Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

An Experimental Demonstration of a Distributed and Event-Based State Estimation Algorithm [★]

Sebastian Trimpe [*] **Raffaello D'Andrea** [*]

[*] *Institute for Dynamic Systems and Control (IDSC), ETH Zurich, Sonneggstr. 3, 8092 Zurich, Switzerland (e-mail: strimpe@ethz.ch, rdandrea@ethz.ch).*

Abstract: A distributed state estimation algorithm that makes use of model-based predictions to reduce communication requirements in a networked control architecture is tested on an unstable system. A cube balancing on one of its edges serves as the test platform, and six rotating bodies on the cube's inner faces constitute the agents in the control network. Each agent carries a computational unit, which runs estimation and control algorithms, and is associated with local sensors and an actuator. Measurement data is shared among the agents over a broadcast network. Each agent maintains two estimates of the system state: the first reflecting the common knowledge in the network, and the second additionally including all local sensor information. An agent's sensor measurement is only broadcast if it deviates from the common estimate of that measurement by more than a specified threshold. Experimental results show that the number of communicated measurements required for stabilizing the system can be significantly reduced with this event-based communication protocol.

Keywords: State estimation, Kalman filters, communication networks, distributed control, discrete-event systems.

1. INTRODUCTION

In most traditional control systems, a communication medium is dedicated exclusively to the exchange of data between the plant and the controller. However, recent developments in the area of networked control systems (see Hespanha et al. [2007] for an overview) suggest a different view and regard communication as a shared resource. Multiple agents share a network and may exchange data of various content, in addition to sensor data and control commands of typical feedback control systems.

Viewing communication as a resource may also be useful from an architectural point of view. On a system level, one may distinguish between time-critical tasks (e.g. real-time feedback control) and non-time-critical tasks (e.g. adaptation). Clearly, sufficient resources (such as communication bandwidth) must be allocated to the critical tasks in order to maintain the operation of the system in times of need. On average, however, the full capacity of the resource is rarely exploited. In such cases, unused resources may be reallocated to non-critical tasks, resulting in improved system performance in the long run.

From control theory it is known that sensor feedback is critical, for example, for stabilization of unstable systems, disturbance rejection, and when dealing with uncertainties. In other situations, a system may operate satisfactorily open-loop, at least for limited time periods. In event-based control, sensor feedback caused by certain *events*

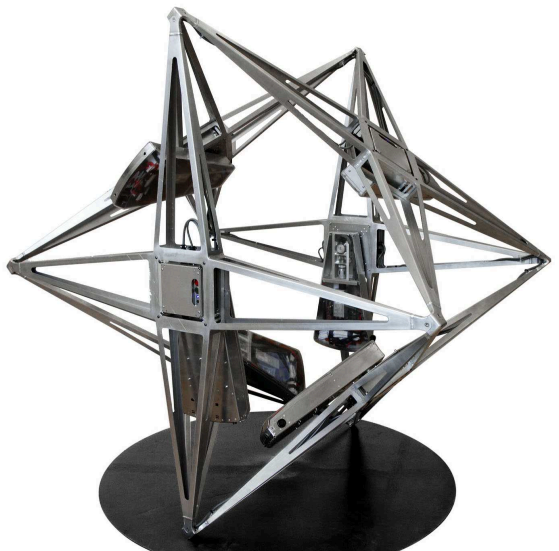

Fig. 1. The Balancing Cube is an example of a networked control system: six rotating modules, each having sensors, actuation, and computational unit, share information over a network to balance the cube on its edge.

(such as a measurement exceeding a threshold) is used as an alternative to time-triggered sending (see Åström [2008] and references therein). Lunze and Lehmann [2010] use the deviation of the actual state from the trajectory of

[★] This work was partially funded by the Swiss National Science Foundation (SNSF).

the closed-loop system with continuous sensor feedback to decide if an event should be triggered. Hence, the feedback loop is closed only when required.

At a high level, a similar idea can be applied to reduce the communication in networked control systems with multiple agents, each associated with local sensors and actuators. The agents may use a process model to make predictions about all other agents' sensor measurements. Using the same model, each agent also makes predictions about its own state or measurements. If the prediction of its local data deviates significantly from the true data, the local data is broadcast to all other agents, which can then update their estimates. Communication schemes like these (where, in order to reduce network traffic, sensor data is not sent at every time step) are referred to as *controlled communication*, cf. Hespanha et al. [2007]. The distributed estimation algorithm of this paper falls into this class of algorithms, which have previously been proposed by Yook et al. [2002], Xu and Hespanha [2004, 2005].

The implementation of the method for distributed state estimation in this work makes use of two independent discrete-time Kalman filters: one uses the measurement data that has been broadcast and is thus globally known and identical for all agents, and the other additionally exploits all local measurements. Each agent bases its decision to broadcast its local measurements on a comparison of the actual measurement to the common estimate of the system. Using this communication protocol, the Kalman filters receive a varying number of measurements (including none) at every time step.

A related problem for state estimation in networked control systems is considered in Sinopoli et al. [2004]. The authors analyze Kalman filtering for the scenario where non-delivery of measurement data results from packet drops rather than a decision made by some agent on the network. They model the arrival of a measurement as a binary random variable that is independent of the process data (states and measurements). In contrast, the delivery of a measurement in the presented approach is purposefully chosen to depend on past measurements.

The distributed estimators exploit the prediction capabilities of the Kalman filter to compensate for the deliberate non-communication of measurements. Zhang et al. [2001] use a similar idea for a different purpose in that they exploit model-based predictions to compensate for network-induced delays.

The Balancing Cube, a 1.2 m cube that can balance autonomously on any of its edges or corners[1] (see Fig. 1), serves as the platform for testing the distributed estimation algorithm presented in this paper. Six rotating *modules* located on each inner face of the cube enable the cube to balance. The modules are self-contained units that carry sensors, an actuator, a battery, and a computer. The modules share information over a communication network. Experiments demonstrate that the distributed estimation algorithm combined with the event-based communication protocol allows a significant reduction of the average number of communicated sensor measurements while keeping the cube balanced.

