance ( $\Omega_i = 0$ ):

$$P_{ss}^{0,i} = P_{ss} - P_{sv_i} A_i^T (A_i P_{v_i v_i} A_i^T + R_i)^{-1} A_i P_{v_i s}. \quad (22)$$

For this  $i$ th measurement, we look at how much the determinant of the covariance decreases, if that measurement was linear

$$nl[i] = \frac{\det P_{ss}^{+,i}}{\det P_{ss}^{0,i}}. \quad (23)$$

We choose to update state (and remove these measurements from stack) with measurements for which  $nl$  is close to 1, or if the number of variables in state is too large then we remove the measurements with smallest  $nl$ .

#### B. Prediction

We use the augmented prediction for state that contains the original variables in beginning, the older added state elements next, and last we add those variables that we want to store from  $i$ th measurement

$$\hat{x}_k = \hat{F}_k \hat{x}_{k-1} + \varepsilon_{\hat{Q}} = \begin{bmatrix} F_k x_{k-1,1:n_{LS}} \\ x_{k-1,n_{LS}+1:m_{LS}} \\ x_{k-1,v_i} \end{bmatrix} + \varepsilon_{\hat{Q}}, \quad (24)$$

where  $v_i$  refer to variables that are used in a new measurement from  $k$ th time and which will be used also later.

We can now write

$$\hat{\mu}_k^- = \hat{F}_k \hat{\mu}_{k-1}, \quad (25)$$

$$\hat{P}_k^- = \hat{F}_k \hat{P}_{k-1} \hat{F}_k^T + \hat{Q}_k. \quad (26)$$

Here matrices with hat ( $\hat{\cdot}$ ) are modified so that they will be used with augmented state with older variables and the stack of measurements. The transformation matrix  $\hat{F}_k$  is

$$\hat{F}_k = \begin{bmatrix} F_k & 0 \\ 0 & I \\ V_k & 0 \end{bmatrix}, \quad (27)$$

where matrix  $V_k$  moves the new state variables, that will be stored, to the end of the stacked state. It has as many rows as the state has new variables and it has ones on the columns that refer to the new variables. The process noise covariance of the stacked state is now

$$\hat{Q}_k = \begin{bmatrix} Q_k & 0 \\ 0 & 0 \end{bmatrix}, \quad (28)$$

where  $Q$  is the state process noise covariance matrix of the normal state transition. One can note that the augmented state variables or their covariances do not change in time transition. This is just because these variables are in principle already estimated in the past and in the augmented state are from that time.

We propose to separate new measurements, which can be multidimensional, so that different measurements refer only to different state variables. This enables to have multiple separate low dimensional measurements and makes it simple to remove measurements when necessary. This also helps implementation when all measurement functions refer to first variables and the  $v$  is used to define which variables of the state these are.

This algorithm is given in Algorithm 1. Its inputs are the prior  $prior$ , the LS that may also be used as a posterior estimate, and the stack of measurements  $mStack$ .

---

**Algorithm 1: Prediction**

---

```

Function [prior, LS, mStack] = pred(prior, LS, mStack)
/* Initialize matrix V that is
   used to move variables that are
   stored in following time steps
   to end of state vector and
   covariance */
u = 0 // How many variable moved
for all measurement indices i do
  // If variables have small indices
  if any(v[i] ≤ nLS) then
    for J=1 to length(v[i]) do
      u = u + 1
      // Initialize row with zeros
      V(u,:) = 0
      // Set Jth element
      V(u, v[i](J)) = 1
      /* Change the Jth reference
         index to new variables */
      v[i](J) = mLS + v
    end
  end
end
// State transition
Ĥ = [ F          0nLS × mLS - nLS
      0mLS - nLS × nLS  ImLS - nLS × mLS - nLS
      V          0v × mLS - nLS ]
// Process covariance
Ĥ = [ Q          0nLS × mLS + v - nLS
      0mLS + v - nLS × nLS  0mLS + v - nLS × mLS + v - nLS ]
mLS = mLS + u
// Predict prior
ĥ- = Ĥμ // mean
Ĥ- = ĤĤT + Ĥ // covariance
// Predict LS
ĥLS- = ĤĥLS} // mean
ĤLS- = ĤĤLS}ĤT + Ĥ // covariance
end

