INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D3.1.1 - Method to extend models for system design to models for system operation

WP 3.1 Continuous Systems

WP 3 State estimation and system monitoring

MODRIO (11004)

Version 3.0

Date April 15, 2016

Authors

Jonathan Brembeck DLR

Executive summary

There are two main application areas that are dealt with in the scope of MODRIO regarding state estimation:

1. Non-linear model predictive control (NMPC),
2. System diagnosis (assessing and predicting systems performance degradation in order to plan more efficiently maintenance operations).

The mathematical and algorithmic background are essentially the same for both applications, although the vocabulary may differ in some aspects. The first application uses vocabulary from the systems control community, whereas the second application uses vocabulary from the physics community, but may also use vocabulary from the systems control community as well.

The main difference between the two applications lies in the fact that the assessment of uncertainties is sought for diagnosis and not for systems control. On the other hand, real time capabilities are essential for systems control, and not for system diagnosis (where extra computing time may be used to compute the uncertainties).

In this document the methods for the first application is presented.

Contents

1. Principals of state estimation in discrete systems

For many of today's control tasks like Model Predictive or State Feedback Control, the knowledge of the actual full state vector of the controller's synthesis model is necessary. In most of the control applications not all states can be measured directly via sensors, this might come from the unavailability (i.e. unavailability of direct charge measurement of a Lithium cell) or cost considerations (i.e Force-Momentum sensors in each axis of an industrial robot). Therefore the missing states must be reconstructed from the available measurement (i.e. motor current and angular velocity of a robot axis) in combination with the knowledge of the system dynamics of the controlled plant. Due to today nearly all controllers are executed on real-time capable micro controllers, the estimation algorithms must be performed in discrete time algorithms. Thus we focus in the next chapters on methods in such representation and do not cope with the continuous time theory of state estimation. Furthermore we focus on Kalman Filter based algorithms which have a large experience and good results for causal systems in the last decades.

1.1. Basic structure of an estimation setup

In Figure 1 a signal flow diagram in Modelica style is sketched that shows the configuration of a state estimator in a discrete time system (i.e. on a micro controller). From a controller a system input u_k is commanded to the real system (by help of a D/A converter) and also feed to the inputs of the prediction model of the state estimator. Note that we always assume that the inputs have no direct feed through to the outputs this is a valid assumption for all physical models and controlled plants. Due to this input and the system dynamics we can measure some outputs y of the real system that may be disturbed through a measurement noise z; in an additive case this yields to the sampled (via D/A converter) measurement vector $\hat{y}_k$. Also from the discrete prediction model $\hat{f}_k$ and the output function $\hat{h}_k$ we get an estimate of the output. To estimate the true states of the real system we try to reduce the error between measurement and prediction. This is coped via calculation of the gain K in the estimator algorithm. It is determined in a way that it minimizes the error between $\hat{x} - x$ under statistical assumption in the believe of the underlying prediction model and the measured values.

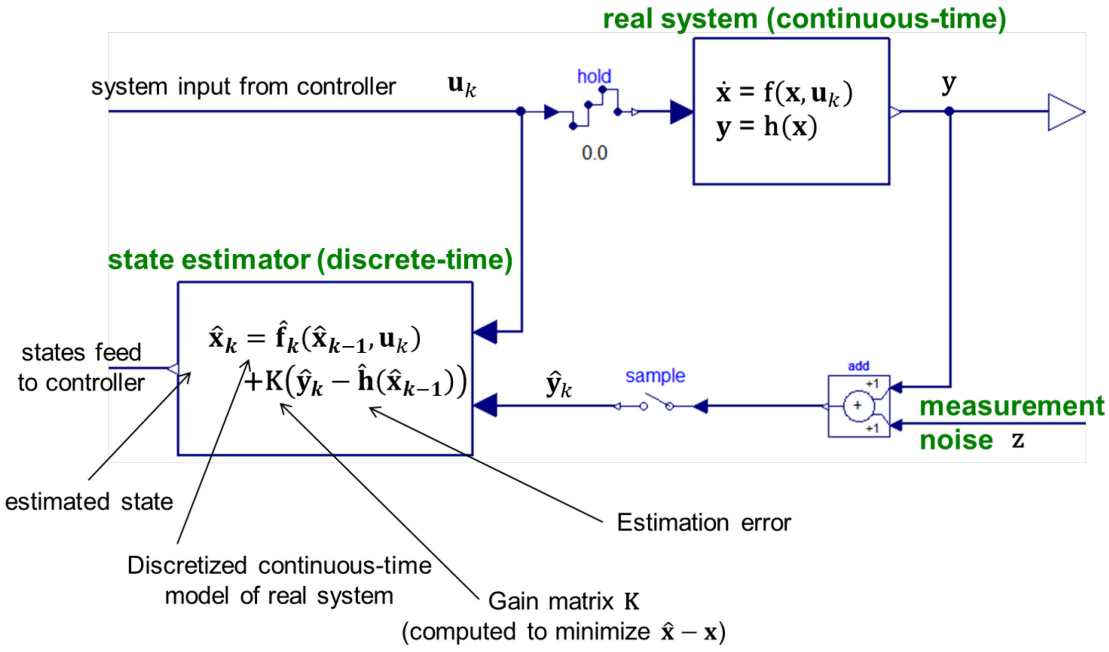

Figure 1: State Estimator in discrete time environment

1.2. Content of this document

Beginning from the principal need of state estimation and the sketch of the configuration in real world micro controllers in this chapter, the upcoming is organized as follows. In chapter 2 an introduction to recursive estimation techniques is given. Recursive means, in every time step only the information from the last one are

taken into account for the estimation algorithm. This technique is wide spread due to limited memory and computing capacity of most common micro controllers. Here the focus is on the connection between (recursive) weighted least squares estimation of a constant and the (linear) *Kalman Filter* technique for state estimation. Afterwards (chapter 3) several extensions to nonlinear systems and the most important techniques, from the view of the author, are shown. Here two different types of approaches are explained, on the one hand side algorithms that need Jacobians and on the other side derivative free method.

