Neural Network State-Space Estimators

Minxing Sun[1,2,3,4], Li Miao[1,2,3], Qingyu Shen[1,2,3,4], Yao Mao[1,2,3*], Qiliang Bao[1,2,3]

[1*]National Key Laboratory of Optical Field Manipulation Science and Technology, Chinese Academy of Science, Chengdu, 610209, Sichuan, China.

[2]Key Laboratory of Optical Engineering, Chinese Academy of Science, Chengdu, 610209, Sichuan, China.

[3]Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu, 610209, Sichuan, China.

[4]School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, 101408, China.

*Corresponding author(s). E-mail(s): iiiauthor@gmail.com; Contributing authors: iauthor@gmail.com; iiauthor@gmail.com; iiiauthor@gmail.com; iiiauthor@gmail.com;

Abstract

Classical state estimation algorithms rely on predefined state-space models, complicating mathematical modeling and limiting applicability when those models change. Artificial neural network–based estimators offer an alternative, but they seldom incorporate classical estimation techniques intrinsically and depend heavily on pre-collected training data. To address these challenges, we introduce a state-space model in which the input-layer nodes and all weight parameters are treated as estimated states, effectively simulating a neural network. We implement this nonlinear model using the canonical representatives of three estimator classes—the Extended Kalman Estimator, the Unscented Kalman Estimator, and the Particle Filter—to validate its generality. The model's accuracy is then compared against seven neural network–based methods across three test scenarios. Results show that our neural network state-space estimators combine robust learning capability with higher accuracy than traditional model-based methods, while matching the performance of pre-trained neural network approaches.

Keywords: Kalman estimator, Neural network, Learning capability, Adaptability

1 Introduction

State estimation algorithms are extensively utilized across various engineering disciplines. For example, in electro-optical tracking systems, they are essential for mitigating optical path disturbances caused by atmospheric turbulence, compensating for delays in hardware and image processing pipelines, and filtering out signal noise generated by sensors and target detection mechanisms. The fundamental principle of these algorithms comprises three stages: predicting the next state using the current estimate and the state-space model, predicting the corresponding observation via the observation equation, and correcting the state estimate based on the deviations between the actual observation and its prediction.

Estimators can be categorized based on how they establish the relationship between observation deviations and state deviations. One category comprises methods such as the Extended Kalman Estimator (EKE), which computes partial derivatives from the state-space model. The primary limitation of EKE is the complexity involved in deriving analytical solutions for these partial derivatives, particularly in highly nonlinear models. Another category includes the Unscented Kalman Estimator (UKE), which approximates the state distribution by assigning predefined deviations to each estimated state and evaluating the resulting variations in observations. However, UKE faces challenges in accurately capturing the relationships between coupled estimated states and observations. The third category is represented by Particle Estimator (PE), which introduces random perturbations to the estimated states and assesses the resulting observation variations. While PE is better suited for handling nonlinearities, its main drawback is the high computational cost associated with the large number of particles required, which can impede real-time performance.

All three estimation methods rely on predefined state-space models, and even the robust estimators designed to mitigate model inaccuracies can only compensate for a limited range of errors. In practical applications, defining an accurate state-space model for the target system is often inherently challenging, particularly when tracking non-cooperative targets.

The rapid advancement and widespread adoption of artificial neural network (ANN) methodologies have profoundly impacted time-domain signal processing, enabling significant breakthroughs in estimation and prediction tasks. Recurrent Neural Networks (RNNs), introduced by Hopfield (1982), were enhanced by Rumelhart, Hinton, and Williams (1986) with the backpropagation algorithm, facilitating more efficient training. Subsequent improvements by Jordan (1986) and Elman (1990) incorporated feedback loops, enhancing the ability to model temporal dependencies. However, RNNs struggled with vanishing and exploding gradients, challenges addressed by Hochreiter (1997) with Long Short-Term Memory (LSTM) networks, and further refined by Schmidhuber (1999) through the addition of forget gates. Gated Recurrent Units (GRUs), proposed by Kyunghyun (2014) and validated by Junyoung (2014), offered a simplified yet effective alternative to LSTMs. Convolutional Neural Networks (CNNs), initially developed by LeCun (1989) for handwritten digit recognition, have been extended to Recurrent Convolutional Neural Networks (Ming, 2015) to capture spatial and temporal dependencies, significantly improving prediction accuracy in applications such as vehicle and human trajectory forecasting

(Xiaoyu, 2020; Dapeng, 2020, 2021). Temporal Convolutional Networks (TCNs), introduced by Colin (2018), utilize pooling and upsampling to capture long-term motion patterns, enhancing capabilities in action segmentation and detection (Chen, 2020; Kaiyu, 2022). Neural Ordinary Differential Equations (NeuralODEs), proposed by Ricky (2018), represent neural network layers through differential equations, facilitating deeper and more efficient networks for tasks like robotic path planning and travel time estimation (Suhan, 2022; Yijun, 2023). Transformer architectures, introduced by Ashish (2017), leverage self-attention mechanisms to model long-term dependencies while reducing computational complexity. Their applications span maneuvering target tracking, autonomous vehicle trajectory prediction, and 3D human pose estimation (Yushu, 2017; Divya, 2022; Jihua, 2024).

Traditional estimation algorithms have also been integrated into neural networks to optimize training processes. Early efforts by Singhal (1988), Puskorius (1994), and Pérez-Ortiz (2003) applied EKE to train multilayer perceptrons and recurrent neural networks (RNNs), enhancing training efficiency. More recent integrations of UKE and PE have improved the training of feedforward and recurrent networks in applications such as voice classification and evapotranspiration estimation (Zaqiatud, 2017; Nazari, 2020). However, in these studies, estimation algorithms primarily treated the back-propagation algorithm as state-space equations, serving an auxiliary role rather than being the primary mechanism for state estimation. Consequently, the effectiveness of estimation algorithms remains limited and warrants further enhancement.