```

---

For example, the range measurement in the example in Section II-E contains only position variables. If we had a larger state that contains position and velocity then we would not need the velocity estimate from time 1 even we would reprocessed the range measurement. The initial  $v_1$  would be  $[1 \ 2]^T$  and after prediction it would be  $[5 \ 6]^T$ . Matrix  $V_1$  would then be

$$V_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (29)$$

so it would allow us to compute the predicted variances and state variables with (27) and (28).

*C. Update*

In update with measurement stack  $mStack$  we will have to compute the measurement function  $\hat{h}(\cdot)$ , measurement value  $\hat{y}$ , additive measurement error covariances  $\hat{R}$ , and the information of which state elements are used in it  $v$ . In our notation the newest variables are moved to the end at each time instance. The measurement augmentation happens in prediction if some new measurements are to be stored for later time steps.

Assume we have  $n_M$  measurements in the stack. Then the augmented measurement is

$$\hat{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_M} \end{bmatrix}. \quad (30)$$

We have measurement functions which have been selected to be stored, due to the non-linearity, into the stack and the augmented measurement function is

$$\hat{h}(x) = \begin{bmatrix} h_1(x_{v_1}) \\ \vdots \\ h_{n_M}(x_{v_{n_M}}) \end{bmatrix}. \quad (31)$$

The additive total covariance is now

$$\hat{R} = \begin{bmatrix} R_1 & \mathbf{0} & \dots & \dots \\ \mathbf{0} & R_2 & \mathbf{0} & \dots \\ \vdots & \mathbf{0} & \ddots & \ddots \\ \vdots & \ddots & \ddots & R_{n_M} \end{bmatrix}. \quad (32)$$

The total update algorithm is written in Algorithm 2.

*D. Reducing the number of processed measurements*

So far we have explained how to add measurements and the variables referenced by them. The measurements can be used at multiple time steps and multiple times within an update. However, to reduce the total computational cost and to avoid an ever increasing state dimensionality, we propose to check the measurement non-linearity and then remove the linear enough or at least the most linear measurement from the measurement stack as presented in Section III-A. In the removal the prior is updated using an update that is computed using the measurement to be removed. Pseudo-code for this is given in Algorithm 3.

---

**Algorithm 2: Update**

---

**Function**  $[LS] = \text{update}(\text{prior}, LS, mStack)$   
Merge measurements from stack:

$$\hat{y} = \begin{bmatrix} y[1] \\ \vdots \\ y[n_M] \end{bmatrix} \quad \begin{array}{l} \text{Total} \\ // \text{ measurement} \\ \text{value} \end{array}$$
$$\hat{h}(x) = \begin{bmatrix} h(x[v[1]]) \\ \vdots \\ h(x[v[n_M]]) \end{bmatrix} \quad \begin{array}{l} \text{Total} \\ // \text{ measurement} \\ \text{function} \end{array}$$
$$\hat{R} = \begin{bmatrix} R[1] & \mathbf{0} & \cdots & \cdots \\ \mathbf{0} & R[2] & \mathbf{0} & \cdots \\ \vdots & \mathbf{0} & \ddots & \ddots \\ \vdots & \ddots & \ddots & R[n_M] \end{bmatrix}$$

// Total measurement covariance  
Do linearization with mean  $\hat{\mu}_{LS}$  and covariance  $\hat{P}_{LS}$  of measurement  $\hat{h}(x)$   
Do Kalman update of *prior* to get new *LS*. See Section II-B and Section II-C

