applied sciences

MDPI

Article

Deep Data Assimilation: Integrating Deep Learning with Data Assimilation

Rossella Arcucci [1,*], Jiangcheng Zhu [2], Shuang Hu [3] and Yi-Ke Guo [1,4]

[1] Data Science Institute, Imperial College London, London SW72AZ, UK; y.guo@imperial.ac.uk
[2] State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China; jiangchengzhu@zju.edu.cn
[3] Ningbo Joynext Technology Inc., Ningbo 315000, China; shuang.hu@joynext.com
[4] Department of Computer Science, Hong Kong Baptist University, Hong Kong
* Correspondence: r.arcucci@imperial.ac.uk

Abstract: In this paper, we propose Deep Data Assimilation (DDA), an integration of Data Assimilation (DA) with Machine Learning (ML). DA is the Bayesian approximation of the true state of some physical system at a given time by combining time-distributed observations with a dynamic model in an optimal way. We use a ML model in order to learn the assimilation process. In particular, a recurrent neural network, trained with the state of the dynamical system and the results of the DA process, is applied for this purpose. At each iteration, we learn a function that accumulates the misfit between the results of the forecasting model and the results of the DA. Subsequently, we compose this function with the dynamic model. This resulting composition is a dynamic model that includes the features of the DA process and that can be used for future prediction without the necessity of the DA. In fact, we prove that the DDA approach implies a reduction of the model error, which decreases at each iteration; this is achieved thanks to the use of DA in the training process. DDA is very useful in that cases when observations are not available for some time steps and DA cannot be applied to reduce the model error. The effectiveness of this method is validated by examples and a sensitivity study. In this paper, the DDA technology is applied to two different applications: the Double integral mass dot system and the Lorenz system. However, the algorithm and numerical methods that are proposed in this work can be applied to other physics problems that involve other equations and/or state variables.

Keywords: data assimilation; deep learning; neural network

Citation: Arcucci, R.; Zhu, J.; Hu, S.; Guo, Y.-K. Deep Data Assimilation: Integrating Deep Learning with Data Assimilation. *Appl. Sci.* **2021**, *11*, 1114. https://doi.org/10.3390/app11031114

Academic Editor: João M. F. Rodrigues
Received: 8 December 2020
Accepted: 21 January 2021
Published: 26 January 2021

1. Introduction and Motivations

The present work is placed in the context of the design of reliable algorithms for solving Data Assimilation (DA) for dynamic systems applications. DA is an uncertainty quantification technique by which measurements and model predictions are combined in order to obtain an accurate estimation of the state of the modelled system [1]. In the past 20 years, DA methodologies, used, in principle, mainly in atmospheric models, has become a main component in the development and validation of mathematical models in meteorology, climatology, geophysics, geology, and hydrology (often these models are referred to with the term predictive to underline that these are dynamical systems). Recently, DA has also been applied to numerical simulations of geophysical applications [2], medicine, and biological science [3,4] for improving the reliability of the numerical simulations. In practice, DA provides the answers to questions, such as: how well does the model under consideration represent the physical phenomena? What confidence can one have that the numerical results produced by the model are correct? How far can the calculated results be extrapolated? How can the predictability and/or extrapolation limits be extended and/or improved?

Accuracy in numerical models of dynamic systems does not come easily. The simulation errors come from three main sources: inaccurate input data, inaccurate physics models, and limited accuracy of the numerical solutions of the governing equations. Our ability to predict any physical phenomenon is determined by the accuracy of our input data and our modelling approach [5]. Errors arise at the different stages of the solution process, namely, the uncertainty in the mathematical model, in the model's solution, and in the measurements. These are the errors intrinsic to the DA problem. Moreover, there are the approximation errors that are introduced by the linearization, the discretization, and the model reduction. These errors incur when infinite-dimensional equations are replaced by a finite-dimensional system (that is, the process of discretization), or when simpler approximations to the equations are developed (e.g., by model reduction). Finally, given the numerical problem, an algorithm is developed and implemented as a mathematical software. At this stage, the inevitable rounding errors introduced by working in finite-precision arithmetic occur. These approaches are unable to fully overcome their unrealistic assumptions, particularly linearity, normality, and zero error covariances, despite the improvements in the complexity of the DA models to better match their application requirements and circumvent their implementation problems [6].

With the rapid developments in recent years, DL shows great capability in approximating nonlinear systems, and extracting high dimensional features. As the foundation and main driving force of deep learning, DNN is concerned less about numerical modelling. DNN took a data driven approach, where the models are built by learning from data. Instead of being deterministic [7,8], DL models such as NN are stochastic. Thus, they can well succeed when applied to deterministic systems, but without ever learning the relationship between the variables. For e.g., ML algorithms do not know when they violate the physics laws of [9,10] in weather forecasting. Before they begin to produce accurate results, machine learning methods often need large quantities of data, and, the bigger the architecture, the more data are required. In a wide variety of applications, DL is successfully used when the conditions allow. However, in cases where the dimensions are either very large, the data are noisy or the data do not cover the entire domain adequately; these approaches still fail. Furthermore, the network will not perform well if the available information is too noisy, too scarce, or if there is a lack of prominent features to reflect the problem. This can occur either if there is a lack of good characteristics or a lack of data on good ground reality. For this reason, a good quality of the data (smaller errors) can help to generate a more reliable DNN. DA is the Bayesian approximation of the true state of some physical system at a given time by combining time-distributed observations with a dynamic model in an optimal way. In some sense, DA increases the meaningfulness of the data and reduces the forecasting errors. Therefore, it is interesting to investigate the mechanism where two data driven paradigms, DA and DNN, can be integrated.

In this context, we developed the Deep Data Assimilation (DDA) model. DDA is a new concept that combines the DNN and DA. It faces the problem to reduce both input data error (by DA) and modelling error (by ML).

Related Works and Contribution of the Present Work

This paper is placed in the imperfect models and sensitivity analysis context for data assimilation and deep neural network. Sensitivity Analysis (SA) refers to the determination of the contributions of individual uncertainty on data to the uncertainty in the solution [11]. The sensitivity of the variational DA models has been studied in [12], where an Adjoint modeling is used in order to obtain first and second-order derivative information. In [13], sensitivity analysis is based on the Backward Error Analysis (B.E.A.), which figure out how much the errors propagation in DA process depend on the condition number of the background error covariance matrices. Reduced-order approaches are formulated in [12] in order to alleviate the computational cost that is associated with the sensitivity estimation. This method makes rerunning less expensive, the parameters must still be selected a priori and, consequently, important sensitivities may be missed [14]. For imperfect models, weak

constraint DA (WCDA) can be employed [15], in which the model error term in the covariance evolution equation acts to reduce the dependence of the estimate on observations and prior states that are well separated in time. However, the time complexity of the WCDA models is a big limit that makes these models often unusable.

ML algorithms are capable of assisting or replacing, without the assumptions of traditional methods, the aforementioned conventional methods in assimilating data, and producing forecasts. Because any linear or nonlinear functions can be approximated by neural networks, DA has been integrated as a supplement in various applications. The technology and the model presented in this paper are very general and they can be applied to any kind of dynamical systems. The use of ML in correcting model forecasts is promising in several geophysics applications [16,17]. Babovic [18–20] applies neural network for error correction in forecasting. However, in this literature, the error correction neural network has not a direct relation with the system update model in each step and it does not train the results of the assimilation process. In [21], DA and ML are also integrated and a surrogate model is used in order to replace the dynamical system. In this paper we do not replace the the dynamical system with a surrogate models, as this would add approximation errors. In fact, in real case scenarios (i.e., in some operational centres), data driven models are welcome to support Computational Fluid Dynamics simulations [9,10,22], but the systems of Partial Differential Equations that are stably implemented to predict (weather, climate, ocean, air pollution, et al.) dynamics are not replaced due to the big approximations that this replacement would introduce. The technology that we propose in this paper integrates DA with ML in a way that can be used in real case scenarios for real world applications without changing the already implemented dynamical systems.

Other works have already explored the relationship between the learning, analysis, and implementation of data in ML and DA. These studies have concentrated on fusing methods from the two fields instead of taking a modular approach like the one implemented in this paper, thereby developing methods that are unique to those approaches. The integration of the DA and ML frameworks from a Bayesian perspective illustrates the interface between probabilistic ML methods and differential equations. The equivalence is formally presented in [23] and it illustrates the parallels between the two fields. Here, the equivalence of four-dimensional VarDA (4D-Var) and Recurrent NN, and how approximate Bayesian inverse methods (i.e., VarDA in DA and back-propagation in ML) can be used to merge the two fields. These methods are also especially suited to systems involving Gaussian process, as presented in [24–26]. These are developed data-driven algorithms that are capable, under a unified approach, of learning nonlinear, space-dependent cross-correlations, and of estimating model statistics. A DA method can be used via the DL framework to help the dynamic model (in this case, the ML model) to find optimal initial conditions. Although DA can be computationally costly, the recent dimensional reduction developments using ML substantially reduce this expense, while retaining much of the variance of the original [27] model. Therefore, DA helps to develop the ML model, while [28,29] also benefits from the ML techniques. Methods that merge DA models, such as Kalman filters and ML, exist to overcome noise or to integrate time series knowledge. The authors account for temporal details in human motion analysis in [30] by incorporating a Kalman filter into a neural network of LSTM. In [31], the authors suggest a methodology that combines NN and DA for model error correction. Instead, in [32], the authors use DA, in particular Kalman tracking, to speed up any learning-based motion tracking method to real-time and to correct some common inconsistencies in motion tracking methods that are based on the camera. Finally, in [33], the authors introduce a new neural network for speed and replace the whole DA process.

