Kalman Filter-based Estimators for Dual Adaptive Neural Control
A Comparative Analysis of Execution Time and Performance Issues

Simon G. Fabri and Marvin K. Bugeja
Department of Systems and Control Engineering, University of Malta, Msida, Malta

Keywords: Dual Adaptive Control, Kalman Filter, Neural Networks, Intelligent Control.

Abstract: The real time implementation of neural network-based dual adaptive control for nonlinear systems can become significantly demanding because of the amount of network parameters requiring estimation. This paper explores the effect of three different estimation algorithms for dual adaptive control of a class of multiple-input, multiple-output nonlinear systems in terms of tracking performance and execution time. It is shown that the Unscented and Square-root Unscented Kalman filter estimators lead to a significant improvement in tracking performance when compared with the Extended Kalman filter, but with an appreciable increase in execution time. Such issues need to be given due consideration when implementing controllers for on-line operation.

1 INTRODUCTION

The use of dual adaptive techniques for neural network closed-loop control of nonlinear systems has been investigated from different perspectives in the recent past (Bugeja and Fabri, 2007; Bugeja et al., 2009; Šimandl et al., 2005). Inspired by the pioneering work of Fel'dbaum (Fel'dbaum, 1965), these dual adaptive methods handle joint estimation and control of nonlinear systems that are subject to functional uncertainty by characterizing them within a stochastic framework.

Neural networks are used to learn the unknown nonlinear functions in real-time, whereas the dual controller generates system inputs that exhibit two desirable properties: (a) caution; whereby the uncertainty of the neural network estimates is taken into consideration by the controller so as to maintain good tracking capabilities, and (b) probing; whereby a component of the input is used to excite the system so that the neural network training algorithm reduces the functional uncertainty rapidly and efficiently (Fabri and Kadirkamanathan, 2001; Filatov and Unbehauen, 2004).

The dual adaptive control paradigm is rooted in the methodology of stochastic estimation and control. Stochastic algorithms are employed for estimation of the neural network parameters which often appear in nonlinear form. Such estimators effectively "train" the neural networks to capture the system's nonlinear functions recursively in real-time, based on measurements of its inputs and outputs. Nonlinear stochastic estimators which have been proposed in the dual adaptive control literature include the Extended Kalman filter (Fabri and Kadirkamanathan, 1998), the Unscented Kalman filter (Bugeja and Fabri, 2009) and the Gaussian Sum filter (Šimandl et al., 2005).

The aim of this paper is to investigate and compare the computational demand of a select set of nonlinear estimation algorithms when applied within the context of neural network, dual adaptive control of a class of uncertain, nonlinear, multiple-input/multiple-output dynamic systems. Computational demand analysis is a crucial consideration for the implementation of control algorithms which operate on computer hardware in real-time (Åström and Wittenmark, 2011). The discrete-time nature of such digital control systems entails that the control law, including the estimation algorithm, executes, calculates and generates a fresh control signal with minimal delay at every sampling instant. As a consequence, the estimation and control algorithms must execute well within the sampling period at which the discrete-time control law is operating.

The analysis reported in this work would be of help for designers to judge the feasibility of implementing a particular estimation algorithm for real-time, dual adaptive neuro-control. Three nonlinear estimation techniques are considered and analysed in this paper - the Extended, Unscented and Square-root Unscented Kalman Filter algorithms - all three being variations of the well-known Kalman filter method-

Fabri S. and Bugeja M..
Kalman Filter-based Estimators for Dual Adaptive Neural Control - A Comparative Analysis of Execution Time and Performance Issues.
DOI: 10.5220/0004455601690176
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 169-176
ISBN: 978-989-8565-70-9
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

ology for the nonlinear case. These algorithms are evaluated by testing their effect on the execution time and on the capability of the dual adaptive controller to meet the control system's performance requirements when embedded within the closed loop control system.

The rest of the paper is organized as follows: Section 2 presents the background of the dual adaptive control problem for nonlinear systems and describes the control and estimation algorithms. This is followed by Section 3 where the performance and computational demands of the different estimation algorithms are tested and the results are analysed. Conclusions are provided in Section 4.

2 ESTIMATION AND CONTROL

In this work we consider multiple-input, multiple-output (MIMO) systems with nonlinear dynamics of the following form:

$$\boldsymbol{y}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{G}(\boldsymbol{x}_{k-1})\boldsymbol{u}_{k-1} + \boldsymbol{\varepsilon}_k \tag{1}$$

where $\boldsymbol{y}_k \in \mathbb{R}^s$ is a vector of s outputs, $\boldsymbol{u}_k \in \mathbb{R}^s$ is an s-input control vector, $\boldsymbol{x}_{k-1} := \left[\boldsymbol{y}_{k-n}^T \cdots \boldsymbol{y}_{k-1}^T \quad \boldsymbol{u}_{k-1-p}^T \cdots \boldsymbol{u}_{k-2}^T\right]^T \in \mathbb{R}^{s(n+p)}$ is the system state vector (N.B. $0 \leq p \leq n$ and the $\boldsymbol{u}$ terms vanish from the state vector if $p = 0$), vector field $\boldsymbol{f}(\boldsymbol{x}_{k-1}) : \mathbb{R}^{s(n+p)} \mapsto \mathbb{R}^s$ and matrix $\boldsymbol{G}(\boldsymbol{x}_{k-1}) : \mathbb{R}^{s(n+p)} \mapsto \mathbb{R}^{s \times s}$ contain the unknown nonlinear functionals governing the system dynamics, and $\boldsymbol{\varepsilon}_k \in \mathbb{R}^s$ represents an additive output white noise signal assumed to be zero-mean Gaussian of covariance $\boldsymbol{R}_{\boldsymbol{\varepsilon}}$.

The control objective is for the output vector $\boldsymbol{y}_k$ to track a reference input vector $\boldsymbol{y}_{\boldsymbol{d}k} \in \mathbb{R}^s$ despite the uncertainty in the nonlinear functionals comprising $\boldsymbol{f}$ and $\boldsymbol{G}$.

2.1 Neural Network Estimation Model

Two multilayer perceptron (MLP) neural networks are used to estimate and approximate the nonlinear system functionals within an arbitrarily large compact set $\boldsymbol{\chi}$ in which the state vector is known to be contained. The estimates from the neural networks will be utilized within a dual adaptive control law.