2. Recursive state estimation

In this chapter, the principle ideas of recursive state estimation are summarized, and its (historical) development leading to the Kalman Filter is outlined. Further background information, alternative formulations, and recent developments are provided in the standard book (Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, 2006) that is also the starting point for the following explanations.

2.1. Weighted Least Squares Estimation

At first, we consider an estimation of a *constant signal* on the basis of several noisy measurements. This *Weighted Least Squares Estimation* problem is well-known in system identification tasks (see, e.g., (Ljung, 1998)). Through the weighted formulation, the user can assign different levels of confidence to certain measurements (or observations). This feature is crucial for tuning Kalman Filters. The corresponding minimization problem is formulated as follows:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} H_{11} & \dots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{kn} & \cdots & H_{kn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}$$
$$E\left(v_i^2\right) = \sigma_i^2 \quad (i = 1, \dots, k) \tag{2.1}$$
$$x \in \mathbb{R}^n, y \in \mathbb{R}^k, v \in \mathbb{R}^k$$

The unknown vector x is constant and consists of n elements, y is a k-element noisy measurement vector and usually $k \gg n$. Each element of y - y_i - is a linear combination (H_{k*}) with the unknown vector x and the variance of the measurement noise of the i-th measurement v_i. The noise of each measurement is zero-mean and independent from each other, therefore the measurement covariance matrix is

$$R = (vv^T) = diag(\sigma_1^2, \dots, \sigma_k^2) \tag{2.2}$$

The residual

$$\epsilon_y = \underbrace{(Hx + v)}_{=y} - H\hat{x} \tag{2.3}$$

is the difference of all measured values y with the (unknown) x-vector minus the estimated vector $\hat{y}$ that is computed from the estimated vector $\hat{x}$. The goal is to compute the estimated vector $\hat{x}$ such that the weighted residual is as small as possible, i.e., to minimize the cost function J:

$$J = \frac{\epsilon_{y1}^2}{\sigma_1^2} + \cdots + \frac{\epsilon_{yk}^2}{\sigma_k^2} \tag{2.4}$$

To minimize J, it is useful to compute the partial derivative with respect to the estimated $\hat{x}$ vector and set it to zero. In this way, an optimal solution for $\hat{x}$ can be calculated:

$$\frac{\partial J}{\partial \hat{x}} = 2 \cdot (-y^T R^{-1} H + \hat{x}^T H^T R^{-1} H) = 0$$
$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} y \tag{2.5}$$

(2.5) requires that R is nonsingular and H has full rank. This is the "textbook" version of the algorithm. It is inefficient and numerically not reliable. Alternatively, (2.4) can be formulated as:

$$J = \begin{bmatrix} \frac{\epsilon_{y1}}{\sigma_1} & \dots & \frac{\epsilon_{yk}}{\sigma_k} \end{bmatrix} \cdot \begin{bmatrix} \frac{\epsilon_{y1}}{\sigma_1} \\ \vdots \\ \frac{\epsilon_{yk}}{\sigma_k} \end{bmatrix} \tag{2.6}$$

To solve the following standard linear least squares problem that minimizes the Euclidian norm of the weighted residue vector:

$$\begin{aligned} &\min_{\hat{x}} \left\| \begin{bmatrix} \frac{\epsilon_{y1}}{\sigma_1} & \dots & \frac{\epsilon_{yk}}{\sigma_k} \end{bmatrix} \right\|^2 \\ &= \min_{\hat{x}} \|W(y - H\hat{x})\|^2 \\ &= \min_{\hat{x}} \|WH\hat{x} - Wy\|^2 \\ &= \min_{\hat{x}} \|A\hat{x} - b\|^2 \\ &W = diag(1/\sigma_1, \dots, 1/\sigma_k) \end{aligned} \tag{2.7}$$

This minimization problem has a unique solution, if $A=WH$ has *full rank*. If A is *rank deficient*, an infinite number of solutions $\hat{x}$ exists. The usual approach is to select from the infinite number of solutions the unique one that additionally minimizes the norm of the solution vector: $\|\hat{x}\|^2 \rightarrow min$. Given $A = WH$ and $b = Wy$, this solution vector can be computed with a least squares solver like the LAPACK function DGELSX (Anderson, et al., 1999).This function uses a QR decomposition of A with column pivoting together with a right multiplication of an orthogonal matrix Z to arrive at:

$$\min_{\hat{x}} \left\| [Q_1 \quad Q_2] \begin{bmatrix} U & 0 \\ 0 & 0 \end{bmatrix} ZP\hat{x} - b \right\|^2 \tag{2.8}$$

where Q and Z are orthogonal matrices, P is a permutation matrix, U is a regular, upper triangular matrix and the dimension of the quadratic matrix U is identical to the rank of A. Since the norm of a vector is invariant against orthogonal transformations, this equation can be transformed to:

$$\min_{\hat{x}} \left\| \begin{bmatrix} U & 0 \\ 0 & 0 \end{bmatrix} ZP\hat{x} - \begin{bmatrix} Q_1^T b \\ Q_2^T b \end{bmatrix} \right\|^2 \tag{2.9}$$