In this paper, we proposed a new concept, called DDA, which combines the DL into the conventional DA. In recent years, DL gained great success both in academic and industrial areas. In function approximation, which has unknown model and high nonlinearity, it shows great benefit. Here, we use DNN to describe the model uncertainty and the assimilation process. In an end-to-end approach, the neural networks are introduced and

their parameters are iteratively modified by applying the DA methods with upcoming observations. The resulting DNN model includes the features of the DA process. We prove that this implies a reduction of the model error, which decreases at each iteration.

We also prove that the DDA approach introduces an improvement in both DA and DNN. We introduce a sensitivity analysis that is based on the backward error analysis to compare the error in DDA result and the error in DA. We prove that the error in the results obtained by DDA is reduced with respect the DA. We also prove that the use of the results of DA to train the DL model introduces a novelty in the SA techniques used to evaluate which inputs are important in NN. The implemented approach can be compared to the 'Weights' Method [34]. The distinction from the classical 'Weights' method is that the weights between the nodes are given in our approach by the error covariance matrices that are determined in the DA phase.

In summary, we prove that the DDA approach

- includes the features of the DA process into the DNN model with a consequent reduction of the model error;
- introduces a reduction in the overall error in the assimilation solution with respect to the DA solution; and,
- increases the reliability of the DNN model, including the error covariance matrices as weight in the loss function.

This paper is structured, as follows. Section 2 provides preliminaries on DA. We proposed the DDA methodology in Section 3, where we introduce the integration of deep learning with data assimilation. In Section 4, numerical examples are provided in order to implement and verity the DDA algorithm and, in Section 5, conclusions and future work are summarized.

2. Data Assimilation Process

Let

$$\dot{u} = \mathcal{M}(u, t, \theta) \tag{1}$$

be a forecasting model, where u denotes the state, $\mathcal{M}$ is a nonlinear map, θ is a parameter of interest, and $t \in [0, T]$ denotes the time. Let

$$v = \mathcal{H}(u) + \epsilon \tag{2}$$

be an observation of the state u, where $\mathcal{H}$ is an observation function and ϵ denotes the measurement error.

Data Assimilation is concerned with how to use the observations in (2) and the model in (1) in order to obtain the best possible knowledge of the system as a function of time.The Bayes theorem conducts the estimation of a function u^{DA}, which maximizes a probability density function, given the observation v and a prior from u [1,35].

For a fixed a time step $t_k \in [0, T]$, let u_k denote the estimated system state at time t_k:

$$u_k = M\, u_{k-1} \tag{3}$$

where the operator M represents a discretization of a first order approximation of $\mathcal{M}$ in (1). Let v_k be an observation of the state at time t_k and let

$$H : u_k \mapsto v_k. \tag{4}$$

be the discretization of a first order approximation of the linearization of $\mathcal{H}$ in (2). The DA process consists in finding u^{DA} (called analysis) as an optimal tradeoff between the prediction made based on the estimated system state (called background) and the available observation. The state u is then replaced by the function u^{DA}, which includes the information from the observation v in a prediction-correction cycle that is shown in Figure 1.

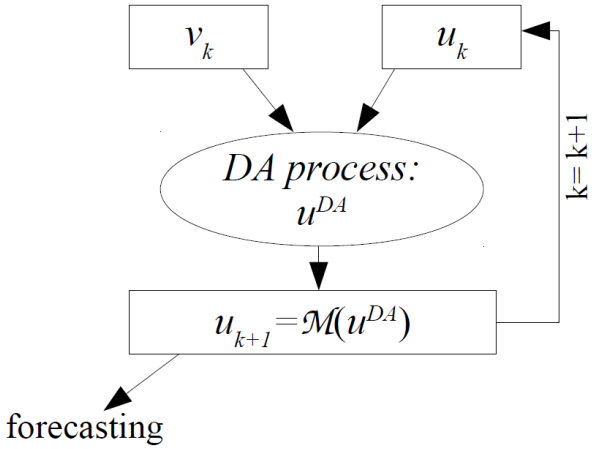


Figure 1. The prediction-correction cycle.

The analysis u^{DA} is computed as inverse solution of

$$v_k = Hu^{DA} + e_{R_k}, \tag{5}$$

subject to the constraint that

$$u^{DA} = u_k + e_{B_k}. \tag{6}$$

where $e_{R_k} = \mathcal{N}(\mu_R, R_k)$ and $e_{B_k} = \mathcal{N}(\mu_B, B_k)$ denote the observation error and model error, respectively, $\mathcal{N}(\cdot)$ denotes Gaussian distribution [1], and e_{R_k} includes the measurement error ϵ introduced in (2). Both of the errors include the discretization, approximation and the other numerical errors.

Because H is typically rank deficient, the (5) is an ill posed inverse problem [36,37]. The Tikhonov formulation [38] leads to an unconstrained least square problem, where the term in (6) ensures the existence of a unique solution of the (5). The DA process can be then described in Algorithm 1, where:

$$u^{DA} = argmin_u \left\{ \|u - u_k\|^2_{B_k^{-1}} + \|v_k - Hu\|^2_{R_k{}^{-1}} \right\} \tag{7}$$

where R_k and B_k are the observation and model error covariance matrices, respectively:

$$R_k := \sigma_0^2 I, \tag{8}$$

with $0 \leq \sigma_0^2 \leq 1$ and I be the identity matrix,

$$B_k = V_k V_k^T \tag{9}$$

with V_k background error deviation matrix [13] being computed by a set of n_k historical data $S = \{u_j\}_{j=1,\ldots,n_k}$ available at time t_k and such that

$$V_k = \{V_{jk}\}_{j=1,\ldots,n_k},$$

where

$$V_{jk} = u_j - \bar{u}$$

and

$$\bar{u} = mean_{j=1,\ldots,n_k}\{u_j\}.$$

Algorithm 1 DA

Input: for $k = 0, \ldots, m$ temporal steps: observations v_k, matrices R_k, model M, H, background u_0, historical data $S = \{u_j\}_{j=1,\ldots,n_k}$,
Compute B_0 from u_0 and S
Initialize iteration $k = 1$
while $k < m$ **do**
Compute $u_k = M\, u_{k-1}$
Compute B_k
Compute

$$u_k^{DA} = argmin_u \left\{ \|u - u_k\|_{B_k^{-1}} + \|v_k - Hu\|_{R_k^{-1}} \right\} \tag{10}$$

Count up k for the next iteration
end
Output: u^{DA}

The DA process, as defined in (7), can be solved by several methods. Two main approaches that have gained acceptance as powerful methods for data assimilation on the last years are the variational approach and Kalman Filter (KF). The variational approach [39] is based on the minimization of a functional, which estimates the discrepancy between numerical results and measures. The Kalman Filter [40] is a recursive filtering instead. In both cases we seek an optimal solution. Statistically, KF seeks a solution with minimum variance. Variational methods seek a solution that minimizes a suitable cost function. In certain cases, the two approaches are identical and they provide exactly the same solution [1]. However, the statistical approach, though often complex and time-consuming, can provide a richer information structure, i.e., an average and some characteristics of its variability (probability distribution).

Assuming the acquisition of the observations in a fixed time steps, the variational approach is named 3DVar [1,6], and it consists in computing the minimum of the cost function:

$$J(u) = (u - u_k)^T B_k^{-1} (u - u_k) + (v_k - Hu)^T R_k^{-1} (v_k - Hu) \tag{11}$$

where

$$u^{DA} = argmin_u\, J(u).$$

Potential problems of a variational method are mainly given by three main points: it heavily relies on correct B_k matrices, it does not take system evolution into account, and it can end up in local minima.

Kalman Filter consists in computing the explicit solution of the normal equations:

$$u^{DA} = u_k + K_k(v_k - Hu_k) \tag{12}$$

where the matrix

$$K_k = B_k H^T (H B_k H^T + R_k)^{-1} \tag{13}$$

is called Kalman gain matrix and where B_k is updated at each time step [6]:

$$B_k = M((1 - K_{k-1}H)B_{k-1})M^T. \tag{14}$$

A potential problem of Kalman Filter is that K_k is too large to store for large-dimensional problems.

A common point of 3DVar and KF is that the observations that are defined in the window $[0, T]$ (circle points in Figure 2) are assimilated in the temporal point, where the state is defined (grey rhombus in Figure 2 at time t_k). After the assimilation, a new temporal window $[T, 2T]$ is considered and new observations are assimilated. Each process starts at the end point of the previous process and the two assimilation processes are defined

in almost independent temporal windows. The result of the assimilation process u^{DA} (green pentagons in Figure 2), as computed by 3DVar or KF, is called analysis. The explicit expression for the analysis error covariance matrix, denoted as A_k, is [1]:

$$A_k = (I - K_k H) B_k. \tag{15}$$

where I denotes the identity matrix.