Neural network $\hat{\boldsymbol{f}} = \left[\hat{f}_1 \cdots \hat{f}_s\right]^T$ is used to approximate the functionals in $\boldsymbol{f} := [f_1 \cdots f_s]^T$, where $\hat{f}_i$ approximates f_i. Neural network $\hat{\boldsymbol{g}} = [\hat{g}_1 \cdots \hat{g}_{s^2}]^T$ is used to approximate the functionals in $\boldsymbol{G}$, where $\hat{g}_{(i-1)s+j}$ approximates $G_{i,j}$ according to the notation:

$$\boldsymbol{G} := \begin{bmatrix} G_{1,1} & \cdots & G_{1,s} \\ \vdots & & \vdots \\ G_{s,1} & \cdots & G_{s,s} \end{bmatrix}, \hat{\boldsymbol{G}} = \begin{bmatrix} \hat{g}_1 & \cdots & \hat{g}_s \\ \vdots & & \vdots \\ \hat{g}_{s^2-s+1} & \cdots & \hat{g}_{s^2} \end{bmatrix}.$$

Each of the two networks contains one hidden layer of sigmoidal neurons whose outputs are represented by vectors $\boldsymbol{\phi_f}, \boldsymbol{\phi_g}$ for the $\hat{\boldsymbol{f}}, \hat{\boldsymbol{g}}$ networks respectively such that:

$$\begin{aligned} \hat{f}_i &= \boldsymbol{\phi}_{\boldsymbol{f}}^T \hat{\boldsymbol{w}}_{\boldsymbol{f}_i};\ i = 1 \cdots s \\ \hat{g}_i &= \boldsymbol{\phi}_{\boldsymbol{g}}^T \hat{\boldsymbol{w}}_{\boldsymbol{g}_i};\ i = 1 \cdots s^2 \end{aligned} \tag{2}$$

where $\hat{\boldsymbol{w}}_{\boldsymbol{f}_i}, \hat{\boldsymbol{w}}_{\boldsymbol{g}_i}$ denote the synaptic weight vectors of networks $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ respectively. The individual MLP sigmoidal activation functions are given by

$$\phi_{f_i} = \left[1 + \exp(-\hat{\boldsymbol{s}}_{\boldsymbol{f}_i}\check{\boldsymbol{x}})\right]^{-1};\ i = 1 \cdots L_f, \check{\boldsymbol{x}} = \left[\boldsymbol{x}^T\ 1\right]^T$$

$$\phi_{g_i} = \left[1 + \exp(-\hat{\boldsymbol{s}}_{\boldsymbol{g}_i}\check{\boldsymbol{x}})\right]^{-1};\ i = 1 \cdots L_g, \check{\boldsymbol{x}} = \left[\boldsymbol{x}^T\ 1\right]^T$$

with L_f, L_g denoting the number of neurons in the hidden layer of the $\hat{\boldsymbol{f}}, \hat{\boldsymbol{g}}$ networks respectively, and $\hat{\boldsymbol{s}}_{\boldsymbol{f}_i}, \hat{\boldsymbol{s}}_{\boldsymbol{g}_i}$ denoting the corresponding hidden layer weight vectors that shape the sigmoid of the i^{th} activation function.

Let us group together all the unknown network weights into one vector $\hat{\boldsymbol{z}} = \left[\hat{\boldsymbol{p}}_{\boldsymbol{f}}^T\ \hat{\boldsymbol{p}}_{\boldsymbol{g}}^T\right]^T$, where $\hat{\boldsymbol{p}}_{\boldsymbol{f}} = \left[\cdots \hat{\boldsymbol{w}}_{\boldsymbol{f}_i}^T \cdots \hat{\boldsymbol{s}}_{\boldsymbol{f}_i}^T \cdots\right]^T$ and $\hat{\boldsymbol{p}}_{\boldsymbol{g}} = \left[\cdots \hat{\boldsymbol{w}}_{\boldsymbol{g}_i}^T \cdots \hat{\boldsymbol{s}}_{\boldsymbol{g}_i}^T \cdots\right]^T$ are the weights associated with the $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ networks respectively and which require on-line estimation. According to the Neural Network Universal Approximation Theorem (Haykin, 1999), there exists a set of optimal (constant) weights such that an appropriately sized neural network can approximate any smooth nonlinear function within a compact subset of its input space, up to any desired degree of accuracy. For our case, let us denote this optimal set of weights as $\boldsymbol{z}^* = \left[\boldsymbol{p_f}^{*T}\ \boldsymbol{p_g}^{*T}\right]^T$, these being the unknown optimal values of $\hat{\boldsymbol{z}}, \hat{\boldsymbol{p}}_{\boldsymbol{f}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{g}}$ respectively. Let us assume that the accuracy achieved with this set of optimal weights is such that the network approximation error can be considered negligible. In this case, system equation (1) can be equivalently re-written in terms of the following optimal neural network estimation model:

$$\begin{aligned} \boldsymbol{z}_{k+1}^* &= \boldsymbol{z}_k^* + \boldsymbol{\rho}_k \\ \boldsymbol{y}_k &= \hat{\boldsymbol{f}}(\boldsymbol{x}_{k-1}, \boldsymbol{p}_{\boldsymbol{f}_k}^*) + \hat{\boldsymbol{G}}(\boldsymbol{x}_{k-1}, \boldsymbol{p}_{\boldsymbol{g}_k}^*)\boldsymbol{u}_{k-1} + \boldsymbol{\varepsilon}_k \\ &= \boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{z}_k^*) + \boldsymbol{\varepsilon}_k \end{aligned} \tag{3}$$

where
$\boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{z}_k^*) := \hat{\boldsymbol{f}}(\boldsymbol{x}_{k-1}, \boldsymbol{p}_{\boldsymbol{f}_k}^*) + \hat{\boldsymbol{G}}(\boldsymbol{x}_{k-1}, \boldsymbol{p}_{\boldsymbol{g}_k}^*)\boldsymbol{u}_{k-1}$
and $\boldsymbol{\rho}_k$ is a Gaussian process noise with known covariance $\boldsymbol{Q}_{\boldsymbol{\rho}}$, generally set to have very small magnitude. The latter noise signal was not present in the original system equation but is included in the estimation model because it aids the weight estimation process.