[1] Videos may be found online at `http://www.cube.ethz.ch`.

This paper is organized as follows: The distributed estimation algorithm with event-based communication is presented in Sec. 2. An overview of the physical test bed – the Balancing Cube – is given in Sec. 3, where the experimental results are also presented. The paper concludes with remarks in Sec. 4.

2. STATE ESTIMATION ALGORITHM

Figure 2 shows the network control architecture considered in this work. The block *Algorithm* represents a computational unit that runs the estimation and control algorithm and also handles the communication with the other units. Each unit is associated with local sensors (S) and actuators (A). Hence, the algorithm determines the appropriate commands for its actuator based on local sensor data, as well as data possibly received from the network. The combination of computational unit, sensor, and actuator will be referred to as *agent* for the remainder of this paper. The total number of agents is denoted by N.

The data unit that is sent over the network is a scalar sensor measurement. We assume that data is broadcast over the network; that is, if an agent sends a measurement, then all other agents receive this measurement. Furthermore, we take an abstract view of the network and assume the communication to be ideal; that is, we assume that the communication of measurements is instantaneous and no data is lost. This may be partly ensured by low level communication protocols. It is recognized however that sending only scalar measurements may not always be the best approach. For network protocols that require a minimum data length, it may be more efficient to send the full measurement vector. The method presented in the following can readily be adapted to this scenario.

We consider the common setup of a linear stochastic process given by a discrete-time state space model,

$$x(k) = A\,x(k-1) + B\,u(k-1) + v(k-1) \tag{1}$$
$$y(k) = C\,x(k) + w(k), \tag{2}$$

where k is the time index; $x(k),\ v(k) \in \mathbb{R}^n$; $u(k) \in \mathbb{R}^m$; $y(k),\ w(k) \in \mathbb{R}^p$; and all matrices are of corresponding dimensions. The process and measurement noise, $v(k)$ and $w(k)$, are random variables with $v(k) \sim \mathcal{N}(0, Q)$ and $w(k) \sim \mathcal{N}(0, R)$, where $\mathcal{N}(m, V)$ denotes a normally distributed random variable with mean m and covariance matrix V. All noise sources are assumed temporally inde-

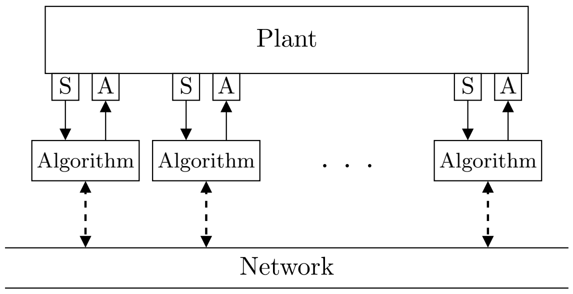

Fig. 2. The considered networked control architecture: the blocks A and S denote (possibly heterogeneous) actuator and sensor units; the Algorithm block includes estimation and control algorithms as well as the communication logic.

pendent; and R is assumed diagonal. The initial state $x(0)$ is also assumed to be normally distributed with known mean x_0 and covariance P_0.

In the proposed setup, each agent maintains an estimate of the full system state $x(k)$, which is input to a state feedback controller. Hence, the question of effective data communication rests with the state estimator: the objective is to maintain an estimate of the full system state $x(k)$ on each agent with a limited exchange of data between the agents. Two key issues will be addressed: (1) how to make use of the varying sets of measurements that arrive at an agent (*receiver algorithm*), and (2) how to decide if local sensor data should be broadcast over the network (*sender algorithm*).

Before addressing the distributed estimation problem with these two points in Sec. 2.2 and 2.3, the standard results for centralized state estimation (i.e. with access to the full measurement vector $y(k)$) are presented in Sec. 2.1. The centralized case will serve as a baseline for the distributed estimation method.

2.1 Centralized estimation

It is well known that the *optimal* state estimator of the system given by (1) and (2) is the time-varying Kalman filter. It is optimal in the sense that it keeps track of the entire conditional probability density of the system state $x(k)$ conditioned on all measurements and control inputs up to time k (cf. Anderson and Moore [1979]). The Kalman filter can be given in the following recursive form,

$$\hat{x}(k|k-1) = A\hat{x}(k-1|k-1) + Bu(k-1) \tag{3}$$
$$P(k|k-1) = AP(k-1|k-1)A^T + Q \tag{4}$$
$$K(k) = P(k|k-1)\, C^T\big(CP(k|k-1)C^T + R\big)^{-1} \tag{5}$$
$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\big(y(k) - C\hat{x}(k|k-1)\big) \tag{6}$$
$$P(k|k) = \big(I-K(k)C\big)P(k|k-1) \tag{7}$$

where $\hat{x}(k|k-1)$ denotes the expected value of the state $x(k)$ given all measurements and inputs up to time $k-1$, $\hat{x}(k|k)$ is the expected value of $x(k)$ given all data up to time k, and $P(k|k-1)$ and $P(k|k)$, respectively, are their covariance matrices. The filter is initialized by $\hat{x}(0|0) = x_0$ and $P(0|0) = P_0$. There are many different variants of the Kalman filter; which implementation of the Kalman filter is used does not matter for the method presented below.

For simplicity, static state feedback is considered,

$$u(k-1) = F\,\hat{x}(k-1|k-1), \tag{8}$$

where F denotes the matrix feedback gain. The exposition of the proposed method can, however, be adapted to controllers with states. Each agent is responsible for a subset of the control input vector $u(k)$. To ease the exposition, an index denoting the elements of the vector is not used unless otherwise noted.

It is assumed that both the Kalman state observer (3)–(7) and the feedback controller (8) are designed such that the given control objective is satisfied.

In a network with sufficient communication bandwidth, a conceptually equivalent implementation of the centralized Kalman filter (3)–(7) on the distributed control network in Fig. 2 would be to communicate at every time step all measurement data to all agents. Each agent can then simply run a copy of the estimator (3)–(7) and the controller (8). With this design as a starting point, the objective is to develop a distributed and event-driven estimation scheme that utilizes less communication bandwidth on average, but that may revert to the centralized design if required.