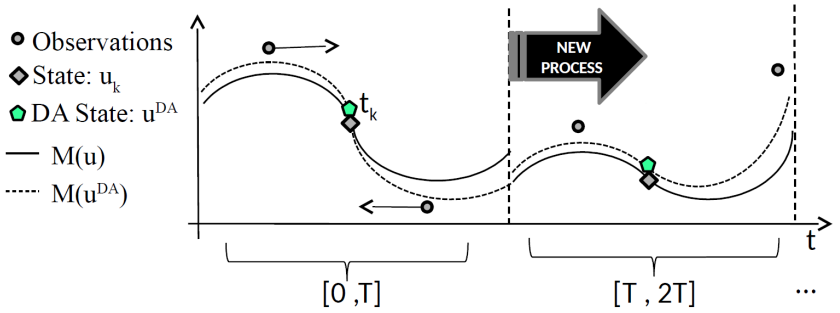


Figure 2. The Data Assimilation (DA) process.

Even if the DA process takes information in a temporal window into account, it does not really take the past into account. This is due to the complexity of the real application forecasting models for which a long time window would be too computationally expensive. In next section, we introduce the Deep Data Assimilation (DDA) model, in which we replace the forecasting model M (black line in Figure 3) with a new Machine Learning model $\mathcal{M}$ (red line in Figure 3) trained while using DA results and taking the past into account.

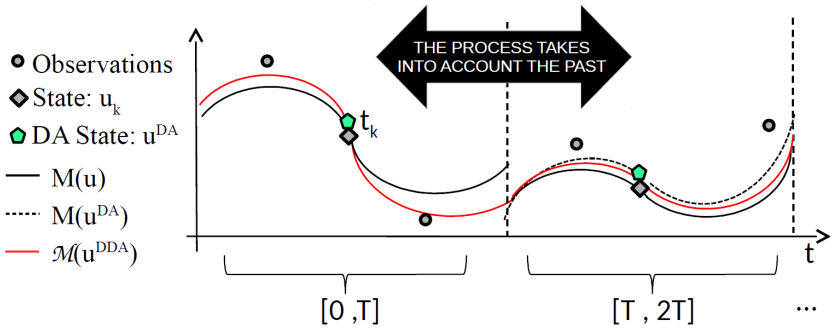


Figure 3. The Deep Data Assimilation (DDA) process.

3. The Deep Data Assimilation (DDA) Model

Data assimilation (DA) methodologies improve the levels of confidence in computational predictions (i.e., improve numerical forecasted results) by incorporating observational data into a prediction model. However, the error propagation into the forecating model is not improved by DA, so that, at each step, correction have to be based from scratch without learning from previous experience of error correction. The strongly nonlinear character of many physical processes of interest can result in the dramatic amplification of even small uncertainties in the input, so that they produce large uncertainties in the system behavior [5]. Because this instability, as many observations are assimilated as possible to the point where a strong requirement to DA is to enable real-time utilization of data to improve predictions. Therefore, it is desirable to use machine learning method in order to learn a function which accumulates the process of previous assimilation process. We use NN to model this process. The key concept is in recording each step of state correction during an assimilation period and then learn a function in order to capture this updating mechanism. The system model is then revised by composing this learned updating

function with the current system model. Such a process continues by further learning the assimilation process with the updated system model.

The resulting NN-forecasting model is then a forecasting model with an intrinsic assimilation process.

We introduce a neural network $\mathcal{G}_\omega$, starting by a composition with the model M, as described in Figure 4. The training target is:

$$u_k = \mathcal{G}_\omega(Mu_{k-1}). \tag{16}$$

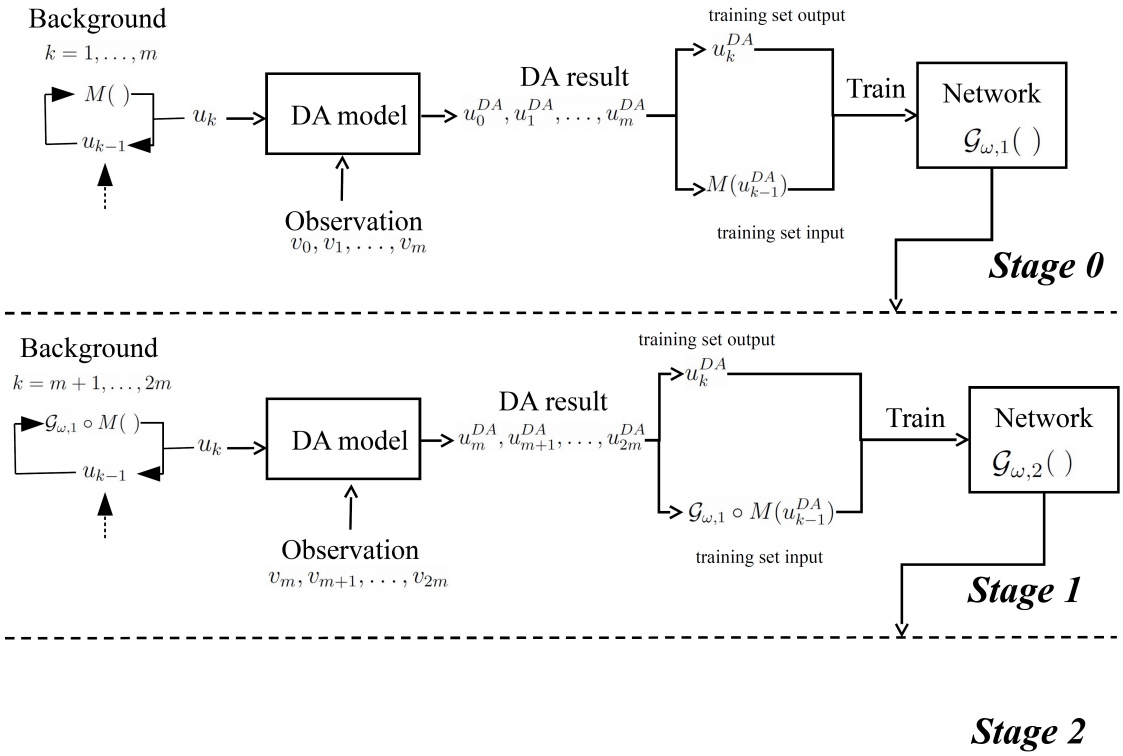


Figure 4. The schematic diagram of DDA.

We train $\mathcal{G}_\omega$ by exploiting the results of DA, as described in Algorithm 2. The $\mathcal{G}_\omega$ model can be the iterative form $\mathcal{G}_{\omega,i} \circ \mathcal{G}_{\omega,i-1} \circ ... \circ \mathcal{G}_{\omega,1} \circ M$, where $\mathcal{G}_{\omega,i}$ is defined as the network that was trained in ith iteration and $\circ$ denotes the composition function. Let $\mathcal{M}_i$ be the model that was implemented in the ith iteration, it is:

$$\mathcal{M}_i = \mathcal{G}_{\omega,i} \circ \mathcal{G}_{\omega,i-1} \circ ... \circ \mathcal{G}_{\omega,1} \circ M, \tag{17}$$

from which $\mathcal{M}_i = \mathcal{G}_{\omega,i} \circ \mathcal{M}_{i-1}$, and $\mathcal{M}_0 = M$.

For iteration $i > 1$, the training set for $\mathcal{G}_{\omega,i}$ is $\{(u_k, u_k^{DDA})\}$, which exploits the $\{(u_{k-1}^{DDA}, u_k^{DDA})\}$ from the last updated model $\mathcal{M}_i$. In this paper, we implement a Recurrent Neural Network (RNN) based on Elman networks [41]. It is a basic RNN structure that loops through the hidden layer output as an internal variable (that corresponds to the Jordan network [42] that outputs $u_{t+\ldots}$ as a variable). As shown in Figure 5, the RNN is specifically expressed as

$$\begin{aligned} h_t &= tanh(w_1[u_k, h_{t-1}] + b_1) \\ u_k^{DDA} &= w_2(tanh(w_1[u_k, h_{t-1}] + b_1)) + b_2 \end{aligned} \tag{18}$$

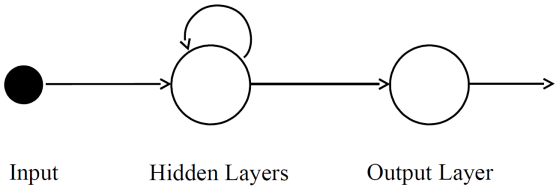


Figure 5. Recurrent Neural Network.

Algorithm 2 $\mathcal{A}(DDA)$

Input: for $k = 0, \dots, m$ temporal steps: observations v_k, matrices R_k, background u_0, historical data $S = \{u_j\}_{j=1,\dots,n_k}$, model M, H

Compute B_0 from u_0 and S
Initialize the $u_0^{DDA} = u_0$
Initialize the $\mathcal{M}_1 = M$
Initialize iteration $k = 1$
while $k < m$ **do**
Compute $u_k = \mathcal{M}_k u_{k-1}^{DDA}$

$$u_k^{DDA} = argmin_u \left\{ \|u - u_k\|_{B_k^{-1}} + \|Hu - v_k\|_{R_k^{-1}} \right\} \tag{19}$$

Generate the training set $\mathcal{D}_{k_i} = \{u_{k_i}, u_{k_i}^{DDA}\}$ with $k_i \in [k \cdot m, (k+1) \cdot m]$
Train a neural networks $\mathcal{G}_{\omega,k}$ with $\mathcal{D}_{k_i}$
Put trained $\mathcal{G}_{\omega,k}$ in the model: $\mathcal{M}_{k+1} \leftarrow \mathcal{G}_{\omega,k} \circ \mathcal{M}_k$
Put $B_{k+1} \leftarrow A_k$ (where A_k is defined in (15))
Count up k for the next iteration
end
Output: u^{DDA}

This choice to train the NN using data from DA introduces a reduction of the model's error at each time step that can be quantified as reduction of the background error covariance matrix, as proved in next theorem.