2.2 Parameter Estimation

The optimal neural network parameters vector $\boldsymbol{z}^*$ appearing in model (3) is unknown and needs to be estimated recursively in real-time while the control actions are being executed. In this work, $\boldsymbol{z}^*$ is treated as a random variable having an initial condition that is assumed to be Gaussian distributed with known mean $\hat{\boldsymbol{z}}_0$ and covariance $\boldsymbol{P}_0$. These parameters appear nonlinearly in function $\boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{z}_k^*)$ of the model's output equation (3). As a consequence, even with Gaussian additive noise, the distribution of the parameters will no longer remain Gaussian because of the nonlinear effects of the system dynamics. Thus nonlinear stochastic estimation methods need to be applied. Three such approaches are considered here:

- the Extended Kalman filter (EKF),
- the Unscented Kalman filter (UKF),
- the Square-root Unscented Kalman filter (SRUKF).

There exist other nonlinear estimation algorithms such as Particle filters and Gaussian Sum filters but these fall outside the scope of the analysis reported in this work.

2.2.1 The EKF Estimator

Use of the EKF algorithm as a nonlinear estimator for the parameters of neural networks is well documented (Haykin, 2001). The EKF is based on a rather crude, first-order linearization of the system dynamics, upon which a Kalman filter is applied. It effectively ignores the non-Gaussian distribution of the random variables and propagates the density functions by means of the standard Kalman filter equations which would only be correct in a linear scenario, where Gaussianity is preserved. This assumption often leads to a well-known disadvantage of the EKF: divergence of the parameter estimates when propagated through time.

Let us denote the estimate to vector $\boldsymbol{z}^*$ calculated at time step k as $\hat{\boldsymbol{z}}_k$. The EKF algorithm, operating in predictive mode, recursively generates estimates for the parameters of the neural network controller being considered in this work, as follows:

Algorithm 1 - EKF:
Initialize with $\hat{\boldsymbol{z}}_0$ and $\boldsymbol{P}_0$. Then at every time step k...

1. Calculate $\nabla_{\boldsymbol{h}_{k-1}}$, the Jacobian of $\boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{z}_k^*\right)$ with respect to $\boldsymbol{z}_k^*$ evaluated at $\hat{\boldsymbol{z}}_k$:
$$\nabla_{\boldsymbol{h}_{k-1}} = \left[\nabla_{\boldsymbol{f}_{k-1}} \ \ \nabla_{\boldsymbol{g}_{k-1}}\right] = \left[\left.\frac{\partial \hat{\boldsymbol{f}}_{k-1}}{\partial \boldsymbol{p}^*_{\boldsymbol{f}_k}}\right|_{\hat{\boldsymbol{p}}_{\boldsymbol{f}_k}} \quad \left.\frac{\partial\left(\hat{\boldsymbol{G}}_{k-1}\boldsymbol{u}_{k-1}\right)}{\partial \boldsymbol{p}^*_{\boldsymbol{g}_k}}\right|_{\hat{\boldsymbol{p}}_{\boldsymbol{g}_k}} \right]$$
2. Approximate the covariance of the output measurement's distribution:
$$\boldsymbol{P}_{yy_k} = \nabla_{\boldsymbol{h}_{k-1}} \boldsymbol{P}_k \nabla^T_{\boldsymbol{h}_{k-1}} + \boldsymbol{R}_{\boldsymbol{\varepsilon}}$$
3. Calculate the Kalman gain:
$$\boldsymbol{K}_k = \boldsymbol{P}_k \nabla^T_{\boldsymbol{h}_{k-1}} \boldsymbol{P}_{yy_k}^{-1}$$
4. Use the current output measurement to generate the innovations:
$$\boldsymbol{i}_k = \boldsymbol{y}_k - \boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \hat{\boldsymbol{z}}_k\right)$$
5. Predict the parameter estimate:
$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k + \boldsymbol{K}_k \boldsymbol{i}_k$$
6. Predict the covariance of the estimate:
$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{K}_k \nabla_{\boldsymbol{h}_{k-1}} \boldsymbol{P}_k + \boldsymbol{Q}_{\boldsymbol{\rho}}.$$

Step 1 of this algorithm highlights a second crucial disadvantage of the EKF algorithm, namely that it entails a prior, off-line derivation of the Jacobian matrix for calculation in step 1. This derivation can become rather complex in the context of our neural network control schemes that typically demand hundreds of network parameters (Bugeja, 2011). Space limitations preclude us from showing the derivation and final form of this Jacobian matrix. Suffice to say that it will be of size $s \times \left(L_f(s + L_a) + L_g(s^2 + L_a)\right)$, where $L_a = s(n+p)+1$ denotes the length of vector $\check{\boldsymbol{x}}$.

2.2.2 The UKF Estimator

The UKF is based on the Unscented Tranformation which propagates the mean and covariance of the random variables through appropriate nonlinear transformations (Julier and Uhlmann, 2004). A set of so-called sigma points are chosen from the statistics of the nonlinear transformation so as to propagate second-order properties of the probability distribution. As a consequence, the UKF is a more accurate estimator than the EKF (Wan and van der Merwe, 2001). In addition, unlike the EKF, it does not require any derivations or calculation of complex Jacobian matrices. When applied within the context of the neural network controller discussed in this paper, the algorithm takes the following form:

Algorithm 2 - UKF:
Initialize with $\hat{\boldsymbol{z}}_0$ and $\boldsymbol{P}_0$. Then at every time step k...

1. Form a matrix $\bar{\boldsymbol{Z}}_k$ whose columns are the $2N+1$ sigma vectors $\bar{\boldsymbol{z}}_{i_k}$ generated as follows:
$$\begin{aligned} \bar{\boldsymbol{z}}_{1_k} &= \hat{\boldsymbol{z}}_k \\ \bar{\boldsymbol{z}}_{i_k} &= \hat{\boldsymbol{z}}_k + \gamma\boldsymbol{\sigma}_{i_k} \quad i = 2, \ldots, N+1 \\ \bar{\boldsymbol{z}}_{i_k} &= \hat{\boldsymbol{z}}_k - \gamma\boldsymbol{\sigma}_{i_k} \quad i = N+2, \ldots, 2N+1 \end{aligned}$$

where $\boldsymbol{\sigma}_{i_k}$ is the i^{th} column of $\boldsymbol{\Sigma}_k$, the latter denoting the lower-triangular Cholesky factorization of $\boldsymbol{P}_k$ such that $\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_k^T = \boldsymbol{P}_k$. In practice, this is obtained by calling a Cholesky factorization function in software, denoted in this paper as $\text{chol}\{\boldsymbol{P}_k\}$.

