# A Users Exploration of Data Assimilation Techniques for Parameter Fitting on a Simple ODE Model

An Do[a], Lisette de Pillis[b,*], Christina J. Edholm[c], Blerta Shtylla[d], Christina Catlett[c], Daniel Shenker[e], Rachel Wander[b], Maya Watanabe[f]

[a] *Claremont Graduate University, Claremont, CA, USA*
[b] *Harvey Mudd College, Claremont, CA, USA*
[c] *Scripps College, Claremont, CA, USA*
[d] *Pomona College, Claremont, CA, USA*
[e] *Johns Hopkins University, Baltimore, MD, USA*
[f] *University of Massachusetts Amherst, Amherst, MA, USA*

**Abstract**

We are trying to make this data assimilation approach accessible, which is why we include code. You know we want people working at all levels to be able to take advantage of these techniques. For example, Kalman filters, historically confined to the realm of some engineers, not very often introduced to mathematics students (doing modeling). There have been recent forays into using Data Assimilation techniques (Kalman filters which are part of that) in biological models (Albers) But because there are challenges with the stability of the numerics (among other things) these approaches have not become widespread. We have found certain challenges. Certain implementation challenges such as stability of the numerics and even clarity of explanation as to what these algorithms do. Covariance matrices have to be positive definite. But they lose this characteristic all the time. Many have tried different ways to adjust this problem. We want to mention the issues and share the practical experiences in implementing these techniques, what is working and what is not.

*Keywords:* `elsarticle.cls`, LaTeX, Elsevier, template

*2010 MSC:* 00-01, 99-00

[☆] Fully documented templates are available in the elsarticle package on CTAN.

[*] Corresponding author
*Email address:* `email.edu` ()

**Contents**



*0.1. Unscented Kalman Filter*

The Kalman filter (KF) was developed by Kalman in 1960 [**?** ]. It is an iterative prediction-correction scheme that uses a set of equations and a series of measurements observed over time, including their statistical noises and measurement errors to produce estimates of the objects of interest. A unique feature about the Kalman filter approaches is that it can estimate the value of unobservable variables (latent) given a series of data of the observable state variables [**?** ].

In the context of mechanistic modeling, latent states are the unknown parameters while the observables are the state variables whose data has been measured or observed from experimental studies. The objective goal is to use the available data of state variables to estimate the model parameters that yield the best estimates of the model outputs [**?** ].

The Kalman filter is normally used for linear systems. When system dynamics are nonlinear, the extended Kalman filter is normally executed. However, the extended kalman filter suffers from divergence issue [**?** ] because linearization does not always capture the correct dynamics of the underlying system [**?** ]. As a result, several new filtering methods have recently been introduced on the basis of the Kalman filter such as dual unscented Kalman filters [**?** **?** ], joint unscented Kalman filters [**?** **?** ], ensemble Kalman filters [**?** **?** ].

The advantage of these "derivativeless" approaches is that rather than seeking to linearize the nonlinear dynamics of the system, they deterministically sample the joint density of the states in such a way that the mean and covariance are preserved. The full nonlinear system dynamics are then applied to these sample points in order to propagate the density through the prediction step of the filter. In this paper, we will use the joint unscented kalman filter along with other parameter estimation techniques to estimate unknown parameters in a mechanistic biological model.

The joint unscented algorithm is a two-step process: predict and update. In the prediction step, algorithm uses the system dynamics to update the previous measurement before any new data input becomes available. Then, these estimates are updated once the new data becomes available. This two-step recursion is then applied at each successive time period, using only the present data and the previously states estimates and its uncertainty matrix without any additional past information is required.

*Implementation*

Before outline the implementation steps of Joint UKF, we want to review some critical assumptions (illustrated in Equation (1)- (2)) about the algorithm as these features play a critical role in the algorithm.

$$x_{k+1} = F(x_k) + q_{k+1} \tag{1}$$

$$y_{k+1} = H(x_{k+1}) + r_{k+1} \tag{2}$$

- The time prediction of the state variables and parameters $x_{k+1}$ at each time step driven by a nonlinear dynamical system $F$.
- There is some noise/uncertainty $q$ associated with the ODE model solver (*"process noise"*). This known quantity is defaulted in our paper and discussed in the Supplementary information.
- The actual estimated observables $y_{k+1}$ comprises of only the state variables of the model (excluding the model parameters).
- Function $H$ in equation (1) represents the relationship between the projected states variables and the actual observables. For example, rather than a single or all state variables can be observed, only some algebraic combination of them can be observed [**?** ].
- $r$ represents the *"measurement noise"* in obtaining the actual data.