Theorem 1. *Let B_{k-1} and B_k be the error covariance matrices at step $k-1$ and step k in Algorithm 2, respectively. We prove that*

$$\|B_k\|_\infty \leq \|B_{k-1}\|_\infty \tag{20}$$

i.e., the error in background data is reduced at each iteration.

Proof. We prove the thesis by a reductio ad absurdum. We assume

$$\|B_k\|_\infty > \|B_{k-1}\|_\infty \tag{21}$$

At step $k-1$, from Step 10 of Algorithm 2 we have $B_k = A_{k-1}$, i.e., from (15) and (13), the (21) gives

$$\left\| \left(I - \frac{H_{k-1} B_{k-1} H_{k-1}^T}{H_{k-1} B_{k-1} H_{k-1}^T + R_{k-1}} \right) B_{k-1} \right\|_\infty > \|B_{k-1}\|_\infty \tag{22}$$

i.e., from the property of $\|\cdot\|_\infty$: $\|a\|_\infty \|b\|_\infty > \|a\, b\|_\infty$, we have

$$\left\| I - \frac{H_{k-1} B_{k-1} H_{k-1}^T}{H_{k-1} B_{k-1} H_{k-1}^T + R_{k-1}} \right\|_\infty \|B_{k-1}\|_\infty > \|B_{k-1}\|_\infty \tag{23}$$

which means that

$$\left\| I - \frac{H_{k-1} B_{k-1} H_{k-1}^T}{H_{k-1} B_{k-1} H_{k-1}^T + R_{k-1}} \right\|_\infty > 1 \tag{24}$$

Becaue of the definition of $\|\cdot\|_\infty$, we can then assume that $\exists\, ij$, such that

$$1 - \frac{b_{ij}}{b_{ij} + r_{ij}} > 1 \tag{25}$$

i.e., such that

$$\frac{b_{ij}}{b_{ij} + r_{ij}} < 0. \tag{26}$$

which is absurd. In fact, if we consider the two possible conditions $i = j$ and $i \neq j$. In case $i = j$, b_{ii} and r_{ii} are diagonal elements of the errors covariance matrices. In other words, they are values of variances that mean that they are positive values and the (26) is not satisfied. In case $i \neq j$, $r_{ij} = 0$ as the error covariance matrix R_k is diagonal, as defined in (8). Subsequently, we have that $\frac{b_{ij}}{b_{ij}+r_{ij}} = \frac{b_{ij}}{b_{ij}} = 1$ and the (26) is also not satisfied. Subsequently, the (20) is proven. □

Another important aspect is that the solution of the DDA Algorithm 2 is more accurate than the solution of a standard DA Algorithm 1.

In fact, even if some sources of errors cannot be ignored (i.e., the round off error), then the introduction of the DDA method reduces the error in the numerical forecasting model (below denoted as ξ_k). The following result held.

Theorem 2. *Let δ_k denote the error in the DA solution:*

$$u_k^{DA} = u_k^{true} + \delta_k\,,$$

and let $\hat{\delta}_k$ be the error in DDA solution:

$$u_k^{DDA} = u_k^{true} + \hat{\delta}_k\,,$$

it is:

$$\|\hat{\delta}_k\|_\infty \leq \|\delta_k\|_\infty \tag{27}$$

Proof. Let ξ_k be error in the solution of the dynamic system M in (3), i.e., $u_k = M(u_{k-1}) + \xi_k$. From the backward error analysis that was applied to the DA algorithm [13], we have:

$$\|\delta_k\|_\infty = \mu_{DA}\, \|\xi_k\|_\infty \tag{28}$$

and

$$\|\hat{\delta}_k\|_\infty = \mu_{DDA}\, \|\xi_k\|_\infty \tag{29}$$

where μ_{DA} and μ_{DDA} denote the condition numbers of the DA numerical model and the DDA numerical model, respectively.

It has been proved, in [37], that the condition number of the DA and DDA numerical models are directly proportional to the condition numbers of the related background error covariance matrices B_k. Subsequently, thanks to the (20), we have $\mu_{DDA} \leq \mu_{DA}$, from which:

$$\|\hat{\delta}_k\|_\infty = \mu_{DDA}\, \|\xi_k\|_\infty \leq \mu_{DA}\, \|\xi_k\|_\infty = \|\delta_k\|_\infty \tag{30}$$

□

The DDA framework in this article implement a Recurrent neural network (RNN) structure as $\mathcal{G}_\omega$. The loss function is defined, as described in Theorem 3 and the network is updated at each step by the gradient loss function on its weights and parameters:

$$\omega_{i+1} \leftarrow \omega_{i+1} + \alpha \frac{\partial \mathcal{L}_i}{\partial \omega}. \tag{31}$$

Theorem 3. *Let* $\{(u_k, u_k^{DDA})\}$ *be the training set for* $\mathcal{G}_{\omega,i}$*, which exploits the data* $\{(u_{k-1}^{DDA}, u_k^{DDA})\}$ *from the past updated model* $\mathcal{M}_i$ *in (17). The loss function for the Back-propagation, being defined as mean square error (MSE) for the DA training set, is such that*

$$\mathcal{L}_k(\omega) = \left\| B_k H^T O_k^{-1} d_k - \mathcal{G}_{\omega,k}(B_{k-1} H^T O_{k-1}^{-1} d_{k-1}) \right\|_2 \tag{32}$$

where $d_k = v_k - H_k u_k$ *is the misfit between observed data and background data, and* $O_k = H_k B_k H_k^T + R_k$ *is the Hessian of the DA system.*

Proof. The mean square error (MSE)-based loss function for the Back-propagation is such that

$$\mathcal{L}_i = ||u_k^{DDA} - \mathcal{G}_{\omega,i} \circ \mathcal{M}_i(u_{k-1}^{DDA})||_2, \tag{33}$$

The solution u_k^{DDA} of the DDA system, which is obtained by a DA function, can be expresses as [1]:

$$u_k^{DDA} = B_k H^T (H B_k H^T + R_k)^{-1} (v_k - H u_k) \tag{34}$$

Let posed $d_k = v_k - H u_k$ and $O_k = H B_k H^T + R_k$, (34) gives:

$$u_k^{DDA} = B_k H^T O_k^{-1} d_k \tag{35}$$

Substituting, in (33), the expression of u_k^{DDA} given by (35), the (32) is proven. □

4. Experiments

In this section, we apply the DDA technology to two different applications: the Double integral mass dot system and the Lorenz system:

- Double integral mass dot system:
  For the physical representation of a generalized motion system, the double-integral particle system is commonly used. The method, under the influence of an acceleration regulation quantity, can be seen as the motion of a particle. The position and velocity are the variables that affect the state at each time step. The status as a controlled state gradually converges to a relatively stable value, due to the presence of a PID controller in the feedback control system. The performance of the controller is the system-acting acceleration, or it can be interpreted as a force that linearly relates to the acceleration. Double integral system is a mathematic abstraction of Newton's Second Law that is often used as a simplified model of several controlled systems. It can be represented as a continuous model, as:
  $$\begin{aligned} \dot{u} &= Au + Bz + \xi, \\ v &= Cu + r \end{aligned} \tag{36}$$
  where the state $u = [u_1, u_2]^T$ is a two-dimensional vector containing position u_1 and velocity u_2 and where $z = \dot{u}_2$ is the controlled input. The coefficients matrices A, B, and C are time-invariant system and observations matrices. r is the noise of the observations, which is Gaussian and two dimensional. The system disruption ξ comprises two aspects: random system noise and instability of the structural model.
- Lorenz system:
  Lorenz developed a simplified mathematical model for atmospheric convection in 1963. It is a common test case for DA algorithms. The model is a system of three ordinary differential equations, named Lorenz equations. For some parameter values and initial conditions, it is noteworthy for having a chaotic behavior. The Lorenz equations are given by the nonlinear system:

$$\begin{aligned} \frac{dp}{dt} &= -\sigma(p-q), \\ \frac{dq}{dt} &= \rho p - q - pr, \\ \frac{dr}{dt} &= pq - \beta r. \end{aligned} \tag{37}$$

where p, q and r are coordinates, and σ, ρ, and β are parameters. In our implementation, the model has been discretized by second order Runge–Kutta method [43] and, for this test case, we posed $\sigma = 10$, $\rho = 8/3$ and $\beta = 28$.

We mainly provide testing in order to validate the results that we proved in the previous section. Afterwards, we mainly focus on:

4.1 DDA accuracy: we prove that the accuracy of the forecasting result increases as the number of time steps increases, as proved in Theorem 1;

4.2 DDA vs. DA: in order to address the true models, we face the circumstances where DA is not sufficient to reduce the modeling uncertainty and we show that the DDA algorithm, we have proposed is applicable to solve this issue. Such findings validate what we showed in Theorem 2; and,

4.3 R-DDA vs. F-DDA: we are demonstrating that the accuracy of the R-DDA is better than that of the F-DDA. It is also due to the way the weight is calculated, i.e., by including the DA covariance matrices, as shown in Theorem 3.

