State Estimation of Partially Unknown Dynamical Systems with a Deep Kalman Filter

Erik Chinellato[1][0009−0007−4628−5581] and Fabio Marcuzzi[1][0000−0003−1757−3019]

Department of Mathematics "Tullio Levi Civita", University of Padova, Via Trieste 63, 35121 Padova, Italy marcuzzi@math.unipd.it

Abstract. In this paper we present a novel scientific machine learning reinterpretation of the well-known Kalman Filter, we explain its flexibility in dealing with partially-unknown models and show its effectiveness in a couple of situations where the classic Kalman Filter is problematic.

Keywords: Kalman Filter · scientific machine learning · deep unfolding · deep learning

1 Introduction

Often, in data-driven model discovery and in partially-unknown-model learning from experimental data, a limited set of measured variables is given and one aims at a model that describes a more complex process than that directly represented by measured variables. Typically, this means to include in the estimate an additional set of unmeasurable variables that give a more extended representation of the true system's dynamics. As an example, that we will use in this paper as a model problem, we measure some temperatures on the boundary but we want our model to represent also the behaviour of the internal temperature field. This is paradigmatic for a great number of computational intensive applications, where one needs to indirectly measure constants and variables that have a precise physical meaning (e.g. mechanical, thermal, etc.) from data obtained by measurements of other physical quantities (e.g. displacements, temperatures, etc.) or in different locations. This estimate represents a virtual measurement of these variables and, if performed by an algorithm in real-time, it is usually called a *soft-sensor* [3]. When these variables belong to the state-vector of a dynamical system, it is common to adopt a state estimator and quite often this is the Kalman Filter (KF) [7] [9]. It is a predictor-corrector algorithm where the predictor is based on a reference model, which can be black-box or conveniently a physico-mathematical model; the corrector acts on the basis of the description of the modelling and measurement errors, represented by their covariance matrices, and must operate a few matrix inversions. Actually, the Kalman Filter is often used also to estimate unknown inputs and/or parameters of the model; this is frequently done through a proper state-augmentation, see e.g. [12].

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

The aim of this paper is to present a state/parameters estimation method that inherits the structure of the Kalman Filter but it is blended with data-driven model discovery, and we call it the Deep Kalman Filter.

Let us see a couple of motivations that lead us to combine the Kalman Filter with machine learning, and in particular neural/back-propagation learning. First, we are interested in obtaining less demanding algorithms, suitable e.g. to be executed in real-time on embedded systems [3], while simultaneously maintaining a satisfactory interpretation of the physico-mathematical model properties and the inferred statistical description of the model errors, which is not the case with plain neural computing. In this regard, it can be convenient to substitute covariance matrices determination and numerical inversion with neural learning, an idea that has been already presented in [14] but here we formulate in a quite different way. Secondly, we strive to develop a state-estimation algorithm that allows to include general model uncertainties (also deterministic ones, even nonlinear) that can be learned from data together with state estimation, since both linearity of the reference state-space model and accurate knowledge of it are often not encountered in practice.

There is an increasing literature about combining Kalman filtering and neural networks. In particular, we refer to KalmanNet [14], where the overall information required to generate the Kalman gain is learned by a Recurrent Neural Network. Here we propose a novel, highly interpretable learning scheme for Kalman filtering inspired by the *unfolding* technique [8], originally invented in the context of nonnegative matrix factorizations. Similar approaches can be found also in the Data Assimilation (DA) context; in the taxonomy developed in a recent review [2], the Deep Kalman here presented is an end-to-end learning scheme for the whole DA system and, precisely, a "Sequential-DA-inspired neural scheme" that employs the *unfolding* technique instead of a Recurrent Neural Network, thus providing a higher degree of interpretability to KF operations.

The paper is organized as follows: in Sec. 2 we briefly recall the Kalman Filter equations. in sec. 3 we present the Deep Kalman Formulation and its flexibility to deal with partially unknown models. In sec. 4 we present a few relevant experiments and a Conclusions section ends the paper.

2 Kalman Filter estimation

Let us consider a parametric Discrete, Linear, Time-Invariant (DLTI) dynamical system in state-space form:

$$\begin{aligned} x(k+1) &= A(k,p)x(k) + B(k,p)u(k) + v(k) \\ y(k) &= Cx(k) + w(k) \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $y \in \mathbb{R}^m$ is the observation vector, and the dependence of A and B from p is in general nonlinear. This would arise, for example, from the discretization with an implicit method of a physico-mathematical model (see an example in sec. 4). To estimate the state vector from measurement

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

data it is a common choice to adopt a Kalman Filter, that we recall here in its one-step version [9]:

$$
\begin{aligned}
P(k) &= \left[\left(Q(k-1) + A(k-1,p)P(k-1)A(k-1,p)^\top\right)^{-1} + C^\top R^{-1} C\right]^{-1} && (2)\\
e_{pred} &= C\left(A(k-1,p)\ \hat{x}(k-1) + B(k,p)\ u(k-1)\right) - y(k) && (3)\\
\delta\hat{x}(k) &= -P(k)\ C^\top R^{-1} e_{pred} && (4)\\
\hat{x}(k) &= A(k-1,p)\ \hat{x}(k-1) + B(k-1,p)\ u(k-1) + \delta\hat{x}(k) && (5)
\end{aligned}
$$

From equation (4) it is evident that the KF operates a proportional feedback action on the output prediction error e_{pred}, with a gain:

$$
\mathcal{K}_G^{(k)} = -P(k)\ C^\top R^{-1} \tag{6}
$$

3 The Deep Kalman formulation

Let us generalize equations (3)-(4) for a nonlinear state-space system.
Given inputs $\{u(k)\}_{k=0,\dots,N-1}$, measurements $\{y(k)\}_{k=1,\dots,N}$, and an evolution map $f = f(x,p,u)$ depending on a state x, unknown dynamics vector p and input u, we have:

$$
\begin{aligned}
x(k+1) &= f\left(x(k),p,u(k)\right) + v(k)\\
y(k) &= Cx(k) + w(k)
\end{aligned} \tag{7}
$$

For an initial state estimate $\hat{x}(0)$, the forward run of the filter is thus given by:

$$
\hat{x}(k) = \hat{x}(k\,|\,k-1) + \mathcal{K}_G^{(k)}\left(y(k) - C\hat{x}(k\,|\,k-1)\right) \quad \forall k = 1,\dots,N \tag{8}
$$

where $\hat{x}(k\,|\,k-1) = f\left(\hat{x}(k-1), p^{(k)}, u(k)\right)$ is the k-th state prediction, C is the observation matrix, $\mathcal{K}_G^{(k)}$ is the k-th Kalman gain matrix and $p^{(k)} \equiv p$ is a shared vector modelling unknown dynamics.