This is equivalent to

$$\min_{\hat{x}} \left\| \begin{bmatrix} U \\ 0 \end{bmatrix} \bar{\hat{x}}_1 - \begin{bmatrix} Q_1^T b \\ Q_2^T b \end{bmatrix} \right\|^2, \overline{\hat{x}} = ZP\hat{x} \tag{2.10}$$

from which the solution can be directly computed as (taking into account $b = Wy$):

$$\hat{x} = PZ^T U^{-1} Q_1^T W y \tag{2.11}$$

In the following, only textbook versions of algorithms will be shown, such as (2.5). Their implementation is, however, performed in an efficient and numerically reliable way, such as (2.11), where matrices R and H can be rank deficient. The sketched approach, both (2.5) and (2.11), can be used for *offline estimation* with a predetermined number of measurements k. In real-time applications, new measurements arrive in each sample period to improve the estimation. Using (2.11) would require a complete recalculation with $O(k^3)$-flops. One approach could be to use a moving horizon and to forget the older measurements (still costly). Another option is to reformulate the problem into a recursive form that is updated at every sample instant with the new measurements. A linear recursive estimator can be written in the following representation:

$$\begin{aligned} y_k &= H_k x + v_k \\ \hat{x}_k &= \hat{x}_{k-1} + K_k \cdot (y_k - H_k \hat{x}_{k-1}) \end{aligned} \tag{2.12}$$

We compute $\hat{x}_k$ based on the estimation from the last time step $\hat{x}_{k-1}$ and the information from the new measurement y_k. K_k is the estimator gain vector that weights the correction term $y_k - H_k \hat{x}_{k-1}$. Hence, we have to compute an optimal K_k in a recursive way. To this end, it is necessary to formulate another cost function that minimizes the covariance in a recursive way.

$$\frac{\partial J_k}{\partial K_k} = \frac{\partial Tr P_k}{\partial K_k} = 0 \tag{2.13}$$

$$\begin{aligned} P_k &= (I - K_k H_k) P_{k-1} (I - K_k H_K)^T \\ &\quad + K_k R_k K_k^T \end{aligned} \tag{2.14}$$

$$K_k = P_{k-1} H_k^T \cdot \left(H_k P_{k-1} H_k^T + R_k\right)^{-1} \tag{2.15}$$

This results in a recursive formula to update the estimation of the unknown, but constant, vector x in every sample with the latest measurements, based only on the estimation from the last sample. Table 1 summarizes the whole algorithm.

Table 1: Recursive weighted least squares algorithm

Initialization

$$\hat{x}_0 = E(x)$$
$$P_0 = E[(x - \hat{x}_0)(x - \hat{x}_0)]$$

For $k = 1,2, \ldots$

$$y_k = H_k x + v_k$$
$$K_k = P_{k-1} H_k^T \cdot \left(H_k P_{k-1} H_k^T + R_k\right)^{-1}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k \cdot (y_k - H_k \hat{x}_{k-1})$$
$$P_k = (I - K_k H_k) P_{k-1} (I - K_k H_K)^T + K_\nu R_\nu K_\nu^T$$

2.2. Recursive linear Kalman Filter

For many real-time control problems, it is more interesting to estimate the system states rather than some constant parameters. Therefore the *linear Kalman Filter* was developed in the 60's. It enables to estimate the system states of a linear discrete-time model in a recursive way. The fundamental assumption is that the system and the output equations are disturbed by white Gaussian noise. Both of these noise processes are regarded as uncorrelated with zero mean. This results in the following equations:

$$\begin{aligned} x_k &= F_{k-1} x_{k-1} + G_{k-1} u_{k-1} + w_{k-1} \\ y_k &= H_k x_k + v_k \\ E\left(w_k w_j^T\right) &= Q_k \delta_{k-j} \\ E(v_k v_k^T) &= R_k \delta_{k-j} \\ E(w_k v_k^T) &= 0 \end{aligned} \tag{2.16}$$

At this point, we introduce the principle of every Kalman Filter derivation (compare Figure 2). Subsequent to filter initialization, the first step in every sample is the a-priori estimation of the mean (system states) and the covariance (a gauge for the confidence in them). This is called the prediction step and all of the equations that are related with it contain a "-" in the superscript.

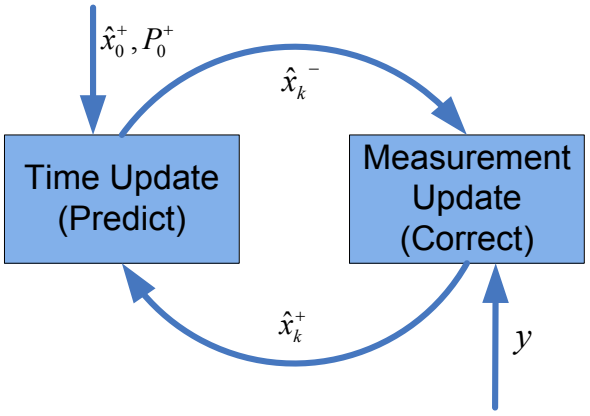

Figure 2: Principle of recursive Kalman filter.