We implemented the algorithms on the Simulink (Simulation and Model-Based Design) in Matlab 2016a.

4.1. DDA Accuracy

4.1.1. Double Integral Mass Dot System

When considering model uncertainty, a system disturbance in (36) is introduced so that:

$$\xi_k = \xi_{smu}(u_k) + \xi_{iid,k}, \tag{38}$$

where $\xi_{smu}(\cdot)$ denotes the structural model uncertainty and ξ_{iid} denotes the i.i.d random disturbance.

One typical model error is from the parallel composition as:

$$\xi_{smu}(u_k) = D\frac{e^{u_k+1}}{1+e^{u_k+1}}. \tag{39}$$

where the amplitude vector $D = [d_1, d_2]$ is adjustable, according to the reality.

A cascade controller is designed to prevent divergence from the device in order to accomplish the model simulation. As a sinusoidal function, the tracking signal is set. 10,000 samples are collected from a 100 s simulation with a 100 Hz sample frequency. The training set for the DDA Algorithm 2 is made of the time series $\{u_k, u_k^{DDA}\}$ of m couples. First of all, we run the framework for the dynamic system Equation (36) and record the observation v, control input z. A corresponding DA runs alongside program updates, and outputs a u^{DA} prediction sequence. The network $\mathcal{G}_{\omega,1}$ is trained on the dataset of m steps.

Subsequently, the system and the DA update the data from the $m+1$ step to $2m$ step. Although the device upgrade process in DA now incorporates the qualified neural network $\mathcal{G}_{\omega,1}$, as:

$$u_k = \mathcal{G}_{\omega,1}(Mu_{k-1} + Gu_k) \tag{40}$$

Figure 6 shows what we expected regarding the accuracy of DDA. In fact, as it is shown, the mean forecasting error decrease as the training window length increases. The Figure also shows a comparison of the mean forecasting error values with respect the results that were obtained from DA. Figure 7 is a convergence curve when matlab trains a

network using the Levenberg–Marquardt backpropagation [44,45] method. It can be seen that, with just few epochs, the best value for the mean square error is achieved.

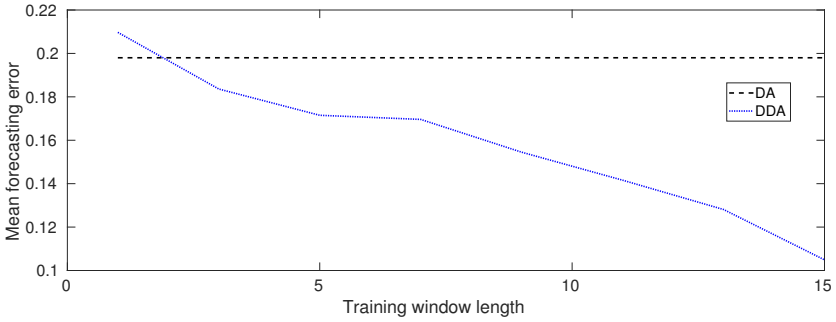


Figure 6. Mean forecasting error-first test case.

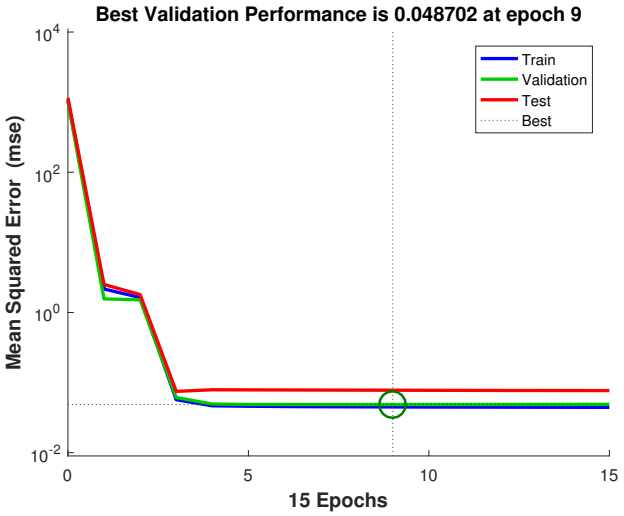


Figure 7. Training process-first test case.

4.1.2. Lorenz System

We implement a Lorenz system with structural model uncertain as:

$$u_k = M_k(u_{k-1}) + \xi_{smu,k} \tag{41}$$

where $u_k = [p_k, q_k, r_k]^T$ denotes the state and $M_k(\cdot)$ denotes the discrete function. The structural model uncertainty is denoted here as $\xi_{smu,k} = c\, u_{k-1}$, with $c = 0.01$, and $M_k(\cdot)$, such that:

$$\begin{aligned} p_{k+1} =& p_k + \sigma\frac{\Delta t}{2}[2(q_k - p_k) + \Delta t(\rho p_k - q_k - p_k r_k) - \sigma\Delta t(q_k - p_k)], \\ q_{k+1} =& q_k + \frac{\Delta t}{2}[\rho p_k - q_k - p_k r_k + \rho(p_k + \sigma\Delta t(q_k - p_k)) - q_k \\ & - \Delta t(\rho p_k - q_k - p_k r_k) - (p_k + \sigma\Delta t(q_k - p_k))(r_k + \Delta t(p_k q_k - \beta r_k))], \\ r_{k+1} =& r_k + \frac{\Delta t}{2}[p_k q_k - \beta r_k + (p_k + \Delta t\sigma(q_k - p_k))(q_k + \Delta t(\rho p_k - q_k - p_k r_k)) \\ & - \beta r_k - \Delta t(p_k q_k - \beta r_k)]. \end{aligned} \tag{42}$$

First of all, for the dynamic system, Equation (37), we run the system and record the true value of the system u. Subsequently, by adding Gaussian noise to the true value of u, v is created. The DA algorithm is then applied to produce a sequence with a DA result of u^{DA} and a length of m. The training set is made of the time series $\{M_k(u_{k-1}^{DA}), u_k^{DA}\}$. Afterwards, the neural network $\mathcal{G}_\omega$ is trained.

Figure 8 shows the accuracy results of DDA for Lorenz test case. Additionally, in this case, the results confirm our expectation. In fact, as it is shown, the mean forecasting error decrease as the training window length increases. Additionally, in this figure, a comparison of the mean forecasting error values with respect the results obtained from DA is shown. Figure 9 is a convergence curve when matlab trains a network using the Levenberg–Marquardt backpropagation method and, also in this case, the best value for the mean square error is achieved with just a few epochs.

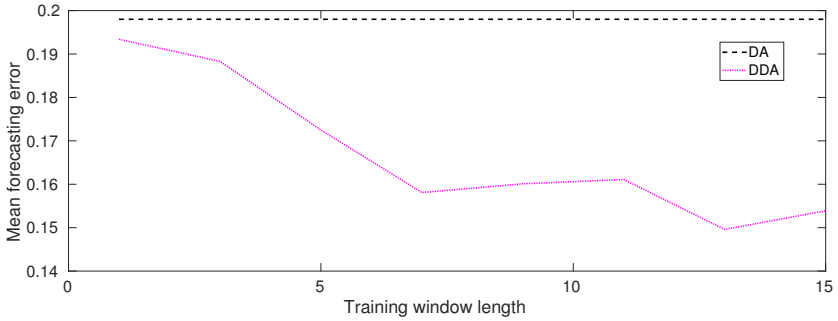


Figure 8. Mean forecasting error-second test case.

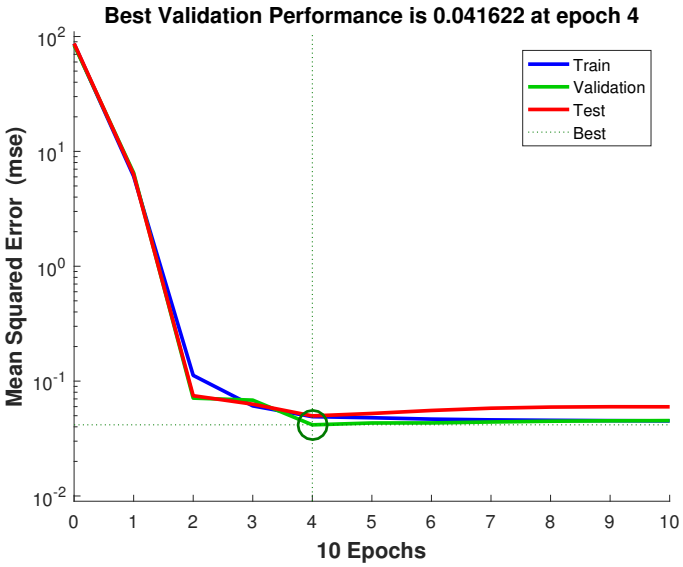


Figure 9. Training process-second test case.

4.2. DDA vs. DA

In this section, we compare the forecasting results based on DA model and the DDA model, respectively. For the DA, the system model update is formed as:

$$u_k = M_k(u_{k-1}). \tag{43}$$

where M_k is the discrete forecasting system in (3). For DDA, the system update model is as:

$$u_k = \mathcal{M}_k(u_{k-1}). \tag{44}$$

where $\mathcal{M}_k$ is defined in (17).