2.2 Distributed estimation: receiver algorithm

The *receiver* of agent i denotes the algorithm that uses all information available at time k to compute an estimate of the system state $x(k)$. The counterpart in the event-driven estimation scheme, i.e. the *sender algorithm*, is derived in the next section. It should be noted, however, that both the receiver and sender algorithms run on each agent in parallel.

The following notation is used to distinguish the different types of measurements that are available to agent i at time k:

$\bar{y}_i(k) \in \mathbb{R}^{\bar{p}_i}$	local sensor data
$\tilde{y}_i(k) \in \mathbb{R}^{\tilde{p}_i(k)}$	data received over the network
$y_i(k) \in \mathbb{R}^{p_i(k)}$	all available data, $y_i(k) = (\tilde{y}_i(k), \bar{y}_i(k))$.

The dimension of $\tilde{y}_i(k)$ and hence $y_i(k)$ is time-varying because of the varying and a-priori unknown number of measurements received. In particular, $\tilde{p}_i(k) = 0$ in the case where no measurement is received at time k by agent i.

The elements of the vectors $\bar{y}_i(k)$, $\tilde{y}_i(k)$, and $y_i(k)$ are subsets of the elements of the full measurement vector $y(k)$ in (2). Analogous notation is used to denote the output matrices and the measurement noise,

$$\bar{y}_i(k) = \bar{C}_i\, x(k) + \bar{w}_i(k) \tag{9}$$
$$\tilde{y}_i(k) = \tilde{C}_i(k)\, x(k) + \tilde{w}_i(k) \tag{10}$$
$$y_i(k) = C_i(k)\, x(k) + w_i(k), \tag{11}$$

where $\bar{C}_i$, $\tilde{C}_i(k)$, $C_i(k)$ are of appropriate dimensions, and $\bar{w}_i(k) \sim \mathcal{N}(0, \bar{R}_i)$, $\tilde{w}_i(k) \sim \mathcal{N}(0, \tilde{R}_i(k))$, and $w_i(k) \sim \mathcal{N}(0, R_i(k))$. Note that the dimension of the matrices $\tilde{C}_i(k)$, $C_i(k)$, $\tilde{R}_i(k)$, and $R_i(k)$ are time-varying due to the varying dimension of the corresponding measurement vector. This includes the case where $\tilde{C}_i(k)$ and $\tilde{R}_i(k)$ have zero rows when no measurement is received. In order to avoid special treatment of this case, the Kalman filter equations below should be understood such that the measurement update step is skipped if no measurement is available.

A time-varying Kalman filter that determines an estimate $\hat{x}_i(k|k)$ of the system state $x(k)$ on agent i, taking into account all available measurements up to and including time k, is designed analogously to the filter (3)–(7) for the centralized case:

$$\hat{x}_i(k|k-1) = A\hat{x}_i(k-1|k-1) + B\hat{u}_i(k-1) \tag{12}$$
$$P_i(k|k-1) = AP_i(k-1|k-1)A^T + Q \tag{13}$$
$$K_i(k) = P_i(k|k-1)\, C_i^T(k) \cdot \big(C_i(k)P_i(k|k-1)C_i^T(k) + R_i(k)\big)^{-1} \tag{14}$$

$$\hat{x}_i(k|k) = \hat{x}_i(k|k-1) + K_i(k)\big(y_i(k) - C_i(k)\hat{x}_i(k|k-1)\big) \tag{15}$$

$$P_i(k|k) = \big(I - K_i(k)C_i(k)\big)P_i(k|k-1). \tag{16}$$

To avoid misunderstanding, it should be reemphasized here that the index i in $\hat{x}_i(k|k)$ does not denote the i-th element of $\hat{x}(k|k)$, but agent i's state estimate. Similarly, $\hat{u}_i$, given by

$$\hat{u}_i(k-1) = F\,\hat{x}_i(k-1|k-1), \tag{17}$$

is agent i's estimate of what the control input to the whole system should be; in particular, agent i uses the components corresponding to its actuator as actual control commands.

We note that the distributed estimator (12)–(16) is the same as its centralized counterpart (3)–(7) if all sensor data is communicated.

2.3 Distributed estimation: sender algorithm

In the previous section, we assumed that each agent receives a varying number of sensor measurements from the other agents in the network. This section addresses the sending decision: agent i's sender algorithm determines if the local sensor data $\bar{y}_i(k)$ should be sent to all other agents on the network.

Following the key idea discussed in the introduction, agent i only sends its local measurements if it determines it is necessary; that is, if the other agents' expectation of the measurement is significantly different. One way for agent i to estimate the other agents knowledge is to construct another state estimate $\check{x}_i(k|k)$ that uses only measurements that have been broadcast to all agents. Hence, this estimator reflects the knowledge that is common to all agents. This requires that the local sensor data $\bar{y}_i(k)$ is only included in $\check{x}_i(k|k)$ if it is also broadcast to the network. The estimate $\check{x}_i(k|k)$ is again obtained from a Kalman filter,

$$\check{x}_i(k|k-1) = A\check{x}_i(k-1|k-1) + B\check{u}_i(k-1) \tag{18}$$

$$\check{P}_i(k|k-1) = A\check{P}_i(k-1|k-1)A^T + Q \tag{19}$$

$$\check{K}_i(k) = \check{P}_i(k|k-1)\,\tilde{C}_i^T(k) \cdot \big(\tilde{C}_i(k)\check{P}_i(k|k-1)\tilde{C}_i^T(k) + \tilde{R}_i(k)\big)^{-1} \tag{20}$$

$$\check{x}_i(k|k) = \check{x}_i(k|k-1) + \check{K}_i(k)\big(\tilde{y}_i(k) - \tilde{C}_i(k)\check{x}_i(k|k-1)\big) \tag{21}$$

$$\check{P}_i(k|k) = \big(I - \check{K}_i(k)\tilde{C}_i(k)\big)\check{P}_i(k|k-1) \tag{22}$$

with the corresponding estimate of the control input

$$\check{u}_i(k-1) = F\,\check{x}_i(k-1|k-1). \tag{23}$$

The estimate $\check{x}_i(k|k)$ ensures consistency in the network, since it is the same for all agents (whereas the estimate from (12)–(16) is generally different).