N denotes the length of $\hat{\mathbf{z}}_k$ (in this case $N = L_f(s+L_a)+L_g(s^2+L_a)$) and $\gamma = \sqrt{N+\lambda}$ where $\lambda = \alpha^2(N+\kappa)-N$, α being a constant that determines the spread of the points in the sigma vectors around $\hat{\mathbf{z}}_k$, typically set within the range $1 \geq \alpha \geq 10^{-4}$. κ is a constant scaling parameter normally set to $3-N$.

2. Propagate the sigma vectors through the neural network estimation model's output equation:

$$\bar{\boldsymbol{y}}_{i_k} = \bar{\boldsymbol{f}}_{i_{k-1}} + \bar{\boldsymbol{G}}_{i_{k-1}}\boldsymbol{u}_{k-1}$$

where

$$\bar{\boldsymbol{f}}_{i_{k-1}} = \hat{\boldsymbol{f}}\left(\boldsymbol{x}_{k-1}, \bar{\mathbf{z}}_{\boldsymbol{q}_{i_k}}\right);\ \bar{\boldsymbol{G}}_{i_{k-1}} = \hat{\boldsymbol{G}}\left(\boldsymbol{x}_{k-1}, \bar{\mathbf{z}}_{\boldsymbol{r}_{i_k}}\right);$$

$i = 1,\ldots,2N+1$; vectors $\bar{\mathbf{z}}_{\boldsymbol{q}_{i_k}}, \bar{\mathbf{z}}_{\boldsymbol{r}_{i_k}}$ are the columns of matrices $\bar{\boldsymbol{Z}}_{\boldsymbol{q}_k}, \bar{\boldsymbol{Z}}_{\boldsymbol{r}_k}$ which in turn are submatrices of size $\left(L_f(s+L_a)\times 2N+1\right)$ and $\left(L_g(s^2+L_a)\times 2N+1\right)$ respectively, taken from the matrix of sigma vectors $\bar{\boldsymbol{Z}}_k$ after repartitioning it as follows:

$$\bar{\boldsymbol{Z}}_k = \begin{bmatrix}\bar{\boldsymbol{Z}}_{\boldsymbol{q}_k}\\ \bar{\boldsymbol{Z}}_{\boldsymbol{r}_k}\end{bmatrix}.$$

3. Approximate the mean and covariance of the distribution of the output measurement using the propagated sigma vectors, calculated by the following weighted summations:

$$\hat{\boldsymbol{y}}_k = \sum_{i=1}^{2N+1} W_{m(i)}\bar{\boldsymbol{y}}_{i_k}$$

$$\boldsymbol{P}_{yy_k} = \sum_{i=1}^{2N+1} W_{c(i)}\left(\bar{\boldsymbol{y}}_{i_k}-\hat{\boldsymbol{y}}_k\right)\left(\bar{\boldsymbol{y}}_{i_k}-\hat{\boldsymbol{y}}_k\right)^T + \boldsymbol{R}_{\boldsymbol{\varepsilon}}$$

$$\boldsymbol{P}_{zy_k} = \sum_{i=1}^{2N+1} W_{c(i)}\left(\bar{\mathbf{z}}_{i_k}-\hat{\mathbf{z}}_k\right)\left(\bar{\boldsymbol{y}}_{i_k}-\hat{\boldsymbol{y}}_k\right)^T$$

where the so-called unscented transform weights are given by

$$W_{m(1)} = \frac{\lambda}{N+\lambda},\quad W_{c(1)} = \frac{\lambda}{N+\lambda}+1-\alpha^2+\beta$$

$$W_{m(i)} = W_{c(i)} = \frac{1}{2(N+\lambda)},\quad i = 2,\ldots,2N+1.$$

β is a constant parameter that depends upon prior knowledge of the estimate's distribution where, for the Gaussian prior case, $\beta = 2$ is optimal.

4. The Kalman gain is calculated as:

$$\boldsymbol{K}_k = \boldsymbol{P}_{zy_k}\boldsymbol{P}_{yy_k}^{-1}$$

5. The innovations are calculated as:

$$\boldsymbol{i}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$$

6. Predict the parameter estimate:

$$\hat{\mathbf{z}}_{k+1} = \hat{\mathbf{z}}_k + \boldsymbol{K}_k\boldsymbol{i}_k$$

7. Predict the covariance of the estimate:

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{K}_k\boldsymbol{P}_{yy_k}\boldsymbol{K}_k^T + \boldsymbol{Q}_{\boldsymbol{\rho}}$$

In its credit, and in contrast with the EKF, the UKF algorithm does not make use of the Jacobian matrix. Additionally, several studies have been published which illustrate the improved performance of the UKF over the EKF for estimation of variables in nonlinear systems (Julier and Uhlmann, 2004; Wan and van der Merwe, 2001).

2.2.3 The SRUKF Estimator

The numerical properties of the UKF algorithm can be further improved by a square-root implementation, leading to the SRUKF algorithm (Wan and van der Merwe, 2001). This guarantees that the covariance matrices remain positive semi-definite.

Additionally, whereas the UKF needs to perform a Cholesky factorization of the covariance matrix at every time step in order to compute its square root $\boldsymbol{\Sigma}_k$ (refer to Step 1 in the UKF algorithm), the SRUKF propagates $\boldsymbol{\Sigma}_k$ directly, avoiding the recursive Cholesky factorization operations. In the parameter estimation scenario, this leads to a computational complexity which has the same order as that of the EKF in terms of floating point instructions per iteration (Wan and van der Merwe, 2001). However this does not necessarily translate to an equal or a similar execution time, especially when the amount of parameters is high, leading to a large number of sigma points (LaViola, 2003). When applied to the neural network controller developed in this paper, the SRUKF algorithm takes the following form:

Algorithm 3 - SRUKF:
Initialize with $\hat{\mathbf{z}}_0$ and $\boldsymbol{\Sigma}_0 = \text{chol}\{\boldsymbol{P}_0\}$. Then at every time step k...