**end**

---

---

**Algorithm 3: Removing a measurement from stack and update the prior**

---

**Function**  $[\text{prior}, LS, mStack] = \text{remove}(\text{prior}, LS, mStack, i)$   
Do linearization of  $mStack[i]$  at point with mean  $\hat{\mu}_{LS}$  and covariance  $\hat{P}_{LS}$  using  $mStack[i]$  for  $y = h(x) + \varepsilon^R$   
Use a KF update to get new prior  $\hat{\mu}^-$ , and  $\hat{P}^-$  using  $i$ th measurement  
**if**  $v[i] > n_{LS}$  **then**  
    Remove  $v[i]$ th variables from means  $\hat{\mu}^-$  and  $\hat{P}_{LS}$ .  
    Remove  $v[i]$ th rows and columns from  $\hat{P}^-$  and  $\hat{P}_{LS}$ .  
    Correct all  $v[i]$ s so that variable references are to correct state indices.  
**end**  
Delete  $i$ th measurement from  $mStack$

**end**

---

### E. Complete S-IPLF

The state of the resulting S-IPLF algorithm consists, in general, of variables for multiple time steps. Basically, we could do the linearization using the prior for all these variables, but as shown in (11), it is better to use posterior estimates in linearization. This is why we keep the prior and the linearization point in prediction and move them to the next phase, update.

The pseudo-code of the S-IPLF is given in Algorithm 4. Variables with subscript LS are the point where the linearization is done. Because algorithm is based IPLF the LS estimates are updated often to make better linearizations all the time and

---

**Algorithm 4: Stacked Iterated Posterior Linearization Filter (S-IPLF)**

---

**for**  $k=1$  to  $t$  // For loop over time  
**do**  
    // Algorithm 1  
     $[\text{prior}, LS] = \text{pred}(\text{prior}, LS, mStack)$   
    // Add measurement to stack  
     $mStack[\text{end}+1] = \text{meas}[k]$   
     $n_M = n_M + 1$   
    **for**  $i=1$  to  $n$  **do**  
        // Algorithm 2  
         $[LS] = \text{update}(\text{prior}, LS, mStack)$   
    **end**  
    Compute nonlinearities using (23)  
    **for**  $i=\text{measurement to remove from stack}$  **do**  
        // Algorithm 3  
         $[\text{prior}, LS, mStack] = \text{remove}(\text{prior}, LS, mStack, i)$   
    **end**  
**end**

---

the prior that is basis for each update is changed only more seldom. Basically the posterior is the first  $n_{LS}$  variables of LS.

## IV. EXAMPLES

We test the performance of the S-IPLF in two examples. In these two examples, we use similar motion models. The state has model that has two dimensional position  $x_{1:2}$  with velocity  $x_{3:4}$ . The state transition model used is

$$x_k = \begin{bmatrix} x_{1,k-1} + x_{3,k-1} \\ x_{2,k-1} + x_{4,k-1} \\ x_{3,k-1} \\ x_{4,k-1} \end{bmatrix} + \varepsilon_k^Q = \begin{bmatrix} I_2 & I_2 \\ \mathbf{0} & I_2 \end{bmatrix} x_{k-1} + \varepsilon_k^Q, \quad (33)$$

In these examples, all filters use the unscented transform in (15) with weight 0.5 for the sigma point located at the mean.

### A. Short time series

In this first example, we have non-linear range measurements from 5 different sources. One measurement from one source at different time steps. The number of time step in the simulation is 5 and each measurement is done on its own time step. We compare the proposed algorithm (S-IPLF), with the UKF and the IPLF. We also test a variant of the S-IPLF that does not limit the number of stored measurements in the stack. The IPLF and S-IPLF use 10 iterations at each time step. The state transition noise  $\varepsilon_Q$  has zero mean and variance  $0.1^2 I_4$ . The initial state has mean  $[0 \ -1 \ 0 \ 0]^T$  and unit diagonal covariance  $I_4$ .