Let us now *unfold* these equations and *untie* the Kalman gains. More precisely, the unfolding process considers the k-th state-update of equation (8) as the action of the k-th layer of a neural network. The untying then regards the Kalman gain matrices $\mathcal{K}_G^{(k)}$ as the network's weights, which are therefore no longer tied together by the recursive relation (2). The resulting scheme is shown in Figure 1: each grey rectangle represents one layer of the network, embedding equation (8). Note that this formulation is quite simpler than the one proposed in [14] and does not require the knowledge of the measured state vector at each time instant during the training phase, which is a hard constraint for many applications, but rather the only required quantity is the target final state $x(N)$. Also, there is no explicit learning of the initial condition $x(0)$, but the covariance of the initial-state estimation error $P(0)$ (see eqns. (6) and (2)) is learned and embedded directly in the first parameter matrix $\mathcal{K}_G^{(1)}$ during training.

The weights of the proposed network also include the vector $p^{(k)} \equiv p$.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

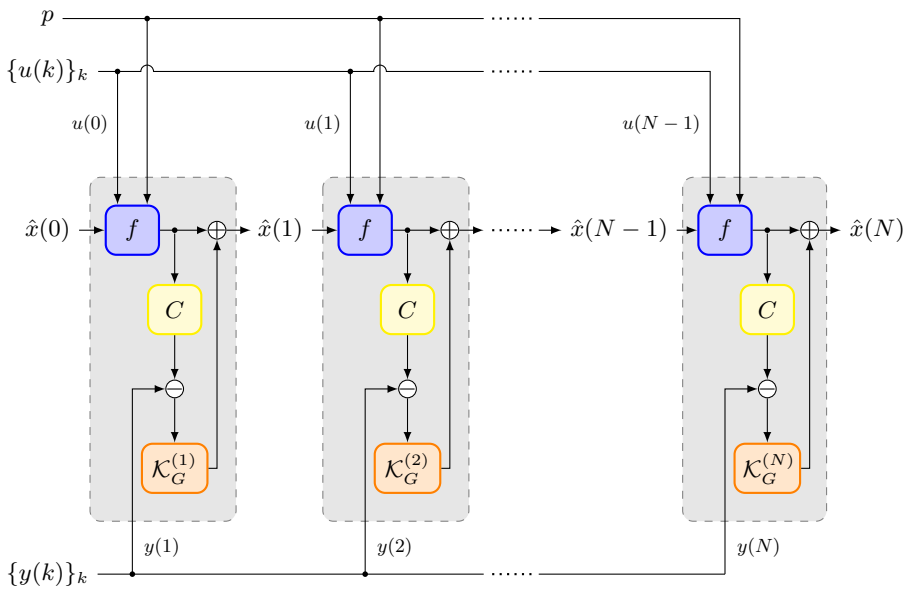

Fig. 1: Structure of the DeepKalman network.

3.1 Different parametrizations for $\mathcal{K}_G^{(k)}$

The most straightforward formulation for the parametrization of the gain matrices $\mathcal{K}_G^{(k)}$ is that each component is a weight of the net. Another possible network architecture that provides more structure to the Kalman gain matrices and is more closely related to the original Kalman Filter algorithm can be obtained by splitting the former as:

$$\mathcal{K}_G^{(k)} = C_x^{(k)} C^\top C_y^{(k)} \quad \forall k = 1, \dots, N \tag{9}$$

where $C_x^{(k)}$ plays the role of the state covariance matrix and $C_y^{(k)}$ the inverse of the measurement covariance matrix. Since the positive definiteness of these matrices should be preserved, this alternative architecture replaces the weights $\{\mathcal{K}_G^{(k)}\}_{k=1,\dots,N}$ with $\{L_x^{(k)}\}_{k=1,\dots,N}$ and $\{L_y^{(k)}\}_{k=1,\dots,N}$ where $L_x^{(k)}$, $L_y^{(k)}$ are lower triangular matrices satisfying:

$$C_x^{(k)} = L_x^{(k)} L_x^{(k)^\top} \qquad C_y^{(k)} = L_y^{(k)} L_y^{(k)^\top} \tag{10}$$

If $\mathcal{K}_G^{(k)} \in \mathbb{R}^{n \times m}$ and $p \in \mathbb{R}^q$, then the number of parameters in this second architecture increases from $nmK + q$ to $\frac{1}{2}\left[\, n(n+1) + m(m+1) \,\right] K + q$.