With the common estimate $\check{x}_i(k|k)$, agent i can now estimate what all other agents assume its measurement $\bar{y}_i(k)$ is: simply $\bar{C}_i\check{x}_i(k|k-1)$. This estimate can be used with some communication logic to decide if $\bar{y}_i(k)$ is broadcast. Here, a simple threshold logic, applied for each element l of the measurement vector individually, is used:

$$\text{send } \bar{y}_{i,l}(k) \Leftrightarrow |\bar{y}_{i,l}(k) - \bar{C}_{i,l}\,\check{x}_i(k|k-1)| \geq \delta_{i,l}, \tag{24}$$

where $\bar{y}_{i,l}(k)$ denotes the l-th element of $\bar{y}_i(k)$, $\bar{C}_{i,l}$ the l-th row of $\bar{C}_i$, and $\delta_{i,l} \in [0, \infty)$ is a design parameter capturing the tolerated deviation. Because of its impact on the network communication, we hereafter refer to it as the *communication threshold*.

Obviously, the choice $\delta_{i,l} = 0$ means that the measurement $\bar{y}_{i,l}(k)$ is always sent; on the contrary, $\delta_{i,l} \to \infty$ corresponds to never sending the measurement. Again, if all sensor data is communicated, the distributed estimators (12)–(16) and (18)–(22) yield the same estimates as the centralized Kalman filter in (3)–(7). Thus, the performance of the optimal state estimation can be recovered by choosing all $\delta_{i,l}$ to zero. This turns out to be very handy for practical implementation and tuning of the algorithm.

We remark that other communication logics are possible: a condition involving a bound on $\check{P}_i(k|k-1)$ or a combination of a condition on $\check{x}_i(k|k-1)$ and on $\check{P}_i(k|k-1)$ may be used instead.

2.4 The complete algorithm

Both Kalman filters (12)–(16) and (18)–(22) run in parallel on each agent. Using the condition (24) and the estimate $\check{x}_i(k|k-1)$, agent i decides if its associated sensor data is sent. For the purpose of computing the control input for its actuator, the estimate $\hat{x}_i(k|k)$ is used, since it makes use of all information locally available.

A control loop step using the distributed estimation method of Sec. 2.2 and 2.3 is summarized in Algorithm 1.

Algorithm 1 Control step on agent i, executed every time step k.

```
apply component of $\hat{u}_i(k-1)$ to local actuator
acquire local measurement $\bar{y}_i(k)$
receive $\tilde{y}_i(k)$ from network (possibly empty)
compute estimate $\hat{x}_i(k|k)$ from (12)–(16), (17)
compute estimate $\check{x}_i(k|k)$ from (18)–(22), (23)
for $l = 1$ to $\bar{p}_i$
    if $|\bar{y}_{i,l}(k) - \bar{C}_{i,l}\,\check{x}_i(k|k-1)| \geq \delta_{i,l}$
        send $\bar{y}_{i,l}(k)$
    end if
end for
compute control $\hat{u}_i(k) = F\,\hat{x}_i(k|k)$
```

3. APPLICATION TO THE BALANCING CUBE

In this section, the effectiveness of the distributed estimation algorithm is demonstrated on the Balancing Cube, which represents an unstable system. Six agents are used to stabilize the system. A brief system description is provided in Sec. 3.1. Even though a complete treatment is beyond the scope of this paper, a brief explanation of the modeling technique and the control design is provided. Remarks on the implementation of the state estimation method are given in Sec. 3.2 and experimental results are presented in Sec. 3.3.

3.1 System description

The Balancing Cube is a multi-body system consisting of a rigid body in the shape of a cube and six rotating bodies (called *modules*) on the inner faces of the cube, see Fig. 3. Each of the six faces of the cube is made of an X-shaped aluminum structure (cf. Fig. 1 and 3); the edge length of the cube is 1.2 m. The total mass of the cube is 21.4 kg and the modules have a base mass of 3.7 kg. In the setup presented here, three of the modules carry extra weights adding up to a total mass of 5.8 kg.

Though the cube can balance on a corner (presented in Trimpe and D'Andrea [2010]), for the purpose of this study it balances on its edge as shown in Fig. 1 and 3. In this configuration, the cube body has only one rotational degree of freedom (the rotational axis is the edge the cube is standing on), which results in a simpler dynamic model and eases the exposition presented below.

The modules are actuated by a DC motor and rotate relative to the cube structure. A drawing of a module with its functional parts is shown in Fig. 4. When the modules rotate, they exert reactional and gravitational forces (by shifting the center of mass) on the cube structure. Each module carries a single-board computer (SBC) that receives data from the sensors and sends commands to the motor. The computers themselves are connected over a Controller Area Network (CAN), whose wires run through a slip ring and along the cube structure. The low level CAN protocols allow each module to broadcast its local measurements to all other modules on the network. All components on a module are powered by a battery, which allows for a normal balancing operation of around 4 hours.

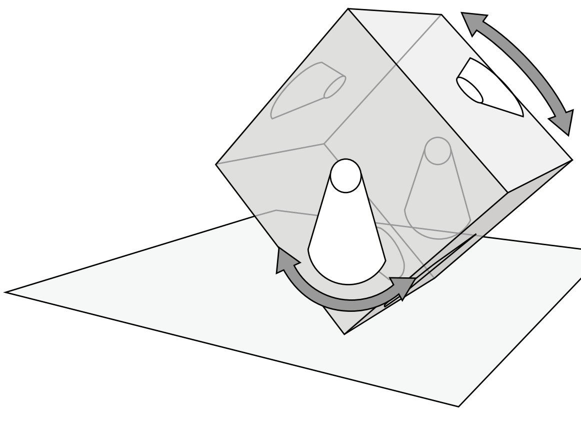

Fig. 3. Rendering of the Balancing Cube, shown in the same orientation as in Fig. 1. The cube has six rotating modules, one on each face.