Unscented Transformation. This section writing is very confusing. The general idea of sigma points are just some random samples around the previously estimated states/parameter vector. Details of this UT has been mentioned in greater details in [**? ? ?** ]. UT is not a bottleneck of this algorithm according to our experience .However, it is the vehicle for the covariance matrix and error

covariance matrix to show up.

**Algorithm 1:** Say something about Joint UKF algorithm

**Result**: Estimated states and model parameters over time

Initialize states and model parameters in a vector $x_0$ and covariance matrix $P_0$;

**for** *every time data point* **do**

  At time $t = k$;

  Draw as subset of sigma points around $_1$;

  Estimate the next time update (both states and parameters) via the dynamical system ;

  Predict $\hat{x_k}, \hat{P}_k$, the projected states/parameter at time $t = k$ and projected covariance matrix, which is the weighted average of all the transformed sigma points Update the prediction when new data becomes available ;

  Update the prediction when new data becomes available

**end**

### *0.1.1. Initialization*

To create these initial guesses $x_0$ and $P_0$, some prior knowledge of the states and their plausible ranges is necessary. Assuming this information is available, one may initialize with:

$$\hat{x}_0 = E[x_0], P_0 = \text{identity matrix scaled by } 10^{-4}, \text{ why?} \tag{3}$$

### *0.1.2. Projection Step*

Now, we proceed to do the projection portion of the algorithm. Having chosen a set of sigma points via equations **??**, **??**, and **??** , we now project them in time by applying the transition matrix to each point in the vector $\chi_{k-1}$:

$$\chi_{k|k-1} = F(\chi_{k-1}). \tag{4}$$

It is important to note that here $\hat{x}_{k-1}$ plays the role of $\bar{v}$ in equations **??**, **??**, and **??**. Next we calculate a prior prediction of the state $\hat{x}_k^-$ with:

$$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \chi_{i,k|k-1}. \tag{5}$$

Similarly we can calculate a prior covariance $P_k^-$ by using our weights as well as the process noise:

$$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\chi_{i,k|k-1} - \hat{x}_k^-][\chi_{i,k|k-1} - \hat{x}_k^-]^T + Q. \tag{6}$$

We can see that the calculation of $\hat{x}_k^-$ and $P_k^-$ are both *applications of the Unscented Transformation* as we created, transformed, and weighted sigma vectors. Next, we apply the UT transformation to understand our observables with:

$$Y_{k|k-1} = H[\chi_{k|k-1}]. \tag{7}$$

This allows us to know calculate a prior estimate of the observables $y_k^-$ by utilizing:

$$\hat{y}_k^- = \sum_{i=0}^{2L} W_i^{(m)} Y_{i,k|k-1}. \tag{8}$$

Once again, we have made use of the *UT transformation* here by passing our sigma vectors through $H$ this time, as opposed to $F$ as before.

### *0.1.3. Update Step*

Having created prior estimates for both states and observables, we now proceed to the update step where the observed data is brought in [**?** ].

We begin by calculating the covariance matrix $P_{\tilde{y}_k,\tilde{y}_k}$, which is the covariance matrix for the error between the projected and actual values of the observables.

Moreover, this calculation takes into account the measurement noise through $R$:

$$P_{\tilde{y}_k,\tilde{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [Y_{i,k|k-1} - \hat{y}_k^-][Y_{i,k|k-1} - \hat{y}_k^-]^T + R. \tag{9}$$

Next we need a covariance between our states and observables, which is calculated by:

$$P_{x_k,y_k} = \sum_{i=0}^{2L} W_i^{(c)} [\chi_{i,k|k-1} - \hat{x}_k^-][Y_{i,k|k-1} - \hat{y}_k^-]^T. \tag{10}$$

