SoftwareX 7 (2018) 28–33

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

The EKF-AUS-NL algorithm implemented without the linear tangent model and in presence of parametric model error

Luigi Palatella [a,*], Fabio Grasso [b]

[a] Liceo Scientifico Statale "C. De Giorgi", viale De Pietro 14, I-73100 Lecce, Italy
[b] Institute of Science of Atmosphere and Climate ISAC-CNR, Str. Prov. Lecce-Monteroni km 1, 200, I-73100, Lecce, Italy

ARTICLE INFO

Article history:
Received 8 September 2017
Received in revised form 14 December 2017
Accepted 3 January 2018

Keywords:
Kalman-Filter
AUS
Lorenz96
SLAM

ABSTRACT

In this paper we propose a C++-software package implementing the algorithm EKF-AUS-NL (Extended Kalman Filter with Assimilation in the Unstable Space with NonLinear evolution) designed to perform data assimilation in the unstable space when the Jacobian of the differential equation cannot be calculated. We also propose a simple approach to take into account the presence of the model error in the framework of the EKF-AUS-NL. The software performs the data assimilation using the EKF-AUS-NL algorithm with a dynamical systems defined as a generic time evolution routine separately implemented. We present two illustrative examples based on the Lorenz96 and SLAM systems.

© 2018 Published by Elsevier B.V.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00068
Legal Code License	Mozilla Public License, 2.0 (MPL-2.0) https://mozilla.org/MPL/2.0/
Code versioning system used	none
Software code languages, tools, and services used	C++
Compilation requirements, operating environments	gcc 4.2.1, Eigen package http://eigen.tuxfamily.org [16], MacOSX, Linux
Link to developer manual	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00068
Support email for questions	luigi.palatella@yahoo.it

1. Motivation and significance

In some recent works [1–3], a particular formulation (called EKF-AUS-NL) of the Extended Kalman Filter with Assimilation confined in the Unstable Space (AUS) was proposed and tested. This approach also leads to a quite natural nonlinear extension of the algorithms thanks to the computation of the self- and cross-nonlinear interaction between the leading Lyapunov vectors [3]. This previous work deals with systems provided with the tangent linear models computed thanks to the analytical expression of the Jacobian $\partial F_i(\mathbf{x})/\partial x_j$ of the deterministic differential equation $\dot{x}_i = F_i(\mathbf{x})$. In this paper we propose a variation of the algorithm EKF-AUS-NL designed to perform the data assimilation process when $\partial F_i(\mathbf{x})/\partial x_j$ cannot be defined or calculated. Moreover we propose a simple approach to be followed in order to take into account the presence of parametric model error in the framework of the EKF-AUS-NL routines. We present a C++ software package that performs the data assimilation using the EKF-AUS-NL algorithm with the dynamical systems defined as a generic time evolution routine. The package can perform both perfect model scenario tests (handling the parameters affected by the model error as additional state variables) and actual data assimilation tests using a routine designed to get the actual measurements. The time evolution is implemented separately from the EKF-AUS-NL routine, so it can be written in whatever language, while the input–output processes of the dynamical system is performed through a state-file where the EKF-AUS-NL routines read/write the state variables of the system.

As illustrative examples, we show two perfect model tests applied to two models: the Lorenz96 model [4] and the Simultaneous Localization and Map Model (SLAM) [5–7].

* Corresponding author.
E-mail addresses: luigi.palatella@yahoo.it, luigi.palatella@liceodegiorgi.gov.it (L. Palatella).

https://doi.org/10.1016/j.softx.2018.01.001
2352-7110/© 2018 Published by Elsevier B.V.

The software package is implemented in the C++ object oriented language. To maintain the separation between the software architecture and the dynamical system code, the package presented includes two routines (PrepareForEvolution and PrepareForAnalysis) that, starting from the analysis and forecast error perturbations $\mathbf{X}_{a,f}$, read/write the initial condition file used by the time evolution routine (that is run using the *system* C++ command). The analysis/forecast error perturbations $\mathbf{X}_{a,f}$ are a suitable number of column vectors representing the best estimate of the error just after and before the assimilation procedure.

