Geosci. Model Dev., 14, 5957–5976, 2021
https://doi.org/10.5194/gmd-14-5957-2021
© Author(s) 2021. This work is distributed under
the Creative Commons Attribution 4.0 License.

Geoscientific Model Development
Open Access
EGU

SymPKF (v1.0): a symbolic and computational toolbox for the design of parametric Kalman filter dynamics

Olivier Pannekoucke[1,2,3] **and Philippe Arbogast**[4]

[1]INPT-ENM, Toulouse, France
[2]CNRM, Université de Toulouse, Météo-France, CNRS, Toulouse, France
[3]CERFACS, Toulouse, France
[4]Météo-France, Toulouse, France

Correspondence: Olivier Pannekoucke (olivier.pannekoucke@meteo.fr)

Received: 22 March 2021 – Discussion started: 23 April 2021
Revised: 29 July 2021 – Accepted: 16 August 2021 – Published: 4 October 2021

Abstract. Recent research in data assimilation has led to the introduction of the parametric Kalman filter (PKF): an implementation of the Kalman filter, whereby the covariance matrices are approximated by a parameterized covariance model. In the PKF, the dynamics of the covariance during the forecast step rely on the prediction of the covariance parameters. Hence, the design of the parameter dynamics is crucial, while it can be tedious to do this by hand. This contribution introduces a Python package, SymPKF, able to compute PKF dynamics for univariate statistics and when the covariance model is parameterized from the variance and the local anisotropy of the correlations. The ability of SymPKF to produce the PKF dynamics is shown on a nonlinear diffusive advection (the Burgers equation) over a 1D domain and the linear advection over a 2D domain. The computation of the PKF dynamics is performed at a symbolic level, but an automatic code generator is also introduced to perform numerical simulations. A final multivariate example illustrates the potential of SymPKF to go beyond the univariate case.

1 Introduction

The Kalman filter (KF) (Kalman, 1960) is one of the backbones of data assimilation. This filter represents the dynamics of a Gaussian distribution all along the analysis and forecast cycles and takes the form of two equations representing the evolution of the mean and of the covariance of the Gaussian distribution.

While the equations of the KF are simple linear algebra, the large dimension of linear space encountered in the realm of data assimilation makes the KF impossible to handle, and this is particularly true for the forecast step. This limitation has motivated some approximation of covariance matrix to make the KF possible. For instance, in the ensemble method (Evensen, 2009), the covariance matrix is approximated by a sample estimation, whereby the time evolution of the covariance matrix is then deduced from the forecast of each individual sample. In the parametric Kalman filter (PKF) (Pannekoucke et al., 2016, 2018b, a), the covariance matrix is approximated by a parametric covariance model; the time evolution of the matrix is deduced from the time integration of the parameters' evolution equations.

One of the major limitations for the PKF is the design of the parameter evolution equations. Although not difficult from a mathematical point of view, this step requires the calculation of many terms that are difficult to calculate by hand and may involve errors in the calculation. To facilitate the derivation of the parametric dynamics and certify the correctness of the resulting system a symbolic derivation of the dynamics would be welcome.

The goal of the package SymPKF 1.0[1] is to facilitate the computation of the PKF dynamics for a particular class of covariance model, the VLATcov model, which is parameterized by the variance and the anisotropy. The symbolic computation of the PKF dynamics relies on a computer algebra system (CAS) able to handle abstract mathematical ex-

[1]https://github.com/opannekoucke/sympkf, last access: 22 September 2021.

pressions. A preliminary version has been implemented with Maxima[2] (Pannekoucke, 2021a). However, in order to create an integrated framework that would include the design of the parametric system, as well as its numerical evaluation, the symbolic Python package SymPy (Meurer et al., 2017) has been preferred for the present implementation. In particular, SymPKF comes with an automatic code generator to provide an end-to-end exploration of the PKF approach from the computation of the PKF dynamics to their numerical integration.

The paper is organized as follows. The next section provides the background on data assimilation and introduces the PKF. Section 3 focuses on the PKF for univariate VLATcov models in the perspective of symbolic computation by a CAS. Then, the package SymPKF is introduced in Sect. 4 from its use on the nonlinear diffusive advection (the Burgers equation) over a 1D domain. A numerical example illustrates the use of the automatic code generator provided in SymPKF. Then, the example of the linear advection over a 2D domain shows the ability of SymPKF to handle 2D and 3D domains. The section ends with a simple illustration of a multivariate situation, which also shows that SymPKF applies to a system of prognostic equations. The conclusion is given in Sect. 5.

2 Description of the PKF

2.1 Context of the numerical prediction

Dynamics encountered in geosciences are given as a system of partial differential equations (PDEs):

$$\partial_t \mathcal{X} = \mathcal{M}(t, \partial\mathcal{X}), \tag{1}$$

where $\mathcal{X}(t,\mathbf{x})$ is the state of the system and denotes either a scalar field or multivariate fields in a coordinate system $\mathbf{x} = (x^i)_{i\in[1,d]}$, where d is the dimension the geographical space, $\partial\mathcal{X}$ represents the partial derivatives with respect to the coordinate system at any order, with the convention that order zero denotes the field $\mathcal{X}$ itself, and $\mathcal{M}$ denotes the trend of the dynamics. A spatial discretization (e.g., by using finite differences, finite elements, finite volumes, spectral decomposition) transforms Eq. (1) into

$$\partial_t \mathcal{X} = \mathcal{M}(t, \mathcal{X}), \tag{2}$$

where, this time, $\mathcal{X}(t)$ is a vector, and $\mathcal{M}$ denotes the discretization of the trend in Eq. (1). Thereafter, $\mathcal{X}$ can be seen either as a collection of continuous fields with dynamics given by Eq. (1) or a discrete vector of dynamics as in Eq. (2).

Because of the sparsity and the error of the observations, the forecast $\mathcal{X}^{\mathrm{f}}$ is only an estimation of the true state $\mathcal{X}^{\mathrm{t}}$, which is known to within a forecast error defined by $e^{\mathrm{f}} = \mathcal{X}^{\mathrm{f}} - \mathcal{X}^{\mathrm{t}}$. This error is often modeled as an unbiased random variable, $\mathbb{E}\left[e^{\mathrm{f}}\right] = 0$. In the discrete formulation of the dynamics in Eq. (2), the forecast error covariance matrix is given by $\mathbf{P}^{\mathrm{f}} = \mathbb{E}\left[e^{\mathrm{f}}(e^{\mathrm{f}})^{\mathrm{T}}\right]$, where the superscript $^{\mathrm{T}}$ denotes the transpose operator. Since this contribution is focused on the forecast step, hereafter the upper script $^{\mathrm{f}}$ is removed for the sake of simplicity.

We now detail how the error covariance matrix evolves during the forecast by considering the formalism of the second-order nonlinear Kalman filter.

2.2 Second-order nonlinear Kalman filter

A second-order nonlinear Kalman filter (KF2) is a filter that extends the Kalman filter (KF) to nonlinear situations in which the error covariance matrix evolves tangent-linearly along the trajectory of the mean state and the dynamics of this mean are governed by the fluctuation–mean interacting dynamics (Jazwinski, 1970; Cohn, 1993). Hence, we first state the dynamics of the mean under the fluctuation–mean interaction, then the dynamics of the error covariance. Note that the choice of the following presentation is motivated by the perspective of using a computer algebra system to perform the computation.

2.2.1 Computation of the fluctuation–mean interaction dynamics

Because of the uncertainty in the initial condition, the state $\mathcal{X}$ is modelized as a Markov process $\mathcal{X}(t,\mathbf{x},\omega)$, where ω stands for the stochasticity, while $\mathcal{X}$ evolves by Eq. (1). Hence, ω lies within a certain probability space $(\Omega,\mathcal{F},P)$, where $\mathcal{F}$ is σ algebra on Ω (a family of subsets of Ω, which contains Ω and which is stable for the complement and the countable union) and P is a probability measure (see, e.g., Øksendal, 2003, chap.2). $\mathcal{X}(t,\mathbf{x},\cdot) : (\Omega,\mathcal{F}) \to (\mathbb{R}^n, \mathcal{B}_{\mathbb{R}^n})$ is an $\mathcal{F}$-measurable function wherein $\mathcal{B}_{\mathbb{R}^n}$ denotes the Borel σ algebra on $\mathbb{R}^n$, where the integer n is either the dimension of the multivariate field $\mathcal{X}(t,x)$ or the dimension of its discretized version $\mathcal{X}(t)$. The connection between the Markov process and the parameter dynamics is obtained using the Reynolds averaging technique (Lesieur, 2007, chap. 4).

To perform the Reynolds averaging of Eq. (1), the first step is to replace the random field by its Reynolds decomposition $\mathcal{X}(t,\mathbf{x},\omega) = \mathbb{E}[\mathcal{X}](t,\mathbf{x}) + \eta e(t,\mathbf{x},\omega)$. In this modeling of the random state, $\mathbb{E}[\mathcal{X}]$ is the ensemble average or the mean state; e is an error or a fluctuation to the mean, and it is an unbiased random field, $\mathbb{E}[e] = 0$. Then, Eq. (1) reads as

$$\partial_t \mathbb{E}[\mathcal{X}] + \eta\partial_t e = \mathcal{M}(t, \partial\mathbb{E}[\mathcal{X}] + \eta\partial e), \tag{3}$$

where η is a control of magnitude introduced to facilitate Taylor's expansion when using a computer algebra system. At the second order, the Taylor's expansion in η of Eq. (3)

[2] https://maxima.sourceforge.io/, last access: 22 September 2021.

reads

$$\partial_t \mathbb{E}[\mathcal{X}] + \eta \partial_t e = \mathcal{M}(t, \partial\mathbb{E}[\mathcal{X}]) + \eta \mathcal{M}'(t, \partial\mathbb{E}[\mathcal{X}])(\partial e) + \eta^2 \mathcal{M}''(t, \partial\mathbb{E}[\mathcal{X}])(\partial e \otimes \partial e), \quad (4a)$$

where $\mathcal{M}'$ and $\mathcal{M}''$ are two linear operators; the former (the latter) refers to the tangent-linear model (the Hessian), and both are computed with respect to the mean state $\mathbb{E}[\mathcal{X}]$. The first-order expansion is deduced from Eq. (4a) by setting $\eta^2 = 0$, which then reads as

$$\partial_t \mathbb{E}[\mathcal{X}] + \eta \partial_t e = \mathcal{M}(t, \partial\mathbb{E}[\mathcal{X}]) + \eta \mathcal{M}'(t, \partial\mathbb{E}[\mathcal{X}])(\partial e). \quad (4b)$$

By setting η to 1, the dynamics of the ensemble average are calculated at the second order from the expectation of Eq. (4a) that reads as

$$\partial_t \mathbb{E}[\mathcal{X}] = \mathcal{M}(t, \partial\mathbb{E}[\mathcal{X}]) + \mathcal{M}''(t, \partial\mathbb{E}[\mathcal{X}])(\mathbb{E}[\partial e \otimes \partial e]), \quad (5)$$

where $\partial e \otimes \partial e$ denotes the tensor product of the partial derivatives with respect to the spatial coordinates, i.e., terms such as $\partial^k e \partial^m e$ for any positive integers (k, m). Here, we have used the assumption that the partial derivative commutes with the expectation, $\mathbb{E}[\partial e] = \partial\mathbb{E}[e]$, and that $\mathbb{E}[e] = 0$. Because the expectation is a projector, $\mathbb{E}[\mathbb{E}[\cdot]] = \mathbb{E}[\cdot]$, the expectation of $\mathcal{M}(t, \partial\mathbb{E}[\mathcal{X}])$ is itself. The second term of the right-hand side makes the retro-action of the error appear in the ensemble-averaged dynamics. Hence, Eq. (5) gives the dynamics of the error–mean interaction (or fluctuation–mean interaction).

Note that the tangent-linear dynamics along the ensemble-averaged dynamics in Eq. (5) are obtained as the difference between the first-order Taylor's expansion in Eq. (4b) and its expectation, and they read as

$$\partial_t e = \mathcal{M}'(t, \partial\mathbb{E}[\mathcal{X}])(\partial e). \quad (6)$$

Now it is possible to detail the dynamics of the error covariance from the dynamics of the error, which tangent-linearly evolve along the mean state $\mathbb{E}[\mathcal{X}]$.