At this point, we have sufficient information to calculate the **Kalman Gain**, a crucial term in the UKF algorithm. The Kalman Gain is used to weight the error between the predicted value and the observed value of $x_k$ in order to update the filter's prior predictions. A larger Kalman Gain will give more weight to the error, resulting in a larger correction and a smaller Kalman Gain will give less weight to the error, resulting in a smaller correction. The Kalman Gain is also used to adjust the covariance matrix $P_k$ and is calculated through [**?** ]:

$$K_k = P_{x_k,y_k} P^{-1}_{\tilde{y}_k,\tilde{y}_k}. \tag{11}$$

Calculation of the Kalman Gain allows us to calculate our posterior estimates of both the states, $\hat{x}_k$, and covariance, $P_k$ [**?** ]. For the states the following equation is used

$$\hat{x}_k = \hat{x}^-_k + K_k(y_k - \hat{y}^-_k). \tag{12}$$

And for the covariance we use:

$$P_k = P^-_k - K_k P_{\tilde{y}_k,\tilde{y}_k} K^T_k. \tag{13}$$

We can see that both equations function by adjusting our prior prediction through use of the Kalman Gain.

After having gone through the update step, the next data point would be brought in and we would return to the Projection Step once more.

#### *0.1.4. Summary*

A flow chart depicting one iteration of the entire UKF process is found in figure **??**. To summarize, our goal is to begin with a set of sigma points chosen in a deterministic fashion and arrive at the posterior estimates for the states and covariance. This is done in two main sections, the Projection Step, whose quantities are shaded in red, and the Update Step, displayed in blue.

## 1. Implementation of the DA techniques on Lotka-Volterra

### *1.1. Pros & Cons*

Need the previous sections to be finalized first.

### *1.2. Covariance matrices for UKF*

#### *1.2.1. Stability and Divergence Issues*

When utilizing the UKF, it is important to consider the issue of positive definitiness of the covariance matrix $P_k$. If this matrix becomes non-positive definite, the filter can experience severe divergence issues due to invalid covariance matrices, which can occur as a result of the update made in equation 13. This then can lead the filter to diverge [**?** ].

The solution to the issue lies in using the **Cholesky factorization** [**?** ].

At every iteration of the Kalman Filter, one must perform the following step:

$$P_k = {P_k}^{1/2} {P_k}^{T/2}, \tag{14}$$

where ${P_k}^{1/2}$ is the **Cholesky factor** and is in lower-triangular form, and ${P_k}^{T/2}$ is the Cholesky factor's transpose [**?** ]. Here, $P_k$ is now the product of a square matrix and its transpose, which is **guaranteed to be positive definite**. In order to find the Cholesky factor, a function such as Matlab's *chol* (https://www.mathworks.com/help/matlab/ref/chol.html) can be used. By using this approach, one will have a better chance of maintaining the stability of the Kalman Filter across iterations.

## 2. Numerical results for parameter and state estimation

### *2.1. Lotka-Volterra model*

The model was developed independently by Alfred J. Lotka and Vito Volterra in the 1920s to describe the interaction between hares (prey, denoted as $h$) and lynx (predator, denoted as $h$). The model consists of two ordinary differential equations 15-16 and contains 4 parameters presented in Table 1. We are interested in employing the aforementioned parameter fitting techniques (PSO, DRAM, UKF) to estimate their values from real-life data.

$$\frac{dh}{dt} = \alpha h - \beta hl \tag{15}$$

$$\frac{dl}{dt} = -\gamma l + \delta hl \tag{16}$$

| Parameter | Description |
|---|---|
| $\alpha$ | growth rate of hares |
| $\beta$ | death rate of hares due to being consumed by lynx |
| $\gamma$ | death rate of lynx due to absence of hares |
| $\delta$ | growth rate of lynx due to sufficient food source |

Table 1: Parameter notation and definition of the Lotka-Voterra model described in equation (15)-(16)

In equation (15), the primary growth in the hare population is in proportion to its own population and their loss is due to predation by lynx. Equation (16) describes the change in lynx population. First, their growth depends on sufficient food (hares), which is similar to the death rate for the hare population, Additionally, their loss is presumed to be due to the absence of hares.

This model rests on several simplifying assumptions. First, hare population will grow exponentially in the absence of the predator population. Second, lynx population will starve in the absence of prey. Finally, there is no limit to the number of hares that can be consumed by any lynx.

The model's solution has been shown to experience oscillating behavior, with the peak of the lynx's oscillation lagging behind that of the hare's [**? ?** ].

*2.2. Time-invariant parameters*

*2.3. Time-varying parameters*

*2.4. Large data-set*

**3. Conclusions**

*3.1. PSO & DRAM for time-invariant parameters*

*3.2. UKF for time-varying parameters*

*3.3. Our contributions*