1. Form matrix $\bar{\boldsymbol{Z}}_k$ of $2N+1$ sigma vectors as follows:

$$\bar{\mathbf{z}}_{1_k} = \hat{\mathbf{z}}_k$$
$$\bar{\mathbf{z}}_{i_k} = \hat{\mathbf{z}}_k + \gamma\boldsymbol{\sigma}_{i_k}\quad i = 2,\ldots,N+1$$
$$\bar{\mathbf{z}}_{i_k} = \hat{\mathbf{z}}_k - \gamma\boldsymbol{\sigma}_{i_k}\quad i = N+2,\ldots,2N+1$$

where $\boldsymbol{\sigma}_{i_k}$ is the i^{th} column of $\boldsymbol{\Sigma}_k$. The constants N, γ, λ and α are defined in an identical manner as for the UKF.

2. Propagate the sigma vectors through the neural network output equation:
$$\bar{\boldsymbol{y}}_{i_k} = \bar{\boldsymbol{f}}_{i_{k-1}} + \bar{\boldsymbol{G}}_{i_{k-1}} \boldsymbol{u}_{k-1}$$
where $\bar{\boldsymbol{f}}_{i_{k-1}}$, $\bar{\boldsymbol{G}}_{i_{k-1}}$, $\bar{\boldsymbol{z}}_{\boldsymbol{q}_{i_k}}$, $\bar{\boldsymbol{z}}_{\boldsymbol{r}_{i_k}}$ are defined in the same way as for the UKF.
3. Approximate the mean and the Cholesky factor of the covariance of the output measurement's distribution through the following weighted summations of the propagated sigma vectors:
$$\hat{\boldsymbol{y}}_k = \sum_{i=1}^{2N+1} W_{m(i)} \bar{\boldsymbol{y}}_{i_k}$$
$$\boldsymbol{\Sigma}_{y_k} = \text{qr}\left\{ \left[\sqrt{W_{c(2:2N+1)}} \left(\bar{\boldsymbol{y}}_{2:2N+1_k} - \hat{\boldsymbol{y}}_k \right) \quad \sqrt{\boldsymbol{R}_{\boldsymbol{\varepsilon}}} \right] \right\}$$
$$\boldsymbol{\Sigma}_{y_k} = \text{cholupdate}\left\{ \boldsymbol{\Sigma}_{y_k},\, \bar{\boldsymbol{y}}_{1_k} - \hat{\boldsymbol{y}}_k,\, W_{c(1)} \right\}$$
$$\boldsymbol{P}_{zy_k} = \sum_{i=1}^{2N+1} W_{c(i)} \left(\bar{\boldsymbol{z}}_{i_k} - \hat{\boldsymbol{z}}_k \right) \left(\bar{\boldsymbol{y}}_{i_k} - \hat{\boldsymbol{y}}_k \right)^T$$
where the unscented transform weights $W_{m(i)}$, $W_{c(i)}$ and parameter β are defined as in the UKF algorithm.
4. The Kalman gain is calculated as:
$$\boldsymbol{K}_k = (\boldsymbol{P}_{zy_k}) / \boldsymbol{\Sigma}_{y_k}^T) / \boldsymbol{\Sigma}_{y_k}$$
5. The innovations are calculated as:
$$\boldsymbol{i}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$$
6. Predict the parameter estimate:
$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k + \boldsymbol{K}_k \boldsymbol{i}_k$$
7. Predict the Cholesky factor of the covariance of the estimate as either:

   Option 1:
   $$\boldsymbol{\Sigma}_{k+1} = \lambda_{RLS}^{-0.5}\ \text{cholupdate}\left\{ \boldsymbol{\Sigma}_k,\, \boldsymbol{K}_k \boldsymbol{\Sigma}_{y_k}, -1 \right\}$$
   where λ_{RLS} is a constant forgetting factor parameter set to a value just less than 1 *e.g.* 0.9995,

   or Option 2:
   $$\boldsymbol{\Sigma}_{k+1} = \text{cholupdate}\left\{ \boldsymbol{\Sigma}_k,\, \boldsymbol{K}_k \boldsymbol{\Sigma}_{y_k}, -1 \right\} + \boldsymbol{D}_{\boldsymbol{\rho}_k}$$
   where
   $$\boldsymbol{D}_{\boldsymbol{\rho}_k} = -\text{Diag}\left\{ \boldsymbol{\Sigma}_k \right\} + \sqrt{\text{Diag}^2\left\{ \boldsymbol{\Sigma}_k \right\} + \text{Diag}\left\{ \boldsymbol{Q}_{\boldsymbol{\rho}} \right\}}$$
   with function Diag$\{\cdot\}$ denoting the formation of a diagonal matrix whose diagonal elements are a copy of the diagonal terms in the matrix of the function's argument.

The qr$\{\cdot\}$, $\cdot/\cdot$ and cholupdate$\{\cdot\}$ symbols used in steps 3, 4 and 7 of the above algorithm denote functions that execute QR factorization, efficient least squares and Cholesky factor updating operations respectively. More details on this can be found in (Wan and van der Merwe, 2001).

2.3 The Control Law

The ideal solution to the dual control problem involves the minimization of a cost function whose solution is practically impossible to implement in most situations (Fabri and Kadirkamanathan, 2001). Some adaptive control schemes thus optimize a much more basic cost function which leads to control laws, such as Heuristic Certainty Equivalence (HCE) and Cautious control, that lack the desirable effects of dual control. These lead to an inferior performance typically exhibiting large overshoots in HCE, or long response times in Cautious control. A better alternative, as used in this work, is to adopt a suboptimal cost function but which to a certain extent still retains dual-like properties in its control actions.