2.2.2 Computation of the error covariance dynamics

In the discretized form, the dynamics of the error in Eq. (6) read as the ordinary differential equation (ODE):

$$\frac{\mathrm{d}e}{\mathrm{d}t} = \mathbf{M}e, \quad (7)$$

where $\mathbf{M}$ stands for the tangent-linear (TL) model $\mathcal{M}'(t, \partial\mathbb{E}[\mathcal{X}])$ evaluated at the mean state $\mathbb{E}[\mathcal{X}]$. So the dynamics of the error covariance matrix, $\mathbf{P} = \mathbb{E}\left[ee^{\mathrm{T}}\right]$, are given by the ODE:

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \mathbf{M}\mathbf{P} + \mathbf{P}\mathbf{M}^{\mathrm{T}} \quad (8a)$$

($\mathbf{M}^{\mathrm{T}}$ is the adjoint of $\mathbf{M}$) or its integrated version

$$\mathbf{P}(t) = \mathbf{M}_{t\leftarrow 0}\mathbf{P}_0(\mathbf{M}_{t\leftarrow 0})^{\mathrm{T}}, \quad (8b)$$

where $\mathbf{M}_{t\leftarrow 0}$ is the propagator associated with the time integration of Eq. (7), initiated from the covariance $\mathbf{P}_0$.

2.2.3 Setting of the KF2

Gathering the dynamics of the ensemble mean given by the fluctuation–mean interaction in Eq. (5) and the covariance dynamics in Eq. (8) leads to the second-order closure approximation of the extended KF, which is the forecast step equations of the KF2.

Similarly to the KF, the principal limitation of the KF2 is the numerical cost associated with the covariance dynamics in Eq. (8): living in a discrete world, the numerical cost of Eq. (8) dramatically increases with the size of the problem. As an example, for the dynamics of a simple scalar field discretized with n grid points, the dimension of its vector representation is n, while the size of the error covariance matrix scales as n^2, leading to a numerical cost of Eq. (8) between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$.

We now introduce the parametric approximation of covariance matrices, which aims to reduce the cost of the covariance dynamics in Eq. (8).

2.3 Formulation of the PKF prediction

The parametric formulation of covariance evolution stands as follows. If $\mathbf{P}(\mathcal{P})$ denotes a covariance model featured by a set of parameters $\mathcal{P} = (p_i)_{i\in I}$, then there is a set $\mathcal{P}_t^{\mathrm{f}}$ featuring the forecast error covariance matrix so that $\mathbf{P}(\mathcal{P}_t^{\mathrm{f}})$ approximates the forecast error covariance $\mathbf{P}_t^{\mathrm{f}}$, i.e., $\mathbf{P}(\mathcal{P}_t^{\mathrm{f}}) \approx \mathbf{P}_t^{\mathrm{f}}$. Note that a parameter p_i can be a scalar or a field, e.g., a variance field.

Hence, starting from the initial condition $\mathcal{P}^{\mathrm{f}} = \mathcal{P}_0^{\mathrm{f}}$ so that $\mathbf{P}(\mathcal{P}_0^{\mathrm{f}}) \approx \mathbf{P}_0^{\mathrm{f}}$, if the dynamics of the parameters $\mathcal{P}_t^{\mathrm{f}}$ are known, then it is possible to approximately determine $\mathbf{P}_t^{\mathrm{f}}$ by $\mathbf{P}(\mathcal{P}_t^{\mathrm{f}})$ without solving Eq. (8) explicitly. This approach constitutes the so-called parametric Kalman filter (PKF) approximation introduced by Pannekoucke et al. (2016, 2018a) (P16, P18).

In practice, the parametric covariance models considered in the PKF are such that the number of parameters in $\mathcal{P}$ is much lower than the number of coefficients required to represent the full covariance $\mathbf{P}(\mathcal{P})$. For instance, for the dynamics of a scalar field discretized with n grid points, as introduced in Sect. 2.2.3, the total number of parameters in $\mathcal{P}$ should be of same order as n, e.g., $2n$ or $3n$, so that the cost to predict the evolution of the parameters would represent 2 or 3 times the cost to predict the evolution of the scalar field. Said differently, the cost to predict the parameters should scale in $\mathcal{O}(n)$, which is much lower than the computation of Eq. (8) in $\mathcal{O}(n^2)$ - $\mathcal{O}(n^3)$.

The cost of the PKF can be compared with other low-rank methods such as the reduced-rank Kalman filter (Fisher, 1998) or the ensemble Kalman filter (Evensen, 2009), for which an ensemble size of 100 members is often encountered depending on the dimension of the unstable subspace and to limit the amount of sampling noise. Hence, when each forecast is made at full resolution the cost of these approaches is 100 times the cost of a prediction, which is larger than the cost we expect for the PKF (see P16, P18). Note that low-

rank and ensemble methods often consider the computation of the dynamics at a lower resolution, which leads to a lower cost than the 100 forecast at full resolution. The PKF is computed at the full resolution and is free from sampling noise.

But the frugality of the covariance model is not the only criterion. For instance, the first variational data assimilation systems considered a covariance model based on the spectral diagonal assumption in spectral space (Courtier et al., 1998; Berre, 2000). This covariance model reads as $\mathbf{P}^{\mathrm{s}}(\mathcal{P}^{\mathrm{s}}) = \mathbf{\Sigma}\mathbf{S}^{-1}\mathbf{D}\left(\mathbf{S}^{-1}\right)^{*}\mathbf{\Sigma}^{*}$, where $\mathbf{S}$ denotes the spectral transform with * the conjugate transpose operator, and $\mathbf{\Sigma}$ is the diagonal matrix of standard deviation, i.e., the square root of the variance field V. In this model, the set of parameters $\mathcal{P}^{\mathrm{s}}$ is given by the grid points and the spectral variances, $\mathcal{P}^{\mathrm{s}} = (V, \mathrm{diag}(\mathbf{D}))$. If the shape of $\mathbf{P}^{\mathrm{s}}(\mathcal{P}^{\mathrm{s}})$ is n^2, the number of parameters $\mathcal{P}^{\mathrm{s}}$ for this covariance model is $2n$ (n variances in grid points, stored as the standard deviation in the diagonal of $\mathbf{\Sigma}$; n variances in spectral, stored in the diagonal of $\mathbf{D}$), which is quite economical. However, the resulting correlation functions are homogeneous (there is the same correlation function in each point), which is enough to represent climatologically stationary background error statistics but not the flow-dependent statistics as existing in the KF. While it is possible to write the equations for the dynamics of the spectral variances (e.g., for the linear waves), the limitation that the spectral diagonal approach can only model homogeneous correlations motivated the introduction of other covariance models. For example, the covariance model based on the diagonal assumption in wavelet space (Fisher, 2004; Pannekoucke et al., 2007) can model heterogeneous correlations at a low memory cost. However, the dynamics of the wavelet variances are much more difficult to develop because of the redundancy of the wavelet transform on the sphere.

Hence, a covariance model adapted for the PKF should be able to represent realistic correlations and be such that the dynamics of the parameters can be computed, e.g., a covariance model defined by parameters in grid points. To do so, we now focus on the PKF applied to a particular family of covariance models, whose parameters are defined in grid points by the variance and the anisotropy fields: $\mathcal{P} = (V, \mathbf{g})$, where $\mathbf{g}$ will denotes the local anisotropy tensor of the local correlation function.

3 PKF for VLATcov models

This part introduces a particular family of covariance models parameterized by the fields of variances and of the local anisotropy tensor: the VLATcov models (Pannekoucke, 2021b). What makes this covariance model interesting is that its parameters are related to the error field, and thus it is possible to determine the dynamics of the parameters. To introduce VLATcov models, we first present the diagnosis of the variance and of the local anisotropy tensor; then we present two examples of VLATcov models, and we end the section with a description of the dynamics of the parameters.

3.1 Definition of the fields of variance and of local anisotropy tensor

From now, we will focus on the forecast error statistics, so the upper script $^{\mathrm{f}}$ is removed for the sake of simplicity. Moreover, for a function f, when there is no confusion, the value of f at a point $\mathbf{x}$ is written either as $f(\mathbf{x})$ or as $f_{\mathbf{x}}$.

The forecast error being unbiased, $\mathbb{E}[e] = 0$, its variance at a point $\mathbf{x}$ is defined as

$$V(\mathbf{x}) = \mathbb{E}[e(\mathbf{x})^2]. \tag{9}$$

When the error is a random differentiable field, the anisotropy of the two-point correlation function $\rho(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{V_{\mathbf{x}} V_{\mathbf{y}}}}\mathbb{E}[e(\mathbf{x})e(\mathbf{y})]$ is featured from the second-order expansion,

$$\rho(\mathbf{x}, \mathbf{x} + \delta\mathbf{x}) \approx 1 - \frac{1}{2}||\delta\mathbf{x}||^2_{\mathbf{g}_{\mathbf{x}}}, \tag{10}$$

by the local metric tensor $\mathbf{g}(\mathbf{x})$ and defined as

$$\mathbf{g}(\mathbf{x}) = -\nabla\nabla^{\mathrm{T}}\rho_{\mathbf{x}}, \tag{11}$$

where $\rho_{\mathbf{x}}(\mathbf{y}) = \rho(\mathbf{x}, \mathbf{y})$, e.g.,

$$g_{ij}(\mathbf{x}) = -\left(\partial^2_{y^i y^j}\rho_{\mathbf{x}}(\mathbf{y})\right)_{\mathbf{y}=\mathbf{x}}.$$

The metric tensor is a symmetric positive definite matrix, and it is a 2×2 (3×3) matrix in a 2D (3D) domain.

Note that it is useful to introduce the local aspect tensor (Purser et al., 2003), defined as the inverse of the metric tensor:

$$\mathbf{s}(\mathbf{x}) = \mathbf{g}(\mathbf{x})^{-1}, \tag{12}$$

where the superscript $^{-1}$ denotes the matrix inverse. The aspect tensor at the point x is geometrically interpreted as an ellipse whose shape coincides with that of the local correlation function.

What makes the metric tensor attractive, either at a theoretical or at a practical level, is that it is closely related to the normalized error $\varepsilon = \frac{e}{\sqrt{V}}$ by

$$\mathbf{g}_{ij}(\mathbf{x}) = \mathbb{E}\left[(\partial_{x^i}\varepsilon)(\partial_{x^j}\varepsilon)\right] \tag{13}$$

(see, e.g., Pannekoucke, 2021b, for details).

Hence, using the notation introduced in Sect. 2.3, a VLATcov model is a covariance model, $\mathbf{P}(\mathcal{P})$, characterized by the set of two parameter fields, $\mathcal{P} = (p_1, p_2)$, given by the variance field and by the anisotropy field – the latter being defined either by the metric tensor field $\mathbf{g}$ or by the aspect tensor field $\mathbf{s}$ – i.e., $\mathcal{P} = (V, \mathbf{g})$ or $\mathcal{P} = (V, \mathbf{s})$. Said differently, any VLATcov model reads as $\mathbf{P}(V, \mathbf{g})$ or $\mathbf{P}(V, \mathbf{s})$.

To put some flesh on the bones, two examples of VLATcov models are now presented.

3.2 Examples of VLATcov models

We first consider the covariance model based on the heterogeneous diffusion operator of Weaver and Courtier (2001), which is used in variational data assimilation to model heterogeneous correlation functions, e.g., for the ocean or for air quality. This model has the property that, under the local homogenous assumption (when the spatial derivatives are negligible), the local aspect tensors of the correlation functions are twice the local diffusion tensors (Pannekoucke and Massart, 2008; Mirouze and Weaver, 2010). Hence, by defining the local diffusion tensors as half the local aspect tensors, the covariance model based on the heterogeneous diffusion equation is a VLATcov model.

Another example of a heterogeneous covariance model is the heterogeneous Gaussian covariance model:

$$\mathbf{P}^{\mathrm{he.g}}(V,\nu)(\mathbf{x},\mathbf{y}) = \sqrt{V(\mathbf{x})V(\mathbf{y})}\frac{|\nu_{\mathbf{x}}|^{1/4}|\nu_{\mathbf{y}}|^{1/4}}{\left|\frac{1}{2}(\nu_{\mathbf{x}}+\nu_{\mathbf{y}})\right|^{1/2}} \times \exp\left(-||\mathbf{x}-\mathbf{y}||^2_{(\nu_{\mathbf{x}}+\nu_{\mathbf{y}})^{-1}}\right), \tag{14}$$

where ν is a field of symmetric positive definite matrices, and $|\nu|$ denotes the matrix determinant. $\mathbf{P}^{\mathrm{he.g}}(V,\nu)$ is a particular case of the class of covariance models deduced from Theorem 1 of Paciorek and Schervish (2004). Again, this covariance model has the property that, under local homogenous assumptions, the local aspect tensor is approximately given by ν, i.e., for any point $\mathbf{x}$,

$$\mathbf{s}_x \approx \nu_x. \tag{15}$$

Hence, as for the covariance model based on the diffusion equation, by defining the field ν as the aspect tensor field, the heterogeneous Gaussian covariance model is a VLATcov model (Pannekoucke, 2021b).

At this stage, all the pieces of the puzzle are put together to build the PKF dynamics. We have covariance models parameterized from the variance and the local anisotropy, which are both related to the error field: knowing the dynamics of the error leads to the dynamics of the VLATcov parameters. This is now detailed.