To address these limitations, our proposed algorithm introduces a novel state-space model that treats input layer nodes and all weight parameters as estimated states, effectively simulating a neural network within a state estimation framework. The main contributions of this work are as follows:

- Neural Network State-Space Equation: We propose a state-space equation that simulates a neural network, enabling real-time learning and pretraining. The estimation algorithms can be tailored by adjusting the input layer length and prediction steps to meet engineering requirements.
- Model Generality: The versatility of the proposed state-space equation is validated through the implementation of three representative estimation algorithms—EKF, UKE, and PF—demonstrating its broad applicability.
- Simulation of Diverse Architectures: The state-space equation is applied to neural networks with varying numbers of hidden layers and different configurations regarding the use of activation functions.
- Comparative Performance Evaluation: We compare the estimation performance of our state-space-based algorithms with various neural network architectures, including RNN, LSTM, GRU, TCN, Neural Ordinary Differential Equation(NeuralODE), and Transformer.
- Comprehensive Testing: Extensive tests were conducted on generated sine trajectories, a laboratory dual-reflection mirror platform, and actual unmanned aerial vehicle (UAV) trajectory data, demonstrating the robustness and effectiveness of our proposed method.

2 Problem Statement

In many real–world tracking problems, the target is non-cooperative, so an accurate motion model cannot be specified a priori. A common expedient is the constant-acceleration state–space model in (1), where only position can be observed while velocity and acceleration remain hidden:

$$\begin{aligned} x_{i+1} &= \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} x_i + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u_i, \\ z_i &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x_i + v_i. \end{aligned} \tag{1}$$

The state vector $x_i = [p_i, v_i, a_i]^{\mathsf{T}}$ stacks position, velocity and acceleration at instant i. T is the time interval between successive estimates. The process noise u_i absorbs modelling errors and environmental perturbations, and the measurement noise v_i reflects sensor inaccuracies; both are assumed zero-mean Gaussian. Their second-order statistics together with the initial error covariance $\Pi_{0|0}$ are summarised in (2), in which δ_{ij} denotes the Kronecker delta:

$$E\left(\begin{bmatrix} x_0 \\ u_i \\ v_i \end{bmatrix} \begin{bmatrix} x_0 \\ u_j \\ v_j \end{bmatrix}^{\mathsf{T}} \right) = \begin{bmatrix} \Pi_{0|0} & 0 & 0 \\ 0 & Q\,\delta_{ij} & 0 \\ 0 & 0 & R\,\delta_{ij} \end{bmatrix}. \tag{2}$$

Because the true motion can be highly nonlinear, model (1) rarely yields the desired predictive accuracy. Although the Kalman estimator provides on-line estimates of v_i and a_i, they are ultimately reconstructed from past position measurements. With scalar gains $w_{v1}, w_{v2}, w_{a1}, w_{a2}$ from calculated Kalman gain, the recursive updates read

$$\begin{aligned} v_i =& w_{v1} \frac{p_i - p_{i-1}}{T} + w_{v2} v_{i-1} \\ =& w_{v1} \frac{p_i - p_{i-1}}{T} + w_{v1} w_{v2} \frac{p_{i-1} - p_{i-2}}{T} + w_{v1} w_{v2}^2 \frac{p_{i-2} - p_{i-3}}{T} + \ldots \\ =& w_{v1} \sum_{k=0}^{i-1} w_{v2}^k \, \frac{p_{\,i-k} - p_{\,i-k-1}}{T}, \\ a_i =& w_{a1} \frac{p_i - 2p_{i-1} + p_{i-2}}{T^2} + w_{a2} a_{i-1} \\ =& w_{a1} \frac{p_i - 2p_{i-1} + p_{i-2}}{T^2} + w_{a1} w_{a2} \frac{p_{i-1} - 2p_{i-2} + p_{i-3}}{T^2} \\ &+ w_{a1} w_{a2}^2 \frac{p_{i-2} - 2p_{i-3} + p_{i-4}}{T^2} + \ldots \\ =& w_{a1} \sum_{k=0}^{i-1} w_{a2}^k \, \frac{p_{\,i-k} - 2p_{\,i-k-1} + p_{\,i-k-2}}{T^2}. \end{aligned} \tag{3}$$

Given p_i, v_i, a_i, an n-step-ahead position prediction follows from kinematics:

$$p_{i+n} = p_i + v_i\, nT + \tfrac{1}{2} a_i\, n^2 T^2. \tag{4}$$

Substituting (3) into (4) yields a weighted finite-difference expansion

$$\begin{aligned} p_{i+n} = {}& p_i + nTw_{v1} \sum_{k=0}^{i-1} w_{v2}^k \frac{p_{\,i-k} - p_{\,i-k-1}}{T} \\ & + \tfrac{1}{2} n^2 T^2 w_{a1} \sum_{k=0}^{i-1} w_{a2}^k \frac{p_{\,i-k} - 2p_{\,i-k-1} + p_{\,i-k-2}}{T^2} = \sum_{k=0}^{i} w_k p_k, \end{aligned} \tag{5}$$

Under model (1), long-term prediction ultimately reduces to a weighted sum of past positions, an intrinsic limitation when the target executes nonlinear manoeuvres. To validate the potential of position-only estimated states, we construct the corresponding state–space models.

We begin with the *two–position estimator* (E2P), based on

$$p_{i+1} = p_i + (p_i - p_{i-1}),$$

which yields

$$x_{i+1} = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} x_i + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_i, \quad z_i = \begin{bmatrix} 1 & 0 \end{bmatrix} x_i + v_i, \tag{6}$$

with $x_i = [p_i, p_{i-1}]^\mathsf{T}$.