4.2.1. Double Integral Mass Dot System

Figure 10 shows the result of DA and DDA model on a double integral system that considers the model error. Two vertical dash lines are at which the neural network is trained. Obviously, the forecasting state value based on DDA is more closer to the true

value than DA. The DA results in a larger residual. We also calculate the ratio of residual to true value as a forecasting error metric. The average forecasting error is significantly reduced from 16.7% to 1.5% by applying DDA model. Table 1 lists the run-time cost of every training window in Figure 10.

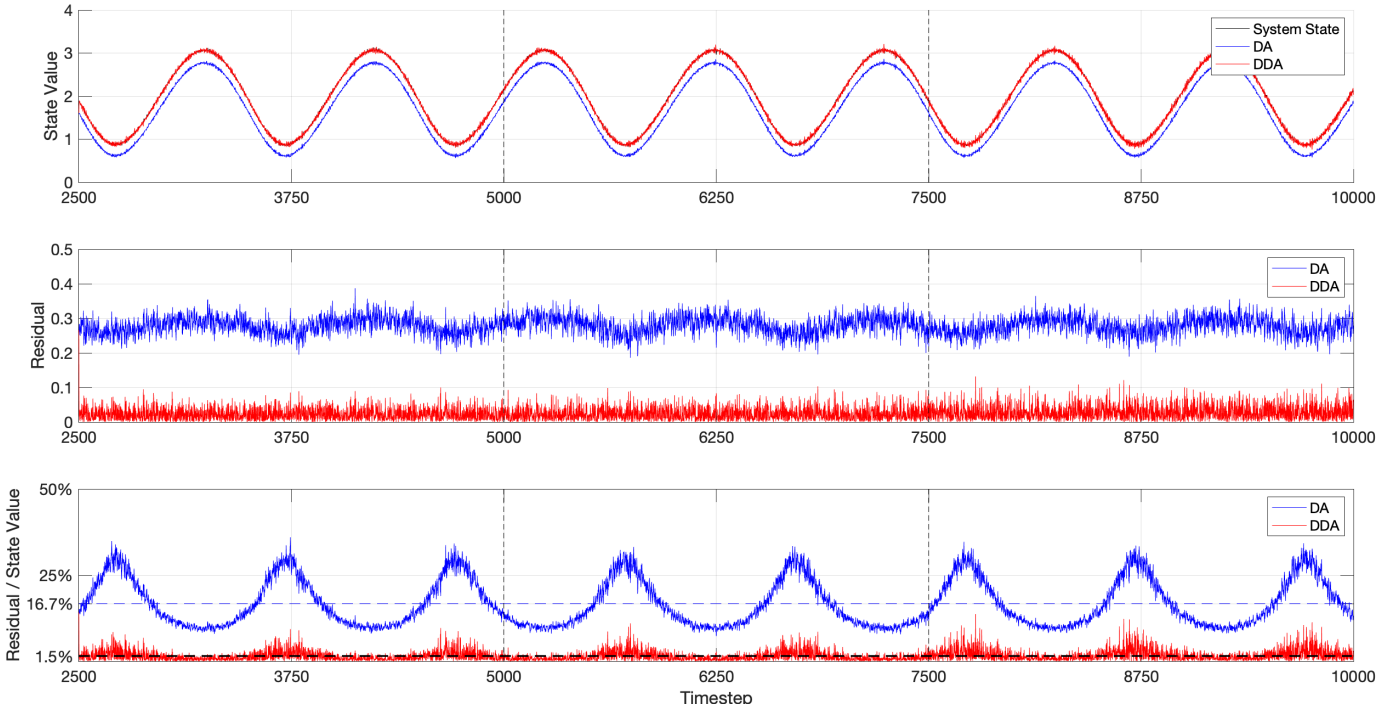


Figure 10. Simulation result of DDA on double integral system considering the model error. The vertical dash-lines refer to the training windows.

4.2.2. Lorenz System

We compare the forecasting results that are shown in Figure 11. The $k \in [0, 1000]$ generated training set is not plotted in full. The forecast starts at $k = 1000$. We can see that the DA model's forecasting errors in all three axes are large. DA model trajectories can not track the real state. Although the DDA model trajectories track the true value very well. Figure 12 is a 3D plot of this forecasting part of the Lorenz system. This demonstrates that the DA model fails to predict the right hand part of this butterfly. In this test, for the right wing trajectory of butterfly, the DDA model outperforms the DA model in forecasting.

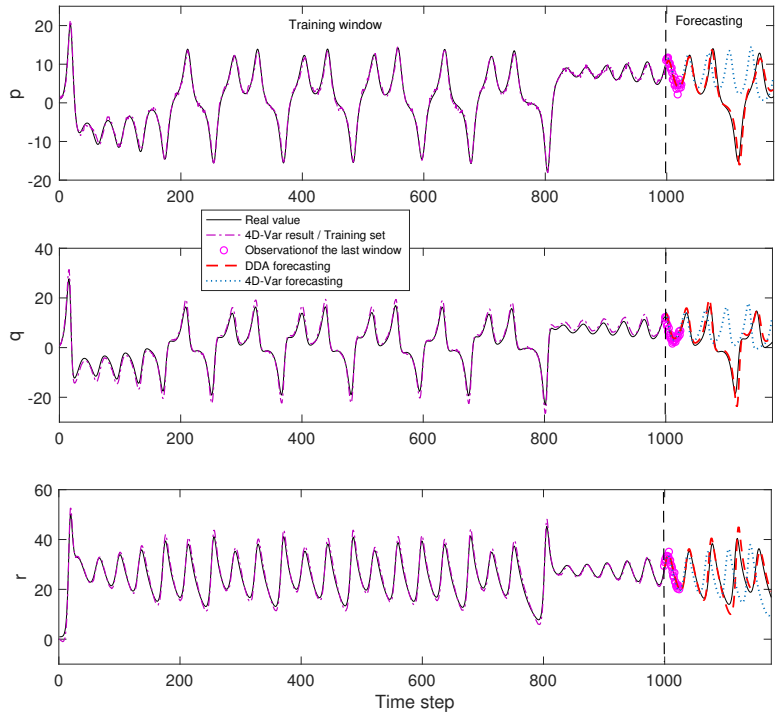


Figure 11. DA and DDA trajectories on 3 axis.

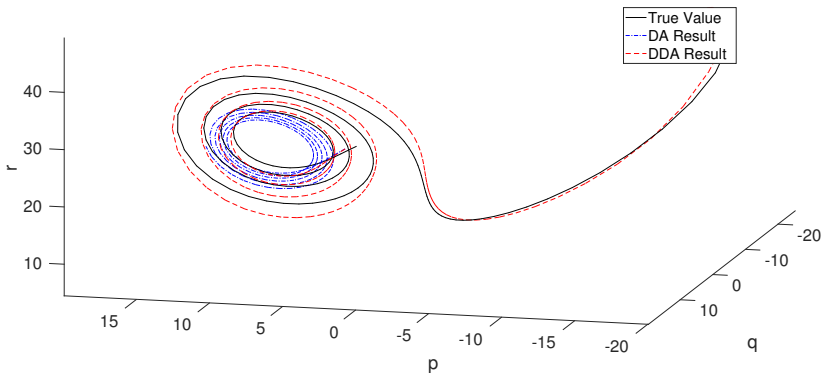


Figure 12. Forecasts of the DA and DDA trajectories. It is the three-dimensional (3D)-plot of Figure 11 for timestep $\in (1000, 1176)$.

4.3. F-DDA vs. R-DDA

In this section, we introduce the DDA using the feedfoward neural network (we named F-DDA) and compare the results with the R-DDA, i.e., the DDA we described in this paper that uses the recurrent neural network. The feed-foward (FC) network is the earliest and most basic neural network structure. This means that in the next layer, the output of each neuron in the upper layer becomes the input of each neuron, and it has a structure with corresponding values of weight. Inside the FC network, there is no cycle or loop. In the fitting of functions, fully conected neural networks are widely cited (especially the output of continuous value functions). The multilayer perceptron (MLP), which contains three hidden layers, is used in this paper.

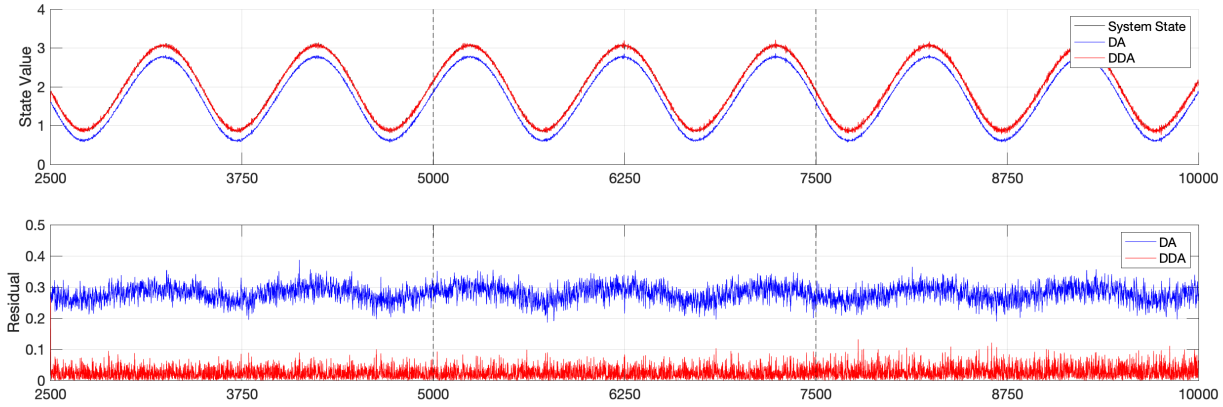
In the training step of the DDA framework, the training set is the DDA result over a period of time. Take the first cycle ($i = 0$) as an example. The training set is $u_0^{DDA}, u_1^{DDA}, u_2^{DDA}, ..., u_m^{DDA}$ (the results of consecutive time series), and this data set generates the model. Because this network is a feedfoward neural network and it is unable to

manage time series, we consider that the training set for the M category is independent of each other.