3.3 PKF prediction step for VLATcov models

When the dynamics of the error e are well approximated from the tangent-linear evolution in Eq. (6), the connection between the covariance parameters and the error, represented in Eqs. (9) and (13), makes it possible to establish the prediction step of the PKF (Pannekoucke et al., 2018a), which reads as the dynamics of the ensemble average (at the second-order closure),

$$\partial_t \mathbb{E}[\mathcal{X}] = \mathcal{M}(t,\partial\mathbb{E}[\mathcal{X}]) + \mathcal{M}''(t,\partial\mathbb{E}[\mathcal{X}])(\mathbb{E}[\partial e\otimes\partial e]), \tag{16a}$$

coupled with the dynamics of the variance and the metric,

$$\partial_t V(t,\mathbf{x}) = 2\mathbb{E}[e\partial_t e], \tag{16b}$$

$$\partial_t \mathbf{g}_{ij}(t,\mathbf{x}) = \mathbb{E}[\partial_t\left((\partial_{x^i}\varepsilon)(\partial_{x^j}\varepsilon)\right)], \tag{16c}$$

where it remains to replace the dynamics of the error (and its normalized version $\varepsilon = e/\sqrt{V}$) from Eq. (6) and where the property that the expectation operator and the temporal derivative commutes, $\partial_t\mathbb{E}[\cdot] = \mathbb{E}[\partial_t\cdot]$, has been used to obtain Eq. (16b) and (16c).

Following the discussion in Sect. 2.3, the set of Eq. (16) is at the heart of the numerical sobriety of the parametric approach since the cost of the prediction of the parameter scales like $\mathcal{O}(n)$. In contrast to the matrix dynamics of the KF, the PKF approach is designed for the continuous world, leading to PDEs for the parameter dynamics in place of ODEs in Eq. (8) for the full matrix dynamics. Moreover, the dynamics of the parameters shed light on the nature of the processes governing the dynamics of covariances, and it does not require any adjoint of the dynamics (Pannekoucke et al., 2016, 2018a).

Note that Eq. (16) can be formulated in terms of aspect tensors thanks to the definition in Eq. (12): since $\mathbf{sg} = \mathbf{I}$, its time derivative $(\partial_t\mathbf{s})\mathbf{g} + \mathbf{s}(\partial_t\mathbf{g}) = 0$ leads to the dynamics $\partial_t\mathbf{s} = -\mathbf{g}^{-1}(\partial_t\mathbf{g})\mathbf{s}$, and then

$$\partial_t\mathbf{s} = -\mathbf{s}(\partial_t\mathbf{g})\mathbf{s}, \tag{17}$$

where it remains to replace occurrences of $\mathbf{g}$ by $\mathbf{s}^{-1}$ in the resulting dynamics of the mean, the variance, and the aspect tensor.

Hence, the PKF forecast step for a VLATcov model is given by either the system in Eq. (16) (in metric) or by its aspect tensor formulation thanks to Eq. (17). Whatever the formulation considered, it is possible to carry out the calculations using a formal calculation language. However, even for simple physical processes, the number of terms in formal expressions can become very large; e.g., it is common to have to manipulate expressions with more than 100 terms. Thus, any strategy that simplifies the assessment of PKF systems in advance can quickly become a significant advantage.

In the following section, we present the splitting method that allows the PKF dynamics to be expressed by bringing together the dynamics of each of the physical processes, calculated individually.

3.4 The splitting strategy

When there are several processes in the dynamics in Eq. (1), the calculation of the parametric dynamics can be tedious even when using a computer algebra system. To better use digital resources, a splitting strategy can be introduced (Pannekoucke et al., 2016, 2018a).

While the theoretical background is provided by the Lie–Trotter formula for Lie derivatives, the well-known idea of time splitting is easily taken from a first-order Taylor expansion of an Euler numerical scheme.

The computation of dynamics,

$$\partial_t \mathcal{X} = f_1(\mathcal{X}) + f_2(\mathcal{X}), \tag{18}$$

over a single time step δt can be done in two steps following the numerical scheme

$$\begin{cases} \mathcal{X}^\star = \mathcal{X}(t) + \delta t\, f_1(\mathcal{X}(t)), \\ \mathcal{X}(t+\delta t) = \mathcal{X}^\star + \delta t\, f_2(\mathcal{X}^\star), \end{cases} \tag{19}$$

where at order δt, this scheme is equivalent to $\mathcal{X}(t+\delta t) = \mathcal{X}(t) + \delta t\,(f_1(\mathcal{X}(t)) + f_2(\mathcal{X}(t)))$, which is the Euler step of Eq. (18). Because f_1 and f_2 can be viewed as vector fields, the fractional scheme, joining the starting point (at t) to the end point (at $t+\delta t$), remains to go through the parallelogram formed by the sum of the two vectors along its sides. Since there are two paths joining the extreme points, starting the computation by f_2 is equivalent to starting by f_1 (at order δt); this corresponds to the commutativity of the diagram formed by the parallelogram.

Appendix A shows that the dynamics given by Eq. (18) imply dynamics of the error, the variance, the metric, and the aspect written as a sum of trends. Hence, it is possible to apply a splitting for all these dynamics.

As a consequence of the calculation of the parametric dynamics, calculating the parametric dynamics of Eq. (18) is equivalent to separately calculating the parametric dynamics of $\partial_t \mathcal{X} = f_1(\mathcal{X})$ and $\partial_t \mathcal{X} = f_2(\mathcal{X})$, then bringing together the two parametric dynamics into a single one by summing the trends for the mean, the variance, the metric, or the aspect dynamics. This splitting applies when there are more than two processes and appears to be a general method to reduce the complexity of the calculation.

3.5 Discussion and intermediate conclusion

However, although the calculation of the system in Eq. (16) is straightforward, as it is similar to the calculation of Reynolds equations (Pannekoucke et al., 2018a), it is tedious because of the many terms involved, and there is a risk of introducing errors during the calculation by hand.

Then, once the dynamics of the parameters are established, it remains to design a numerical code to test whether the uncertainty is effectively well represented by the PKF dynamics. Again, the design of a numerical code is not necessarily difficult, but with numerous terms the risk of introducing an error is important.

To facilitate the design of the PKF dynamics and the numerical evaluation, the package SymPKF has been introduced to perform the VLATcov parameter dynamics and to generate a numerical code used for the investigations (Pannekoucke, 2021c). The next section introduces and details this tool.

4 Symbolic computation of the PKF for VLATcov

In order to introduce the symbolic computation of the PKF for the VLATcov model, we consider an example: the diffusive nonlinear advection in the Burgers equation, which reads

$$\partial_t u + u\partial_x u = \kappa \partial_x^2 u, \tag{20}$$

where u stands for the velocity field and corresponds to a function of the time t and the space of coordinate x and where κ is a diffusion coefficient (constant here). This example illustrates the workflow leading to the PKF dynamics. It consists of defining the system of equations in SymPy, then computing the dynamics with Eq. (16); we now detail these two steps.

4.1 Definition of the dynamics

The definition of the dynamics relies on the formalism of SymPy as shown in Fig. 1. The coordinate system is first defined as instances of the class `Symbols`. Note that the time is defined as `sympkf.t`, while the spatial coordinate is left to the choice of the user, here x. Then, the function u is defined as an instance of the class `Function` as a function of (t, x).

In this example, the dynamics consist of a single equation defined as an instance of the class `Eq`, but in the general situation in which the dynamics are given as a system of equations, the dynamics have to be represented as a Python list of equations.

A preprocessing of the dynamics is then performed to determine several important quantities to handle the dynamics: the prognostic fields (functions for which a time derivative is present), the diagnostic fields (functions for which there is no time derivative in the dynamics), the constant functions (functions that only depend on the spatial coordinates), and the constants (pure scalar terms that are not a function of any coordinate). This preprocessing is performed when transforming the dynamics as an instance of the class `PDESystem` and whose default string output delivers a summary of the dynamics: for the Burgers equation, there is only one prognostic function, $u(t, x)$, and one constant, κ.

The prognostic quantities being known, it is then possible to perform the computation of the PKF dynamics, as discussed now.

4.2 Computation of the VLATcov PKF dynamics

Thanks to the preprocessing, we are able to determine the VLATcov parameters needed to compute the PKF dynamics, which are the variance and the anisotropy tensor associated with the prognostic fields. For the Burgers equation, the VLATcov parameters are the variance V_u and the metric tensor $\mathbf{g}_u = (g_{u,xx})$ or its associated aspect tensor $\mathbf{s}_u = (s_{u,xx})$. Note that, in SymPKF, the VLATcov parameters are labeled by their corresponding prognostic fields to facilitate their

```
# Import of libraries
from sympy import symbols, Function, Derivative, Eq
from sympkf import PDESystem, SymbolicPKF, t

# Set the spatial coordinate system
x = symbols('x')
# Set the constants
kappa = symbols('kappa')
# Define the spatio-temporal scalar field
u = Function('u')(t,x)
```

```
# Definition of the Burgers dynamics
burgers_equation = Eq(Derivative(u,t),
    -u*Derivative(u,x)+kappa*Derivative(u,x,2))
burgers_equation
```

$$\frac{\partial}{\partial t}u(t,x) = \kappa \frac{\partial^2}{\partial x^2}u(t,x) - u(t,x)\frac{\partial}{\partial x}u(t,x)$$

```
# Processing of the PDE system
burgers = PDESystem( burgers_equation )
burgers
```

```
PDE System :
        prognostic functions : u(t, x)
        constant functions   :
        exogeneous functions :
        constants            : kappa
```

Figure 1. Sample of code and Jupyter notebook outputs for the definition of the Burgers dynamics using SymPKF.

identification. This labeling is achieved when the dynamics are transformed as an instance of the class `SymbolicPKF`. This class is at the core of the computation of the PKF dynamics from Eq. (16).

As discussed in Sect. 2.2.1, the PKF dynamics rely on the second-order fluctuation–mean interaction dynamics wherein each prognostic function is replaced by a stochastic counterpart. Hence, the constructor of `SymbolicPKF` converts each prognostic function as a function of an additional coordinate, $\omega \in \Omega$. For the Burgers equation, $u(t,x)$ becomes $u(t,x,\omega)$.

Since the computation of the second-order fluctuation–mean interaction dynamics relies on the expectation operator, an implementation of this expectation operator has been introduced in SymPKF: it is defined as the class `Expectation` built by inheritance from the class `sympy.Function` to leverage the computational facilities of SymPy. The implementation of the class `Expectation` is based on the linearity of the mathematical expectation operator with respect to deterministic quantities and its commutativity with partial derivatives and integrals with respect to coordinates different from ω, e.g., for the Burgers equation $\mathbb{E}[\partial_x u(t,x,\omega)] = \partial_x \mathbb{E}[u(t,x,\omega)]$. Note that $\mathbb{E}[u(t,x,\omega)]$ is a function of (t,x) only: the expectation operator converts a random variable into a deterministic variable.

Then, the symbolic computation of the second-order fluctuation–mean interaction dynamics in Eq. (16a) is performed, thanks to SymPy, by following the steps as described in Sect. 2.2.1. In particular, the computation also leads to the tangent-linear dynamics of the error in Eq. (6), from which it is possible to compute the dynamics of the variance in Eq. (16b) and of the metric tensor in Eq. (16c) (or its associated aspect tensor version). Applying these steps and the appropriate substitutions, this is achieved when calling the `in_metric` or `in_aspect` Python property of an instance of the class `SymbolicPKF`. This is shown for the Burgers equation in Fig. 2, where the background computation of the PKF dynamics leads to a list of the three coupled equations corresponding to the mean, the variance, and the aspect tensor, similar to the system in Eq. (22) first obtained by Pannekoucke et al. (2018a).