Figure 2 shows how the proposed method, with one most nonlinear measurement kept in stack for later state estimation, is more accurate than the UKF or the IPLF. We can see from the errors that the proposed algorithm improves the accuracy in this example. Having of all measurements that have been received so far at once does not improve result much compared

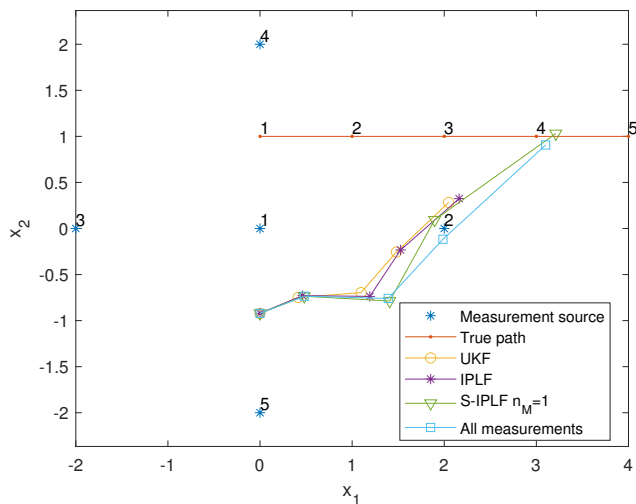


Fig. 2. Five step route estimates. Range measurements origins are marked with \* and each measurement time step is shown with number and corresponding true track step also. Estimates all go from down left to up right.

to having the most nonlinear measurement in memory to be reused.

### B. Longer time series

In this example, we use similar state transition matrix as the previous example, but we generate longer paths with 100 time steps. Measurements are also different. We simulate three kinds of measurements. One measurement is available at each step of the time series. Measurement type order is not pre-defined, but randomly selected in simulation. Each measurement has equal probability to be done.

We have absolute position measurements that are linear, but very inaccurate. They provide the position values plus zero mean error  $\varepsilon_k^1$  that has covariance of  $100^2 \cdot I_2$ . The measurement model is

$$y_{1,k} = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} + \varepsilon_k^1. \quad (34)$$

Then there are range measurements from position  $d_k$ . Position of  $d_k$  is randomly selected. It is located within 10 units wide and tall box centred at the true location. The position  $d_k$  is known, and the second measurement function is

$$y_{2,k} = \|x_{1:2,k} - d_k\| + \varepsilon_k^2, \quad (35)$$

where  $\varepsilon_k^2$  has zero mean and unit variance.

Then the third measurement tells the current speed and its equation is

$$y_{3,k} = \|x_{3:4,k}\| + \varepsilon_k^3, \quad (36)$$

where  $\varepsilon_k^3$  has zero mean and variance  $0.1^2$ .

We performed this 100 time steps long simulation 1000 times. The initial mean is zero and has covariance  $10^2 I_4$ , and the true initial state is simulated from this. The state transition noise is  $Q_k = 0.1^2 I_4$ . The true route is simulated using normal distribution assumptions starting from the true initial state.

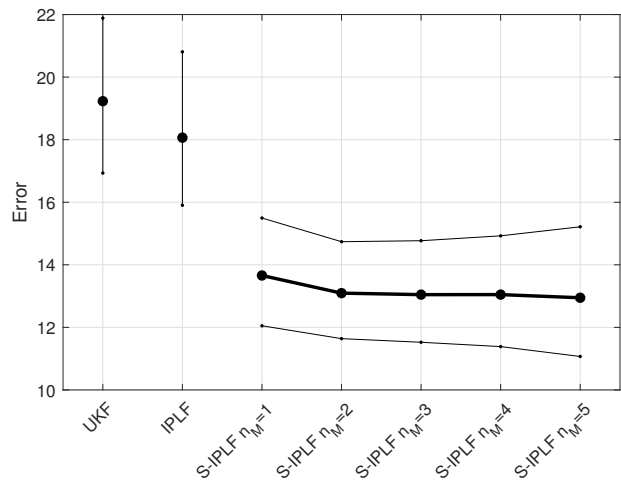


Fig. 3. 25%, 50% and 75% quantiles of error when estimating with UKF, IPLF, and with S-IPLF when estimating with different number of measurements.