By extending the position history, we obtain higher-order variants. Defining $x_i = [p_i, p_{i-1}, p_{i-2}]^\mathsf{T}$ gives the *three–position estimator* (E3P), and $x_i = [p_i, p_{i-1}, p_{i-2}, p_{i-3}]^\mathsf{T}$ yields the *four–position estimator* (E4P):

$$\begin{aligned} p_{i+1} &= p_i + \tfrac{1}{2}(p_i - p_{i-1}) + \tfrac{1}{2}(p_{i-1} - p_{i-2}), \\ p_{i+1} &= p_i + \tfrac{1}{3}(p_i - p_{i-1}) + \tfrac{1}{3}(p_{i-1} - p_{i-2}) + \tfrac{1}{3}(p_{i-2} - p_{i-3}). \end{aligned} \tag{7}$$

A *variable–weight four–position estimator* (E4PVW) further refines the weights:

$$p_{i+1} = p_i + \frac{3(p_i - p_{i-1})}{6} + \frac{2(p_{i-1} - p_{i-2})}{6} + \frac{(p_{i-2} - p_{i-3})}{6}. \tag{8}$$

For even greater flexibility, the weights may be *learned* from data. The offline *four–position estimator with regressed weights* (E4PRW) uses

$$p_{i+1} = 1.2668\, p_i - 0.0152\, p_{i-1} + 0.0103\, p_{i-2} - 0.2618\, p_{i-3}, \tag{9}$$

while its online counterpart (E4PTRW) updates these coefficients at each step using the most recent 50 samples.

Simulation results in Figs. 1 and 2 show that all position-stack estimators outperform classical Kalman filters built on $[p_i]$, $[p_i, v_i]$, $[p_i, v_i, a_i]$, and $[p_i, v_i, a_i, \dot{a}_i]$ (the latter

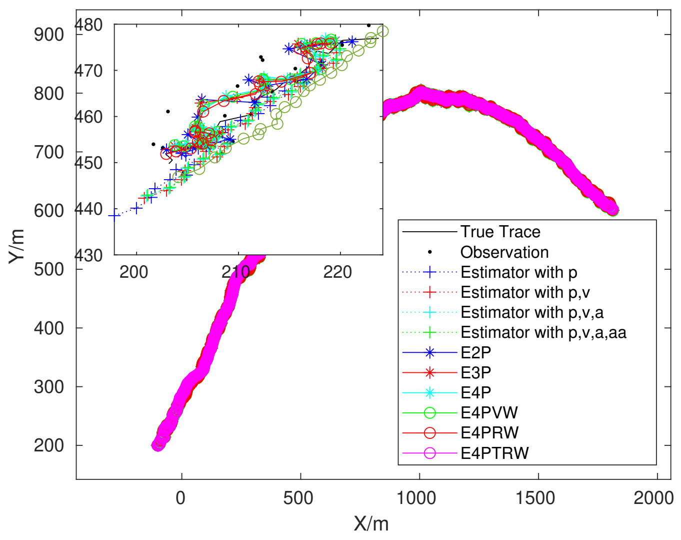

Fig. 1 Position prediction obtained with different estimators.

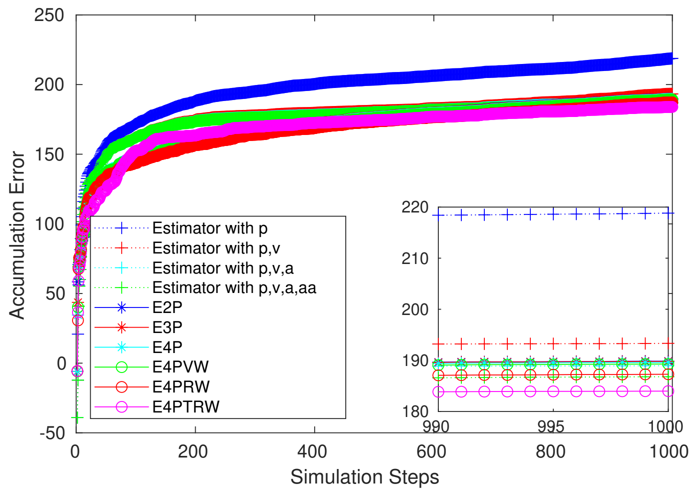

Fig. 2 Accumulated prediction error for the same estimators.

including jerk $\dot{a}_i$). Accuracy increases with longer stacks (E3P, E4P) and peaks for variants with adaptive or regressed weights (E4PVW, E4PRW, E4PTRW), with E4PRW notably narrowing the error band and E4PTRW further enhancing adaptability.

Despite these gains, embedding a dedicated linear-regression step in each prediction is conceptually inelegant and risks overfitting when the target's motion pattern changes abruptly. A more principled solution is to regard the regression coefficients

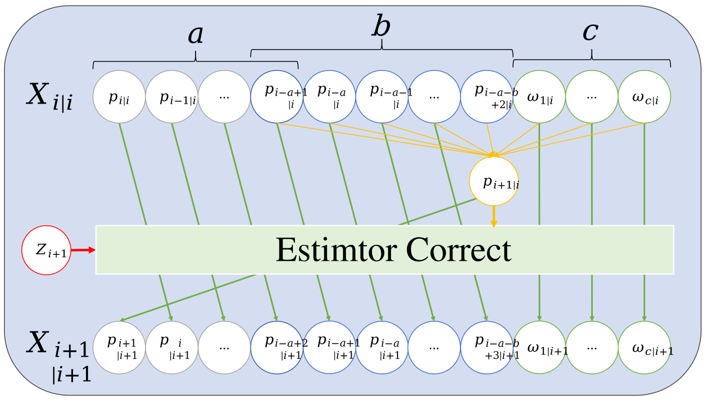

Fig. 3 Neural network state-space model.

themselves as *latent states*, updating them in real time alongside the kinematic variables. Accordingly, we extend the state vector to include not only these coefficients but, when appropriate, the full set of weights of a neural network approximation of the target dynamics. These augmented states are then estimated online using a neural network state–space model(NNSSM).