In both cases the initialization of the parameters can be made from the expression of $\mathcal{K}_G^{(k)}$ (6) and equation (2) (A is a linearization of f in this case). This is really effective, as we will show in the experiments of sec. 4, and provides satisfactory interpretability to the learned parameters $\mathcal{K}_G^{(k)}$.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

3.2 Backpropagation details

Let us now give some details on the backpropagation algorithm in the case of the architecture in equation (8). Given a desired final state $x(N)$ we consider the loss function:

$$\mathcal{L} = \frac{1}{2}\|\hat{x}(N) - x(N)\|_2^2 + \frac{\lambda}{2}\sum_{l=1}^{N}\|C\hat{x}(l) - y(l)\|_2^2 = \mathcal{E} + \sum_{l=1}^{N}\mathcal{F}_\lambda^l \qquad (11)$$

which controls both the reconstructed state as well as the intermediate measurements at each layer.
As a consequence, one obtains the recursive relations:

$$\begin{cases} \nabla_{\hat{x}(N)}\mathcal{E} &= \hat{x}(N) - x(N) \\ \nabla_{\hat{x}(l)}\mathcal{F}_\lambda^l &= \lambda C^\top\left(C\hat{x}(l) - y(l)\right) \quad \forall l = 1,\dots,N \\ \nabla_{\hat{x}(k)}\mathcal{E} &= \left[\left(\mathbb{1} - \mathcal{K}_G^{(k+1)}C\right)\partial_x f\left(\hat{x}(k), p^{(k+1)}, u(k+1)\right)\right]^\top \nabla_{\hat{x}(k+1)}\mathcal{E} \\ & \quad \forall k = 1,\dots,N-1 \\ \nabla_{\hat{x}(k)}\mathcal{F}_\lambda^l &= \left[\left(\mathbb{1} - \mathcal{K}_G^{(k+1)}C\right)\partial_x f\left(\hat{x}(k), p^{(k+1)}, u(k+1)\right)\right]^\top \nabla_{\hat{x}(k+1)}\mathcal{F}_\lambda^l \\ & \quad \forall k = 1,\dots,l-1, \quad \forall l = 1,\dots,N \end{cases}$$

with which to compute the partial gradients:

$$\begin{cases} \nabla_{\mathcal{K}_G^{(k)}}\mathcal{E} &= \nabla_{\hat{x}(k)}\mathcal{E}\left(y(k) - Cf\left(\hat{x}(k-1), p^{(k)}, u(k)\right)\right)^\top \\ & \quad \forall k = 1,\dots,N \\ \nabla_{\mathcal{K}_G^{(k)}}\mathcal{F}_\lambda^l &= \nabla_{\hat{x}(k)}\mathcal{F}_\lambda^l\left(y(k) - Cf\left(\hat{x}(k-1), p^{(k)}, u(k)\right)\right)^\top \\ & \quad \forall k = 1,\dots,l-1, \quad \forall l = 1,\dots,N \\ \nabla_{p^{(k)}}\mathcal{E} &= \left[\left(\mathbb{1} - \mathcal{K}_G^{(k)}C\right)\partial_p f\left(\hat{x}(k-1), p^{(k)}, u(k)\right)\right]^\top \nabla_{\hat{x}(k)}\mathcal{E} \\ & \quad \forall k = 1,\dots,N \\ \nabla_{p^{(k)}}\mathcal{F}_\lambda^l &= \left[\left(\mathbb{1} - \mathcal{K}_G^{(k)}C\right)\partial_p f\left(\hat{x}(k-1), p^{(k)}, u(k)\right)\right]^\top \nabla_{\hat{x}(k)}\mathcal{F}_\lambda^l \\ & \quad \forall k = 1,\dots,l-1, \quad \forall l = 1,\dots,N \end{cases}$$

Adding together all these contributions we obtain the overall gradients for our proposed architecture:

$$\begin{cases} \nabla_{\mathcal{K}_G^{(k)}}\mathcal{L} &= \nabla_{\mathcal{K}_G^{(k)}}\mathcal{E} + \sum_{l=1}^{N}\nabla_{\mathcal{K}_G^{(k)}}\mathcal{F}_\lambda^l \quad \forall k = 1,\dots,N \\ \nabla_p\mathcal{L} &= \sum_{k=1}^{N}\left(\nabla_{p^{(k)}}\mathcal{E} + \sum_{l=1}^{N}\nabla_{p^{(k)}}\mathcal{F}_\lambda^l\right) \end{cases}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

and the weight update relation for some learning rates $\mu_{\mathcal{K}}$, μ_p is given by:

$$\begin{cases} \mathcal{K}_G^{(k)} & \Leftarrow \quad \mathcal{K}_G^{(k)} - \mu_{\mathcal{K}} \nabla_{\mathcal{K}_G^{(k)}} \mathcal{L} \quad \forall k = 1, \dots, N \\ p & \Leftarrow \quad p - \mu_p \nabla_p \mathcal{L} \end{cases} \tag{12}$$

In practice, an optimizer is used in place of the above stochastic gradient descent. For our implementation we adopted Adam [10] with moment parameters β_1 and β_2 depending on the particular experiment being carried out. The learning rates $\mu_{\mathcal{K}}$ and μ_p also depend on the experiment and are adaptively reduced based on the training residue.

As far as the Kalman gains are concerned, we will expand on their initialization in a later section. The unknown dynamics vector p on the other hand is initialized according to its physical meaning in order to not introduce biases.

3.3 Alternative formulations and extensions for the map f

One of the main approaches used to define the map f of (7) is to employ a parametric model governed by differential equations and discretize it, as will be done in our model problem of sec. 4. Possibly, a surrogate model can be used, like e.g. a variationally mimetic operator network (VarMiON) [13].

Since the Kalman Filter operates a mere proportional control of the state-estimation error, there are classes of problems where this gives poor performances and, for example, an additional feed-forward term improves substantially the results of the predictor; see e.g. [12] for an adaptive feed-forward with a gain chosen according to the maximum principle for the heat equation; even this sophisticated term can be actually implemented in the map f of (7) without modifying the interpretation of the Deep Kalman formulation.