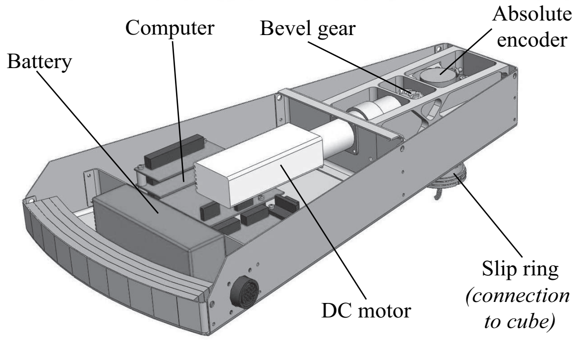

Fig. 4. Rendering of a module with its functional parts. Not shown in this drawing is the inertial measurement unit that sits in the part of the module that is attached rigidly to the cube body.

The local sensors associated with each module are an absolute encoder and an inertial measurement unit (IMU). The absolute encoder measures the module's angle relative to its mounting. The IMU is rigidly mounted to the cube structure (also connected to the SBC through the slip-ring). It has a tri-axis accelerometer and tri-axis rate gyro. For the demonstration of the estimation algorithm of Sec. 2, only the absolute encoder and rate gyro measurements were used. In fact, only one axis of the rate gyro is relevant, namely the axis parallel to the axis of rotation. Hence, each module has access to two local measurements ($\bar{p}_i = 2$).

Since each computational unit is connected locally to sensors and the actuator, and the computers share data over a network, the cube architecture falls into the class of systems considered in Sec. 2. The self-contained modules on the cube play the role of the agents.

In order to stabilize the system about an equilibrium, a cascaded control architecture is applied. On each motor, a local velocity feedback loop operates at 1 kHz. This inner loop tracks module angular velocity commands at a faster rate than the natural dynamics of the cube. With this architecture, nonlinear effects such as friction and backlash in the actuation mechanism are mitigated from the perspective of an outer loop (the full state feedback controller in the form of (8)) which stabilizes the system. The outer control loop is implemented at a frequency of 60 Hz.

Linear model and feedback controller. To obtain a model of the Balancing Cube for the design of the state feedback controller and the state estimator, the time-scale separation technique described in Trimpe and D'Andrea [2009] is applied, where the inner velocity feedback loops are considered as (ideal) high gain feedback loops. The resulting linear discrete-time model with sampling frequency of 60 Hz reads

$$\begin{bmatrix} x_{\mathrm{s}}(k) \\ x_{\mathrm{f}}(k) \end{bmatrix} = \begin{bmatrix} A_{\mathrm{ss}} & A_{\mathrm{sf}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_{\mathrm{s}}(k-1) \\ x_{\mathrm{f}}(k-1) \end{bmatrix} + \begin{bmatrix} B_{\mathrm{s}} \\ I \end{bmatrix} u(k-1) \tag{25}$$

$$y(k) = [C_{\mathrm{s}} \ 0] \begin{bmatrix} x_{\mathrm{s}}(k) \\ x_{\mathrm{f}}(k) \end{bmatrix}. \tag{26}$$

The states [2] $x_{\mathrm{f}}(k)$ are the angular velocities of the modules, $u(k)$ their reference values, and $x_{\mathrm{s}}(k)$ combines all other states. Notice that the approximation of the inner velocity loops as high gain feedback results in tracking of the reference inputs in (25) in one time step, i.e. $x_{\mathrm{f}}(k) = u(k-1)$. It has been verified experimentally that this is a valid approximation. All states and measurements of the linear model are listed in Table 1. The matrices of the state space model may be found in Appendix A.

Each module has access to an encoder and a gyro measurement. For module i, $\bar{y}_{i,1}(k)$ denotes the encoder and $\bar{y}_{i,2}(k)$ the gyro measurement at time k.

The state equation (25) is used to design a stabilizing LQR feedback controller; the feedback law is

[2] The index f corresponds to "fast" and s to "slow."

$$u(k-1) = [F_\mathrm{s} \;\; F_\mathrm{f}] \begin{bmatrix} x_\mathrm{s}(k-1) \\ x_\mathrm{f}(k-1) \end{bmatrix}. \tag{27}$$

The feedback gain matrices F_s and F_f may also be found in Appendix A.

For state estimation, the following reduced state-space representation is used, which follows from (25), (26) with added process and measurement noise,

$$x_\mathrm{s}(k) = A_\mathrm{ss}\, x_\mathrm{s}(k-1) + B_\mathrm{s}\, u(k-1) + A_\mathrm{sf}\, u(k-2) + v(k-1) \tag{28}$$

$$y(k) = C_\mathrm{s}\, x_\mathrm{s}(k) + w(k). \tag{29}$$

The update equations for $\hat{x}(k|k-1)$, $\hat{x}_i(k|k-1)$, and $\check{x}_i(k|k-1)$ in (3), (12), and (18), respectively, are adapted accordingly. The feedback law (27) becomes

$$u(k-1) = [F_\mathrm{s} \;\; F_\mathrm{f}] \begin{bmatrix} x_\mathrm{s}(k-1) \\ u(k-2) \end{bmatrix}. \tag{30}$$

Truth model. The truth model that is used for experimental comparison of the different state estimators is based on the nonlinear state estimation method for the Balancing Cube presented in Trimpe and D'Andrea [2010], which is augmented with further non-causal post-processing. To obtain the "true" state denoted by $x^\mathrm{truth}(k)$, all sensor data (including, in particular, the accelerometer data) is recorded and the state is obtained in post-processing. The estimate of the cube tilt obtained from this method has been verified with a camera-based motion capture system (cf. results in Trimpe and D'Andrea [2010]) and has proven to work well on the cube.