Hence, from SymPKF, for the Burgers equation, the VLATcov PKF dynamics given in the aspect tensor read as

$$\left\{\begin{array}{lcl} \partial_t u & = & \kappa\partial_x^2 u - u\partial_x u - \frac{\partial_x V_u}{2} \\ \partial_t V_u & = & -\frac{2\kappa V_u}{s_{u,xx}} + \kappa\partial_x^2 V_u - \frac{\kappa(\partial_x V_u)^2}{2V_u} \\ & & -u\partial_x V_u - 2V_u\partial_x u \\ \partial_t s_{u,xx} & = & 2\kappa s_{u,xx}^2 \mathbb{E}\left(\varepsilon_u \partial_x^4 \varepsilon_u\right) - 3\kappa\partial_x^2 s_{u,xx} \\ & & -2\kappa + \frac{6\kappa\left(\partial_x s_{u,xx}\right)^2}{s_{u,xx}} - \frac{2\kappa s_{u,xx}\partial_x^2 V_u}{V_u} \\ & & + \frac{\kappa\partial_x V_u \partial_x s_{u,xx}}{V_u} + \frac{2\kappa s_{u,xx}(\partial_x V_u)^2}{V_u^2} \\ & & -u\partial_x s_{u,xx} + 2s_{u,xx}\partial_x u \end{array}\right. , \quad (21)$$

where $s_{u,xx}$ is the single component of the aspect tensor $\mathbf{s}_u$ in 1D domains. Note that in the output of the PKF equations,

```
# Define the PKF system
pkf_burgers = SymbolicPKF(burgers)
```

```
# Compute the PKF system rndeered in metric tensor form (the computatation is only␣
 ↪performed at the first call)
for equation in pkf_burgers.in_metric: display(equation)
```

$$\frac{\partial}{\partial t}u(t,x) = \kappa\frac{\partial^2}{\partial x^2}u(t,x) - u(t,x)\frac{\partial}{\partial x}u(t,x) - \frac{\frac{\partial}{\partial x}\mathrm{V_u}(t,x)}{2}$$

$$\frac{\partial}{\partial t}\mathrm{V_u}(t,x) = -2\kappa\,\mathrm{V_u}(t,x)\,\mathrm{g_{u,xx}}(t,x) + \kappa\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x) - \frac{\kappa\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{2\,\mathrm{V_u}(t,x)} - u(t,x)\frac{\partial}{\partial x}\mathrm{V_u}(t,x) - 2\,\mathrm{V_u}(t,x)\frac{\partial}{\partial x}u(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{g_{u,xx}}(t,x) = 2\kappa\,{\mathrm{g_{u,xx}}}^2(t,x) - 2\kappa\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right) - 3\kappa\frac{\partial^2}{\partial x^2}\mathrm{g_{u,xx}}(t,x) + \frac{2\kappa\,\mathrm{g_{u,xx}}(t,x)\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x)}{\mathrm{V_u}(t,x)} + \frac{\kappa\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\frac{\partial}{\partial x}\mathrm{g_{u,xx}}(t,x)}{\mathrm{V_u}(t,x)} - \frac{2\kappa\,\mathrm{g_{u,xx}}(t,x)\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{{\mathrm{V_u}}^2(t,x)} - u(t,x)\frac{\partial}{\partial x}\mathrm{g_{u,xx}}(t,x) - 2\,\mathrm{g_{u,xx}}(t,x)\frac{\partial}{\partial x}u(t,x)$$

```
# Compute the PKF system rendered in aspect tensor form (the computatation is only␣
 ↪performed at the first call)
for equation in pkf_burgers.in_aspect: display(equation)
```

$$\frac{\partial}{\partial t}u(t,x) = \kappa\frac{\partial^2}{\partial x^2}u(t,x) - u(t,x)\frac{\partial}{\partial x}u(t,x) - \frac{\frac{\partial}{\partial x}\mathrm{V_u}(t,x)}{2}$$

$$\frac{\partial}{\partial t}\mathrm{V_u}(t,x) = -\frac{2\kappa\,\mathrm{V_u}(t,x)}{\mathrm{s_{u,xx}}(t,x)} + \kappa\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x) - \frac{\kappa\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{2\,\mathrm{V_u}(t,x)} - u(t,x)\frac{\partial}{\partial x}\mathrm{V_u}(t,x) - 2\,\mathrm{V_u}(t,x)\frac{\partial}{\partial x}u(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{s_{u,xx}}(t,x) = 2\kappa\,{\mathrm{s_{u,xx}}}^2(t,x)\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right) - 3\kappa\frac{\partial^2}{\partial x^2}\mathrm{s_{u,xx}}(t,x) - 2\kappa + \frac{6\kappa\left(\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\right)^2}{\mathrm{s_{u,xx}}(t,x)} - \frac{2\kappa\,\mathrm{s_{u,xx}}(t,x)\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x)}{\mathrm{V_u}(t,x)} + \frac{\kappa\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)}{\mathrm{V_u}(t,x)} + \frac{2\kappa\,\mathrm{s_{u,xx}}(t,x)\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{{\mathrm{V_u}}^2(t,x)} - u(t,x)\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x) + 2\,\mathrm{s_{u,xx}}(t,x)\frac{\partial}{\partial x}u(t,x)$$

Figure 2. Sample of code and Jupyter notebook outputs: systems of partial differential equations given in metric and in aspect forms produced by SymPKF when applied to the Burgers equation (Eq. 20).

as reproduced in Eq. (21), the expectation in the dynamics of the mean is replaced by the prognostic field; for the Burgers equation, $\mathbb{E}[u](t,x)$ is simply denoted by $u(t,x)$.

While the Burgers equation only contains two physical processes, i.e., the nonlinear advection and the diffusion, the resulting PKF dynamics in Eq. (21) make numerous terms appear, which justifies the use of symbolic computation, as mentioned above. The computation of the PKF dynamics leading to the metric and to the aspect tensor formulation takes about 1 s of computation (Intel Core i7-7820HQ CPU at 2.90 GHz × 8).

In this example, the splitting strategy has not been considered to simplify the computation of the PKF dynamics. However, it can be done by considering the PKF dynamics for the advection $\partial_t u = -u\partial_x u$ and the diffusion $\partial_t u = \kappa\partial_x^2 u$, and computed separately, then merged to find the PKF dynamics of the full Burgers equation. For instance, Fig. 3 shows the PKF dynamics for the advection (first cell) and for the diffusion (second cell); the output can be traced back in Eq. (2), e.g., by the terms in κ for the diffusion.

Thanks to the symbolic computation using the expectation operator, as implemented by the class `Expectation`, it is possible to handle terms such as $\mathbb{E}[\varepsilon_u\partial_x^4\varepsilon_u]$ during the computation of the PKF dynamics. The next section details how these terms are handled during the computation and the closure issue they bring.

```
# PKF for the non-linear advection
nladvection_pkf = SymbolicPKF(
    Eq(Derivative(u,t), -u*Derivative(u,x))
)

for equation in nladvection_pkf.in_aspect: display(equation)
```

$$\frac{\partial}{\partial t}u(t,x) = -u(t,x)\frac{\partial}{\partial x}u(t,x) - \frac{\frac{\partial}{\partial x}\mathrm{V_u}(t,x)}{2}$$

$$\frac{\partial}{\partial t}\mathrm{V_u}(t,x) = -u(t,x)\frac{\partial}{\partial x}\mathrm{V_u}(t,x) - 2\,\mathrm{V_u}(t,x)\frac{\partial}{\partial x}u(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{s_{u,xx}}(t,x) = -u(t,x)\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x) + 2\,\mathrm{s_{u,xx}}(t,x)\frac{\partial}{\partial x}u(t,x)$$

```
# PKF for the diffusion
diffusion_pkf = SymbolicPKF(
    Eq(Derivative(u,t), kappa*Derivative(u,x,2))
)

for equation in diffusion_pkf.in_aspect: display(equation)
```

$$\frac{\partial}{\partial t}u(t,x) = \kappa\frac{\partial^2}{\partial x^2}u(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{V_u}(t,x) = -\frac{2\kappa\,\mathrm{V_u}(t,x)}{\mathrm{s_{u,xx}}(t,x)} + \kappa\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x) - \frac{\kappa\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{2\,\mathrm{V_u}(t,x)}$$

$$\frac{\partial}{\partial t}\mathrm{s_{u,xx}}(t,x) = 2\kappa\,{\mathrm{s_{u,xx}}}^2(t,x)\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right) - 3\kappa\frac{\partial^2}{\partial x^2}\mathrm{s_{u,xx}}(t,x) - 2\kappa + \frac{6\kappa\left(\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\right)^2}{\mathrm{s_{u,xx}}(t,x)} - \frac{2\kappa\,\mathrm{s_{u,xx}}(t,x)\frac{\partial^2}{\partial x^2}\mathrm{V_u}(t,x)}{\mathrm{V_u}(t,x)} + \frac{\kappa\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)}{\mathrm{V_u}(t,x)} + \frac{2\kappa\,\mathrm{s_{u,xx}}(t,x)\left(\frac{\partial}{\partial x}\mathrm{V_u}(t,x)\right)^2}{{\mathrm{V_u}}^2(t,x)}$$

Figure 3. Illustration of the splitting strategy that can be used to compute the PKF dynamics and applied here for the Burgers equation: PKF dynamics of the Burgers equation can be obtained from the PKF dynamics of the advection (first cell) and of the diffusion (second cell).

4.3 Comments on the computation of the VLATcov PKF dynamics and the closure issue

4.3.1 Computation of terms $\mathbb{E}[\partial^\alpha\varepsilon\partial^\beta\varepsilon]$ and their connection to the correlation function

An important point is that terms such as $\mathbb{E}[\varepsilon\partial^\alpha\varepsilon]$, e.g., $\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]$ in Eq. (21) are directly connected to the correlation function $\rho(\mathbf{x},\mathbf{y}) = \mathbb{E}[\varepsilon(\mathbf{x})\varepsilon(\mathbf{y})]$ whose Taylor expansion is written as

$$\rho(\mathbf{x},\mathbf{x}+\delta\mathbf{x}) = \sum_k \frac{1}{k!}\mathbb{E}\left[\varepsilon(\mathbf{x})\partial^k\varepsilon(\mathbf{x})\right]\delta\mathbf{x}^k. \tag{22}$$

However, during their computation, the VLATcov PKF dynamics make the terms $\mathbb{E}[\partial^\alpha\varepsilon\partial^\beta\varepsilon]$ appear, with $|\alpha| \le |\beta|$, where for any α, ∂^α denotes the derivative with respect to the multi-index $\alpha = (\alpha_i)_{i\in[1,n]}$, with α_i denoting the derivative order with respect to the ith coordinate x_i of the coordinate system and where the sum of all derivative order is denoted by $|\alpha| = \sum_i \alpha_i$. The issue it that these terms in $\mathbb{E}\left[\partial^\alpha\varepsilon\partial^\beta\varepsilon\right]$ are not directly connected to the Taylor expansion in Eq. (22).

The interesting property of these terms is that they can be reworded as spatial derivatives of terms in the form $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$. More precisely, any term $\mathbb{E}\left[\partial^\alpha\varepsilon\partial^\beta\varepsilon\right]$ can be written from derivative of terms in $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$, where $|\gamma| < |\alpha| + |\beta|$, and the term $\mathbb{E}\left[\varepsilon\partial^{\alpha+\beta}\varepsilon\right]$ (see Appendix B for the proof). So, to replace any term in $\mathbb{E}\left[\partial^\alpha\varepsilon\partial^\beta\varepsilon\right]$ by terms in $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$, where $|\gamma| < |\alpha| + |\beta|$, a substitution dictionary is computed in SymPKF and stored as the variable `subs_tree`. The computation of this substitution dictionary is performed thanks to a dynamical programming strategy. Thereafter, the integer $|\alpha| + |\beta|$ is called the order of the term $\mathbb{E}\left[\partial^\alpha\varepsilon\partial^\beta\varepsilon\right]$. Figure 4 shows the substitution dictionary computed for the Burgers equation. It appears that terms of order lower than 3 can be explicitly written from the metric (or its derivatives), while terms of order larger than 4 cannot: this is known as the closure issue (Pannekoucke et al., 2018a).

```
# Display the automatically generated dictonary used to substitute E[D^\alpha␣
 ↪\varepsilon D^\beta \varepsilon]
# as terms in E[\varepsilon D^\gamma \varepsilon]
for key, value in pkf_burgers.subs_tree.items():
    display({key:value})
```

$$\left\{\mathbb{E}\left(\left(\frac{\partial}{\partial x}\varepsilon_u(t,x,\omega)\right)^2\right) : \mathrm{g_{u,xx}}(t,x)\right\}$$

$$\left\{\mathbb{E}\left(\frac{\partial}{\partial x}\varepsilon_u(t,x,\omega)\frac{\partial^2}{\partial x^2}\varepsilon_u(t,x,\omega)\right) : \frac{\frac{\partial}{\partial x}\mathrm{g_{u,xx}}(t,x)}{2}\right\}$$

$$\left\{\mathbb{E}\left(\frac{\partial}{\partial x}\varepsilon_u(t,x,\omega)\frac{\partial^3}{\partial x^3}\varepsilon_u(t,x,\omega)\right) : -\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right) - \frac{3\frac{\partial^2}{\partial x^2}\mathrm{g_{u,xx}}(t,x)}{2}\right\}$$

$$\left\{\mathbb{E}\left(\left(\frac{\partial^2}{\partial x^2}\varepsilon_u(t,x,\omega)\right)^2\right) : \mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right) + 2\frac{\partial^2}{\partial x^2}\mathrm{g_{u,xx}}(t,x)\right\}$$

Figure 4. Substitution dictionary computed in SymPKF to replace terms such as $\mathbb{E}[\partial^\alpha\varepsilon\partial^\beta\varepsilon]$ by terms in $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$, where $|\gamma| < |\alpha| + |\beta|$.

The term $\mathbb{E}[\varepsilon\partial_x^4\varepsilon]$, which features long-range correlations, cannot be related to the variance or to the metric and has to be closed. We detail this point in the next section.