3 Neural Network State-Space Model

Based on the concept of treating neural network weights as estimated states, a new state-space model to achieve state estimation for maneuverable targets is introduced as (10) and Figs. 3.

$$\begin{aligned} x_{i+1} &= f_i(x_i, u_i) = f_i([p_i, ..., p_{i-a+1}, ..., p_{i-a-b+2}, w_1, ..., w_c]^\mathsf{T}, u_i) \\ &= [p_{i+1}, p_i, ..., p_{i-a-b+3}, w_1, ..., w_c]^\mathsf{T} + u_i \\ z_i &= h_i(x_i, v_i) = p_i + v_i \end{aligned} \tag{10}$$

In the proposed model, the state vector

$$x_{i+1} = [p_i, ..., p_{i-a+1}, ..., p_{i-a-b+2}, w_1, ..., w_c]^\mathsf{T} \tag{11}$$

comprises $(a-1)+b+c$ elements in total, namely $a+b-1$ historical position states and c weight parameters. Here:

- a is the prediction horizon in discrete steps. For example, if the sampling interval is $T = 0.01$s and a 0.1s ahead prediction is required, then $a = 10$;
- b is the number of input-layer nodes of the neural network being simulated;
- c is the total number of weight parameters in that network.

These $(a-1)+b+c$ states are jointly estimated online via the NNSSM.
Let

$$x_i = [P_{i,1\times(a+b-2)}, p_{i-a-b+2}, W_{1\times c}]^\mathsf{T}, \tag{12}$$

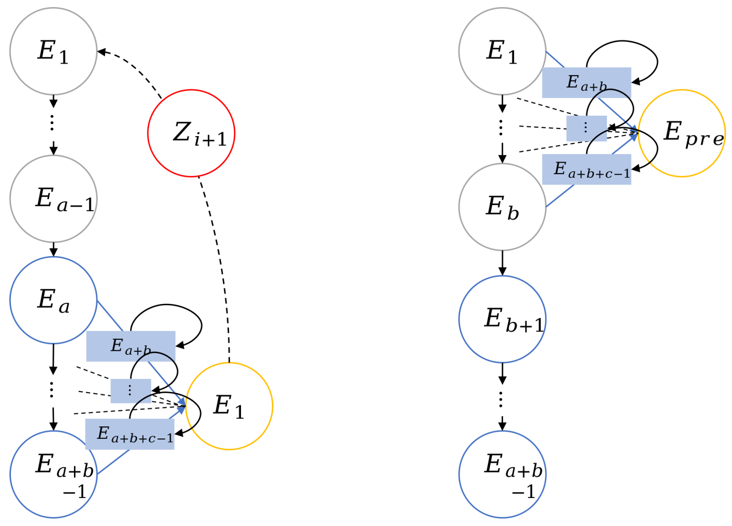

Fig. 4 Simplest neural–network state–space model: weighted-sum iteration.

and denote by $f_{\mathrm{NN}}(\cdot)$ the neural–network mapping from the previous state to the new predicted position p_{i+1}. A more detailed state–space model can then be written as

$$\begin{aligned} x_{i+1} &= f_i(x_i, u_i) \\ &= \left[f_{\mathrm{NN}}\big(P_{i,\,1\times(a+b-2)},\, W_{1\times c}\big),\ P_{i,\,1\times(a+b-2)},\ W_{1\times c}\right]^{\mathsf{T}} + u_i \\ &= \left[p_{i+1},\ P_{i,\,1\times(a+b-2)},\ W_{1\times c}\right]^{\mathsf{T}}, \\ z_i &= p_i + v_i. \end{aligned} \tag{13}$$

Figure 4 illustrates the iterative structure of the simplest neural network, namely a single weighted–sum node. Let E_j denote the j-th element of the estimated state vector x_i and let E_{pre} represent the future position to be predicted at the each step. The corresponding state–space representation is

$$\begin{aligned} x_{i+1} &= f_i\big(x_i, u_i\big) \\ &= \Big[P_{i,\,1\times(a+b-2)}\, W_{1\times c}^{\mathsf{T}},\ P_{i,\,1\times(a+b-2)},\ W_{1\times c}\Big]^{\mathsf{T}} + u_i \\ &= \big[p_{i+1},\ P_{i,\,1\times(a+b-2)},\ W_{1\times c}\big]^{\mathsf{T}}, \\ z_i &= p_i + v_i. \end{aligned} \tag{14}$$

At time instant i, the algorithm forms a weighted sum of the estimated historical positions $E_a, \ldots, E_{a+b-1}$ and the estimated weights $E_{a+b}, \ldots, E_{a+b+c-1}$ to obtain the prior predicted state $x_{i+1|i}$, specifically its first element E_1.

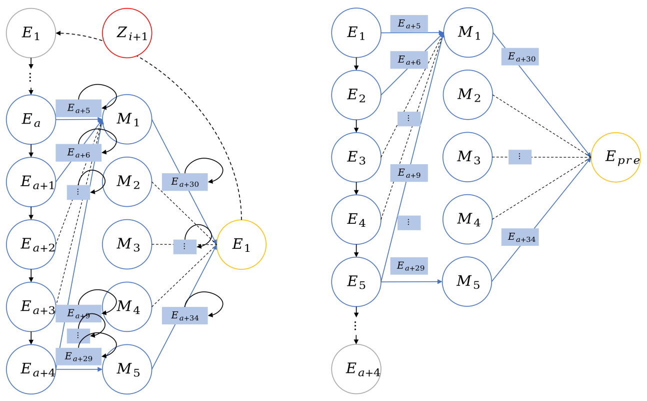

Fig. 5 Simple neural–network state–space model: 5–5–1 architecture.