3.2 Implementation of the state estimation algorithm

The noise parameters Q and R of the Kalman filter (3)–(7) applied to the system given by (28) and (29) were treated as tuning parameters to obtain satisfactory centralized closed loop performance on the Balancing Cube. The following parameters were chosen:

$$Q = \mathrm{diag}\left([1\ 1\ 1\ 1\ 1\ 1\ 0.01\ 1]\right) \tag{31}$$

$$R = \mathrm{diag}\left([0.1\ 1\ 0.1\ 1\ 0.1\ 1\ 0.1\ 1\ 0.1\ 1\ 0.1\ 1]\right). \tag{32}$$

The Kalman filter was initialized with $x_0 = 0$ and $P_0 = Q/10$.

Table 1. The states and utilized measurements of the Balancing Cube. (Note that in this table the indices denote elements of a vector; for example, y_1 is the first component of y.)

state	physical meaning
$x_{\mathrm{s},1}$	angle module 1
$x_{\mathrm{s},2}$	angle module 2
$x_{\mathrm{s},3}$	angle module 3
$x_{\mathrm{s},4}$	angle module 4
$x_{\mathrm{s},5}$	angle module 5
$x_{\mathrm{s},6}$	angle module 6
$x_{\mathrm{s},7}$	cube angle
$x_{\mathrm{s},8}$	cube ang. vel.
$x_{\mathrm{f},1}$	ang. vel. module 1
$x_{\mathrm{f},2}$	ang. vel. module 2
$x_{\mathrm{f},3}$	ang. vel. module 3
$x_{\mathrm{f},4}$	ang. vel. module 4
$x_{\mathrm{f},5}$	ang. vel. module 5
$x_{\mathrm{f},6}$	ang. vel. module 6

meas.	sensor
y_1	encoder module 1
y_2	rate gyro module 1
y_3	encoder module 2
y_4	rate gyro module 2
y_5	encoder module 3
y_6	rate gyro module 3
y_7	encoder module 4
y_8	rate gyro module 4
y_9	encoder module 5
y_{10}	rate gyro module 5
y_{11}	encoder module 6
y_{12}	rate gyro module 6

The same parameters were used for the distributed Kalman filters (12)–(16) and (18)–(22). The only additional parameters that had to be chosen for the distributed implementation are the communication thresholds: for all agents i, they were set to $\delta_{i,1} = 0.02\,\mathrm{rad}$ for the absolute encoder measurements and, for the gyro measurements, to about 2.5 times the standard deviation of the measurement noise, that is, $\delta_{i,2} = 0.01\,\mathrm{rad/s}$.

3.3 Experimental results

Generally, it is expected that communicating less than all data will affect the performance of the feedback control system. If the communication thresholds $\delta_{i,l}$ are all zero, then the performance is equivalent to implementing the centralized state estimator (3)–(7). Below we define the performance and communication measures used in this work.

For evaluating the closed loop performance, a performance index $\mathcal{P}$ is defined as the root-mean square (RMS) value of the truth model state x^truth,

$$\mathcal{P} := \sqrt{\frac{1}{K}\sum_{k=1}^{K} (x^\mathrm{truth}(k))^T x^\mathrm{truth}(k)}, \tag{33}$$

for data of length K. For a system with state output that is driven by white noise with unit variance, the RMS value of the system state is equivalent to the $\mathcal{H}_2$ system norm (see e.g. Skogestad and Postlethwaite [2005]), which is a standard performance measure for stochastic control systems.

In network control systems the communication rate is commonly measured as the number of packets sent per time interval, (cf. Xu and Hespanha [2005]). Similarly, we consider the number of measurements sent per M steps as a measure for the amount of communication. The *communication rate* is computed as a moving average over M steps, that is, for the measurement $\bar{y}_{i,l}(k)$ (l denoting the element of the vector, i the agent), we define

$$\mathcal{R}_{i,l}(k) := \frac{\text{number of } \bar{y}_{i,l}(k) \text{ sent in } [(k-M+1)T_s, kT_s]}{MT_s}, \tag{34}$$

with the sampling time $T_s = 1/60$ s. The horizon is chosen as $M = 100$. Furthermore, the time average $\bar{\mathcal{R}}_{i,l}$ of $\mathcal{R}_{i,l}(k)$ and the *average total rate* $\mathcal{R}$ are defined by

$$\bar{\mathcal{R}}_{i,l} := \frac{1}{K}\sum_{k=1}^{K} \mathcal{R}_{i,l}(k),\ \mathcal{R} := \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\bar{p}_i}\sum_{l=1}^{\bar{p}_i} \bar{\mathcal{R}}_{i,l}\right). \tag{35}$$

The communication rates $\mathcal{R}_{i,l}(k)$, $\bar{\mathcal{R}}_{i,l}$, and $\mathcal{R}$ all lie in the interval $[0,1]$ by definition. In particular, $\mathcal{R} = 1$ corresponds to the case where at each time step all data is exchanged between the agents, while $\mathcal{R} = 0$ means no data is exchanged.

Experiment: steady-state balancing. The distributed estimation method from Sec. 2.2 and 2.3 was implemented on the Balancing Cube in order to stabilize the cube about the equilibrium configuration shown in Fig. 3.

Data was recorded over a period of five minutes of balancing. The obtained measures of performance and communication, $\mathcal{P}$ and $\mathcal{R}$, are given in Table 2; they are compared

Table 2. Communication and performance measures for centralized (eq. (3)–(7)) and distributed state estimation (Algorithm 1).

	$\mathcal{R}$	$\mathcal{P}$
centralized estimation	1.000	0.192
distributed estimation	0.060	0.285

Table 3. Average communication rates for the encoder measurements (top row) and for the gyro measurements (bottom row).

(i, l)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)
$\mathcal{R}_{i,l}$	0.0048	0.0137	0.0028	0.0014	0.0052	0.0011
(i, l)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)
$\mathcal{R}_{i,l}$	0.1059	0.1076	0.0913	0.1374	0.1157	0.1276

to a run with the centralized Kalman filter of Sec. 2.1. The average communication rates for the distributed implementation are listed in Table 3.