The map f can also be extended with a generic, parametric term to do data-driven model discovery during training, or even an ensemble of extensions. This can be done e.g. with the approach of sparse identification of nonlinear dynamics (SINDy) [1] and e-SINDy [5], i.e. to maintain the interpretability of the extension as an explicit function of the state variables. Simply, if the known part of the model is given by a linear system of differential equations like:

$$M\dot{x} \;=\; Kx + d \tag{13}$$

and a collection of states $X = [\, x(1) \cdots x(T) \,] \in \mathbb{R}^{n \times T}$ is provided (e.g. the states estimated by the Deep Kalman at each layer, so that $T = N$) together with their derivatives $\dot{X} = [\, \dot{x}(1) \cdots \dot{x}(T) \,] \in \mathbb{R}^{n \times T}$, which can be approximated numerically, we can build an appropriate (overcomplete) dictionary $\Phi(X) \in \mathbb{R}^{T \times L}$ and do sparse regression on:

$$\left(M\dot{X} - KX - D \right)^{\top} \;=\; \Phi(X) s \tag{14}$$

where $D = [\, d(1) \cdots d(T) \,] \in \mathbb{R}^{n \times T}$ and $s \in \mathbb{R}^{L \times n}$ is the sparse solution matrix. Finally, we can actually set the map f of the Deep Kalman Filter as the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

discretization of the data-driven extended model:

$$M\dot{x} \;=\; Kx + d + \left(\,\Phi(x)s\,\right)^{\top} \tag{15}$$

where now $\Phi(x) \in \mathbb{R}^{1\times L}$. Note that this approach can be applied to partially unknown nonlinear models as well.

In summary, if we lack an accurate model of the underlying dynamics and we want to improve it by learning from observations, we have briefly described how the Deep Kalman can be a framework embracing three core categories of unknown dynamics: (a) parametric dynamical models with unknown parameter values; (b) complex/unknown dynamics captured by interpretable surrogate models, e.g. implemented using neural networks; and (c) inaccurate or partially-known dynamical models that can be improved using data-driven extensions.

Finally, note that data-driven extensions can be physically interpretable even when they describe a fictitious term in the model, if it has been proven to be equivalent to a physical model property which is difficult to formulate or to estimate; see e.g. how the estimation of inner cavities (a nonlinear geometric inverse problem) can be reformulated as the estimation of fictitious heat sources (linear inverse problem) in [6].

4 Numerical experiments

As a model problem, let us consider the heat equation:

$$\begin{cases} \rho C\,\partial_t T \;=\; \kappa\,\Delta T + f_\vartheta & \text{in } D\times[0,t_f] \\ \kappa\,\nabla T\cdot\mathbf{n}_S \;=\; q(t) & \text{on } S\times[0,t_f] \\ \kappa\,\nabla T\cdot\mathbf{n} \;=\; 0 & \text{on } (\delta D\setminus S)\times[0,t_f] \\ T(0,\cdot) \;=\; T_0(\cdot) & \text{in } D \end{cases} \tag{16}$$

where $D(x,y)=[0,1]\times[0,L]$ is a 2-dimensional domain, $S=\{(x,0):\ x\in[0,1]\}$ is the measurable border, ρC is the heat capacity of the material, κ is its thermal conductivity and $\mathbf{n}_S$, $\mathbf{n}$ are the outward normal vectors to S, $\delta D\setminus S$ respectively. The restriction to a 2D problem is only for simplicity, the method here proposed can be used in higher dimensions.

Let us discretize problem (16) in space using the Finite Element Method (FEM) with Lagrangian elements P1, i.e. piecewise first-degree polynomials, and in time with the implicit Euler method, so that at iteration k we get:

$$\begin{aligned} M\,(\rho C)\,\frac{\tilde{T}_k-\tilde{T}_{k-1}}{dt} \;&=\; K(\kappa)\tilde{T}_k + f_k \\ \Rightarrow \left(\,\mathbb{1} - dt\;M\,(\rho C)^{-1}\,K(\kappa)\,\right)\tilde{T}_k \;&=\; \tilde{T}_{k-1} + dt\;M\,(\rho C)^{-1}\,f_k \end{aligned}$$

where $M(\rho C)\in R^{n\times n}$ and $K(\kappa)\in R^{n\times n}$ are the mass and stiffness matrices of the FEM discretization, dt is the time step chosen in the time-discretization and f_k is the heat source at time t_k.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

Since we can measure only a few components of the temperature field T, we reformulate this model as a state-space dynamical system with the aim of estimating its state from output measurements. Let us consider the following state-space discrete model in physical coordinates:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u_m(k) + v_m(k) \\ y_m(k) &= C_m x_m(k) + w(k) \end{aligned} \tag{17}$$

where $x_m(k) = \tilde{T}_k$ is the state-vector, $u_m(k) = f_k$ the input, $v_m(k)$ and $w(k)$ are, respectively, the model and measurement errors, supposed i.i.d. Gaussian processes, C_m is a matrix built with the rows of the identity matrix corresponding to measured nodes, and lastly:

$$\begin{aligned} A_m &= \left(\mathbb{1} - dtM\,(\rho C)^{-1} K(\kappa) \right)^{-1} \\ B_m &= A_m dtM\,(\rho C)^{-1} \end{aligned}$$

Now, to estimate the state vector we apply equations (2)-(5).

In this section, some numerical experiments are described to give a practical evidence of the algorithmic ideas previously presented. In all the examples, experimental temperatures are simulated numerically, while intensive tests have been done in our previous work to experimentally validate the model settings (see [4]). A note about "inverse crimes": here we are solving the inverse problem using the same model that has generated the data, which is considered an inverse crime. Actually, we are interested in analyzing the algebraic operations made during the reconstruction, which are non trivial, and the simplified setting we use is adequate to make significant comparisons, see the interesting discussion in the white-paper of Wirgin [15]. In a real, specific application one should then consider also model and measurement errors, to validate the practical accuracy of his method in the specific application. In the next subsection we provide an example of the robustness of Deep Kalman to model noise.

Let us describe the model settings: $t_f = 1.51\,\mathrm{s}$, $L = 0.1\,\mathrm{m}$, $\rho C = 3.2 \times 10^6\,\mathrm{J/(m^3\,{}^\circ C)}$, $k = 3.77 \times 10^3\,\mathrm{W/(m\,{}^\circ C)}$, and

$$q(t) = \frac{Wt}{\sigma_q^2} e^{-\frac{\sqrt{t}}{\sigma_q}},\ t \in (0, t_f] \tag{18}$$

where $\sigma_q = 1.06 \times 10^{-2}$, $W = 2.9511 \times 10^7\,\mathrm{J}$. The initial condition is set to $T_0(\cdot) = 20\,{}^\circ\mathrm{C}$. In this section an Implicit Euler method is adopted for the time discretization, using a temporal step $\Delta t = 0.0005\,\mathrm{s}$ in $(0, 0.1]$ and $\Delta t = 0.05\,\mathrm{s}$ in $(0.1, t_f]$. A P_1-FE method is used for space discretization, whose step length along y is $h_y = 0.01\,\mathrm{m}$ and $h_x = h_y$. The sensors are supposed to be in the middle of each mesh edge in the instrumented boundary segment. Numerical experiments have been carried out using MATLAB. As a general forcing term we have used a gaussian forcing term f_ϑ with arbitrary variance and point of application.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

In the following subsections we see a couple of relevant parametrizations for just as many inverse heat transfer problems.