This forms the basis for the calculation of the optimal Kalman gain that is used to correct the estimated state vector with the information from the actual measurements. Finally, the covariance matrix is updated. This is called the correction step. In the next sample, these values are used to restart again at the subsequent prediction step. The algorithm can be formulated as follows:

Table 2: Linear discrete Kalman Filter

Initialization

$$\hat{x}_0 = E(x_0)$$
$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)]$$

For $k = 1,2, \ldots$

$$\hat{x}_k^- = F_{k-1}x_{k-1}^+ + G_{k-1}u_{k-1}$$
$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q$$
$$K_k = P_k^-H_k^T \cdot \left(H_kP_k^-H_k^T + R\right)^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot (y_k - H_kx_k)$$
$$P_k^+ = (I - K_K \cdot H_k) \cdot P_k^-$$

To determine the relationship between the *Kalman Filter* and *recursive weighted lest squares,* we should have a closer look at Table 2. The matrix $Q(ww^T) = diag(\sigma_{w1}^2, \ldots, \sigma_{wn}^2)$ represents the covariance of the system states (w denotes the variance of the system states). Its entries represent the confidence in the a-priori estimation and can be tuned by the application engineer. Large values represent high uncertainty (probably due to an imprecise model), whereas small values indicate good trust. The second tuning matrix R represents the confidence in the actual measurements. Its effect resembles our first estimation problem (eq. (2.1) to (2.5)). Furthermore, it can be shown that if x_k is a constant vector then $F_k = I, Q_k = 0$ and $u_k = 0$. In this case, the *Linear discrete Kalman Filter algorithm* (Table 2) reduces to the *recursive weighted least squares algorithm* (Table 1). This property is often exploited in the formulation of parameter estimation problems using *Kalman Filter algorithms*.

3. Kalman based Estimation Algorithms and Extensions

So far, we have discussed estimation problems for linear discrete systems. This is generalized to nonlinear systems starting from a continuous-time representation in state space form:

$$\dot{x} = f(x,u)$$
$$y = g(x) \qquad (3.17)$$

3.1. Recursive nonlinear linearization based methods

3.1.1. EKF

In Table 3, the widely used extension of the *discrete linear Kalman Filter* to the *discrete nonlinear Kalman Filter with additive noise* is presented. The dynamic system is represented as follows:

$$\begin{aligned} x_k &= f_{k-1}(x_{k-1}, u_{k-1}) + w_{k-1} \\ y_k &= h_k(x_k) + v_k \\ \mathrm{w_k} &\cong (0, \mathrm{Q_k}) \\ \mathrm{v_k} &\cong (0, \mathrm{R_k}) \end{aligned} \tag{3.18}$$

The algorithm is very similar to a purely linear one. To handle the nonlinearity, the system is linearized around the last estimation point using a Taylor Series Expansion up to the first term. This can be performed numerically by the use of a forward difference formula.

Table 3: Extended Kalman Filter Algorithm

Initialization

$$\hat{x}_0 = E(x_0)$$
$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)]$$

For $k = 1,2, \ldots$

$$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1})$$
$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q$$
$$where\ F_{k-1} = \left.\frac{\partial f_{k-1}}{\partial x}\right|_{\hat{x}_{k-1}^+}$$
$$K_k = P_k^- H_k^T \cdot \left(H_k P_k^- H_k^T + R\right)^{-1}$$
$$where\ H_k = \left.\frac{\partial h_k}{\partial x}\right|_{\hat{x}_k^-}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot (y_k - h_k(\hat{x}_k^-))$$
$$P_k^+ = (I - K_K \cdot H_k) \cdot P_k^-$$

Since we have a nonlinear continuous-time system representation, we have to linearize and discretize our system at every sample instant. Discretization means to integrate the system in the prediction step from the last sample instant to the new one, e.g. with the *Trapezoidal* or the *Runge-Kutta 4* integration method. The transition matrix F_{k-1} is calculated by an analytic derivation of the system state Jacobian. An alternative is the numerical calculation with, e.g., a forward difference formula:

$$For\ i = 1,2, \ldots, n$$
$$J_{\hat{x}_{k-1}^+}^{[:,i]} = \frac{f(\hat{x}_{k-1}^+ + h \cdot E(:,i), u) - f(\hat{x}_{k-1}^+, u)}{h} \tag{3.19}$$

The transition matrix can be computed with a matrix exponential function resulting in:

$$F_{k-1} = e^{\left(J_{\hat{x}_{k-1}^+} \cdot T_s\right)} \tag{3.20}$$

3.1.2. Numerically stable and reliable EKF implementations

The sketched Extended Kalman Filter algorithm in Table 3 has several difficulties, when it should be implemented on micro controllers with a limited word length of the data types (i.e. a double). So the numerical approximation of the Jacobian matrix can cause inaccuracies due to cancelling effects and therefore a wrong approximation of the nonlinearity in the surrounding of the working point in the actual step. A more serve problem is the calculation of the propagated covariance matrix via the difference
$P_k^+ = P_k^- - P_k^- \cdot K_K \cdot H_k$ (compare Table 3). The so called *divergence phenomenon* (Haykin, 2001) may cause that the propagated matrix P_k^+ may no longer be positive definite (what's a necessary condition of a covariance matrix).