4.3.2 Analytical and data-driven closure

A naïve closure for the PKF dynamics in Eq. (21) would be to replace the unknown term $\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]$ by zero. However, in the third equation that corresponds to the aspect tensor dynamics, the coefficient -3κ of the diffusion term $\partial_x^2 s_u$ being negative, it follows that the dynamics of s_u numerically explode at an exponential rate. Of course, because the system represents the uncertainty dynamics of the Burgers equation in Eq. (20) that is well posed, the parametric dynamics should not explode. Hence, the unknown term $\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]$ is crucial: it can balance the negative diffusion to stabilize the parametric dynamics.

For the Burgers equation, a closure for $\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]$ has been previously proposed (Pannekoucke et al., 2018a), given by

$$\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right] \sim \frac{2}{s_u^2}\partial_x^2 s_u + \frac{3}{s_u^2} - 4\frac{(\partial_x s_u)^2}{s_u^3}, \tag{23}$$

where the symbol $\sim$ is used to indicate that this is not an equality but a proposal of closure for the term in the left-hand side and that leads to the closed system

$$\left\{\begin{array}{rcl}
\partial_t u & = & -u\partial_x u + \kappa\partial_x^2 u - \frac{1}{2}\partial_x V_u,\\
\partial_t V_u & = & -u\partial_x V_u - 2(\partial_x u)V_u + \kappa\partial_x^2 V_u\\
 & & -\frac{\kappa}{2}\frac{1}{V_u}(\partial_x V_u)^2 - \frac{2\kappa}{s_{u,xx}}V_u,\\
\partial_t s_{u,xx} & = & -u\partial_x s_{u,xx} + 2(\partial_x u)s_{u,xx} + 4\kappa\\
 & & -2\frac{\kappa s_{u,xx}}{V_u}\partial_x^2 V_u + 2\frac{\kappa s_{u,xx}}{V_u^2}(\partial_x V_u)^2\\
 & & +\kappa\frac{1}{V_u}\partial_x V_u\partial_x s_{u,xx} + \kappa\partial_x^2 s_{u,xx}\\
 & & -2\kappa\frac{1}{s_{u,xx}}(\partial_x s_{u,xx})^2.
\end{array}\right. \tag{24}$$

The closure in Eq. (23) results from a local Gaussian approximation of the correlation function. Previous numerical experiments have shown that this closure is well adapted to the Burgers equation (Pannekoucke et al., 2018a). But the approach that has been followed to find this closure is quite specific, and it would be interesting to design a general way to find such a closure.

In particular, it would be interesting to search for a generic way to design closures that leverage the symbolic computation, which could be plugged with the PKF dynamics computed from SymPKF at a symbolic level. To do so, we propose an empirical closure that leverages a data-driven strategy to hybridize machine learning with physics, as proposed by Pannekoucke and Fablet (2020) with their neural network generator `PDE-NetGen`.

The construction of the proposal relies on the symbolic computation shown in Fig. 5.

The first step is to consider an analytical approximation for the correlation function. For the illustration, we consider the local correlation function to be well approximated by the quasi-Gaussian function

$$\rho(x, x+\delta x) \approx \exp\left(-\frac{\delta x^2}{s_u(x) + s_u(x+\delta x)}\right). \tag{25}$$

Then, the second step is to perform the computation of the Taylor's expansion of Eq. (22) at a symbolic level. This is done thanks to SymPy with the method `series` applied to Eq. (25) for δx near the value 0 and at a given order; e.g., for the illustration expansion is computed as the sixth order in Fig. 5.

Then, the identification with the Taylor's expansion in Eq. (22) leads to the closure

$$\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right] \sim \frac{3}{s_{u,xx}^2}\partial_x^2 s_{u,xx} + \frac{3}{s_{u,xx}^2} - 3\frac{\left(\partial_x s_{u,xx}\right)^2}{s_{u,xx}^3}. \tag{26}$$

```
import sympy
from sympkf import remove_eval_derivative, Expectation

s = pkf_burgers.fields[u].aspect[0]
eps = pkf_burgers.fields[u].epsilon
dx = sympy.Symbol('\delta x')

# definition of the correlation
rho = sympy.exp(- dx*dx / ( s + s.subs(x,x+dx) ) )

# Taylor expansion of the correlation with respect to dx, at order 6
taylor_order = 6
taylor = rho.series(dx,0,taylor_order)
taylor = remove_eval_derivative(taylor.removeO())

# Definition of pattern for the design of closure proposals
a = sympy.Wild("a",properties=[lambda k: k.is_Rational])
b = sympy.Wild("b")

for order in [4,5]:
    # Extract the term of a given order
    expr = taylor.coeff(dx,order)*sympy.factorial(order)
    if order==4: display(expr)
    # Create a proposal for the term at the order from pattern matching
    expr = sum([sympy.symbols(f'a_{k}^{order}')*term.match(a*b)[b]
                for k,term in enumerate(expr.args)] )
    display({Expectation(eps*Derivative(eps,x,order)):expr})
```

$$\frac{3\frac{\partial^2}{\partial x^2}\mathrm{s_{u,xx}}(t,x)}{\mathrm{s_{u,xx}}^2(t,x)}+\frac{3}{\mathrm{s_{u,xx}}^2(t,x)}-\frac{3\left(\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\right)^2}{\mathrm{s_{u,xx}}^3(t,x)}$$

$$\left\{\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^4}{\partial x^4}\varepsilon_u(t,x,\omega)\right):\frac{a_0^4}{\mathrm{s_{u,xx}}^2(t,x)}+\frac{a_1^4\left(\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\right)^2}{\mathrm{s_{u,xx}}^3(t,x)}+\frac{a_2^4\frac{\partial^2}{\partial x^2}\mathrm{s_{u,xx}}(t,x)}{\mathrm{s_{u,xx}}^2(t,x)}\right\}$$

$$\left\{\mathbb{E}\left(\varepsilon_u(t,x,\omega)\frac{\partial^5}{\partial x^5}\varepsilon_u(t,x,\omega)\right):\frac{a_0^5\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)}{\mathrm{s_{u,xx}}^3(t,x)}+\frac{a_1^5\frac{\partial^3}{\partial x^3}\mathrm{s_{u,xx}}(t,x)}{\mathrm{s_{u,xx}}^2(t,x)}+\frac{a_2^5\left(\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\right)^3}{\mathrm{s_{u,xx}}^4(t,x)}+\frac{a_3^5\frac{\partial}{\partial x}\mathrm{s_{u,xx}}(t,x)\frac{\partial^2}{\partial x^2}\mathrm{s_{u,xx}}(t,x)}{\mathrm{s_{u,xx}}^3(t,x)}\right\}$$

Figure 5. Example of a symbolic computation leading to a proposal for the closure of the unknown terms of order 4 and 5.

While it looks like the closure in Eq. (23), the coefficients are not the same. But this suggests that the closure of $\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]$ can be expanded as

$$\mathbb{E}\left[\varepsilon_u\partial_x^4\varepsilon_u\right]\sim\frac{a_0^4}{s_{u,xx}^2}\partial_x^2 s_{u,xx}+\frac{a_1^4}{s_{u,xx}^2}+a_3^4\frac{\left(\partial_x s_{u,xx}\right)^2}{s_{u,xx}^3},\qquad(27)$$

where $\mathbf{a}^4=(a_0^4,a_1^4,a_2^4)$ represents three unknown reals. A data-driven strategy can be considered to find an appropriate value of $\mathbf{a}^4$ from experiments. This has been investigated by using the automatic generator of a neural network `PDE-NetGen`, which bridges the gap between the physics and the machine learning (Pannekoucke and Fablet, 2020) and with which the training has led to the value $\mathbf{a}^4\approx(1.86,3.0,-3.6)$. Since this proposal is deduced from symbolic computation, it is easy to build some proposals for higher-order unknown terms as is shown in Fig. 5 for the term $\mathbb{E}[\varepsilon_u\partial_x^5\varepsilon_u]$.

Whatever closure has been obtained in an analytical or an empirical way, it remains to compute the closed PKF dynamics to assess their performance. To do so a numerical implementation of the system of partial differential equations has to be introduced. As for the computation of the PKF dynamics, the design of a numerical code can be tedious, with a risk of introducing errors in the implementation due to the numerous terms occurring in the PKF dynamics. To facilitate the research on the PKF, SymPKF comes with a Python numerical code generator, which provides an end-to-end investigation of the PKF dynamics. This code generator is now detailed.

4.4 Automatic code generation for numerical simulations

While compiled language with appropriate optimization should be important for industrial applications, we chose to implement a pure Python code generator, which offers a

```
from sympy import Integer
from sympkf import FDModelBuilder

g = pkf_burgers.fields[u].metric[0] # metric tensor
s = pkf_burgers.fields[u].aspect[0] # aspect tensor

# loc. Gaussian closure of P18 in metric form, then in aspect form
P18_closure = Integer(3)*g**Integer(2)-Integer(2)*Derivative(g,x,2)
P18_closure = P18_closure.subs(g,1/s).doit().expand()

# Introduction of the closure as a dictionary
unclosed_term = list(pkf_burgers.unclosed_terms)[0]
pkf_burgers.set_closure({unclosed_term:P18_closure})

# Build a numerical code at runtime
exec(FDModelBuilder(pkf_burgers.in_aspect, class_name='ClosedPKFBurgers').code)
closed_pkf_burgers = ClosedPKFBurgers(shape=(241,))
```

```
Warning: constant `kappa` has to be set

Sample of the numerical code generated in the class ClosedPKFBurgers

[..]
# Compute derivatives
#------------------------
DV_u_x_o1 = (-V_u[np.ix_(self.index('x',-1))] +
      V_u[np.ix_(self.index('x',1))])/(2*self.dx[self.coordinates.index('x')])

[..]
# Implementation of the trend
#--------------------------------
du[:] = -DV_u_x_o1/2 - Du_x_o1*u + Du_x_o2*kappa

dV_u[:] = -DV_u_x_o1**2*kappa/(2*V_u) - DV_u_x_o1*u + DV_u_x_o2*kappa -
    2*Du_x_o1*V_u - 2*V_u*kappa/s_u_xx

ds_u_xx[:] = 2*DV_u_x_o1**2*s_u_xx*kappa/V_u**2 +
     DV_u_x_o1*Ds_u_xx_x_o1*kappa/V_u -
    2*DV_u_x_o2*s_u_xx*kappa/V_u - 2*Ds_u_xx_x_o1**2*kappa/s_u_xx -
     Ds_u_xx_x_o1*u +
     Ds_u_xx_x_o2*kappa + 2*Du_x_o1*s_u_xx + 4*kappa
[..]
```

Figure 6. Introduction of a closure and automatic generation of a numerical code in SymPKF.

simple research framework for exploring the design of PKF dynamics. It would have been possible to use a code generator already based on SymPy (see, e.g., Louboutin et al., 2019), but with such code generators being domain-specific, it was less adapted to the investigation of the PKF for arbitrary dynamics. Instead, we consider a finite-difference implementation of partial derivatives with respect to spatial coordinates. The default domain to perform the computation is the periodic unit square with the dimension of the number of spatial coordinates. The length of the domain can be specified along each direction. The domain is regularly discretized along each direction, while the number of grid points can be specified for each direction.

The finite difference takes the form of an operator $\mathcal{F}$ that approximates any partial derivate at a second order of consistency: for any multi-index α, $\mathcal{F}^{\alpha}u \underset{0}{=} \partial^{\alpha}u + \mathcal{O}(|\delta\mathbf{x}|^2)$, where $\mathcal{O}$ is Landau's big O notation. For any f, the notation $f(\delta x) \underset{0}{=} \mathcal{O}(\delta x^2)$ means that $\lim_{\delta x \to 0} \frac{f(\delta x)}{\delta x^2}$ is finite. The operator $\mathcal{F}$ computed with respect to independent coordinates commutes, e.g., $\mathcal{F}_{xy} = \mathcal{F}_x \circ \mathcal{F}_y = \mathcal{F}_y \circ \mathcal{F}_x$, where $\circ$ denotes the composition, but it does not commute for dependent coordinates, e.g., $\mathcal{F}_x^2 \neq \mathcal{F}_x \circ \mathcal{F}_x$. The finite difference of the par-

tial derivative with respect to the multi-index is computed sequentially, e.g., $\mathcal{F}_{xxy} = \mathcal{F}_x^2 \circ \mathcal{F}_y = \mathcal{F}_y \circ \mathcal{F}_x^2$. The finite difference of order α with respect to a single spatial coordinate is the centered finite difference based on $\alpha + 1$ points.

For instance, Fig. 6 shows how to close the PKF dynamics for the Burgers equation following P18 and how to build a code from an instance of the class `sympkf.FDModelBuilder`: it creates the class `ClosedPKFBurgers`. In this example, the code is rendered from templates thanks to Jinja[3]; then it is executed at runtime. Note that the code can also be written in an appropriate Python module for adapting the code to a particular situation or to check the correctness of the generated code. At the end, the instance `closed_pkf_burgers` of the class `ClosedPKFBurgers` is created, raising a warning to indicate that the value of constant κ has to be specified before performing a numerical simulation. Note that it is possible to set the value of kappa as a keyword argument in the class `ClosedPKFBurgers`. Figure 6 also shows a sample of the generated code with the implementation of the computation of the first-order partial derivative $\partial_x V_u$, which appears as a centered finite difference. Then, the sample of code shows how the partial derivatives are used to compute the trend of the system of partial differential equations in Eq. (24).