This software is, in our opinion, very valuable because it permits to perform data assimilation using the EKF-AUS-NL technique for very complex models simply writing down an interface routine that run the dynamical system ensemble members. Moreover, the possibility to manage the parametric model error allows one to use our algorithm even in those cases where some model parameters are affected by uncertainty. This happens, e.g., for the SLAM algorithm presented in the following.

Here we briefly review the EKF-AUS-NL algorithm in order to introduce the correct notation and to explain how we handle the model error.

1.1. Theoretical background

The EKF-AUS filter is a particular square root implementation [8] of the Extended Kalman Filter (EKF) formally introduced in [9]. The EKF-AUS algorithm is obtained by confining the assimilation in a manifold of dimension m. When m is equal to the number N of degrees of freedom of the system, the algorithm solves the standard EKF equations. When $m = N^+ + N^0$ the reduced form, with Assimilation in the Unstable Subspace (EKF-AUS) is obtained, where N^+ and N^0 are the number of positive and neutral Lyapunov exponents, respectively. Notice that for general dynamical systems the relation $N^+ + N^0 \ll N$ holds.

In this paper and in the code we use the unifying notation of [10]: the total number of degrees of freedom of the system is N, the number of measure is p. The analysis error covariance matrix is expressed as the matrix product $\mathbf{P}^a = \mathbf{X}_a\mathbf{X}_a^T$, with $\mathbf{X}_a$ one of the "square root" of $\mathbf{P}^a$. We then have that during the forecast step the linearized time evolution acts on the columns of $\mathbf{P}^a$ (the *perturbations*) as

$$\mathbf{P}^f = \mathbf{M}\mathbf{X}_a(\mathbf{M}\mathbf{X}_a)^T = \mathbf{X}_f\mathbf{X}_f^T. \tag{1}$$

In the algorithm here presented the time evolution is performed with the whole dynamical system generating an ensemble of trajectories $\mathbf{x}_j^a$, each given by

$$\mathbf{x}_j^a(t_k) = \mathbf{x}^a(t_k) + \delta\mathbf{x}_j^a \tag{2}$$

where $\delta\mathbf{x}_j^a$ is the jth column of the matrix $\mathbf{X}_a$ and $\mathbf{x}^a(t_k)$ is the analysis at the kth analysis time t_k, i.e. the best estimate of the state of the system. After the time evolution, we obtain the best estimate of the state $\mathbf{x}^f(t_{k+1})$ (the *forecast*) and the $\mathbf{X}_f$ matrix build up collecting together the column vectors $\delta\mathbf{x}_i^f$ given by

$$\delta\mathbf{x}_i^f = \mathbf{x}_i^f(t_{k+1}) - \mathbf{x}^f(t_{k+1}), \tag{3}$$

where $\mathbf{x}_i^f(t_{k+1})$ is the time evolution of the states $\mathbf{x}_j^a(t_k)$ defined in Eq. (1). After that we orthonormalize the perturbations $\delta\mathbf{x}_i^f$, i.e. the columns of $\mathbf{X}^f$, and then we follow the standard EKF-AUS approach as described in [9]. As final result of the routine *Assimilate* we obtain the analysis state $\mathbf{x}^a(t_{k+1})$ and the analysis perturbations given by the matrix $\mathbf{X}_a = [\delta\mathbf{x}_1^a, \delta\mathbf{x}_2^a, \ldots, \delta\mathbf{x}_m^a]$. The vectors $\delta\mathbf{x}_i^a$, columns of $\mathbf{X}_a$, are the new (orthogonal) perturbations.

These results hold when observations are sufficiently dense, accurate and frequent [11] that error dynamics is linear, a necessary condition for the EKF to work properly without being subject to divergence episodes. The convergence hypothesis on which EKF-AUS relies has been recently proved analytically for linear systems by [12]. To handle the situations where the linear approximation does not hold exactly, we use the nonlinear generalization EKF-AUS-NL described in [3].