Upon receiving the new measurement z_{i+1}, the estimator updates all state components E_j, yielding the posterior estimated state $x_{i+1|i+1}$ and completing the $i \to i+1$ cycle.

For multi–frame prediction, the most recent position estimates $E_1, \dots, E_b$ together with the current weight estimates $E_{a+b}, \dots, E_{a+b+c-1}$ are fed into the same weighted-sum relation to generate the required forecast.

Although the weighted-sum model provides a conceptual bridge to neural networks, it lacks the representational richness of even a modest multi-layer perceptron such as the 5–5–1 architecture shown in Fig. 5, which depicts a simple fully connected feedforward network with five input nodes, one hidden layer of five neurons, and a single output node (a "5–5–1" architecture).

To perform an a–frame-ahead prediction, the augmented state vector must include $a+4$ past positions plus the network's weight parameters, for a total of $a+34$ estimated states. Here, M_j denotes the activation of the jth hidden neuron. The mapping $f_{\mathrm{NN}}(\cdot)$ in (13) is a nonlinear function implemented by a multi-layer recurrent architecture.

The precise layer sizes, connectivity pattern, and choice of activation functions should be determined based on the target application and available computational resources.

4 Implementation of the Neural Network State–Space Model

Having formulated the NNSSM, we now turn to its realization. This section presents an implementation based on the neural network state–space model and classical unscented Kalman estimator(NNSSUKE).

4.1 Parameter Initiate

Based on the needs of target tracking, NNSSUKE needs to preinstall the length of the estimated state $\hat{x}_i$ with n-dimension, as well as the spread parameter α , incorporating parameter β, and scaling parameter κ to calculate the scaling parameter λ (15). We also need to set the initial value of the state estimation $\hat{x}_0$, the initial value of the posterior covariance of the state estimation error $\Pi_{0|0}$, the covariance of the observation noise R, and the covariance of the process noise Q.

$$\lambda = \alpha^2(n+\kappa) - n \tag{15}$$

4.2 Obtain Sigma Points and Their Weights Based on Posterior Estimated State

For the estimated n-dimensional state $\hat{x}_i$ and its prior estimation error covariance Π_i, a set of Sigma points $X_{(2n+1)}$ is obtained using the unscented transform rule with the calculated parameter λ. The set contains 2n+1 Sigma points, and the state value weight and estimation error covariance weight of each Sigma point are calculated using following formulas, where the i-th column of the square root of the matrix A is denoted as $(\sqrt{A})_i$. The Sigma point values are calculated as (16):

$$X_{(l)} = \begin{cases} X_{(l)} = \hat{x}_i, & l = 0; \\ X_{(l)} = \hat{x}_i + (\sqrt{(n+\lambda)\Pi_i})_l, & l = 1 \sim n; \\ X_{(l)} = \hat{x}_i - (\sqrt{(n+\lambda)\Pi_i})_l, & l = n+1 \sim 2n. \end{cases} \tag{16}$$

The state weight of Sigma points are calculated as (17):

$$w_S^{(0)} = \frac{\lambda}{(n+\lambda)},\ w_S^{(l)} = \frac{\lambda}{2(n+\lambda)},\ l = 1 \sim 2n. \tag{17}$$

The estimation error covariance weight of Sigma points are calculated as (18):

$$\begin{aligned} w_\Pi^{(0)} &= \frac{\lambda}{(n+\lambda)} + (1-\alpha^2-\beta), \\ w_\Pi^{(l)} &= \frac{\lambda}{2(n+\lambda)},\ l = 1 \sim 2n. \end{aligned} \tag{18}$$

4.3 State Transit on Sigma Points

The state transition equation $f_i(\cdot)$ as (13) should be used to predict the state of $2n+1$ Sigma points and obtain a new set of points $\hat{X}_{(2n+1)}$ through (19).

$$\begin{aligned} &\hat{X}_{(l)} = f(X_{(l)}) = f([p_i, p_{i-1}, p_{i-2}, ..., w_1, w_2, w_3, ...]^T) \\ &= [w_1 p_i + w_2 p_{i-1} + w_3 p_{i-2}, p_i, p_{i-1}, ..., w_1, w_2, w_3, ...]^T \end{aligned} \tag{19}$$

4.4 Calculate the Transited State and Its Covariance

Based on the state value weights and covariance weights of the point set $\hat{X}_{(2n+1)}$, the weighted sum of the state $\hat{x}_{i+1|i}$ and covariance $\Pi_{i+1|i}$ is obtained by (20).

$$
\begin{aligned}
\hat{x}_{i+1|i} &= \sum_{l=0}^{2n} w_S^{(l)} \hat{X}_{(l)} \\
\Pi_{i+1|i} &= \sum_{l=0}^{2n} w_\Pi^{(l)} [\hat{x}_{i+1|i} - \hat{X}_{(l)}][\hat{x}_{i+1|i} - \hat{X}_{(l)}]^T + Q
\end{aligned}
\tag{20}
$$

4.5 Obtain Sigma Points and Their Weights Based on Prior Estimated State

Appling the same unscented transform as (16) to the prior state estimate $\hat{x}_{i+1|i}$ and its estimated error covariance $\Pi_{i+1|i}$, the point set $\tilde{X}_{(2n+1)}$ is obtained (21). The weights of the estimated state or error covariance for each sigma point are the same as the value in the (17)(18), and do not need to be calculated again.