To cope with this problem there are different approaches that enhance the numeric accuracy of the standard EKF algorithm. The most common one are Square Root (SR) filtering by means of *Cholesky* or *U-D Decomposition*. In SR filtering algorithms the square root of the covariance matrix

$$P_k = S_k' \cdot S_k \tag{3.21}$$

is propagated in the recursive formulation. Defining the condition number of a matrix as

$$\kappa(P_k) = \frac{\sigma_{max}(P_k)}{\sigma_{min}(P_k)} \geq 1 \tag{3.22}$$

wherein $\sigma_{max/min}$ are the largest and smallest singular value, it can be shown that the condition number of the square root S_k is the square root of the condition number the matrix P_k (see (Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, 2006) for more details)

$$\begin{gathered} \sigma^2(P_k) = [\sigma^2(S_k)]^2 \\ \frac{\sigma_{max}(P_k)}{\sigma_{min}(P_k)} = \frac{\sigma_{max}^2(S_k)}{\sigma_{min}^2(S_k)} \\ \kappa(P_k) = \kappa^2(S_k) \end{gathered} \tag{3.23}$$

At this point no further detail on the implementation is given. The most common publications regarding the numeric efficient and reliable implementation of the different approaches are given in (Grewal & Andrews, 2001).

3.2. Recursive nonlinear derivative free method

In this chapter we show up algorithms that do not need the calculation of the Jacobians of the nonlinear prediction model. Besides the in the following discussed *Unscented Transformation* (UT) based algorithm there have to be mentioned the *Central Difference Filtering* (CDF) (i.e. (Nørgaard, Poulsen, & Ravn, 1998)) and the *Gaussian Quadrature Kalman Filter* (CGQ) (Arasaratnam, Haykin, & Elliott, 2007). Both methods have been shown to be strongly connected with the UT and are therefore not considered separately (Haykin, 2001). The so called *Sigma Point Transformation* bases on the idea that it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function or transformation ((Julier & Uhlmann, 2004), (Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, 2006)). In other words it is difficult to find a nonlinear transformation of a *Propability Density Function* (PDF), but it is easy to perform a nonlinear transformation of a vector. Therefore this algorithm chooses a set of points around the actual operation points in a heuristic manner and then transforms them through the underlying nonlinear function. With the help of weighting factors and a transformed set of points in the state space a sampled PDF can be approximated (Simon, Kalman filtering with state constraints: a survey of linear and nonlinear algorithms, 2010). In this way a more appropriate solution, in comparison to the EKF, of the mean and covariance can be found. So in general this approximation (for Gaussian assumption) is accurate to third terms of the tailor series expansion. For non-Gaussian assumption it is still valid to the second order (der & Wan, 2001). For a proof of this refer to (Haykin, 2001) – Appendix A. One can easily see the advantage of UT in Figure 3 where a comparison between a Monte Carlo simulation, the unscented approach and the extended Kalman Filter is shown. It is quite clear that the truncation of the tailor series expansion after the first term (EKF) leads to the most inaccurate results.

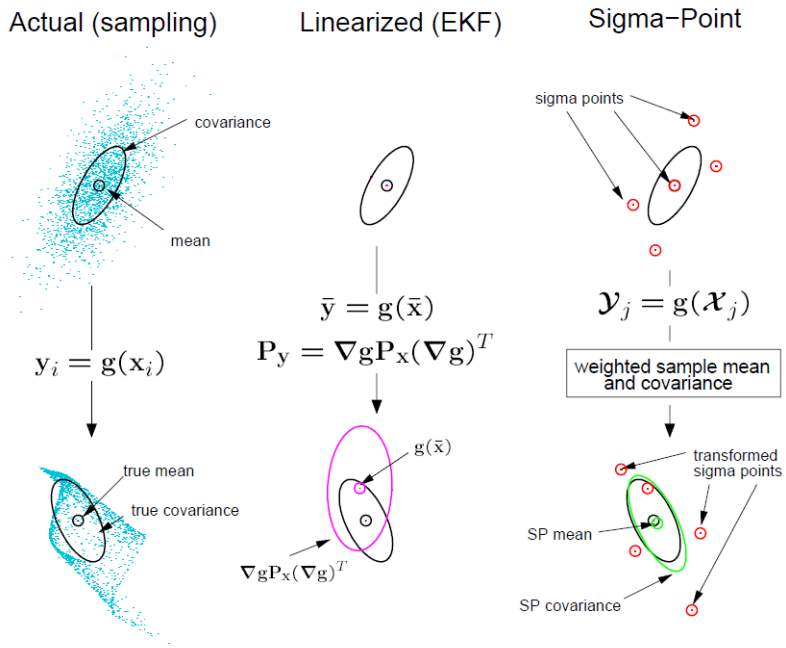

Figure 3: Advantages of Unscented Transformation (Merwe, 2004)

In the following some more details are given regarding the principles of the UT and its implicit context with *Weighted Statistical Linear Regression* (WSLR) (Merwe, 2004). The aim of the WSLR is to find a linear regression

$$y = g(x) \approx Ax + b \tag{3.24}$$

for the nonlinear function $y = g(x)$ through the evaluation of it in N points $(X_i, i = 1..N)$. The set of X_i is chosen that it can represent the statistical properties of x like mean $\bar{x}$ and covariance P_x

$$\begin{aligned} \bar{x} &= \Sigma_{i=1}^{N} w_i X_i \\ P_x &= \Sigma_{i=1}^{N} w_i (X_i - \bar{x})(X_i - \bar{x})^T \\ \Sigma_{i=1}^{N} w_i &= 1 \; (regression\; weights) \end{aligned} \tag{3.25}$$