This work is based on the suboptimal dual cost function originally proposed in (Milito et al., 1982) for linear stochastic systems, but generalized to the case of nonlinear MIMO systems as follows:

$$\begin{aligned} J_{inn} \;=\; & E\Big\{ \left(\boldsymbol{y}_{k+1} - \boldsymbol{y}_{\boldsymbol{d}k+1} \right)^T \boldsymbol{Q}_{\boldsymbol{1}} \left(\boldsymbol{y}_{k+1} - \boldsymbol{y}_{\boldsymbol{d}k+1} \right) \\ & + \left(\boldsymbol{u}_k^T \boldsymbol{Q}_{\boldsymbol{2}} \boldsymbol{u}_k \right) + \left(\boldsymbol{i}_{k+1}^T \boldsymbol{Q}_{\boldsymbol{3}} \boldsymbol{i}_{k+1} \right) \Big| I^k \Big\}, \end{aligned}$$

where $E\{\cdot\}$ denotes mathematical expectation over all random variables, I^k is the information state at timestep k defined as $I^k := \{\boldsymbol{y}_k \ldots \boldsymbol{y}_0 \;\; \boldsymbol{u}_{k-1} \ldots \boldsymbol{u}_0\}$, and $\boldsymbol{i}$ is the innovations vector from the estimator. Design parameters $\boldsymbol{Q}_{\boldsymbol{1}}$ and $\boldsymbol{Q}_{\boldsymbol{2}}$ are $(s \times s)$ positive definite diagonal matrices with real positive elements. $\boldsymbol{Q}_{\boldsymbol{3}}$ is an $(s \times s)$ diagonal matrix satisfying $-\boldsymbol{Q}_{\boldsymbol{1}} \leq \boldsymbol{Q}_{\boldsymbol{3}} \leq \boldsymbol{0}$ element-wise. Matrix $\boldsymbol{Q}_{\boldsymbol{1}}$ imposes a penalty on tracking errors and matrix $\boldsymbol{Q}_{\boldsymbol{2}}$ induces a penalty on large control inputs. $\boldsymbol{Q}_{\boldsymbol{3}}$ affects the innovations vector. Its setting determines whether the control law will act in HCE mode, Cautious mode or alternatively induce the desired dual adaptive control characteristics.

The optimization of cost function J_{inn} subject to the dynamics of the system as represented by the optimal neural network model (3), leads to a dual control law of the following form:

$$\begin{aligned} \boldsymbol{u}_k = & \left(\boldsymbol{G}_k'^T \boldsymbol{Q}_{\boldsymbol{1}} \boldsymbol{G}_k' + \boldsymbol{Q}_{\boldsymbol{2}} + \boldsymbol{N}_k \right)^{-1} \\ & \times \left(\boldsymbol{G}_k'^T \boldsymbol{Q}_{\boldsymbol{1}} \left(y_{d_{k+1}} - \boldsymbol{f}_k' \right) - \boldsymbol{\kappa}_k \right) \end{aligned} \qquad (4)$$

where $\boldsymbol{f}_k'$, $\boldsymbol{G}_k'$, $\boldsymbol{N}_k$ and $\boldsymbol{\kappa}_k$ are terms which depend on the variables of the selected estimator. A detailed derivation and full description of this control law falls outside the scope of this paper. Further details on the control aspect of the work may be obtained from (Bugeja, 2011). $\boldsymbol{N}_k$ and $\boldsymbol{\kappa}_k$ depend on the covariance estimates from the estimator and embody information regarding the uncertainty of the estimates at every time instant k. When matrix $\boldsymbol{Q}_{\boldsymbol{3}}$ is set equal to $-\boldsymbol{Q}_{\boldsymbol{1}}$, the terms $\boldsymbol{N}_k$ and $\boldsymbol{\kappa}_k$ will become null and vanish from control law (4). The controller will thus ignore the uncertainty of the estimates, leading to an HCE controller which tends to generate aggressive inputs that may be beneficial for probing the system but at the expense of excessive overshoot and a low degree of stability. At the other extreme, with $\boldsymbol{Q_3} = \boldsymbol{0}$, the control law gives maximum attention to the uncertainty terms, which leads to Cautious control. Whereas this does not give rise to excessive overshoot and a low degree of stability, it is typically too weak to make the system react promptly and generate probing signals which enhance the estimation process. Both these extremes are inferior to Dual control in terms of performance. When $\boldsymbol{Q_3}$ is set in between these two extremes, the controller will exhibit dual-like characteristics obtained by reasonably balancing out the aggressive probing of HCE and the sluggish actions of Cautious control.

3 TESTING AND RESULTS

The structure of the complete control algorithm can be subdivided in terms of the following tasks:

TASK A - Measurement of the system outputs $\boldsymbol{y}_k$.

TASK B - Model Estimation (EKF, UKF, SRUKF):

I. Calculation of Jacobian matrix (for EKF) or sigma vectors matrix (for UKF, SRUKF), followed by the output measurement covariance or its Cholesky factor (for SRUKF).

II. Calculation of Kalman gain, innovations, parameter prediction, and parameter covariance prediction.

TASK C - Dual Control Law Calculations:

I. Calculation of $\boldsymbol{f}'_k$, $\boldsymbol{G}'_k$, $\boldsymbol{N}_k$, $\boldsymbol{\kappa}_k$.

II. Calculation of $\boldsymbol{u}_k$.

TASK D - Input of $\boldsymbol{u}_k$ to the system.

In the EKF estimator, TASK B(I) is covered in steps 1, 2 and TASK B(II) is implemented in steps 3, 4, 5, 6 of Algorithm 1. In the UKF and SRUKF estimators, TASK B(I) is implemented in steps 1, 2, 3 and TASK B(II) is covered in steps 4, 5, 6, 7 of Algorithms 2 and 3.

Whereas TASK B(I) for the EKF involves an extensive prior off-line effort to derive equations for the Jacobian matrix $\boldsymbol{\nabla}_{\boldsymbol{h}_{k-1}}$ that can be rather complex in a neural networks scenario, calculation of the numerical value of $\boldsymbol{\nabla}_{\boldsymbol{h}_{k-1}}$ and $\boldsymbol{P}_{yy_k}$ during run-time in steps 1 and 2 is rather straightforward. By contrast, the Jacobian-free UKF and SRUKF estimators require no prior off-line effort at all for TASK B(I), but instead shift the effort within steps 1 to 3 during run-time through relatively more intense calculations for Cholesky or QR factorizations and weighted summations across sigma vectors. This computational effort is not trivial in a neural networks scenario where the amount of parameters, and consequently also the number of sigma vectors, is large.