4.1 Example: discovering a distributed forcing term

Let us suppose that the heat source term f_ϑ is an unknown function, in general, except that it is assumed different from zero only in a few disconnected regions of compact support. This is a common situation in many applications. In this example, we estimate f_ϑ from a limited number of temperature measurements $\tilde{T}_{(f_\vartheta)}$, taken at the boundary. The estimate of f_ϑ can be seen as an indirect measurement of f_ϑ from physical/direct temperature measurements and, with this interpretation, it is usually called a *soft-sensor* [3]. This is quite a difficult problem for Kalman filtering and, more precisely, for the Augmented Kalman Filter (AKF): see [12] and Figure 2, where it is clear that the standard AKF misses completely the right location, shape and intensity of the physical forcing term.

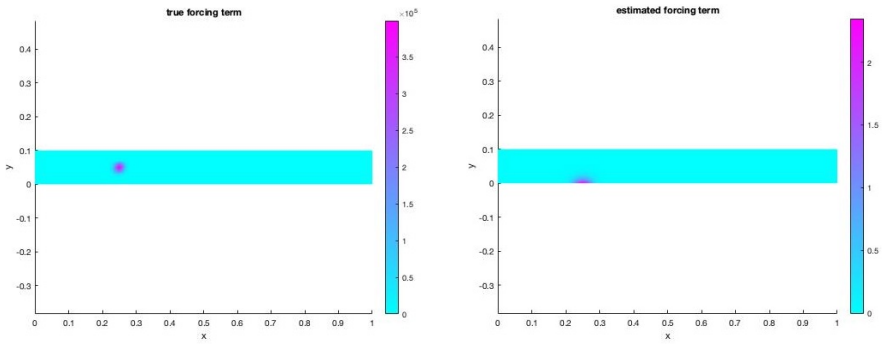

Fig. 2: Comparison between the true forcing term (left) and estimated (right) by the traditional (augmented) Kalman Filter algorithm.

With the method here presented, we set the unknown-dynamics vector of parameters p of sec. 3 to $u_m(k)$ in the reference model (17) and $W = 0$ in (18). Moreover, the training/discovery process was carried out using $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\mu_{\mathcal{K}} = 5 \cdot 10^{-3}$, $\mu_p = 6 \cdot 10^{-3}$. We performed this experiment in the ideal noiseless setting as well as in the presence of model noise significantly corrupting the measured nodes' temperatures. The added model noise is assumed to be an i.i.d. Gaussian process with zero mean and variance $\sigma^2 = 1 \cdot 10^{-4}$. See Figure 3 and Figure 4 respectively for the results. In both cases the Deep Kalman algorithm is able to estimate the forcing term with high accuracy. Note that in the case with model noise, despite the relatively high output-prediction error, the Deep Kalman is able to estimate with a good precision the internal temperature field.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

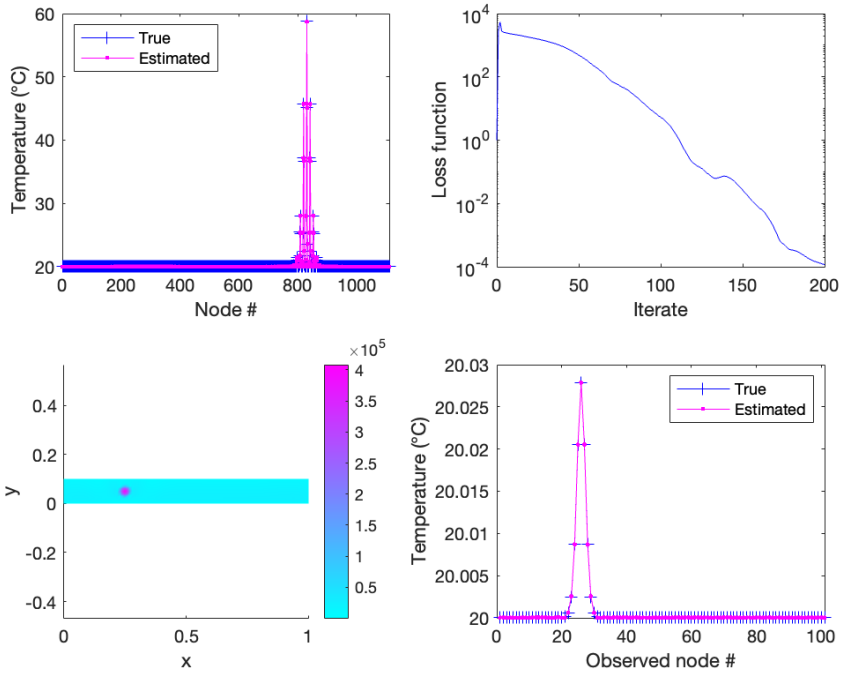

Fig. 3: Forcing term estimation in absence of noise. Nodal values of the temperature field T at time $T_k = N$ (top-left). Loss-function (11) minimization (top-right). Rectangular domain with the estimated source term f_ϑ (bottom-left). Nodal temperatures at the measured boundary S (bottom-right).

4.2 Example: discovering internal material properties

Let us suppose that the ρC, the heat capacity of the material, is an unknown function of (x, y) and, in particular, it is constant along x. This is an interesting situation in applications. In this example, we estimate $(\rho C)(y)$ from a limited number of temperature measurements $\bar{T}_{(\rho C)}$, taken at the boundary. It is another example of *soft-sensor* [3] and a quite difficult problem for Kalman filtering, see in [11] an analogous application to the estimation of the internal material stiffness through mechanical vibration experiments.