$$
\begin{cases}
X_{(l)} = \hat{x}_i, & l = 0; \\
X_{(l)} = \hat{x}_i + (\sqrt{(n+\lambda)\Pi_i})_l, & l = 1 \sim n; \\
X_{(l)} = \hat{x}_i - (\sqrt{(n+\lambda)\Pi_i})_l, & l = n+1 \sim 2n.
\end{cases}
\tag{21}
$$

4.6 State Observe on Sigma Points

The observation equation $h(\cdot)$ as (13) should be used to observe the state of each Sigma point $\tilde{X}_{(l)}$, resulting in a new set of points $Z_{(2n+1)}$, which can be expressed as:

$$
Z_{(l)} = h(\tilde{X}_{(l)}).
\tag{22}
$$

4.7 Calculate the Estimated Observation and Its Covariance

Similarly, based on the state weight $w_S^{(l)}$ and covariance weight $w_\Pi^{(l)}$ of point set $Z_{(2n+1)}$, the estimated observation $\hat{z}_{i+1}$, intermediate covariance matrix Π_{zz} and Π_{xz} are obtained through weighted summation.

$$
\begin{aligned}
\hat{z}_{i+1} &= \sum_{l=0}^{2n} w_S^{(l)} Z_{(l)} \\
\Pi_{zz} &= \sum_{l=0}^{2n} w_\Pi^{(l)} [Z^{(l)} - \hat{z}_{i+1}][Z^{(l)} - \hat{z}_{i+1}]^T + R \\
\Pi_{xz} &= \sum_{l=0}^{2n} w_\Pi^{(l)} [\tilde{X}^{(l)} - \hat{z}_{i+1}][Z^{(l)} - \hat{z}_{i+1}]^T
\end{aligned}
\tag{23}
$$

4.8 Update the Estimated State and Its Covariance

The Kalman gain matrix K is obtained by:

$$
K = \Pi_{xz}\Pi_{zz}^{-1}.
\tag{24}
$$

Then the updated estimation of the state $\hat{x}_{i+1}$ and its covariance $\Pi_{i+1|i+1}$ can be obtained as follow:

$$\begin{aligned}
&\hat{x}_{i+1} = \hat{x}_{i+1|i} + K\left(z_{i+1} - \hat{z}_{i+1}\right) \\
&\Pi_{i+1|i+1} = \Pi_{i+1|i} - K\Pi_{zz}K^{T}.
\end{aligned} \tag{25}$$

4.9 Obtain Target Position Prediction

After obtaining the posterior state estimate $\hat{x}_{i+1}$, the position sequence p_i and weight sequence w_i can be extracted to perform state transition and observation based on the state space model, which results in a position prediction $\hat{p}$ of the tracked target. If multiple-step predictions are required, multiple state transition equations could be nested and followed with state observation equation. Once the target position prediction is obtained, the algorithm should proceed to step 2 for another iteration of computation.

$$\hat{p} = h(f(\hat{x}_{i+1})) \tag{26}$$

The complete steps of NNSSUKE is demonstrated above. It's clear, reliable, easy to understand and apply.

5 This is an example for first level head—section head

5.1 This is an example for second level head—subsection head

5.1.1 This is an example for third level head—subsubsection head

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

6 Equations

Equations in LaTeX can either be inline or on-a-line by itself ("display equations"). For inline equations use the `$...$` commands. E.g.: The equation $H\psi = E\psi$ is written via the command `$H \psi = E \psi$`.

For display equations (with auto generated equation numbers) one can use the equation or align environments:

$$\|\tilde{X}(k)\|^2 \leq \frac{\sum\limits_{i=1}^{p}\left\|\tilde{Y}_i(k)\right\|^2 + \sum\limits_{j=1}^{q}\left\|\tilde{Z}_j(k)\right\|^2}{p+q}. \tag{27}$$

where,

$$D_\mu = \partial_\mu - ig\frac{\lambda^a}{2}A^a_\mu$$

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^a_\nu \tag{28}$$

Notice the use of \nonumber in the align environment at the end of each line, except the last, so as not to produce equation numbers on lines where no equation numbers are required. The \label{} command should only be used at the last line of an align environment where \nonumber is not used.

$$Y_\infty = \left(\frac{m}{\text{GeV}}\right)^{-3} \left[1 + \frac{3\ln(m/\text{GeV})}{15} + \frac{\ln(c_2/5)}{15}\right] \tag{29}$$

The class file also supports the use of \mathbb{}, \mathscr{} and \mathcal{} commands. As such \mathbb{R}, \mathscr{R} and \mathcal{R} produces $\mathbb{R}$, $\mathscr{R}$ and $\mathcal{R}$ respectively (refer Subsubsection 5.1.1).

7 Tables

Tables can be inserted via the normal table and tabular environment. To put footnotes inside tables you should use \footnotetext[]{...} tag. The footnote appears just below the table itself (refer Tables 1 and 2). For the corresponding footnotemark use \footnotemark[...]

Table 1 Caption text

Column 1	Column 2	Column 3	Column 4
row 1	data 1	data 2	data 3
row 2	data 4	data 5[1]	data 6
row 3	data 7	data 8	data 9[2]

Source: This is an example of table footnote. This is an example of table footnote.

[1]Example for a first table footnote. This is an example of table footnote.

[2]Example for a second table footnote. This is an example of table footnote.

The input format for the above table is as follows:

```
\begin{table}[<placement-specifier>]
\caption{<table-caption>}\label{<table-label>}%
\begin{tabular}{@{}llll@{}}
\toprule
Column 1 & Column 2 & Column 3 & Column 4\\
\midrule
row 1 & data 1 & data 2  & data 3 \\
row 2 & data 4 & data 5\footnotemark[1] & data 6 \\
row 3 & data 7 & data 8  & data 9\footnotemark[2]\\
\botrule
\end{tabular}
\footnotetext{Source: This is an example of table footnote.
This is an example of table footnote.}
\footnotetext[1]{Example for a first table footnote.
This is an example of table footnote.}
\footnotetext[2]{Example for a second table footnote.
This is an example of table footnote.}
\end{table}
```

Table 2 Example of a lengthy table which is set to full textwidth

	Element 1[1]			Element 2[2]		
Project	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote. This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]Example for a first table footnote.