A number of simulation trials were performed in order to validate the effects of the three estimators on the control system. The objective of these trials is twofold, namely to quantify across the three estimators: (a) the ability of the control algorithm to track the reference input vector $\boldsymbol{y}_{\boldsymbol{d}_k}$, (b) the time taken for execution of the complete control algorithm, with specific focus on the impact of TASKS B and C.

Simulation trials were performed on MATLAB using a 2-input, 2-output dynamic MIMO system having the form of Equation (1) with $n = 2$, $p = 1$ and nonlinear functions:

$$\boldsymbol{f} = \begin{bmatrix} \frac{0.7x_1x_3}{1+x_2^2+x_3^2} + 0.25x_5 + 0.5x_6 \\ \frac{0.5x_4\sin x_2}{1+x_1^2+x_4^2} + 0.5x_6 + 0.3x_5 \end{bmatrix},$$

$$\boldsymbol{G} = \begin{bmatrix} \cos^2 x_3 & \frac{0.1}{1+3x_1^2+x_4^2} \\ x_1^2 & 0.1x_6 - 5.5 \end{bmatrix}.$$

The measurement noise covariance $\boldsymbol{R}_{\boldsymbol{\varepsilon}} = 5 \times 10^{-4}\boldsymbol{I}$ ($\boldsymbol{I}$ denotes the identity matrix) and that of the model process noise $\boldsymbol{Q}_{\rho} = 1 \times 10^{-5}\boldsymbol{I}$. The two neural networks in the estimation model are structured with $L_f = L_g = 7$ hidden layer neurons. This leads to a total of $N = 140$ neural network parameters requiring estimation. The initial condition of the parameter vector $\hat{\boldsymbol{z}}_0$ is generated at random from the interval $[-0.1, 0.1]$ and its covariance $\boldsymbol{P}_0 = 0.8\boldsymbol{I}$. The UKF and SRUKF parameter α is set to 0.9. $2N + 1 = 281$ sigma vectors, each composed of $N = 140$ elements, are required by these two algorithms. The control law was tested under three different settings of $\boldsymbol{Q_3}$ corresponding to HCE, Cautious and Dual control modes ($\boldsymbol{Q_3} = -0.3\boldsymbol{I}$ in the latter case).

Figure 1 shows a sample of results for dual control with a UKF estimator over a 10s interval. The reference input signals are shown in black and the two system outputs in colour. Notice how the proposed adaptive control system results in good tracking of the reference inputs following an initial period of "learning" due to the controller's lack of knowledge of the nonlinear system functions in $\boldsymbol{f}$ and $\boldsymbol{G}$. The single trial results of Figure 1 are however not sufficient to characterize the general performance of the control system due to variations introduced by the random effects of the noise and the initial parameter vector. A Monte Carlo characterization was therefore performed by repeating the experiment over 150 trials,

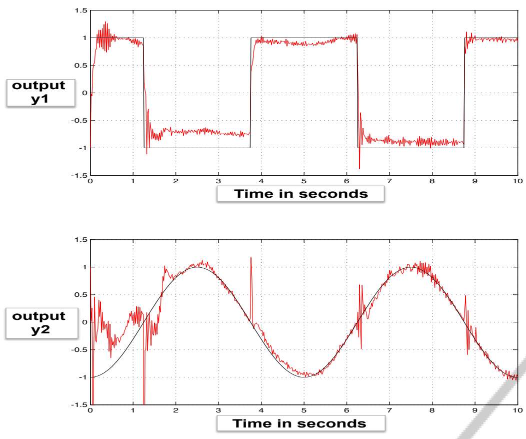

Figure 1: Tracking results with Dual control using UKF.

each time generating fresh realizations of the noise signals and the initial neural network parameters. $\boldsymbol{Q}_2$ was set to $0.1\boldsymbol{I}$ in all three cases. In each individual trial, all three control modes (HCE, Cautious and Dual) and all three estimators (EKF, UKF, SRUKF) were subjected to the same realization of random signals so as to ensure a fair comparison. The SRUKF estimator is tested for both Cholesky factor prediction options in step 7 of the algorithm. The tracking performance at the end of each trial was quantified in terms of an error metric $\mathcal{C} = \sum_{k=0}^{k_{end}} ||\boldsymbol{y}_{d\,k} - \boldsymbol{y}_k||^2$ which captures the squared tracking error over the whole simulation interval which was set to 5s in these trials. Lower values of $\mathcal{C}$ indicate a better tracking performance. The Monte Carlo analysis results are summarized in Table 1, showing the mean value of $\mathcal{C}$ and its variance across all trials for each possible combination of control mode and estimator. The results clearly show the superior performance of Dual control over the Cautious and HCE modes in all estimators, with $\mathcal{C}$ exhibiting the smallest mean and variance in each case. Moreover, comparing across estimators, the inferior control performance arising from the inaccuracies of the EKF estimator is clearly evident in all control modes. On the other hand, the UKF and the two SRUKF options exhibit an error metric having means and variances of the same order, reflecting similar tracking performance.

The simulation experiments were also used to time the main steps of the estimation and control algorithm across all estimator types. The results, shown in Table 2, show the mean execution time in milliseconds (ms) calculated over 250 iterations for the following operations: one complete iteration, Task B(I), Task B(II) and Task C. Figure 2 is a graphical depiction of the same results, but expressed as a factor of the EKF iteration time so as to interpret the measurements independently of any absolute time values which would vary according to the hardware on which the algorithms are executed. Such normalized timings for the separate tasks are shown in different colours as indicated in the legend on Figure 2. The results demonstrate that the execution time of the UKF/SRUKF increases by a factor of approximately 8 to 9 with respect to the EKF. This is mainly due to TASK B(I) in the UKF/SRUKF algorithm, particularly step 2 of the algorithms which propagates the 281 sigma vectors through the neural network. This step takes around 7.5 times the total EKF iteration time, notwithstanding that the UKF/SRUKF algorithms were carefully coded without slow for/next loops in order to maintain efficient timing. The SRUKF's total execution time is of the same order as that of the UKF, albeit slightly longer, especially with Option 2. In fact, TASK B(I) in the SRUKF takes a longer duration than the UKF because of the QR factorization and Cholesky factor update operations in step 3 which takes approximately thrice the time of the corresponding step in the UKF. Although this is somewhat compensated by the shorter duration of step 1 in the SRUKF, due to the absence of Cholesky factorization, this reduction is minimal when com-

Table 1: Results of Monte Carlo analysis.