The numerical integration is handled through the inheritance mechanism: the class `ClosedPKFBurgers` inherits the integration time loop from the class `sympkf.Model` as described by the unified modeling language (UML) diagram shown in Fig. 7. In particular, the class `Model` contains several time schemes, e.g., a fourth-order Runge–Kutta scheme. Note that in the present implementation of SymPKF, only explicit time schemes are considered, but it could be possible to leverage the symbolic computation to implement other schemes more adapted to a given PDE, e.g., an implicit scheme for the transport or the diffusion or a high-order exponential time-differencing method (Kassam and Trefethen, 2005) with which the linear and the nonlinear parts would be automatically determined from the symbolic computation. The details of the instance `closed_pkf_burgers` of the class `ClosedPKFBurgers` make it appear that the closed system in Eq. (24) will be integrated by using a RK4 time scheme on the segment $[0, D]$ (here $D = 1$) with periodic boundaries and discretized by 241 points.

Thanks to the end-to-end framework proposed in SymPKF, it is possible to perform a numerical simulation based on the PKF dynamics in Eq. (23). To do so, we set $\kappa = 0.0025$ and consider the simulation starting from the Gaussian distribution $\mathcal{N}(u_0, \mathbf{P}_h^{\mathrm{f}})$ of mean $u_0(x) = U_{\max}[1 + \cos(2\pi(x - D/4)/D)]/2$ with $U_{\max} = 0.5$ and of covariance matrix

[3]https://jinja.palletsprojects.com/en/2.11.x/, last access: 22 September 2021.

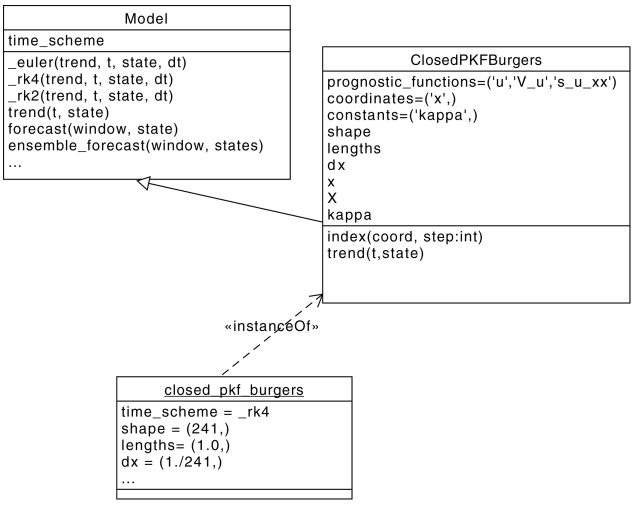


Figure 7. UML diagram showing the inheritance mechanism implemented in SymPKF: the class `ClosedPKFBurgers` inherits from the class `Model`, which implements several time schemes. Here, `closed_pkf_burgers` is an instance of the class `ClosedPKFBurgers`.

$$\mathbf{P}_h^{\mathrm{f}}(x, y) = V_h \exp\left(-\frac{(x-y)^2}{2l_h^2}\right), \tag{28}$$

where $V_h = 0.01 U_{\max}$ and $l_h = 0.02D \approx 5\mathrm{d}x$. The time step of the fourth-order Runge–Kutta scheme is $\mathrm{d}t = 0.002$. The evolution predicted from the PKF is shown in Fig. 8 (solid lines). This simulation illustrates the time evolution of the mean (panel a) and of the variance (panel b); panel (c) represents the evolution of the correlation length scale defined from the aspect tensor as $L(x) = \sqrt{s_{u,xx}(x)}$. Note that at time 0, the length scale field is $L(x) = l_h$. For the illustrations, the variance (the length scale) is normalized by its initial value V_h (l_h).

In order to show the skill of the PKF applied to the Burgers equation, when using the closure of P18, an ensemble validation is now performed. Note that the code generator of SymPKF can be used for arbitrary dynamics, e.g., the Burgers equation itself. Hence, a numerical code solving the Burgers equation is rendered from its symbolic definition. Then an ensemble of 1600 forecasts is computed starting from an ensemble of initial errors at time 0. The ensemble of initial errors is sampled from the Gaussian distribution $\mathcal{N}\left(0, \mathbf{P}_h^{\mathrm{f}}\right)$ of zero mean and covariance matrix $\mathbf{P}_h^{\mathrm{f}}$. Note that the ensemble forecasting implemented in SymPKF as the method `Model.ensemble_forecast` (see Fig. 7) leverages the multiprocessing tools of Python to use the multiple cores of the CPU when present. On the computer used for the simulation, the forecasts are performed in parallel on eight cores. The ensemble estimation of the mean, the variance, and the length scale is shown in Fig. 8 (dashed lines). Since

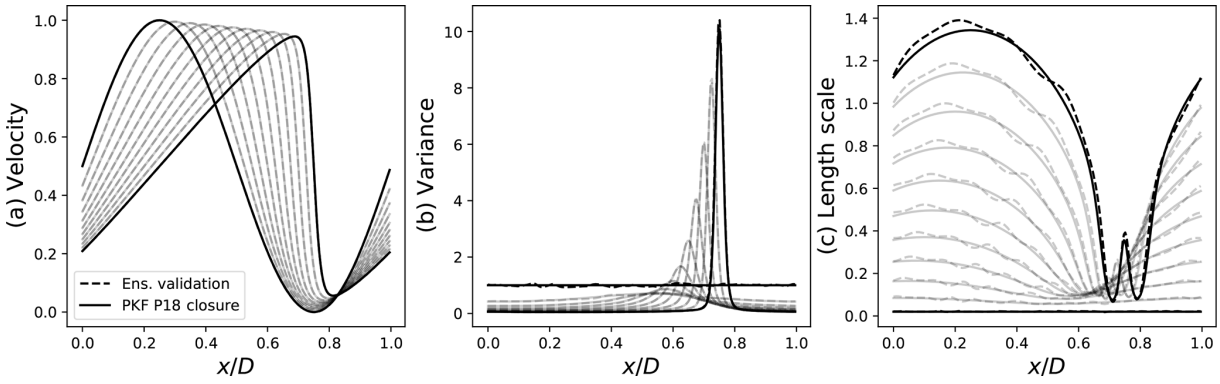


Figure 8. Illustration of a numerical simulation of the PKF dynamics in Eq. (23) (solid line), with the mean **(a)**, the variance **(b)**, and the correlation length scale **(c)**, which is defined from the component $s_{u,xx}$ of the aspect tensor by $L(x) = \sqrt{s_{u,xx}(x)}$. An ensemble-based validation of the PKF dynamics is shown as a dashed line. (Pannekoucke and Fablet, 2020, see their Fig. 7)

the ensemble is finite, sampling noise is visible, e.g., in the variance at the initial time that is not strictly equal to V_h. In this simulation, it appears that the PKF (solid line) coincides with the ensemble estimation (dashed lines), which shows the ability of the PKF to predict the forecast error covariance dynamics. Note that the notebook corresponding to the Burgers experiment is available in the example directory of SymPKF.

While this example shows an illustration of SymPKF in a 1D domain, the package also applies in 2D and 3D domains, as presented now.

4.5 Illustration of dynamics in a 2D domain

In order to illustrate the ability of SymPKF to apply in a 2D or 3D domain, we consider the linear advection of a scalar field $c(t,x,y)$ by a stationary velocity field $\mathbf{u} = (u(x,y), v(x,y))$, which reads as the partial differential equation

$$\partial_t c + \mathbf{u}\nabla c = 0. \tag{29}$$

As for the Burgers equation, the definition of the dynamics relies on SymPy (not shown but similar to the definition of the Burgers equation as given in Fig. 1). This leads to preprocessing the dynamics by creating the instance *advection* of the class `PDESystem`, which transforms the equation into a system of partial differential equations. In particular, the procedure will diagnose the prognostic functions of dynamics, here the function c. Then it identifies the constant functions, which can depend on space but not on time: here, these are the components of the velocity (u, v). The process also identifies exogenous functions and constants, of which there are none here.

The calculation of the parametric dynamics is handled by the class `SymbolicPKF` as shown in the first cell in Fig. 9. The parametric dynamics are a property of the instance *pkf_advection* of the class `SymbolicPKF`, and when it is called, the parametric dynamics are computed once and for all. The parametric dynamics formulated in terms of metric are first computed; see the second cell. For the 2D linear advection, the parametric dynamics are a system of five partial differential equations, as is shown in the output of the second cell: the dynamics of the ensemble average $\mathbb{E}[c]$, which outputs as c for the sake of simplicity (first equation), the dynamics of the variance (second equation), and the dynamics of the local metric tensor (last three equations). In compact form, the dynamics are given by the system

$$\partial_t c + \mathbf{u}\nabla c = 0, \tag{30a}$$
$$\partial_t V_c + \mathbf{u}\nabla V_c = 0, \tag{30b}$$
$$\partial_t \mathbf{g}_c + \mathbf{u}\nabla \mathbf{g}_c = -\mathbf{g}_c(\nabla\mathbf{u}) - (\nabla\mathbf{u})^{\mathrm{T}}\mathbf{g}_c, \tag{30c}$$

which corresponds to the 2D extension of the 1D dynamics first found by Cohn (1993) (Pannekoucke et al., 2016) and validates the computation performed in SymPKF. Due to the linearity of the linear advection in Eq. (29), the ensemble average in Eq. (30a) is governed by the same dynamics as in Eq. (29). While both the variance in Eq. (30b) and the metric are advected by the flow, the metric is also deformed by the shear in Eq. (30c). This deformation more commonly appears in the dynamics written in aspect tensor form, which is given by

$$\partial_t c + \mathbf{u}\nabla c = 0, \tag{31a}$$
$$\partial_t V_c + \mathbf{u}\nabla V_c = 0, \tag{31b}$$
$$\partial_t \mathbf{s}_c + \mathbf{u}\nabla \mathbf{s}_c = (\nabla\mathbf{u})\,\mathbf{s}_c + \mathbf{s}_c(\nabla\mathbf{u})^T, \tag{31c}$$

where Eq. (31c) is similar to the dynamics of the conformation tensor in viscoelastic flow (Bird and Wiest, 1995; Hameduddin et al., 2018).

We do not introduce any numerical simulation of the PKF dynamics in Eq. (30) or Eq. (31), but interested readers are referred to the 2D numerical PKF assimilation cycles of Pannekoucke (2021b), which have been made thanks to SymPKF.

```
# Define the advection and its associated PKF system
x, y = symbols('x y')
c = Function('c')(t,x,y)
u = Function('u')(x,y)
v = Function('v')(x,y)
pkf_advection = SymbolicPKF([ Eq(Derivative(c,t),␣
 ↪-u*Derivative(c,x)-v*Derivative(c,y)) ])
```

```
# Compute the PKF system rendered in metric tensor form (the computation is only␣
 ↪performed at the first call)
for equation in pkf_advection.in_metric:    display(equation)
```

$$\frac{\partial}{\partial t}c(t,x,y) = -u(x,y)\frac{\partial}{\partial x}c(t,x,y) - v(x,y)\frac{\partial}{\partial y}c(t,x,y)$$

$$\frac{\partial}{\partial t}\mathrm{V_c}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{V_c}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{V_c}\,(t,x,y)$$

$$\frac{\partial}{\partial t}\mathrm{g_{c,xx}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{g_{c,xx}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{g_{c,xx}}\,(t,x,y) - 2\,\mathrm{g_{c,xx}}\,(t,x,y)\frac{\partial}{\partial x}u(x,y) - 2\,\mathrm{g_{c,xy}}\,(t,x,y)\frac{\partial}{\partial x}v(x,y)$$

$$\frac{\partial}{\partial t}\mathrm{g_{c,xy}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{g_{c,xy}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{g_{c,xy}}\,(t,x,y) - \mathrm{g_{c,xx}}\,(t,x,y)\frac{\partial}{\partial y}u(x,y) - \mathrm{g_{c,xy}}\,(t,x,y)\frac{\partial}{\partial x}u(x,y) - \mathrm{g_{c,xy}}\,(t,x,y)\frac{\partial}{\partial y}v(x,y) - \mathrm{g_{c,yy}}\,(t,x,y)\frac{\partial}{\partial x}v(x,y)$$

$$\frac{\partial}{\partial t}\mathrm{g_{c,yy}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{g_{c,yy}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{g_{c,yy}}\,(t,x,y) - 2\,\mathrm{g_{c,xy}}\,(t,x,y)\frac{\partial}{\partial y}u(x,y) - 2\,\mathrm{g_{c,yy}}\,(t,x,y)\frac{\partial}{\partial y}v(x,y)$$

```
# Compute the PKF system rendered in aspect tensor form (the computation is only␣
 ↪performed at the first call)
for equation in pkf_advection.in_aspect:    display(equation)
```