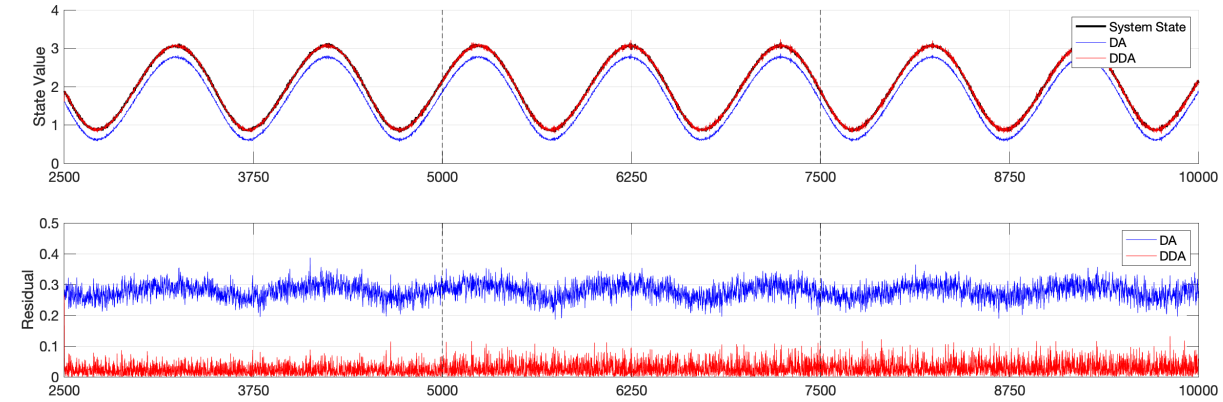
4.3.1. Double Integral Mass Dot System

In this section, we compared the performance of F-DDA and R-DDA in the double integral mass dot system. Because our data were previously generated by simulink, random noise, and observation noise are fixed and can be compared.

Figure 13 shows that F-DDA obtains a similar result when comparing to R-DDA in this case. It is mainly because the double integral mass dot system is not strongly dependent on the time historical states. Meanwhile, when considering the execution time cost that is shown in Table 1, F-DDA is more preferable, as it takes shorter training time than R-DDA.



(**a**) F-DDA.



(**b**) R-DDA.

Figure 13. The simulation result of DDA on double integral system. The vertical dash-lines refer to the training windows.

Table 1. The execution time cost for both F-DDA and R-DDA in every training and forecasting window when considering the model error.

Train	**t1**	**t2**	**t3**
F-DDA	0.9605 s	0.5723 s	0.4181 s
R-DDA	102.2533 s	16.4835 s	15.0160 s
Forecast	**t1**	**t2**	**t3**
F-DDA	17.9610 s	36.2358 s	56.4286 s
R-DDA	18.3527 s	35.9934 s	53.8951 s

4.3.2. Lorenz System

Figure 14 is a comparison of the forecasting results of F-DDA and R-DDA for the Lorenz system. It can be seen that the R-DDA (black dashed line) has a more accurate prediction of the true value (blue solid line) and the curve fits better. In contrast, F-DDA (red dotted line) has a certain time lag and it is also relatively inaccurate in amplitude. This is mainly because the Lorenz system is strongly time-dependent. Table 2 provides the run-time cost for both F-DDA and R-DDA in training and forecasting. It is worth taking more training time for R-DDA to achieve a better prediction than F-DDA.

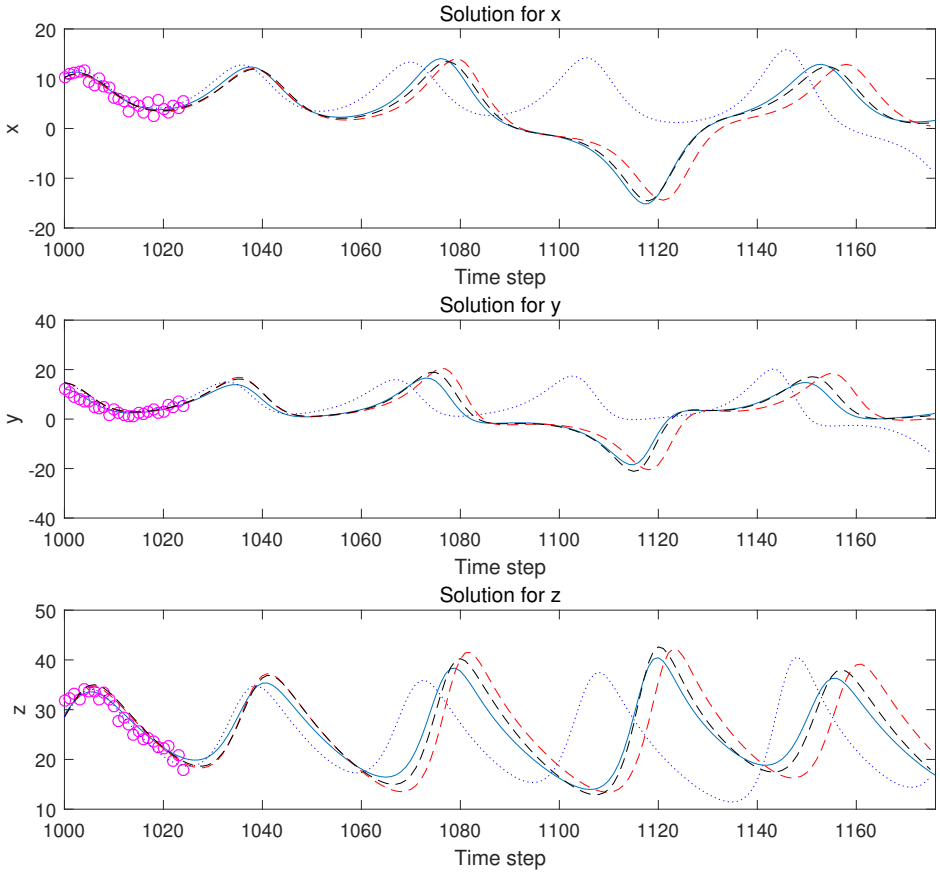


Figure 14. Trajectories of DA and DDA forecasting. It is the timesteps as Figure 11 for $\in (1000, 1176)$. R-DDA (black dashline) vs. F-DDA (red dashline).

Table 2. Run-time cost for F-DDA and R-DDA in training and forecasting of Lorenz system.

	Train Time	Forecast Time
F-DDA	3.7812 s	3.8581 s
R-DDA	24.2487 s	3.2901 s

5. Conclusions and Future Work

In this article, we discuss the DDA: the integration of DL and DA and we validate the algorithm by a numerical examples. The DA methods have increased strongly in complexity in order to better suit their application requirements and circumvent their implementation problems. However, the approaches to DA are unable to fully overcome their unrealistic assumptions, particularly of zero error covariances, linearity, and normality. DL shows great capability in approximating nonlinear systems, and extracting high-dimensional features. Together with the DA methods, DL is capable of helping traditional methods to make forecasts without the conventional methods' assumptions. On the other side, the training data provided to DL technologies include several numerical, approximation,

and round off errors that are trained in the DL forecasting model. DA can increase the reliability of the DL models reducing errors by including information on physical meanings from the observed data.

This paper showed that the cohesion of DL and DA is blended in the future generation of technology that is used in support of predictive models.

So far, the DDA algorithm still remains to be implemented and verified in varies specific domains, which have huge state space and more complex internal and external mechanisms. Future work includes adding more data to the systems. A generalization of DDA could be developed if it is used for different dynamical systems.

The numerical solution of dynamic systems could be replaced by DL algorithms, such as Generative Adversarial Networks or Convolutional Neural Networks combined with LSTM, in order to make the runs faster. This will accelerate the forecast process towards a solution in real time. When combined with DDA, this future work has the potential to be very fast.

DDA is encouraging, as speed and accuracy are typically terms that are mutually exclusive. The results that are shown here are promising and demonstrate how, in computationally demanding physical models, the merger of DA and ML models, especially in computationally demanding physical models.

Author Contributions: R.A. and Y.-K.G. conceived and designed the experiments; J.Z. and S.H. performed the experiments; All the authors analyzed the data; All the authors contributed reagents/materials/analysis tools. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Imperial College-Zhejiang University Joint Applied Data Science Lab. The work is supported by the EP/T000414/1 PREdictive Modelling with QuantIfication of UncERtainty for MultiphasE Systems (PREMIERE) and the EP/T003189/1 Health assessment across biological length scales for personal pollution exposure and its mitigation (INHALE).