The optimization objective to find a solution for equation (3.24) writes as follows,

$$\begin{aligned} \{A, b\} &= \min E[\epsilon^T W \epsilon] \approx \min \Sigma_{i=1}^{N} w_i \epsilon_i^T \epsilon_i \\ \epsilon_i &= g(X_i) - (AX_i + b) \end{aligned} \tag{3.26}$$

where ϵ_i denotes the point-wise linearization error and we assume that the chosen regression points X_i gather the prior mean and covariance of x. According to (Merwe, 2004), A, b can be derived through *Statistical Linearization* as the following expressions:

$$\begin{aligned} A &= P_{xy}^T P_x^{-1} \\ b &= \bar{y} - A\bar{x} \end{aligned} \tag{3.27}$$

To calculate the unknown posteriori Gaussian statistics, the covariance P_{xy}, P_y and mean value $\bar{y}$, the following assessments of the propagated regression point $\gamma_i = g(X_i)$ can be assumed (compare eq. (3.25)):

$$\begin{aligned} \bar{y} \approx \hat{y} &= \Sigma_{i=1}^{N} w_i \gamma_i \\ P_y \approx \hat{P}_y &= \Sigma_{i=1}^{N} w_i (\gamma_i - \hat{y})(\gamma_i - \hat{y})^T \\ P_{xy} \approx \hat{P}_{xy} &= \Sigma_{i=1}^{N} w_i (X_i - \bar{x})(\gamma_i - \hat{y})^T \\ P_\epsilon \approx \Sigma_{i=1}^{N} w_i \epsilon_i^T \epsilon_i &= \hat{P}_y - A P_x A^T \end{aligned} \tag{3.28}$$

Where ϵ_i is defined as the point-wise linearization error:

$$\epsilon_i = g(X_i) - (AX_i + b) \tag{3.29}$$

By help of these assumptions the statistics of y can be expressed as follows:

$$\begin{aligned} \hat{y}^{slr} &= A\bar{x} + b \\ \hat{P}_y^{slr} &= A P_x A^T + P_\epsilon \end{aligned} \tag{3.30}$$

The following graphic repeats the above sketch for interconnection of statistical properties by hand of a simple example.

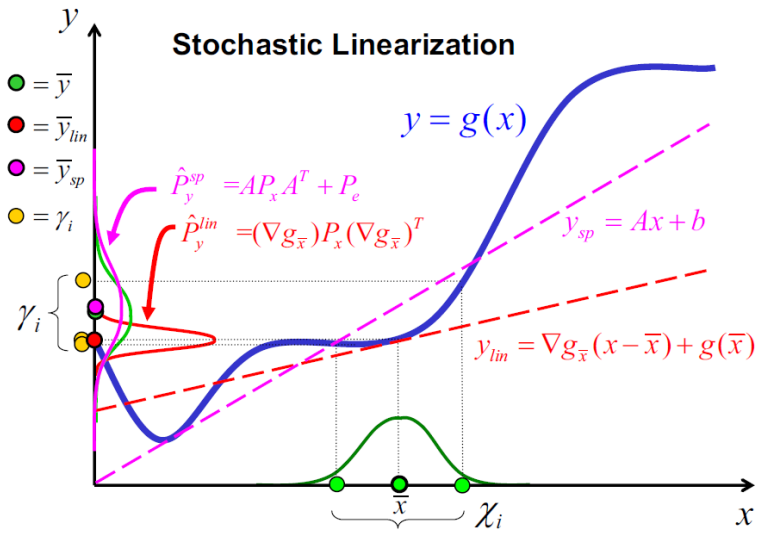

Figure 4: WSLR vs. 1.order Taylor Approx. (Merwe, 2004)

In Figure 4 a one dimensional *Gaussian Random Variable* x (GRV) with a certain distribution (green line) and a mean $\bar{x}$ is transformed through a nonlinear function $y = g(x)$ (blue line) in order to produce the a posteriori random variable y. The green line on the y axis indicates the true mean and covariance of the distribution after the transformation. Considering the two different approximations of the transformation (lin = first order taylor approximation [red], sp = sigma point transformation [magenta]) one can see instantaneous the difference in the accuracy of the approaches. The linearization $y_{lin} = \nabla g_{\bar{x}}(x - \bar{x}) + g(\bar{x})$ of the nonlinear function results in a strongly biased mean (red dot) and a totally different distribution of P_y (red line). This is reasoned on the one hand side that the weighted statistical linearization (WSLR) calculates an approximate expected Jacobian whereas the EKF simply calculates the Jacobian at the prior mean $\bar{x}$. On the other hand WSLR considers a non-zero bias b, whereas the first order Taylor Series approximation of the EKF Jacobian drops this term (Merwe, 2004).

Now we compare the UKF algorithm (see *Table 6*) with the WSLR method, to show up it's implicit context as mentioned in the next to last section. The calculation of the covariance $P_k^-, P_{y_k}, P_{x_k y_k}$ are performed in similar way, besides adding additive uncertainties Q, R, like in eq (3.25), (3.28). Another interesting point is the calculation of the Kalman gain $K_k = P_{x_k y_k} \cdot {P_{y_k}}^{-1}$. It optimally propagates the new information of the measurement reversely in the state space. Comparing it with the WSLR gain A (eq. (3.27)), this can *be interpreted as propagating the innovation information downwards (from y-space to the x-space) using a*

statistically linearized inverse observation function (Merwe, 2004).