With the method here presented, we set the unknown-dynamics vector of parameters p of sec. 3 to the values of a piecewise constant approximation of $(\rho C)(y)$, defined according to the previously assumed discretization of the domain. Moreover, the training/discovery process was carried out using $\beta_1 = 0.05$, $\beta_2 = 0.059$ and $\mu_\mathcal{K} = 5 \cdot 10^{-3}$, $\mu_p = 1 \cdot 10^{-2}$. The Deep Kalman algorithm is able to estimate these parameters p with high accuracy, see Figure 5.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

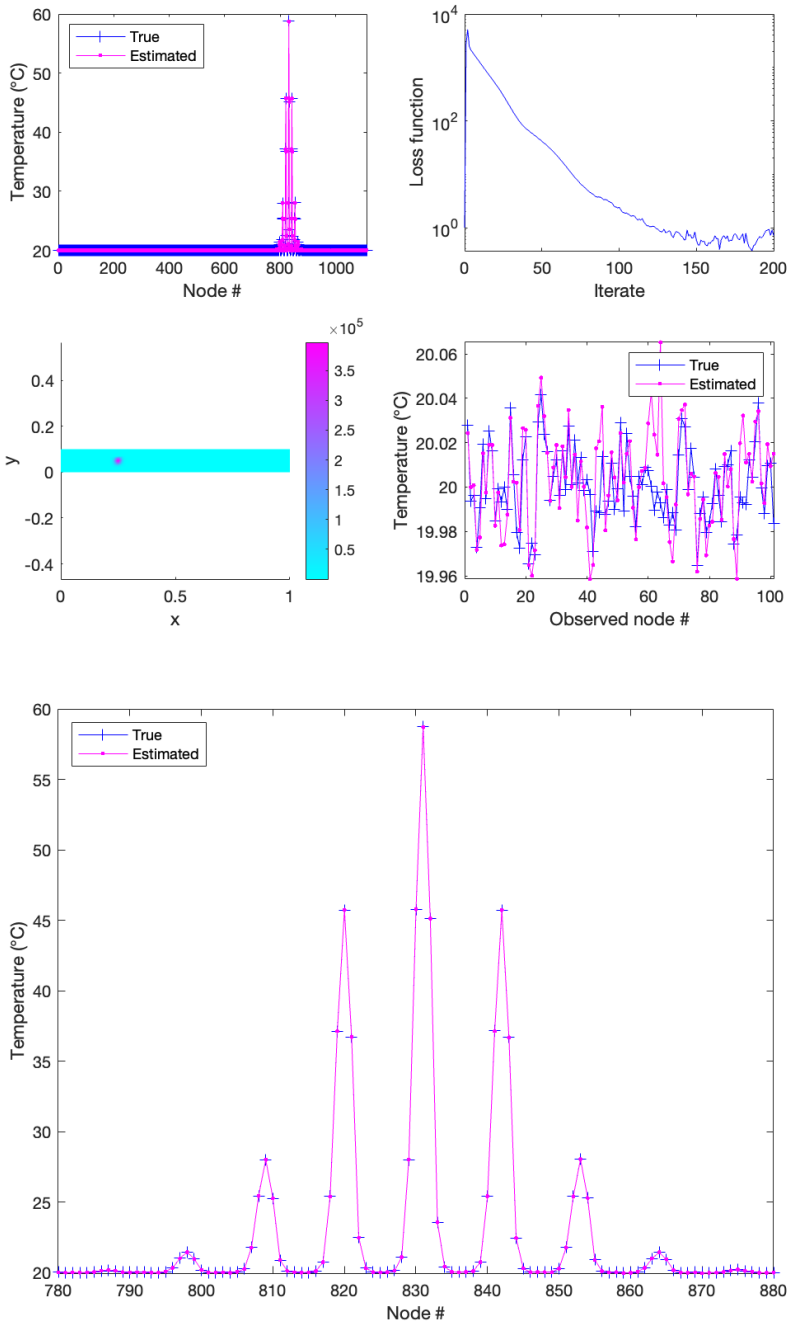

Fig. 4: Forcing term estimation in presence of model noise. Nodal values of the temperature field T at time $T_k = N$ (top-left). Loss-function (11) minimization (top-right). Rectangular domain with the estimated source term f_ϑ (center-left). Nodal temperatures at the measured boundary S (center-right). Zoom on the top-left figure from node 780 to 880 (bottom).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