We compare UKF, IPLF and the proposed S-IPLF with various number of measurements in stack. The median, 25% and 75% quantile errors are shown in Figure 3. The quantiles are computed from Root Mean Square Error (RMSE) errors of all time steps. The IPLF is practically the S-IPLF with 0 measurements in stack. From this example we can see that, at least in this case, the storage of measurement in stack improves the estimation accuracy. More than 1 measurement in stack does not have much more significant improvement in estimation accuracy.

This result is, of course, based only on this example and in some other examples it could be useful to have many measurements in stack. However, in this example, the state with one measurement has 6 variables compared to original 4 element states as every measurement depend only on 2 state elements. The accuracy improves while there is no need to increase the state size significantly.

## V. CONCLUSIONS

In this paper, we proposed the Stacked Iterated Posterior Linearization Filter (S-IPLF). The proposed algorithm can help to improve estimation accuracy. The proposed algorithm stores non-linear measurements in stack to be relinearized in later time instances if necessary. If it is considered that these measurements are not necessary, they will be removed from the stack. This relinearization makes the estimation accuracy better in some non-linear situations.

There are topics that should be investigated in coming up papers. The value for non-linearity (23) was used in this work to select most non-linear measurements for future time steps. We should also make research to determine a suitable limit value for this.

We should also develop the non-linearity computation to be faster. We could also develop algorithm similar to one presented in this paper for non-linear estimation problems that

have non-linear dynamic functions. The generalisation of the S-IPLF to smoothing can also be studied in future.

#### REFERENCES

- [1] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
- [2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [3] A. Gelb, J. F. Kasper, R. A. Nash, C. F. Price, and A. A. Sutherland, Eds., *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.
- [4] E. Wan and R. van der Merwe, "The Unscented Kalman Filter," S. Haykin, Ed. Wiley Publishing, 2001.
- [5] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.
- [6] S. Ungarala, "On the iterated forms of Kalman filters using statistical linearization," *Journal of Process Control*, vol. 22, no. 5, pp. 935–943, 2012.
- [7] A. Kullberg, I. Skog, and G. Hendeby, "Iterated filters for nonlinear transition models," in *Proceedings of the 2023 Information Fusion Conference, 2023*.
- [8] A. 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.
- [9] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proc. F Radar Signal Process. UK*, vol. 140, no. 2, p. 107, 1993.
- [10] S. Ali-Loytty, "Efficient Gaussian mixture filter for hybrid positioning," in *2008 IEEE/ION Position, Location and Navigation Symposium, 2008*, pp. 60–66.
- [11] A. 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.
- [12] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 910–927, 2000.
- [13] H. Arasaratnam, S. Haykin, and R. Elliott, "Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature," *Proceedings of the IEEE*, vol. 95, pp. 953 – 977, 06 2007.
- [14] R. v. d. Merwe and E. A. Wan, "Sigma-point Kalman filters for integrated navigation," in *Proceedings of the 60th annual meeting of the institute of navigation (2004)*, 2004, pp. 641–654.
- [15] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.
- [16] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, 2002.
- [17] F. Faubel, J. McDonough, and D. Klakow, "The split and merge unscented Gaussian mixture filter," *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 786–789, 2009.
- [18] M. R. Morelande and A. F. García-Fernández, "Analysis of Kalman filter approximations for nonlinear measurements," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5477–5484, 2013.
- [19] J. Xavier, S. Patnaik, and R. C. Panda, "Nonlinear measure for nonlinear dynamic processes using convergence area: typical case studies," *Journal of Computational and Nonlinear Dynamics*, vol. 16, no. 5, p. 051002, 2021.
- [20] J. Duník, O. Straka, and A. F. García-Fernández, "Performance evaluation of nonlinearity and non-Gaussianity measures in state estimation," in *2017 20th International Conference on Information Fusion (Fusion)*. IEEE, 2017, pp. 1–10.
- [21] M. Raitoharju, "Linear models and approximations in personal positioning." Ph.D. dissertation, 2014.