3.2.1. Unscented Kalman Filter Algorithm

In order to achieve higher accuracy, the *Unscented Kalman Filter* (UKF) calculates the mean and covariance from disturbed state vectors, by use of the uncented transformation of Sigma Points (see for more details chap. 3.2). Furthermore the Jacobians of f(x) and g(x) are not needed for the approximation of the nonlinear prediction model. The structure of the equation set, containing prediction and update, is similar to the EKF (Table 3). However, the calculation of the covariances requires to integrate the nonlinear system $2n + 1$ times from the last to the actual time instant ($X_{(k|k-1)}$ step in Table 6) and is therefore computationally costly. The symmetry of all the involved matrices is fully exploited to reduce computational costs.

All $\sqrt{...}$ matrix operations are done by the use of a Cholesky Decomposition (Anderson, et al., 1999). This decomposition is used due to the fact that the covariance matrix is symmetric positive definite. For all calculus always only the lower triangular of the solution is used. $X_{(k|k-1)}$ denotes the propagation from the time step k-1 to k through the continuous nonlinear model by the use of discrete integration method (i.e. Runge-Kutta 4). In the following *Table 4* and *Table 5* a list of tuning parameters with typical default values for the UKF are given. Finally the algorithm with additive noise assumption Q, R is sketched in the pseudo code in *Table 6*.

Table 4: List of UKF parameters

n	Number of states
α	Spread of sigma points around the actual mean (e.g. $1 \geq \alpha \geq 10^{-4}$)
κ	Scaling kurtosis of sigma points (Default = 0, or $3 - n$)
β	Characteristic of the statistic distribution of noise process (For Gaussian distribution 2 is optimal)

Table 5: List of pre-calculated weightings

$\lambda = \alpha^2 \cdot (n + \kappa) - n$	Scaling parameter
$a = \alpha^2 \cdot (n + \kappa)$	Denominator argument of weightings
$w_0^m = \frac{\lambda}{a}$	Weighting of unmodified mean prediction
$w_0^c = \frac{\lambda}{\alpha + 1 - \alpha^2 + \beta}$	Weighting of unmodified output mean prediction
$w_i^m = w_i^c = \frac{1}{2 \cdot a}\ ; i = 1..2n$	Weighting of sigma points of states and outputs
$\gamma = \sqrt{\alpha^2 \cdot (n + \kappa)}$	Weighting of square root of covariance from $k - 1$

Table 6: Unscented Kalman Filter Algorithm

Initialization

$$\hat{x}_0 = E(x_0)$$
$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$$

For $k = 1,2, \ldots$

Prediction and time update

$$X_{k-1} = [\hat{x}_{k-1}, \hat{x}_{k-1} + \gamma \cdot \sqrt{P_{k-1}^+}, \hat{x}_{k-1} - \gamma \cdot \sqrt{P_{k-1}^+}]$$
$$X_{k|k-1} = F[X_{k-1}, u_{k-1}]$$
$$\hat{x}_k^- = \sum_{i=0}^{2 \cdot n} w_i^m \cdot X_{i,k|k-1}$$
$$P_k^- = \sum_{i=0}^{2 \cdot n} w_i^c \cdot [X_{i,k|k-1} - \hat{x}_k^-][X_{i,k|k-1} - \hat{x}_k^-]^T + Q$$
$$X_k = [\hat{x}_k^-, \hat{x}_k^- + \gamma \cdot \sqrt{P_k^-}, \hat{x}_k^- - \gamma \cdot \sqrt{P_k^-}]$$
$$Y_k = H[X_k]$$
$$\hat{y}_k^- = \sum_{i=0}^{2 \cdot n} w_i^m \cdot Y_{i,k}$$

Measurement update

$$P_{y_k} = \sum_{i=0}^{2 \cdot n} w_i^c \cdot [Y_{i,k} - \hat{y}_k^-][Y_{i,k} - \hat{y}_k^-]^T + R$$
$$P_{x_k y_k} = \sum_{i=0}^{2 \cdot n} w_i^c \cdot [X_{i,k|k-1} - \hat{x}_k^-][Y_{i,k} - \hat{y}_k^-]^T$$
$$K_k = P_{x_k y_k} \cdot {P_{y_k}}^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot (y_k - \hat{y}_k^-)$$
$$P_k^+ = P_k^- - K_k \cdot P_{y_k} \cdot K_k^T$$

3.2.2. Square Root Unscented Kalman Filter Algorithm

The equations of the SR-UKF are identical to the UKF, but the fact that the covariance matrices must be always postitive defintite is exploited. Although the covariance matrix P_k and the predicted covariance matrix P_k^- are uniquely defined by their Ckolesky factors $\sqrt{P_k}$ and $\sqrt{P_k^-}$ respectively, with UKF the covariance matrices are calculated at each step. Furthermore, the sigma points X_k can be computed with the Cholesky factor $\sqrt{P_k}$, and the updated sigma points of the measurement update with the Cholesky factor $\sqrt{P_k^-}$ without using the covariance matrices. Moreover, the gain matrix K_k is determined as solution of the linear equation system

$$K_k \cdot P_{y_k y_k} = P_{x_k y_k} \tag{3.31}$$

that can be more efficiently solved by utilizing again the Cholesky factorization:

$$K_k = P_{x_k y_k} \cdot \left(S'_{y_k} S_{y_k}\right)^{-1} \tag{3.32}$$

In the SR-UKF implementation, the Cholesky factors are propagated directly and the refactorization of the covariance matrices is avoided through rank one updates of the cholesky matrix (der & Wan, 2001). In this way

it is guarantied that the cholesky factor are always regular, what ensures numerical robustnees of ther algorithm.