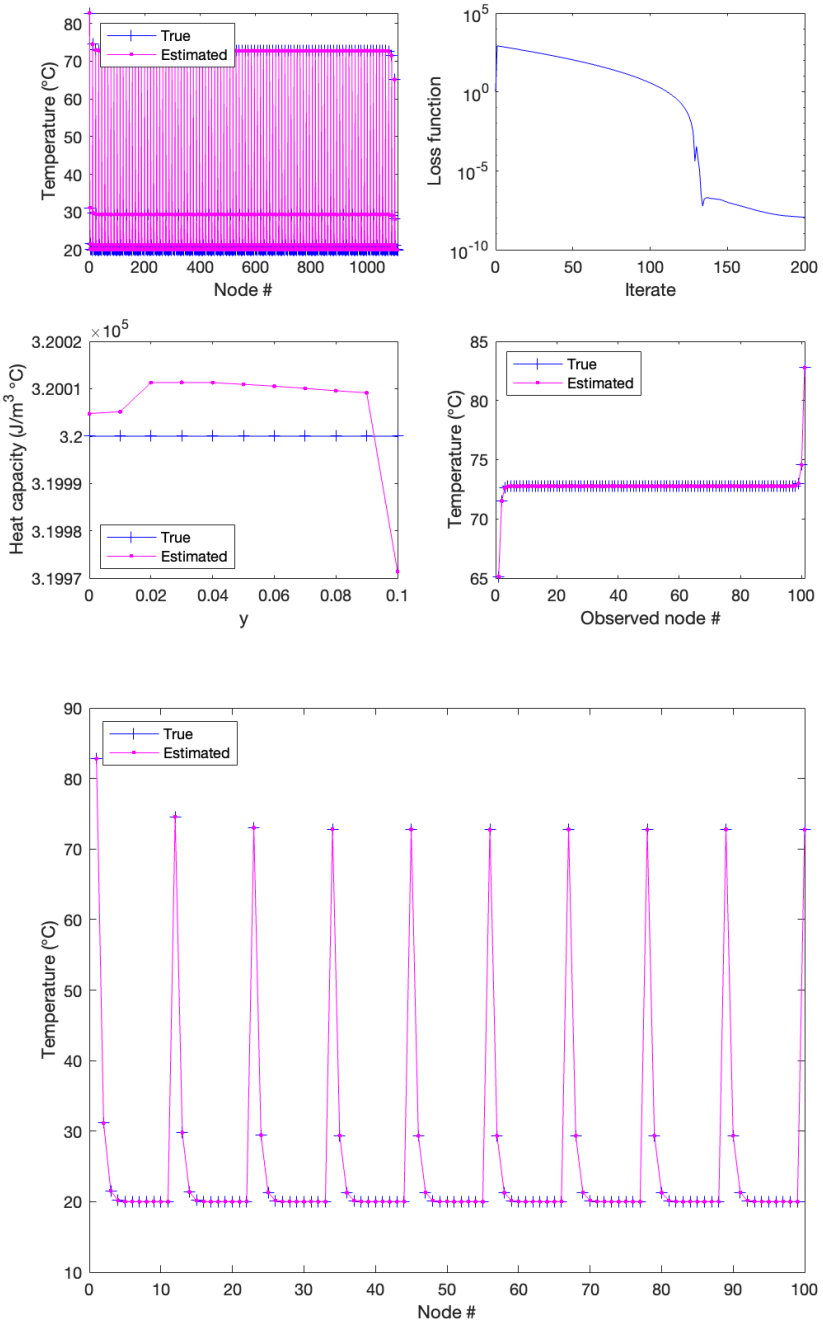

Fig. 5: Heat capacity estimation. Nodal values of the temperature field T at time $T_k = N$ (top-left). Loss-function (11) minimization (top-right). Rectangular domain with the estimated heat capacity $(\rho C)(y)$ for $y = 0, 0.01, 0.02, \ldots, 0.1$ (center-left). Nodal temperatures at the measured boundary S (center-right). Zoom on the top-left figure from node 1 to 100 (bottom).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

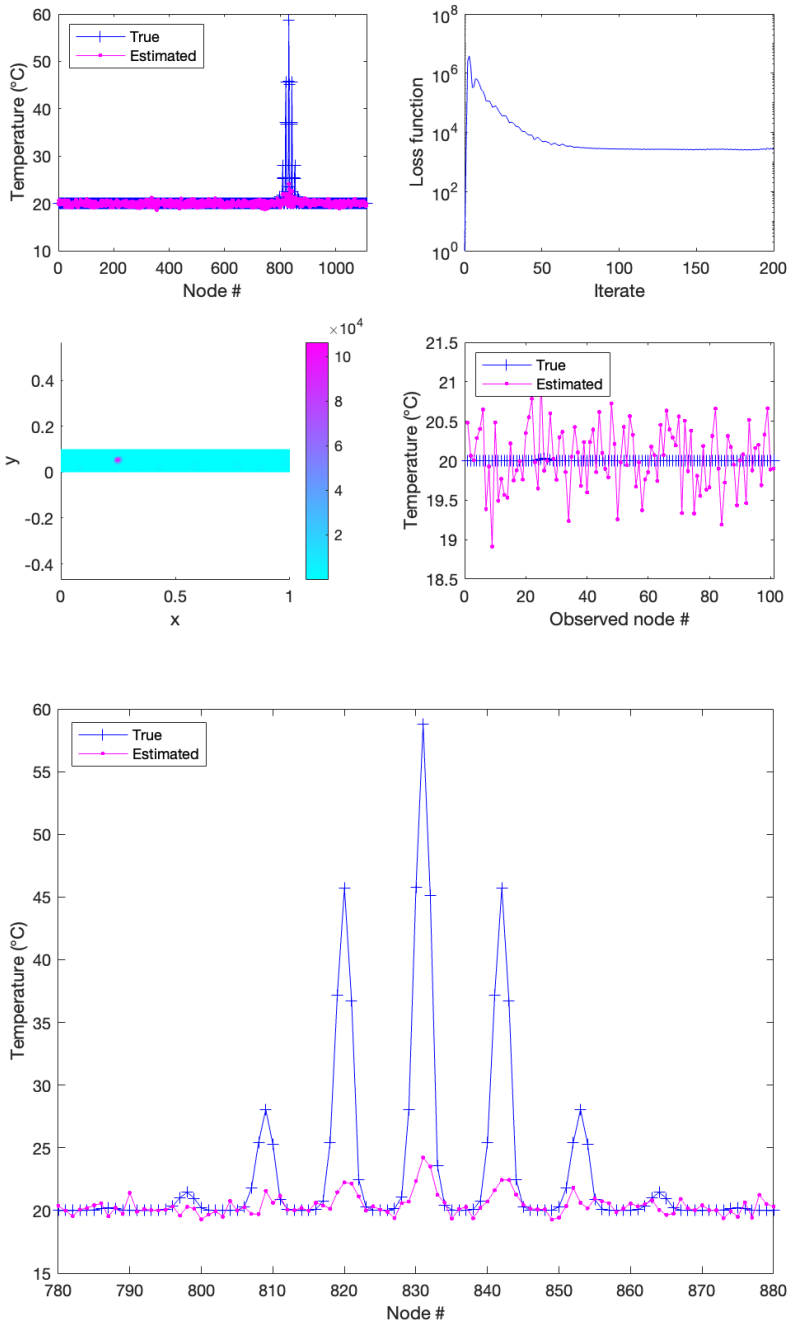

Fig. 6: Forcing term estimation in absence of noise with a random initialization of the parameters $\mathcal{K}_G^{(k)}$. Nodal values of the temperature field T at time $T_k = N$ (top-left). Loss-function (11) minimization (top-right). Rectangular domain with the estimated source term f_ϑ (center-left). Nodal temperatures at the measured boundary S (center-right). Zoom on the top-left figure from node 780 to 880 (bottom).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

4.3 Initialization strategy for parameters $\mathcal{K}_G^{(k)}$

In both previous examples, the initialization of the parameters $\mathcal{K}_G^{(k)}$ in the Deep Kalman learning algorithm was made by choosing reasonable values for the matrices Q, $P(0)$ and R, and precisely those dictated by standard rules to tune the KF [7]. Then, we have set $\mathcal{K}_G^{(k)} = P(1)$, where $P(1)$ is given by (2) with $k = 1$.