Estimator	Mode	Mean of $\mathcal{C}$	Var of $\mathcal{C}$
EKF	HCE	15384.0	1.90×10^{10}
	Cautious	4562.0	3.03×10^{9}
	Dual	43.4	4.32×10^{3}
UKF	HCE	99.8	6.29×10^{4}
	Cautious	42.7	464.8
	Dual	35.6	318.5
SRUKF option 1	HCE	90.8	2.18×10^{4}
	Cautious	44.0	485.2
	Dual	36.2	336.9
SRUKF option 2	HCE	101.0	7.90×10^{4}
	Cautious	43.9	611.4
	Dual	35.6	319.8

Table 2: Execution time in ms (MATLAB running on a 3GHz Pentium 4 CPU with 2GB RAM).

	One complete iteration	TASK B(I)	TASK B(II)	TASK C
EKF	10.9	1.8	1.9	7.0
UKF	89.7	85.7	0.6	0.8
SRUKF option 1	91.3	88.3	1.3	0.7
SRUKF option 2	95.5	88.3	5.3	0.7

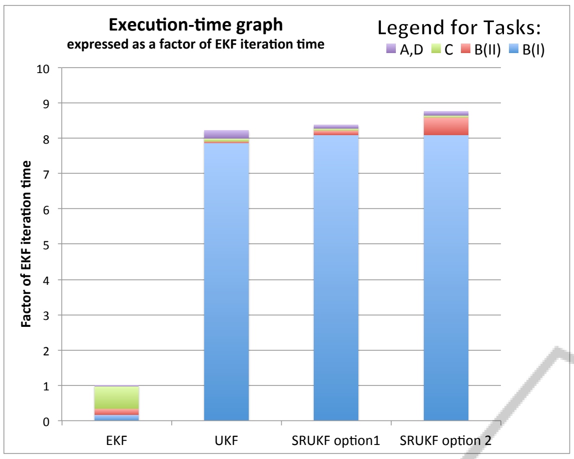

Figure 2: Execution time as a factor of the EKF case.

pared with the increased duration of step 3. TASK B(II) in the two SRUKF cases is also significantly longer than that of the UKF, particularly with option 2. The dominant component here is the Cholesky factor update operation in step 7 and, for the case of option 2, the additional operations required for calculation of $\boldsymbol{D}_{\rho_k}$. The execution time of the EKF algorithm is dominated by TASK C, which is around 9 times longer than the corresponding task in the other estimators. However this duration is still much less pronounced than the contribution of TASK B(I) in the UKF/SRUKF cases.

4 CONCLUSIONS

The results of the previous section clearly illustrate the general tracking superiority of Dual control over Cautious and HCE control. In addition, use of the UKF or SRUKF estimators leads to even better tracking results than the EKF - $\mathcal{C}$ is reduced by *circa* 18% on average in the Dual control case. The SRUKF offers no significant advantages in terms of tracking performance with respect to the UKF, neither with option 1 nor option 2.

However, this improvement in tracking performance comes at the cost of significantly increased execution time. The UKF and SRUKF with option 1 respectively take 8.2 and 8.4 times longer than the EKF-based controller, while the SRUKF with option 2 takes around 8.8 times longer.

One therefore concludes that Dual control with a UKF-based estimator should be used for best tracking performance, provided that the hardware is able to execute the estimation and control algorithm within a small percentage of the sampling interval. If this condition is not satisfied, the faster EKF-based Dual controller could be implemented instead but with a compromise; namely an inferior tracking performance.

REFERENCES

Åström, K. J. and Wittenmark, B. (2011). *Computer-Controlled Systems: theory and design*. Dover Publications, U.S.A, 3rd edition.

Bugeja, M. K. (2011). *Computational Intelligence Methods for Dynamic Control of Mobile Robots*. PhD thesis, University of Malta, Dept. of Systems and Control Engineering, Msida, Malta.

Bugeja, M. K. and Fabri, S. G. (2007). Dual adaptive control for trajectory tracking of mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 07)*, pages 2215–2220, Rome, Italy.

Bugeja, M. K. and Fabri, S. G. (2009). A novel dual adaptive neuro-controller based on the unscented transform for mobile robots. In *Proceedings of the International Conference on Neural Computation (ICNC 2009)*, pages 355–362, Funchal Madeira, Portugal.

Bugeja, M. K., Fabri, S. G., and Camilleri, L. (2009). Dual adaptive control of mobile robots using neural networks. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 39(1):129–141.

Fabri, S. and Kadirkamanathan, V. (1998). Dual adaptive control of nonlinear stochastic systems using neural networks. *Automatica*, 34(2):245–253.

Fabri, S. G. and Kadirkamanathan, V. (2001). *Functional Adaptive Control: An intelligent systems approach*. Springer-Verlag, London.

Fel'dbaum, A. A. (1965). *Optimal Control Systems*. Academic Press, New York.

Filatov, N. and Unbehauen, H. (2004). *Adaptive Dual Control: Theory and applications*. Springer, Berlin.

Haykin, S. (1999). *Neural Networks: A comprehensive foundation*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition.

Haykin, S., editor (2001). *Kalman Filtering and Neural Networks*. John Wiley and Sons.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

LaViola, J. (2003). A comparison of Unscented and Extended Kalman filtering for estimating quaternion motion. In *Proceedings of the 2003 American Control Conference*, pages 2435–2440.

Milito, R., Padilla, C. S., Padilla, R. A., and Cadorin, D. (1982). An innovations approach to dual control. *IEEE Transactions on Automatic Control*, AC-27(1):133–137.

Šimandl, M., Král, L., and Hering, P. (2005). Neural network based bicriterial dual control of nonlinear systems. In *Preprints of the 16th IFAC World Congress*, volume 16, Prague, Czech Republic.

Wan, E. A. and van der Merwe, R. (2001). The Unscented Kalman Filter. In Haykin, S., editor, *Kalman Filtering and Neural Networks*, Adaptive and Learning Systems for Signal Processing, Communications and Control, chapter 7, pages 221–280. John Wiley and Sons.