[2]Example for a second table footnote.

In case of double column layout, tables which do not fit in single column width should be set to full text width. For this, you need to use `\begin{table*}` ... `\end{table*}` instead of `\begin{table}` ... `\end{table}` environment. Lengthy tables which do not fit in textwidth should be set as rotated table. For this, you need to use `\begin{sidewaystable}` ... `\end{sidewaystable}` instead of `\begin{table*}` ... `\end{table*}` environment. This environment puts tables rotated to single column width. For tables rotated to double column width, use `\begin{sidewaystable*}` ... `\end{sidewaystable*}`.

8 Figures

As per the LaTeX standards you need to use eps images for LaTeX compilation and `pdf/jpg/png` images for `PDFLaTeX` compilation. This is one of the major difference between LaTeX and `PDFLaTeX`. Each image should be from a single input .eps/vector image file. Avoid using subfigures. The command for inserting images for LaTeX and `PDFLaTeX` can be generalized. The package used to insert images in `LaTeX/PDFLaTeX` is the graphicx package. Figures can be inserted via the normal figure environment as shown in the below example:

```
\begin{figure}[<placement-specifier>]
\centering
\includegraphics{<eps-file>}
\caption{<figure-caption>}\label{<figure-label>}
```

Table 3 Tables which are too long to fit, should be written using the "sidewaystable" environment as shown here

	Element 1[1]			Element[2]		
Projectile	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40
Element 5	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 6	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]This is an example of table footnote.

```
\end{figure}
```

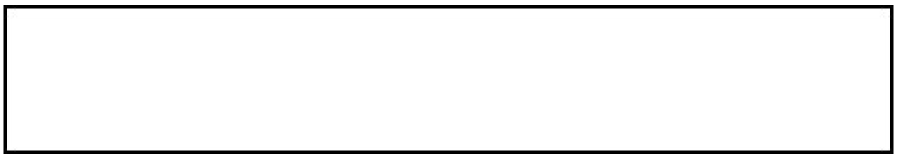

Fig. 6 This is a widefig. This is an example of long caption this is an example of long caption this is an example of long caption this is an example of long caption

In case of double column layout, the above format puts figure captions/images to single column width. To get spanned images, we need to provide `\begin{figure*}` ... `\end{figure*}`.

For sample purpose, we have included the width of images in the optional argument of `\includegraphics` tag. Please ignore this.

9 Algorithms, Program codes and Listings

Packages `algorithm`, `algorithmicx` and `algpseudocode` are used for setting algorithms in LaTeX using the format:

```
\begin{algorithm}
\caption{<alg-caption>}\label{<alg-label>}
\begin{algorithmic}[1]
. . .
\end{algorithmic}
\end{algorithm}
```

You may refer above listed package documentations for more details before setting `algorithm` environment. For program codes, the “verbatim” package is required and the command to be used is `\begin{verbatim}` ... `\end{verbatim}`.

Similarly, for `listings`, use the `listings` package. `\begin{lstlisting}` ... `\end{lstlisting}` is used to set environments similar to `verbatim` environment. Refer to the `lstlisting` package documentation for more details.

A fast exponentiation procedure:

```
begin
  for $i := 1$ to $10$ step $1$ do
      expt($2, i$);
      newline() od                    Comments will be set flush to the right margin
where
proc expt($x, n$) $\equiv$
  $z := 1$;
  do if $n = 0$ then exit fi;
     do if odd($n$) then exit fi;
```

```
            comment:  This is a comment statement;
            $n := n/2$;  $x := x * x$  od;
        {  $n > 0$  };
        $n := n - 1$;  $z := z * x$  od;
    print($z$).
end
```

Algorithm 1 Calculate $y = x^n$

```
Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
 1: $y \Leftarrow 1$
 2: if $n < 0$ then
 3:     $X \Leftarrow 1/x$
 4:     $N \Leftarrow -n$
 5: else
 6:     $X \Leftarrow x$
 7:     $N \Leftarrow n$
 8: end if
 9: while $N \neq 0$ do
10:     if $N$ is even then
11:         $X \Leftarrow X \times X$
12:         $N \Leftarrow N/2$
13:     else[$N$ is odd]
14:         $y \Leftarrow y \times X$
15:         $N \Leftarrow N - 1$
16:     end if
17: end while
```