Acknowledgments: This work is supported by the EP/T000414/1 PREdictive Modelling with QuantIfication of UncERtainty for MultiphasE Systems (PREMIERE) and the EP/T003189/1 Health assessment across biological length scales for personal pollution exposure and its mitigation (INHALE).

Conflicts of Interest: The authors declare no conflict of interest.

Acronyms

DA	Data Assimilation
DDA	Deep Data Assimilation
DL	Deep Learning
DNN	Deep Neural Network
LSTM	Long Short Term Memory
ML	Machine Learning
NN	Neural Network
VarDA	Variational Data Assimilation

References

1. Kalnay, E. *Atmospheric Modeling, Data Assimilation and Predictability*; Cambridge University Press: Cambridge, MA, USA, 2003.
2. Blum, J.; Le Dimet, F.X.; Navon, I.M. Data assimilation for geophysical fluids. In *Handbook of Numerical Analysis*; Elsevier: Amsterdam, The Netherlands, 2009; Volume 14, pp. 385–441.
3. D'Elia, M.; Perego, M.; Veneziani, A. A variational data assimilation procedure for the incompressible Navier-Stokes equations in hemodynamics. *J. Sci. Comput.* **2012**, *52*, 340–359. [CrossRef]
4. Potthast, R.; Graben, P.B. Inverse problems in neural field theory. *SIAM J. Appl. Dyn. Syst.* **2009**, *8*, 1405–1433. [CrossRef]
5. Christie, M.A.; Glimm, J.; Grove, J.W.; Higdon, D.M.; Sharp, D.H.; Wood-Schultz, M.M. Error analysis and simulations of complex phenomena. *Los Alamos Sci.* **2005**, *29*, *6–25*.
6. Asch, M.; Bocquet, M.; Nodet, M. *Data Assimilation: Methods, Algorithms, and Applications*; SIAM: Philadelphia, PA, USA, 2016; Volume 11.
7. Weinan, E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.* **2017**, *5*, 1–11.
8. Li, Q.; Chen, L.; Tai, C.; Weinan, E. Maximum principle based algorithms for deep learning. *J. Mach. Learn. Res.* **2017**, *18*, 5998–6026.

9. Boukabara, S.A.; Krasnopolsky, V.; Stewart, J.Q.; Maddy, E.S.; Shahroudi, N.; Hoffman, R.N. Leveraging Modern Artificial Intelligence for Remote Sensing and NWP: Benefits and Challenges. *Bull. Am. Meteorol. Soc.* **2019**, *100*, ES473–ES491. [CrossRef]
10. Dueben, P.D.; Bauer, P. Challenges and design choices for global weather and climate models based on machine learning. *Geosci. Model Dev.* **2018**, *11*, 3999–4009. [CrossRef]
11. Cacuci, D. *Sensitivity and Uncertainty Analysis*; Chapman & Hall/CRC: New York, NY, USA, 2003.
12. Daescu, D.; Navon, I. Sensitivity analysis in nonlinear variational data assimilation: theoretical aspects and applications. In *Advanced Numerical Methods for Complex Environmental Models: Needs and Availability*; Farago, I., Zlatev, Z., Eds.; Bentham Science Publishers: Sharjah, UAE, 2013.
13. Arcucci, R.; D'Amore, L.; Pistoia, J.; Toumi, R.; Murli, A. On the variational data assimilation problem solving and sensitivity analysis. *J. Comput. Phys.* **2017**, *335*, 311–326. [CrossRef]
14. Cacuci, D.; Navon, I.; Ionescu-Bujor, M. *Computational Methods for Data Evaluation and Assimilation*; CRC Press: Boca Raton, FL, USA, 2013.
15. Fisher, M.; Leutbecher, M.; Kelly, G.A. On the equivalence between Kalman smoothing and weak-constraint four-dimensional variational data assimilation. *Q. J. R. Meteorol. Soc.* **2005**, *131*, 3235–3246. [CrossRef]
16. Gagne, D.J.; McGovern, A.; Haupt, S.E.; Sobash, R.A.; Williams, J.K.; Xue, M. Storm-based probabilistic hail forecasting with machine learning applied to convection-allowing ensembles. *Weather Forecast.* **2017**, *32*, 1819–1840. [CrossRef]
17. Campos, R.M.; Krasnopolsky, V.; Alves, J.H.G.; Penny, S.G. Nonlinear wave ensemble averaging in the Gulf of Mexico using neural networks. *J. Atmos. Ocean. Technol.* **2019**, *36*, 113–127. [CrossRef]
18. Babovic, V.; Keijzer, M.; Bundzel, M. From global to local modelling: A case study in error correction of deterministic models. In Proceedings of the Fourth International Conference on Hydro Informatics, Cedar Rapids, IA, USA, 23–27 July 2000.
19. Babovic, V.; Cañizares, R.; Jensen, H.R.; Klinting, A. Neural networks as routine for error updating of numerical models. *J. Hydraul. Eng.* **2001**, *127*, 181–193. [CrossRef]
20. Babovic, V.; Fuhrman, D.R. Data assimilation of local model error forecasts in a deterministic model. *Int. J. Numer. Methods Fluids* **2002**, *39*, 887–918. [CrossRef]
21. Brajard, J.; Carassi, A.; Bocquet, M.; Bertino, L. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model. *arXiv* **2020**, arXiv:2001.01520.
22. Rasp, S.; Dueben, P.D.; Scher, S.; Weyn, J.A.; Mouatadid, S.; Thuerey, N. WeatherBench: A benchmark dataset for data-driven weather forecasting. *arXiv* **2020**, arXiv:2002.00469.
23. Geer, A.J. *Learning Earth System Models from Observations: Machine Learning or Data Assimilation*? Technical Report 863; ECMWF: Reading, UK, 2020. [CrossRef]
24. Bocquet, M.; Brajard, J.; Carrassi, A.; Bertino, L. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *Found. Data Sci.* **2020**, *2*, 55–80. [CrossRef]
25. Raissi, M.; Perdikaris, P.; Karniadakis, G. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.* **2016**, *335*. [CrossRef]
26. Perdikaris, P.; Raissi, M.; Damianou, A.; Lawrence, N.; Karniadakis, G. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2017**, *473*, 20160751. [CrossRef]
27. Chao, G.; Luo, Y.; Ding, W. Recent advances in supervised dimension reduction: A survey. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 341–358. [CrossRef]
28. Quilodrán Casas, C.; Arcucci, R.; Wu, P.; Pain, C.; Guo, Y.K. A Reduced Order Deep Data Assimilation model. *Phys. D Nonlinear Phenom.* **2020**, *412*, 132615. [CrossRef]
29. Makarynskyy, O. Improving wave predictions with artificial neural networks. *Ocean Eng.* **2004**, *31*, 709–724. [CrossRef]
30. Coskun, H.; Achilles, F.; Dipietro, R.; Navab, N.; Tombari, F. Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5525–5533. arXiv:1708.01885v1. [CrossRef]
31. Zhu, J.; Hu, S.; Arcucci, R.; Xu, C.; Zhu, J.; Guo, Y.k. Model error correction in data assimilation by integrating neural networks. *Big Data Min. Anal.* **2019**, *2*, 83–91. [CrossRef]
32. Buizza, C.; Fischer, T.; Demiris, Y. Real-Time Multi-Person Pose Tracking using Data Assimilation. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Snowmass Village, CO, USA, 1–5 March 2020; pp. 1–8.
33. Arcucci, R.; Moutiq, L.; Guo, Y.K. Neural assimilation. In *International Conference on Computational Science*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 155–168.
34. Foo, Y.W.; Goh, C.; Li, Y. Machine learning with sensitivity analysis to determine key factors contributing to energy consumption in cloud data centers. In Proceedings of the 2016 International Conference on Cloud Computing Research and Innovations (ICCCRI), Singapore, 4–5 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 107–113.
35. Purser, R. A new approach to the optimal assimilation of meteorological data by iterative Bayesian analysis. In Proceedings of the Preprint of the 10th AMS Conference on Weather Forecasting and Analysis, Clearwater Beach, FL, USA, 25–29 June 1984; American Meteorological Society: Boston, MA, USA, 1984; pp. 102–105.
36. Engl, H.W.; Hanke, M.; Neubauer, A. *Regularization of Inverse Problems*; Springer Science & Business Media: Berlin/Heidelberg, Germany,1996; Volume 375.

37. Nichols, N. Mathematical concepts in data assimilation. In *Data Assimilation*; Lahoz, W., Khattatov, B., Menard, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010.
38. Hansen, P. *Rank Deficient and Discrete Ill-Posed Problems*; SIAM: Philadelphia, PA, USA, 1998.
39. Le Dimet, F.; Talagrand, O. Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus* **1986**, *38A*, 97–110. [CrossRef]
40. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
41. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
42. Jordan, M.I. Serial order: A parallel distributed processing approach. In *Advances in Psychology*; Elsevier: Amsterdam, The Netherlands, 1997; Volume 121, pp. 471–495.
43. Lawless, A.S. *Data Assimiliation with the Lorenz Equations*; University of Reading: Reading, UK, 2002.
44. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **1944**, *2*, 164–168. [CrossRef]
45. Marquardt, D.W. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]