1.1.1. The nonlinear extension of EKF-AUS: the EKF-AUS-NL algorithm

The authors of [3] obtain in their Appendix A that if one takes into account nonlinearity up to the second order, the perturbations, namely the columns of $\mathbf{X}$, evolve from $\mathbf{X}^a$ at $t = t_k$ to $\mathbf{X}^f$ at $t_{k+1} = t_k + \tau$ (where τ is the analysis interval) according to the differential equation (Einstein's convention on repeated indexes, $F_{i;j} \equiv \partial_j F_i$, $F_{i;jk} \equiv \partial_j\partial_k F_i$)

$$\begin{aligned} \frac{d}{dt}\mathbf{X}_{is} &= F_{i;j}\mathbf{X}_{js}, \quad s \le m \\ \frac{d}{dt}\mathbf{X}_{is(q,r)} &= F_{i,j}\mathbf{X}_{js(q,r)} + \frac{1}{2}\bar{\alpha}F_{i;jk}\mathbf{X}_{jq}\mathbf{X}_{kr}, \end{aligned} \tag{4}$$

with $q \le r$; $q, r \in [1, m_l]$; $s(q,r) = m + \sum_{r=1}^{m_l}\sum_{q=1}^{q\le r} 1$, where $m_l \le m$ is the number of linear vectors that are involved in the nonlinear interactions and $\bar{\alpha} = \sqrt{3}$. One may of course consider all the possible nonlinear interactions of the m perturbation vectors. This means that the algorithm should involve $m(m+1)/2$ more vectors, i.e. the number of all possible pairs obtained with the m linear vectors. In several instances this approach cannot be followed because the number of vectors to be considered becomes too large. For this reason we limit the nonlinear interactions to the leading (most unstable) m_l vectors. The condition on the indexes s, m_l, q, r is nothing but a convention to keep the vectors evolving according to the nonlinear terms separated from the first $m = N^+ + N^0$ vectors already involved in the EKF-AUS algorithm. Notice that the total number of vectors involved in the analysis process becomes now $m + m_l(m_l+1)/2$.

The background covered in the preceding paragraphs are based on [3]. However, we need to understand how to handle nonlinearity when, as it often happens, the differential equation $\dot{\mathbf{x}} = F(\mathbf{x})$, that drives the dynamical system is known but there are difficulties to define or calculate the first and second derivative of $F(\mathbf{x})$. As a first step we have to find a typical scale η (in the phase-space, not in the physical space of the model) at which the evolution of the difference between two trajectories is essentially linear. There are different techniques to estimate η and we refer the reader to the literature like for example [13,14]. Here is a slight difference between the treatment suggested in the Appendix B of [3]. To better reproduce the linearized dynamics when the linear tangent model is not available, we realize that the approach briefly suggested in Appendix B is not totally correct. We now show the formally correct approach. As in the standard case, we have a trajectory that defines the analysis $\mathbf{x}^a(t_k)$. The nonlinear operator driving the trajectory from the analysis time t_k to the forecast time $t_{k+1} = t_k + \tau$ is $\mathcal{M}(\mathbf{x}^a)$. We thus have $\mathbf{x}^f(t_k + \tau) = \mathcal{M}(\mathbf{x}^a(t_k))$. Given at time t_k the $m + m_l(m_l+1)/2$ perturbations $\mathbf{X}^a$, let us define the jth column of $\mathbf{X}^a$ as $\delta\mathbf{x}_j^a$. We obtain the first $m + m_l(m_l+1)/2$ trajectories to be evolved in this way

$$\mathbf{x}_j^a(t_k) = \mathbf{x}^a(t_k) + \eta\,\delta\mathbf{x}_j^a, \quad j \in [1, m + m_l(m_l+1)/2] \tag{5}$$

then the remaining $m_l(m_l+1)/2$ trajectories are obtained as

$$\mathbf{x}_s^a(t_k) = \mathbf{x}^a(t_k) + \frac{1}{2}[\delta\mathbf{x}_r^a + \delta\mathbf{x}_q^a], \tag{6}$$

where $r, q \in [1, m_l]$, $q \le r$; $s(q,r) = m + m_l(m_l+1)/2 + \sum_{r=1}^{m_l}\sum_{q=1}^{q\le r} 1$. At this point all the trajectories are evolved up to time $t_{k+1} = t_k + \tau$ using the full nonlinear system

$$\mathbf{x}_j^f(t_{k+1}) = \mathcal{M}(\mathbf{x}_j^a(t_k)), \quad j \in [1, m + m_l(m_l+1)]. \tag{7}$$

Download English Version:

https://daneshyari.com/en/article/6964817

Download Persian Version:

https://daneshyari.com/article/6964817

Daneshyari.com