$$\frac{\partial}{\partial t}c(t,x,y) = -u(x,y)\frac{\partial}{\partial x}c(t,x,y) - v(x,y)\frac{\partial}{\partial y}c(t,x,y)$$

$$\frac{\partial}{\partial t}\mathrm{V_c}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{V_c}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{V_c}\,(t,x,y)$$

$$\frac{\partial}{\partial t}\mathrm{s_{c,xx}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{s_{c,xx}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{s_{c,xx}}\,(t,x,y) + 2\,\mathrm{s_{c,xx}}\,(t,x,y)\frac{\partial}{\partial x}u(x,y) + 2\,\mathrm{s_{c,xy}}\,(t,x,y)\frac{\partial}{\partial y}u(x,y)$$

$$\frac{\partial}{\partial t}\mathrm{s_{c,xy}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{s_{c,xy}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{s_{c,xy}}\,(t,x,y) + \mathrm{s_{c,xx}}\,(t,x,y)\frac{\partial}{\partial x}v(x,y) + \mathrm{s_{c,xy}}\,(t,x,y)\frac{\partial}{\partial x}u(x,y) + \mathrm{s_{c,xy}}\,(t,x,y)\frac{\partial}{\partial y}v(x,y) + \mathrm{s_{c,yy}}\,(t,x,y)\frac{\partial}{\partial y}u(x,y)$$

$$\frac{\partial}{\partial t}\mathrm{s_{c,yy}}\,(t,x,y) = -u(x,y)\frac{\partial}{\partial x}\mathrm{s_{c,yy}}\,(t,x,y) - v(x,y)\frac{\partial}{\partial y}\mathrm{s_{c,yy}}\,(t,x,y) + 2\,\mathrm{s_{c,xy}}\,(t,x,y)\frac{\partial}{\partial x}v(x,y) + 2\,\mathrm{s_{c,yy}}\,(t,x,y)\frac{\partial}{\partial y}v(x,y)$$

Figure 9. Sample of code and Jupyter notebook outputs: system of partial differential equations produced by SymPKF when applied to the linear advection in Eq. (29).

```
# Define the dynamics
A = Function('A')(t,x)
B = Function('B')(t,x)
dynamics = [Eq(Derivative(A,t), B), Eq(Derivative(B,t), -A)]

# Compute the associated PKF system (here in metric)
for equation in SymbolicPKF(dynamics).in_metric:    display(equation)
```

$$\frac{\partial}{\partial t}A(t,x) = B(t,x)$$

$$\frac{\partial}{\partial t}B(t,x) = -A(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{V_A}(t,x) = 2\,\mathrm{V_{AB}}(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{V_B}(t,x) = -2\,\mathrm{V_{AB}}(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{V_{AB}}(t,x) = -\mathrm{V_A}(t,x) + \mathrm{V_B}(t,x)$$

$$\frac{\partial}{\partial t}\mathrm{g_{A,xx}}(t,x) = -\frac{2\,\mathrm{V_{AB}}(t,x)\,\mathrm{g_{A,xx}}(t,x)}{\mathrm{V_A}(t,x)} + \frac{2\sqrt{\mathrm{V_B}(t,x)}\mathbb{E}\left(\frac{\partial}{\partial x}\varepsilon_A(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_B(t,x,\omega)\right)}{\sqrt{\mathrm{V_A}(t,x)}} + \frac{\mathbb{E}\left(\varepsilon_B(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_A(t,x,\omega)\right)\frac{\partial}{\partial x}\mathrm{V_B}(t,x)}{\sqrt{\mathrm{V_A}(t,x)}\sqrt{\mathrm{V_B}(t,x)}} - \frac{\sqrt{\mathrm{V_B}(t,x)}\mathbb{E}\left(\varepsilon_B(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_A(t,x,\omega)\right)\frac{\partial}{\partial x}\mathrm{V_A}(t,x)}{\mathrm{V_A}^{\frac{3}{2}}(t,x)}$$

$$\frac{\partial}{\partial t}\mathrm{g_{B,xx}}(t,x) = \frac{2\,\mathrm{V_{AB}}(t,x)\,\mathrm{g_{B,xx}}(t,x)}{\mathrm{V_B}(t,x)} - \frac{2\sqrt{\mathrm{V_A}(t,x)}\mathbb{E}\left(\frac{\partial}{\partial x}\varepsilon_A(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_B(t,x,\omega)\right)}{\sqrt{\mathrm{V_B}(t,x)}} + \frac{\sqrt{\mathrm{V_A}(t,x)}\mathbb{E}\left(\varepsilon_A(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_B(t,x,\omega)\right)\frac{\partial}{\partial x}\mathrm{V_B}(t,x)}{\mathrm{V_B}^{\frac{3}{2}}(t,x)} - \frac{\mathbb{E}\left(\varepsilon_A(t,x,\omega)\frac{\partial}{\partial x}\varepsilon_B(t,x,\omega)\right)\frac{\partial}{\partial x}\mathrm{V_A}(t,x)}{\sqrt{\mathrm{V_A}(t,x)}\sqrt{\mathrm{V_B}(t,x)}}$$

Figure 10. Output of the computation by SymPKF of the PKF dynamics for the simple multivariate periodic chemical reaction, corresponding to the right-hand side of Eq. (32).

This example illustrates a 2D situation and shows the multidimensional capabilities of SymPKF. Similarly to the simulation conducted for the Burgers equation, it is possible to automatically generate a numerical code able to perform numerical simulations of the dynamics in Eq. (31) (not shown here). Hence, this 2D domain example showed the ability of SymPKF to apply in dimensions larger than 1D.

Before concluding, we would like to present a preliminary application of SymPKF in a multivariate situation.

4.6 Towards the PKF for multivariate dynamics

SymPKF can be used to compute the prediction of the variance and the anisotropy in a multivariate situation.

Note that one of the difficulties with the multivariate situation is that the number of equations increases linearly with the number of fields and the dimension of the domain; e.g., for a 1D (2D) domain and two multivariate physical fields, there are two ensemble-averaged fields, two variance fields, and two (six) metric fields. Of course this is no not a problem when using a computer algebra system as done in SymPKF.

To illustrate the multivariate situation, only a very simple example is introduced. Inspired from chemical transport models encountered in air quality, we consider the transport over a 1D domain of two chemical species, whose concentrations are denoted by $A(t,x)$ and $(B(t,x)$, advected by the wind $u(x)$. For the sake of simplicity, the two species interact following periodic dynamics as defined by the coupled system

$$\partial_t A + u\partial_x A = B, \tag{32a}$$

$$\partial_t B + u\partial_x B = -A. \tag{32b}$$

Thanks to the splitting strategy, the PKF dynamics due to the advection have already been detailed in the previous section (see Sect. 4.5), so we can focus on the chemical part of the dynamics, which is given by the processes on the right-hand side of Eq. (32). The PKF of the chemical part is computed thanks to SymPKF and shown in Fig. 10. This time, and as expected, multivariate statistics appear in the dynamics. Here, the dynamics of the cross-covariance $V_{AB} = \mathbb{E}[e_A e_B]$ are given by the fifth equation. The coupling brings up unknown terms, e.g., the term $\mathbb{E}[\partial_x \varepsilon_A \partial_x \varepsilon_B]$ in the sixth equation of the output shown in Fig. 10. Note that by taking into account the multivariate situation with the dynamics of the cross-covariance, the multivariate PKF hybridizes the continuous nature of the multivariate fields with the matrix form

in Eq. (8a), which corresponds here to the dynamics of the variances (V_A, V_B) and the cross-covariance V_{AB}.

To go further, some research is still needed to explore the dynamics and the modeling of the multivariate cross-covariances. A possible direction is to take advantage of the multivariate covariance model based on the balance operator as often introduced in variational data assimilation (Derber and Bouttier, 1999; Ricci et al., 2005). Note that such multivariate covariance models have recently been considered for the design of the multivariate PKF analysis step (Pannekoucke, 2021b). Another way is to consider a data-driven strategy to learn the physics of the unknown terms from a training based on ensembles of forecasts (Pannekoucke and Fablet, 2020).

To conclude, this example shows the potential of SymPKF to tackle the multivariate situation. Moreover, the example also shows that SymPKF is able to perform the PKF computation for a system of partial differential equations. However, all the equations should be prognostic; SymPKF is not able to handle diagnostic equations.

5 Conclusions

This contribution introduced the package SymPKF that can be used to conduct the research on the parametric Kalman filter prediction step for covariance models parameterized by the variance and the anisotropy (VLATcov models). SymPKF provides an end-to-end framework: from the equations of dynamics to the development of a numerical code.

The package has been first introduced by considering the nonlinear diffusive advection dynamics in the Burgers equation. In particular, this example shows the ability of SymPKF to handle abstract terms, e.g., the unclosed terms formulated with the expectation operator. The expectation operator implemented in SymPKF is a key tool for the computation of the PKF dynamics. Moreover, we showed how to handle a closure and how to automatically render numerical codes.

For univariate situations, SymPKF applies in a 1D domain as well as in 2D and 3D domains. This has been shown by considering the computation of the PKF dynamics for the linear advection equation on a 2D domain.

A preliminary illustration with multivariate dynamics showed the potential of SymPKF to handle the dynamics of multivariate covariance. But this point has to be further investigated, and this constitutes the main perspective of development. Moreover, to perform a multivariate assimilation cycle with the PKF, the multivariate formulation of the PKF analysis state is needed. A first investigation of the multivariate PKF assimilation has been proposed by Pannekoucke (2021b).

In its present implementation, SymPKF is limited to computation with prognostic equations. It is not possible to consider dynamics based on diagnostic equations, while these are often encountered in atmospheric fluid dynamics, e.g., the geostrophic balance. This constitutes another topic of research development for the PKF, facilitated by the use of symbolic exploration.

Note that the expectation operator as introduced here can be used to compute Reynolds equations encountered in turbulence. This opens new perspectives for the use of SymPKF for other applications that could be interesting, especially for automatic code generation.

Appendix A: Splitting for the computation of the parametric dynamics

In this section we show that using a splitting strategy is possible for the design of the parametric dynamics. For this, it is enough to show that given dynamics written as

$$\partial_t \mathcal{X} = f_1(\mathcal{X}) + f_2(\mathcal{X}), \tag{A1}$$

the dynamics of the error, the variance, the metric, and the aspect all write as a sum of trends depending on each process f_1 and f_2. We show this starting from the dynamics of the error.

Due to the linearity of the derivative operator, the TL dynamics resulting from Eq. (A1) are written as

$$\partial_t e = f_1'(e) + f_2'(e), \tag{A2}$$

where f_1' and f_2' denote the differential of the two functions, which can be written as the sum of two trends $\partial_t e_1 = f_1'(e)$ and $\partial_t e_2 = f_2'(e)$, depending exclusively on f_1 and f_2, respectively. For the variance's dynamics, $\partial_t V = 2\mathbb{E}[e\partial_t e]$, substitution by Eq. (A2) leads to

$$\partial_t V = \partial_t V_1 + \partial_t V_2, \tag{A3}$$

where $\partial_t V_1 = 2\mathbb{E}\left[e f_1'(e)\right]$ and $\partial_t V_2 = 2\mathbb{E}\left[e f_2'(e)\right]$ depend exclusively on f_1 and f_2, respectively. Then the standard deviation dynamics, obtained by differentiating $\sigma^2 = V$ as $2\sigma\partial_t\sigma = \partial_t V$,

$$\partial_t \sigma = \frac{1}{\sigma}\partial_t V_1 + \frac{1}{\sigma}\partial_t V_2, \tag{A4}$$

read as the sum of two trends $\partial_t\sigma_1 = \frac{1}{\sigma}\partial_t V_1$ and $\partial_t\sigma_2 = \frac{1}{\sigma}\partial_t V_2$, depending exclusively on f_1 and f_2, respectively. It results that the dynamics of the normalized error $\varepsilon = \frac{1}{\sigma}e$, deduced from the time derivative of $e = \sigma\varepsilon$, $\partial_t e = \varepsilon\partial_t\sigma + \sigma\partial_t\varepsilon$, read as

$$\partial_t \varepsilon = \frac{1}{\sigma}\left[f_1'(e) - \frac{\varepsilon}{2\sigma}\partial_t V_1\right] + \frac{1}{\sigma}\left[f_2'(e) - \frac{\varepsilon}{2\sigma}\partial_t V_2\right] \tag{A5}$$

and also expand as the sum of two trends $\partial_t\varepsilon_1 = \frac{1}{\sigma}[f_1'(e) - \frac{\varepsilon}{2\sigma}\partial_t V_1]$ and $\partial_t\varepsilon_2 = \frac{1}{\sigma}[f_2'(e) - \frac{\varepsilon}{2\sigma}\partial_t V_2]$, again depending exclusively on f_1 and f_2, respectively. For the metric terms $g_{ij} = \mathbb{E}[\partial_i\varepsilon\partial_j\varepsilon]$, we deduce that the dynamics $\partial_t g_{ij} = \mathbb{E}[\partial_i(\partial_t\varepsilon)\partial_j\varepsilon] + \mathbb{E}[\partial_i\varepsilon\partial_j(\partial_t\varepsilon)]$ expand as

$$\partial_t g_{ij} = \partial_t g_{ij\,1} + \partial_t g_{ij\,2}, \tag{A6}$$

with $\partial_t g_{ij\,1} = \mathbb{E}[\partial_i(\partial_t\varepsilon_1)\partial_j\varepsilon] + \mathbb{E}[\partial_i\varepsilon\partial_j(\partial_t\varepsilon_1)]$ and $\partial_t g_{ij\,2} = \mathbb{E}[\partial_i(\partial_t\varepsilon_2)\partial_j\varepsilon] + \mathbb{E}[\partial_i\varepsilon\partial_j(\partial_t\varepsilon_2)]$ where each partial trend depends exclusively on f_1 and f_2, respectively. To this end, the dynamics of the aspect tensor $\mathbf{s}$ are deduced from Eq. (17), which expands as

$$\partial_t \mathbf{s} = \partial_t \mathbf{s}_1 + \partial_t \mathbf{s}_2, \tag{A7}$$

where $\partial_t\mathbf{s}_1 = -\mathbf{s}(\partial_t\mathbf{g}_1)\mathbf{s}$ and $\partial_t\mathbf{s}_2 = -\mathbf{s}(\partial_t\mathbf{g}_2)\mathbf{s}$ only depend on f_1 and f_2, respectively.