In *Table 7* the algorithm of the UKF-SR is given, therefore in this section the most important numerical operators are explained. LQ denotes the transpose of the QR decomposition in order to be consistent with the other lower triangular operations (Cholesky). The matrix Q isn't computed directly to minimize the number of necessary operations and only R is calculated ($A = QR \in R^{m \times n}$). The operation $cholupd(CP, v, \pm 1)$ denotes a rank one update of a Cholesky factorized matrix $A = L' \cdot L$ ((Dongarra, Moler, Bunch, & Stewart, 1979), (Seeger, 2007)). It computes a lower triangular matrix $L^* = L \pm vv^T$ that guaranties that the matrix $A^* = L'^* \cdot L^*$ keeps still positive definite and makes the UKF-SR algorithm therefore more robust against numerical instability then the classic UKF (see *Table 6*) algorithm. The parameters and precalculated weights are the same as in the case of the UKF algorithm (see *Table 4* & *Table 5*).

Table 7: Square root unscented Kalman Filter algorithm

Initialization

$$\hat{x}_0 = E(x_0)$$
$$CP_0^+ = \sqrt{E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]}$$

For $k = 1,2, \ldots$

Prediction and time update

$$X_{k-1} = [\hat{x}_{k-1}, \hat{x}_{k-1} + \gamma \cdot CP_{k-1}^+, \hat{x}_{k-1} - \gamma \cdot CP_{k-1}^+]$$
$$X_{k|k-1} = F[\mathrm{X}_{k-1}, u_{k-1}]$$
$$\hat{\mathrm{x}}_k^- = \sum_{i=0}^{2 \cdot n} w_i^m \cdot \mathrm{X}_{i,k|k-1}$$
$$CP_k^- = LQ\left(\left[\sqrt{w_1^c} \cdot (X_{1:2 \cdot n,k|k-1} - \hat{x}_k^-), \sqrt{Q}\right]\right)$$
$$CP_k^- = cholupd\left(CP_k^-, X_{k|k-1} - \hat{x}_k^-, w_0^c\right)$$
$$X_k = [\hat{x}_k^-, \hat{x}_k^- + \gamma \cdot CP_k^-, \hat{x}_k^- - \gamma \cdot CP_k^-]$$
$$Y_k = H[X_k]$$
$$\hat{y}_k^- = \sum_{i=0}^{2 \cdot n} w_i^m \cdot Y_{i,k}$$

Measurement update

$$S_{y_k} = LQ\left(\left[\sqrt{w_1^c} \cdot (Y_{1:2 \cdot n,k} - \hat{y}_k^-), \sqrt{R}\right]\right)$$
$$CP_k^- = cholupd\left(S_{y_k}, Y_{0,k} - \hat{y}_k^-, w_0^c\right)$$
$$P_{x_k y_k} = \sum_{i=0}^{2 \cdot n} w_i^c \cdot \left[\mathrm{X}_{i,k|k-1} - \hat{x}_k^-\right]\left[Y_{i,k} - \hat{y}_k^-\right]^T$$
$$K_k = P_{x_k y_k} \cdot \left(S'_{y_k} S_{y_k}\right)^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot (y_k - \hat{y}_k^-)$$
$$U = K_k \cdot S_{y_k}$$
$$CP_k^+ = cholupd(CP_k^-, U, -1)$$

4. References

Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J. J., et al. (1999). *LAPACK Users' guide (third ed.)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Arasaratnam, I., Haykin, S., & Elliott, R. (2007). Discrete-Time Nonlinear Filtering Algorithms Using Gauss Hermite Quadrature. *Proceedings of the IEEE, 95*(5), 953-977.

der, R. V., & Wan, E. A. (2001). The square-root unscented Kalman filter for state and parameter-estimation., *6*, S. 3461-3464 vol.6.

Dongarra, J. J., Moler, C. B., Bunch, J. R., & Stewart, G. W. (1979). *LINPACK User's Guide*. SIAM.

Grewal, M. S., & Andrews, A. P. (January 2001). *Kalman Filtering : Theory and Practice Using MATLAB* (2 Ausg.). Wiley-Interscience.

Haykin, S. (October 2001). *Kalman Filtering and Neural Networks*. Wiley-Interscience.

Julier, S. J., & Uhlmann, J. K. (2004, March). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE, 92*(3), 401-422.

Ljung, L. (1998). *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR.

Merwe, R. v. (2004). *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. Ph.D. dissertation.

Nørgaard, M., Poulsen, N. K., & Ravn, O. (1998). *Advances in Derivative-Free State Estimation for Nonlinear Systems*. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark,, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Seeger, M. (2007). Low rank updates for the Cholesky Decomposition. *University of California at Berkeley, Tech. Rep*.

Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches* (1. Auflage Ausg.). Wiley & Sons.

Simon, D. (2010). Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications, 4*(8), 1303+.

Wan, E., & van der Merwe, E. (kein Datum). Sigma-Point Kalman Filters (Overview).