In Figure 6 we repeat the experiment of sec. 4.1 but initializing the parameters $\mathcal{K}_G^{(k)}$ with small random values, as is usually done in neural networks learning. We see that the result is much worse, if compared with Figure 3. We can make two considerations: first, the Deep Kalman allows to initialize the weights in a very interpretable way and, in particular, it allows to initialize the weights according to the standard rules adopted for the KF; second, if we do not put a good apriori structure in this initialization, the results are not good.

5 Discussion and Conclusions

We have presented Deep Kalman, a novel scientific machine learning reinterpretation of the well-known Kalman Filter, and shown its effectiveness in a couple of situations where the classic Kalman Filter is problematic.

In this machine learning formulation, we kept the predictor model in its explicit original formulation, usually expressed by a physico-mathematical model; this fact allows to ask for the identifiability of the parametric model learned by the Deep Kalman, i.e. the one-to-one correspondence between the input-output data used in the supervised learning and the value of the learned parameters. This is typical for system identification but not attainable in pure neural learning.

With respect to execution speed, under the assumption that the gains $\mathcal{K}_G^{(k)}$ are entirely given a priori, the Deep Kalman and traditional Kalman Filter are very similar. Indeed, a forward run of both algorithms performs the same algebraic operations. Nevertheless, the difference between the two becomes relevant in the case of partially unknown models, which is the main focus of this contribution. As a matter of fact, in this setting, the traditional Kalman Filter must update the gains at run-time to compensate for the unmodeled dynamics, which involves matrix inversions. The Deep Kalman on the other hand, by virtue of its ability to learn these unknown dynamics during the training phase and the prospect to differentiate the learning of the gains $\mathcal{K}_G^{(k)}$ and the vector p, can offer a potentially non-negligible speed-up if compared to the former algorithm. Moreover, this efficiency in determining the gains for partially known models may contribute to relevant memory savings, especially in comparison e.g. to the demand of employing a filter-bank, a common option for the traditional Kalman Filter, in this case.

The speed-up and memory saving are important when the Deep Kalman must run in real-time and on computers with limited resources, like microcontrollers for embedded applications.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22

References

1. Brunton, S.L., Budisic, M., Kaiser, E., Kutz, J.N.: Modern koopman theory for dynamical systems. SIAM Review **64**(2), 229–340 (2022). https://doi.org/10.1137/21M1401243
2. Cheng, S., et al.: Machine learning with data assimilation and uncertainty quantification for dynamical systems: A review. IEEE/CAA Journal of Automatica Sinica **10** (2023). https://doi.org/10.1109/JAS.2023.123537
3. Chinellato, E., Marcuzzi, F., Pierobon, S.: Physics-aware soft-sensors for embedded digital twins. Springer LNNS, in press (2024)
4. Dessole, M., Marcuzzi, F.: Accurate detection of hidden material changes as fictitious heat sources. Numerical Heat Transfer, Part B: Fundamentals (2023). https://doi.org/10.1080/10407790.2023.2220905
5. Fasel U., Kutz J. N., B.B.W., L., B.S.: Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. Proc. R. Soc. A. (2022). https://doi.org/10.1098/rspa.2021.0904
6. Giusteri, G.G., Marcuzzi, F., Rinaldi, L.: Replacing voids and localized parameter changes with fictitious forcing terms in boundary-value problems. Results in Applied Mathematics **20** (2023). https://doi.org/10.1016/j.rinam.2023.100402
7. Grewal, M.S., Andrews, A.P.: Kalman Filtering: Theory and Practice with MATLAB®: Fourth Edition, vol. 9781118851210. John Wiley & Sons, Inc. (2014). https://doi.org/10.1002/9781118984987
8. Hershey, J., Le Roux, J., Weninger, F.: Deep unfolding: Model-based inspiration of novel deep architectures. arXiv (09 2014)
9. Humpherys, J., Redd, P., West, J.: A fresh look at the kalman filter. SIAM Review **54**(4), 801–823 (2012). https://doi.org/10.1137/100799666
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2014)
11. Marcuzzi, F.: Space and time localization for the estimation of distributed parameters in a finite element model. Computer Methods in Applied Mechanics and Engineering **198**(37), 3020 – 3025 (2009). https://doi.org/10.1016/j.cma.2009.05.007
12. Marcuzzi, F.: A numerical feed-forward scheme for the augmented kalman filter. Springer LNCS, in press (2024). https://doi.org/https://doi.org/10.
13. Patel, D., Ray, D., Abdelmalik, M.R., Hughes, T.J., Oberai, A.A.: Variationally mimetic operator networks. Computer Methods in Applied Mechanics and Engineering **419** (2024). https://doi.org/10.1016/j.cma.2023.116536
14. Revach, G., Shlezinger, N., Ni, X., Escoriza, A., Van Sloun, R.J.G., Eldar, Y.C.: Kalmannet: Neural network aided kalman filtering for partially known dynamics. IEEE Transactions on Signal Processing **70**, 1532–1547 (2022). https://doi.org/10.1109/TSP.2022.3158588
15. Wirgin, A.: The inverse crime (2004). https://doi.org/10.48550/ARXIV.MATH-PH/0401050

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_22