The expected trade-off between communication rate and performance can be observed from these results: tolerating a decrease of performance roughly by a factor of 1.5 compared to the centralized case allows a reduction in communication events roughly by a factor of 16.

The communication rates for some of the absolute encoder measurements are particularly low (significantly less than 1 %, cf. Table 3). Communicating the positions at every time step is obviously not necessary, since this part of the system can apparently be predicted very well from the model. Still, this prediction needs to be updated occasionally with an actual measurement.

For a 30-second sequence, module 1's estimates of the module angles 1 and 3, the cube angle, and the cube angular velocity obtained by (12)–(16) and (18)–(22) are shown in Fig. 5. They are compared to the truth model state x^{truth}. The same module's communication rates are shown in Fig. 6.

4. CONCLUDING REMARKS

Experimental results demonstrate that the algorithm for distributed state estimation presented in this paper is an effective tool for reducing the average communication rate in a networked control system. Moreover, it is a straight-forward tool to implement. First, it is based on the centralized design of the commonly used discrete-time Kalman filter. Second, the communication threshold parameters provide a practical handle for the designer to parametrize the trade-off between communication and estimator performance. In particular, the performance of the centralized design can be recovered and hence used as a starting point for fine-tuning the system performance.

A particularly useful feature of the algorithm presented herein is that the bandwidth required for sensor feedback is determined autonomously by the system, and need not be known beforehand. This encompasses the possibility that the average communication rates may vary for different types of sensors in the system. Furthermore, the system can easily adapt to change in communication requirements, using only the resources needed at any given moment. One may, in fact, view centralized estimation as the "fallback" system, in that it is simply the case where all measurements are communicated.

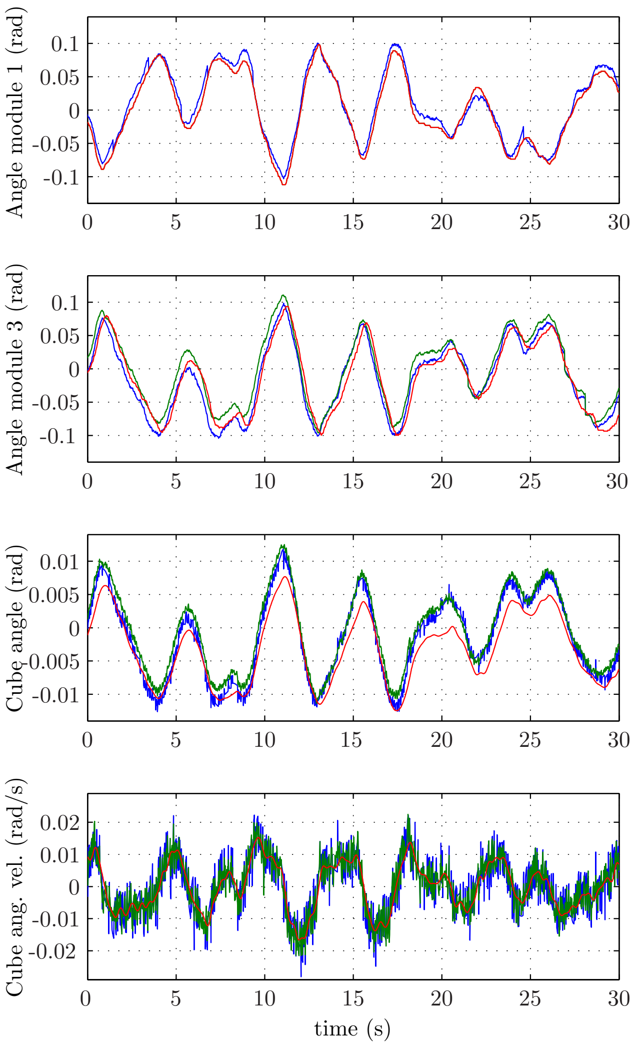

Fig. 5. Agent 1's state estimates $\check{x}_1$ (blue) and $\hat{x}_1$ (green), compared to the truth model state x^{truth} (red) for its own module angle (top), agent 3's module angle, the cube angle, and the cube angular velocity (bottom). The graphs $\hat{x}_1$ and x^{truth} are practically identical in the top diagram.

A theoretical analysis of the presented distributed estimation algorithm is beyond the scope this paper. Likewise, further study of variants of the employed algorithm on the Balancing Cube, such as making the sending decision of a measurement also based on its associated estimation variance, remain for future research.

ACKNOWLEDGEMENTS

The authors would like to thank Jan Lunze for stimulating discussions leading to the initial idea of this work.

REFERENCES

B.D.O. Anderson and J.B. Moore. *Optimal filtering.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

K.J. Åström. Event based control. In Alessandro Astolfi and Lorenzo Marconi, editors, *Analysis and Design of*

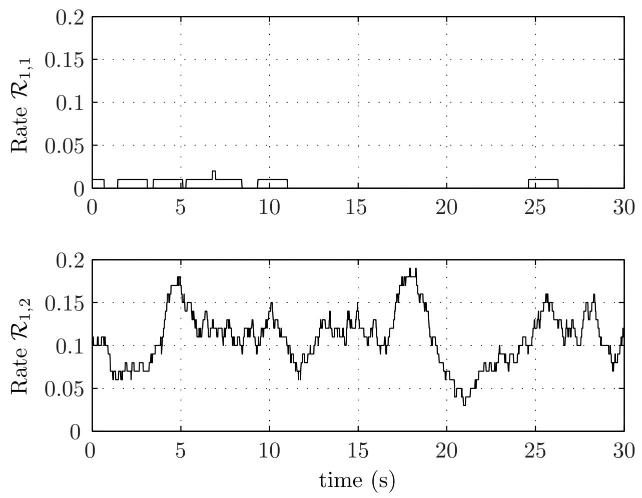

Fig. 6. Agent 1's communication rates for its encoder (top) and rate gyro measurement (bottom).

Nonlinear Control Systems, pages 127–147. Springer Berlin Heidelberg, 2008.