To conclude, the computation of the parametric dynamics for Eq. (A1) can be performed from the parametric dynamics of $\partial_t\mathcal{X} = f_1(\mathcal{X})$ and $\partial_t\mathcal{X} = f_2(\mathcal{X})$ calculated separately, then merged together to obtain the dynamics of the variance in Eq. (A3), of the metric in Eq. (A6), and of the aspect tensors in Eq. (A7).

Appendix B: Computation of terms $\mathbb{E}[\partial^\alpha\varepsilon\partial^\beta\varepsilon]$

In this section we proof the property.

Property 1. Any term $\mathbb{E}[\partial^\alpha\varepsilon\partial^\beta\varepsilon]$ with $|\alpha| \leq |\beta|$ can be related to the correlation expansion term $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$, where $|\gamma| < |\alpha| + |\beta|$, and the term $\mathbb{E}\left[\varepsilon\partial^{\alpha+\beta}\varepsilon\right]$.

– *Proof.* The derivative with respect to a zero α_i is the identity operator. Note that the multi-index forms a semi-group since for two multi-indexes α and β we can form the multi-index $\alpha + \beta = (\alpha_i + \beta_i)_{i\in[1,n]}$.

Now Property 1 can be proven considering the following recurrent process, when assuming that the property is true for all patterns of degree strictly lower than the degree $|\alpha| + |\beta|$.

Without loss of generality we assume $\alpha_i > 0$ and denote $\delta_i = (\delta_{ij})_{j\in[1,n]}$, where δ_{ij} is the Kronecker symbol ($\delta_{ii} = 1$, $\delta_{ij} = 0$ for $j \neq i$). From the formula

$$\partial_{x^i}\left(\partial^{\alpha-\delta_i}\varepsilon\partial^\beta\varepsilon\right) = \partial^\alpha\varepsilon\partial^\beta\varepsilon + \partial^{\alpha-\delta_i}\varepsilon\partial^{\beta+\delta_i}\varepsilon \tag{B1}$$

and from the commutativity of the expectation operator and the partial derivative with respect to the coordinate system, the result is that

$$\mathbb{E}\left[\partial^\alpha\varepsilon\partial^\beta\varepsilon\right] = \partial_{x^i}\mathbb{E}\left[\partial^{\alpha-\delta_i}\varepsilon\partial^\beta\varepsilon\right] - \mathbb{E}\left[\partial^{\alpha-\delta_i}\varepsilon\partial^{\beta+\delta_i}\varepsilon\right], \tag{B2}$$

considering the terms of the left-hand side. On one hand, we observe that the degree of the first term is decreasing to $|\alpha| + |\beta| - 1$; from the recurrence assumption, $\mathbb{E}[\partial^{\alpha-\delta_i}\varepsilon\partial^\beta\varepsilon]$ can be expanded as terms of the form $\mathbb{E}[\varepsilon\partial^\gamma\varepsilon]$. On the other hand, the degree of the second term remains the same, $|\alpha| + |\beta|$, but with a shift of the derivative order. This shift of the order can be done again following the same process, leading after iterations to the term $\mathbb{E}[\varepsilon\partial^{\alpha+\beta}\varepsilon]$.

Code and data availability. The SymPKF package is free and open-source. It is distributed under the CeCILL-B free software license. The source code is provided through a GitHub repository at https://github.com/opannekoucke/sympkf (last access: 22 March 2021). A snapshot of SymPKF is available at https://doi.org/10.5281/zenodo.4608514 (Pannekoucke, 2021c). The data used for the simulations presented here are generated at runtime when using the Jupyter notebooks.

Author contributions. OP introduced the symbolic computation of the PKF dynamics, and OP and PA imagined an end-to-end framework for the design of the PKF dynamics from the equation of the dynamics to the numerical simulation thanks to an automatic code generation. OP developed the codes.

Competing interests. The authors declare that they have no conflict of interest.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Acknowledgements. We would like to thank Sylwester Arabas and the two anonymous referees for their fruitful comments, which have contributed to improving the paper. The UML class diagram has been generated from UMLlet (Auer et al., 2003).

Financial support. This research has been supported by the French national program LEFE/INSU (Étude du filtre de KAlman PAramétrique, KAPA).

Review statement. This paper was edited by Sylwester Arabas and reviewed by two anonymous referees.

References

Auer, M., Tschurtschenthaler, T., and Biffl, S.: A Flyweight UML Modelling Tool for Software Development in Heterogeneous Environments, in: Proceedings of the 29th Conference on EUROMICRO, EUROMICRO '03, pp. 267–272, IEEE Computer Society, Washington, DC, USA, 1–6 September 2003, https://doi.org/10.1109/EURMIC.2003.1231600, 2003.

Berre, L.: Estimation of Synoptic and Mesoscale Forecast Error Covariances in a Limited-Area Model, Mon. Weather Rev., 128, 644–667, 2000.

Bird, R. B. and Wiest, J. M.: Constitutive Equations for Polymeric Liquids, Annu. Rev. Fluid Mech., 27, 169–193, https://doi.org/10.1146/annurev.fl.27.010195.001125, 1995.

Cohn, S.: Dynamics of Short-Term Univariate Forecast Error Covariances, Mon. Weather Rev., 121, 3123–3149, https://doi.org/10.1175/1520-0493(1993)121<3123:DOSTUF>2.0.CO;2, 1993.

Courtier, P., Andersson, E., Pailleux, J., Vasiljević, W. H., Hamrud, D., Hollingsworth, M., Rabier, A. F., and Fisher, M.: The ECMWF implementation of three-dimensional variational assimilation (3D-Var). I: Formulation, Q. J. Roy. Meteor. Soc., 124, 1783–1807, 1998.

Derber, J. and Bouttier, F.: A reformulation of the background error covariance in the ECMWF global data assimilation system, Tellus A, 51, 195–221, https://doi.org/10.3402/tellusa.v51i2.12316, 1999.

Evensen, G.: Data Assimilation: The Ensemble Kalman Filter, Springer-Verlag Berlin Heidelberg, https://doi.org/10.1007/978-3-642-03711-5, 2009.

Fisher, M.: Development of a simplified Kalman filter, Tech. Rep., 260, ECMWF, https://doi.org/10.21957/vz40cqca4, 1998.

Fisher, M.: Generalized frames on the sphere, with application to background error covariance modelling, in: Proc. ECMWF Seminar on "Recent developments in numerical methods for atmospheric and ocean modelling", edited by ECMWF, pp. 87–102, Reading, UK, 2004.

Hameduddin, I., Meneveau, C., Zaki, T. A., and Gayme, D. F.: Geometric decomposition of the conformation tensor in viscoelastic turbulence, J. Fluid Mech., 842, 395–427, https://doi.org/10.1017/jfm.2018.118, 2018.

Jazwinski, A.: Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems, Journal of Basic Engineering, 82, 35–45, https://doi.org/10.1115/1.3662552, 1960.

Kassam, A.-K. and Trefethen, L.: Fourth-order time-stepping for stiff PDEs, SIAM J. Sci. Comput., 26, 1214–1233, 2005.

Lesieur, M.: Turbulence in Fluids, Springer, p. 558, ISBN 978-1-4020-6434-0, 2007.

Louboutin, M., Lange, M., Luporini, F., Kukreja, N., Witte, P. A., Herrmann, F. J., Velesko, P., and Gorman, G. J.: Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration, Geosci. Model Dev., 12, 1165–1187, https://doi.org/10.5194/gmd-12-1165-2019, 2019.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A.: SymPy: symbolic computing in Python, PeerJ Computer Science, 3, e103, https://doi.org/10.7717/peerj-cs.103, 2017.

Mirouze, I. and Weaver, A. T.: Representation of correlation functions in variational assimilation using an implicit diffusion operator, Q. J. Roy. Meteor. Soc., 136, 1421–1443, https://doi.org/10.1002/qj.643, 2010.

Øksendal, B.: Stochastic Differential Equations, Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-14394-6, 2003.

Paciorek, C. and Schervish, M.: Nonstationary Covariance Functions for Gaussian Process Regression, Advances Neural Information Processing Systems, 16, 273–280, https://proceedings.neurips.cc/paper/2003/file/326a8c055c0d04f5b06544665d8bb3ea-Paper.pdf (last access: 22 September 2021), 2004.

Pannekoucke, O.: CAC-PKF-M (v0.1): Computer-aided calculation of PKF dynamics with Maxima, Zenodo [code], https://doi.org/10.5281/ZENODO.4708316, 2021a.

Pannekoucke, O.: An anisotropic formulation of the parametric Kalman filter assimilation, Tellus A, 73, 1–27, https://doi.org/10.1080/16000870.2021.1926660, 2021b.

Pannekoucke, O.: SymPKF: a symbolic and computational toolbox for the design of parametric Kalman filter dynamics, Zenodo [code], https://doi.org/10.5281/zenodo.4608514, 2021c.

Pannekoucke, O. and Fablet, R.: PDE-NetGen 1.0: from symbolic partial differential equation (PDE) representations of physical processes to trainable neural network representations, Geosci. Model Dev., 13, 3373–3382, https://doi.org/10.5194/gmd-13-3373-2020, 2020.

Pannekoucke, O. and Massart, S.: Estimation of the local diffusion tensor and normalization for heterogeneous correlation modelling using a diffusion equation, Q. J. Roy. Meteor. Soc.., 134, 1425–1438, https://doi.org/10.1002/qj.288, 2008.

Pannekoucke, O., Berre, L., and Desroziers, G.: Filtering properties of wavelets for local background-error correlations, Q. J. Roy. Meteor. Soc., 133, 363–379, 2007.

Pannekoucke, O., Ricci, S., Barthelemy, S., Ménard, R., and Thual, O.: Parametric Kalman Filter for chemical transport model, Tellus, 68, 31547, https://doi.org/10.3402/tellusa.v68.31547, 2016.

Pannekoucke, O., Bocquet, M., and Ménard, R.: Parametric covariance dynamics for the nonlinear diffusive Burgers equation, Nonlin. Processes Geophys., 25, 481–495, https://doi.org/10.5194/npg-25-481-2018, 2018a.

Pannekoucke, O., Ricci, S., Barthelemy, S., Ménard, R., and Thual., O.: Parametric Kalman filter for chemical transport models – Corrigendum, Tellus A, 70, 1–2, https://doi.org/10.1080/16000870.2018.1472954, 2018b.

Purser, R., Wu, W.-S., Parrish, D., and Roberts, N.: Numerical aspects of the application of recursive filters to variational statistical analysis. Part II: Spatially inhomogeneous and anisotropic general covariances, Mon. Weather Rev., 131, 1536–1548, https://doi.org/10.1175//2543.1, 2003.

Ricci, S., Weaver, A. T., Vialard, J., and Rogel, P.: Incorporating State-Dependent Temperature–Salinity Constraints in the Background Error Covariance of Variational Ocean Data Assimilation, Mon. Weather Rev., 133, 317–338, https://doi.org/10.1175/mwr2872.1, 2005.

Weaver, A. and Courtier, P.: Correlation modelling on the sphere using a generalized diffusion equation, Q. J. Roy. Meteor. Soc., 127, 1815–1846, https://doi.org/10.1002/qj.49712757518, 2001.