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
```

10 Cross referencing

Environments such as figure, table, equation and align can have a label declared via the `\label{#label}` command. For figures and table environments use the `\label{}` command inside or just below the `\caption{}` command. You can then use the `\ref{#label}` command to cross-reference them. As an example, consider the label declared for Figure 6 which is `\label{fig1}`. To cross-reference it, use the command `Figure \ref{fig1}`, for which it comes up as "Figure 6".

To reference line numbers in an algorithm, consider the label declared for the line number 2 of Algorithm 1 is `\label{algln2}`. To cross-reference it, use the command `\ref{algln2}` for which it comes up as line 2 of Algorithm 1.

10.1 Details on reference citations

Standard LaTeX permits only numerical citations. To support both numerical and author-year citations this template uses `natbib` LaTeX package. For style guidance please refer to the template user manual.

Here is an example for `\cite{...}`: [1]. Another example for `\citep{...}`: [2]. For author-year citation mode, `\cite{...}` prints Jones et al. (1990) and `\citep{...}` prints (Jones et al., 1990).

All cited bib entries are printed at the end of this article: [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] and [13].

11 Examples for theorem like environments

For theorem like environments, we require `amsthm` package. There are three types of predefined theorem styles exists—`thmstyleone`, `thmstyletwo` and `thmstylethree`

`thmstyleone`	Numbered, theorem head in bold font and theorem text in italic style
`thmstyletwo`	Numbered, theorem head in roman font and theorem text in italic style
`thmstylethree`	Numbered, theorem head in bold font and theorem text in roman style

For mathematics journals, theorem styles can be included as shown in the following examples:

Theorem 1 (Theorem subhead)**.** *Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proposition 2. *Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Example 1. *Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Remark 1. *Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Definition 1 (Definition sub head)**.** *Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text.*

Additionally a predefined "proof" environment is available: `\begin{proof} ... \end{proof}`. This prints a "Proof" head in italic font style and the "body text" in roman font style with an open square at the end of each proof environment.

Proof. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proof of Theorem 1. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

For a quote environment, use `\begin{quote}...\end{quote}`

> Quoted text example. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text (refer Figure 6). Sample body text. Sample body text. Sample body text (refer Table 3).

12 Methods

Topical subheadings are allowed. Authors must ensure that their Methods section includes adequate experimental and characterization data necessary for others in the field to reproduce their work. Authors are encouraged to include RIIDs where appropriate.

Ethical approval declarations (only required where applicable) Any article reporting experiment/s carried out on (i) live vertebrate (or higher invertebrates), (ii) humans or (iii) human samples must include an unambiguous statement within the methods section that meets the following requirements:

1. Approval: a statement which confirms that all experimental protocols were approved by a named institutional and/or licensing committee. Please identify the approving body in the methods section
2. Accordance: a statement explicitly saying that the methods were carried out in accordance with the relevant guidelines and regulations
3. Informed consent (for experiments involving humans or human tissue samples): include a statement confirming that informed consent was obtained from all participants and/or their legal guardian/s

If your manuscript includes potentially identifying patient/participant information, or if it describes human transplantation research, or if it reports results of a clinical trial then additional information will be required. Please visit (https://www.nature.com/nature-research/editorial-policies) for Nature Portfolio journals, (https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/publishing-ethics/14214) for Springer Nature journals, or (https://www.biomedcentral.com/getpublished/editorial-policies#ethics+and+consent) for BMC.

13 Discussion

Discussions should be brief and focused. In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Some journals prefer a section 'Results and Discussion' followed by a section 'Conclusion'. Please refer to Journal-level guidance for any specific requirements.

14 Conclusion

Conclusions may be used to restate your hypothesis or research question, restate your major findings, explain the relevance and the added value of your work, highlight any limitations of your study, describe future directions for research and recommendations.

In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Please refer to Journal-level guidance for any specific requirements.

Supplementary information. If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

Acknowledgements. Acknowledgements are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
- Ethics approval and consent to participate
- Consent for publication
- Data availability
- Materials availability
- Code availability
- Author contribution

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

Editorial Policies for:

Springer journals and proceedings: https://www.springer.com/gp/editorial-policies

Nature Portfolio journals: https://www.nature.com/nature-research/editorial-policies

Scientific Reports: https://www.nature.com/srep/journal-policies/editorial-policies

BMC journals: https://www.biomedcentral.com/getpublished/editorial-policies

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

[1] Campbell, S. L. & Gear, C. W. The index of general nonlinear DAES. *Numer. Math.* **72**, 173–196 (1995).

[2] Slifka, M. K. & Whitton, J. L. Clinical implications of dysregulated cytokine production. *J. Mol. Med.* **78**, 74–80 (2000).

[3] Hamburger, C. Quasimonotonicity, regularity and duality for nonlinear systems of partial differential equations. *Ann. Mat. Pura. Appl.* **169**, 321–354 (1995).

[4] Geddes, K. O., Czapor, S. R. & Labahn, G. *Algorithms for Computer Algebra* (Kluwer, Boston, 1992).

[5] Broy, M. in *Software engineering—from auxiliary to key technologies* (eds Broy, M. & Denert, E.) *Software Pioneers* 10–13 (Springer, New York, 1992).

[6] Seymour, R. S. (ed.) *Conductive Polymers* (Plenum, New York, 1981).

[7] Smith, S. E. Zaimis, E. (ed.) *Neuromuscular blocking drugs in man.* (ed.Zaimis, E.) *Neuromuscular junction. Handbook of experimental pharmacology*, Vol. 42, 593–660 (Springer, Heidelberg, 1976).

[8] Chung, S. T. & Morris, R. L. Isolation and characterization of plasmid deoxyribonucleic acid from streptomyces fradiae (1978). Paper presented at the 3rd international symposium on the genetics of industrial microorganisms, University of Wisconsin, Madison, 4–9 June 1978.

[9] Hao, Z., AghaKouchak, A., Nakhjiri, N. & Farahmand, A. Global integrated drought monitoring and prediction system (gidmaps) data sets (2014). Figshare https://doi.org/10.6084/m9.figshare.853801.

[10] Babichev, S. A., Ries, J. & Lvovsky, A. I. Quantum scissors: teleportation of single-mode optical states by means of a nonlocal single photon (2002). Preprint at https://arxiv.org/abs/quant-ph/0208066v1.

[11] Beneke, M., Buchalla, G. & Dunietz, I. Mixing induced CP asymmetries in inclusive B decays. *Phys. Lett.* **B393**, 132–142 (1997).

[12] Stahl, B. deepSIP: deep learning of Supernova Ia Parameters, 0.42. Astrophysics Source Code Library (2020). ascl:2006.023.

[13] Abbott, T. M. C. *et al.* Dark Energy Survey Year 1 Results: Constraints on Extended Cosmological Models from Galaxy Clustering and Weak Lensing. *Phys. Rev. D* **99**, 123505 (2019).