J.P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, January 2007.

J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46(1):211–215, January 2010.

B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, September 2004.

S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley-Interscience, 2nd edition, November 2005.

S. Trimpe and R. D'Andrea. A limiting property of the matrix exponential with application to multi-loop control. In *Proc. of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pages 6419–6425, Shanghai, P.R. China, December 2009.

S. Trimpe and R. D'Andrea. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2630–2636, Anchorage, Alaska, USA, May 2010.

Y. Xu and J.P. Hespanha. Optimal communication logics in networked control systems. In *Proc. of the 43rd IEEE Conference on Decision and Control*, pages 3527–3532, Atlantis, Paradise Island, Bahamas, December 2004.

Y. Xu and J.P. Hespanha. Estimation under uncontrolled and controlled communications in networked control systems. In *Proc. of the 44th IEEE Conference on Decision and Control and the European Control Conference*, pages 842–847, Seville, Spain, December 2005.

J.K. Yook, D.M. Tilbury, and N.R. Soparkar. Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. *IEEE Transactions on Control Systems Technology*, 10(4):503 –518, July 2002.

W. Zhang, M.S. Branicky, and S.M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, February 2001.

Appendix A. STATE SPACE MODEL AND FEEDBACK GAINS OF THE BALANCING CUBE

The matrices of the state space model of the Balancing Cube in (25) and (26) and the static feedback gain matrices in (27) are given here for completeness:

$$A_{\mathrm{ss}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} \end{bmatrix}$$

$$\begin{array}{llll} a_{7,1} = \text{2.8e-5} & a_{7,2} = \text{-5.6e-5} & a_{7,3} = \text{-2.8e-5} & a_{7,4} = \text{-2e-5} \\ a_{7,5} = \text{2.8e-5} & a_{7,6} = \text{2e-5} & a_{7,7} = 1 & a_{7,8} = 0.017 \\ a_{8,1} = 0.0033 & a_{8,2} = \text{-0.0067} & a_{8,3} = \text{-0.0033} & a_{8,4} = \text{-0.0024} \\ a_{8,5} = 0.0033 & a_{8,6} = 0.0024 & a_{8,7} = 0.15 & a_{8,8} = 1 \end{array}$$

$$A_{\mathrm{sf}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \tilde{a}_{7,1} & \tilde{a}_{7,2} & \tilde{a}_{7,3} & \tilde{a}_{7,4} & \tilde{a}_{7,5} & \tilde{a}_{7,6} \\ \tilde{a}_{8,1} & \tilde{a}_{8,2} & \tilde{a}_{8,3} & \tilde{a}_{8,4} & \tilde{a}_{8,5} & \tilde{a}_{8,6} \end{bmatrix}$$

$$\begin{array}{llll} \tilde{a}_{7,1} = \text{6.3e-5} & \tilde{a}_{7,2} = \text{-0.00032} & \tilde{a}_{7,3} = \text{-6.3e-5} & \tilde{a}_{7,4} = \text{-0.00036} \\ \tilde{a}_{7,5} = 0.00018 & \tilde{a}_{7,6} = 0.00036 & \tilde{a}_{8,1} = 0.0038 & \tilde{a}_{8,2} = \text{-0.019} \\ \tilde{a}_{8,3} = \text{-0.0038} & \tilde{a}_{8,4} = \text{-0.022} & \tilde{a}_{8,5} = 0.011 & \tilde{a}_{8,6} = 0.022 \end{array}$$

$$B_{\mathrm{s}} = \begin{bmatrix} 0.0167 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0167 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0167 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0167 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0167 \\ b_{7,1} & b_{7,2} & b_{7,3} & b_{7,4} & b_{7,5} & b_{7,6} \\ b_{8,1} & b_{8,2} & b_{8,3} & b_{8,4} & b_{8,5} & b_{8,6} \end{bmatrix}$$

$$\begin{array}{llll} b_{7,1} = \text{-6.3e-5} & b_{7,2} = 0.00032 & b_{7,3} = \text{6.3e-5} & b_{7,4} = 0.00036 \\ b_{7,5} = \text{-0.00018} & b_{7,6} = \text{-0.00036} & b_{8,1} = \text{-0.0038} & b_{8,2} = 0.019 \\ b_{8,3} = 0.0038 & b_{8,4} = 0.022 & b_{8,5} = \text{-0.011} & b_{8,6} = \text{-0.022} \end{array}$$

$$C_{\mathrm{s}}^{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$F_{\mathrm{s}} = \begin{bmatrix} -0.175 & 0.246 & 0.121 & 0.106 & -0.121 & -0.106 & -6.31 & -2.07 \\ 0.295 & -0.676 & -0.295 & -0.258 & 0.295 & 0.258 & 15.4 & 5.05 \\ 0.121 & -0.246 & -0.175 & -0.106 & 0.121 & 0.106 & 6.31 & 2.07 \\ -0.142 & 0.288 & 0.142 & 0.0165 & -0.142 & -0.124 & -7.39 & -2.43 \\ -0.231 & 0.471 & 0.231 & 0.202 & -0.339 & -0.202 & -12.1 & -3.97 \\ 0.142 & -0.288 & -0.142 & -0.124 & 0.142 & 0.0165 & 7.39 & 2.43 \end{bmatrix}$$

$$F_{\mathrm{f}} = \begin{bmatrix} 0.916 & 0.0567 & 0.0215 & 0.0412 & -0.0292 & -0.0412 \\ 0.0524 & 0.794 & -0.0524 & -0.1 & 0.0713 & 0.1 \\ 0.0215 & -0.0567 & 0.916 & -0.0412 & 0.0292 & 0.0412 \\ -0.0252 & 0.0664 & 0.0252 & 0.974 & -0.0343 & -0.0483 \\ -0.0412 & 0.108 & 0.0412 & 0.0788 & 0.87 & -0.0788 \\ 0.0252 & -0.0664 & -0.0252 & -0.0483 & 0.0343 & 0.974 \end{bmatrix}$$