Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-10 (2015) 246–251

Robust distributed control for a mechanical-electrical demonstrator considering communication constraints

A. Kheirkhah * **F. Kempf** ** **T. Tzschichholz** *** **K. Schilling** ****

* *Zentrum für Telematik e. V., Gerbrunn, Germany (e-mail: ali.kheirkhah@telematik-zentrum.de)*
** *Robotics and Telematics Department, University of Würzburg, Würzburg, Germany (e-mail: kempf@informatik.uni-wuerzburg.de)*
*** *Zentrum für Telematik e. V., Gerbrunn, Germany (e-mail: tristan.tzschichholz@telematik-zentrum.de)*
**** *Robotics and Telematics Department, University of Würzburg, Würzburg, Germany (e-mail: schi@informatik.uni-wuerzburg.de)*

Abstract: Networked control systems are currently very common. In the YETE project, it is attempted to construct a system of nodes, which are interconnected by some means of communication, to perform different tasks. Among these tasks is controlling the formation of satellites or robots, depending on where the system is used. Apart from distributed computing, distributed control is a major goal. In this paper, we present the current state of the distributed control algorithm in the form of a demonstrator, which we have used to perform tests and obtain results. Furthermore, we provide a brief overview of relevant networked control literature.

Keywords: robust control, distributed control, h-infinity control, embedded systems, fault tolerance, simulation

1. INTRODUCTION

With the current trend towards small, networked satellites, the need for distributed control algorithms arises as well, since these formations are typically designed to withstand partial system failure (Schilling (2014),Schilling (2011),Kempf et al. (2014)). Thus, controlling and maintaining a formation becomes a task of multiple interacting nodes.

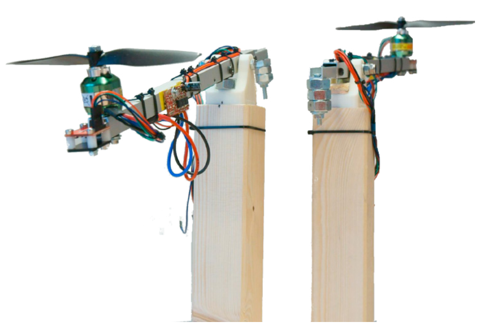

Fig. 1. The current implementation of the demonstrator. A fan is controlled by an ARM board, using a BLDC driver connected via the I^2C bus.

In order to demonstrate developed distributed control algorithms, we have designed a hardware test platform, which is shown in figure 1. The current implementation consists of two pendulums, each equipped with motors and propellers as actuators and an Inertial-Measurement-Unit (IMU) as motion sensor. Two different computational nodes (for which we have used 32-bit ARM processors) are responsible for controlling and thus, stabilizing the position of the pendulums.

In a later stage, both pendulums will be mechanically connected with a bar, on which we intend to put a small ball. The complete scenario then is to control the position of the ball using distributed control.

The paper presents the results of simulations for this complete scenario, starting with a description of the system model. After that, the networked control system is introduced. We conclude the paper with the simulation results and provide an insight in future work.

2. SYSTEM MODEL

As shown in figure 2, the angle is zero in the vertical position what is not an equilibrium for these systems. The objective of the primary control is to stabilize the pendulum around the zero angle.

The equation of movement can be written using the energy functions T and U using the Lagrange method.

$$\begin{aligned} T &= \frac{1}{2}\, J\, \dot{q}^2 \\ U &= m\, g\, l_{\text{COM}}\, (1 + \sin q) \end{aligned} \tag{1}$$

$$J\ddot{q} = -m\, g\, l_{\text{COM}}\, \cos q - b_q\, \dot{q} + l\, F, \tag{2}$$

where l_{COM} is the length from the joint to the center of mass of the arm, J is the total inertia, and b_q is the rotational friction factor.

Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.08.139

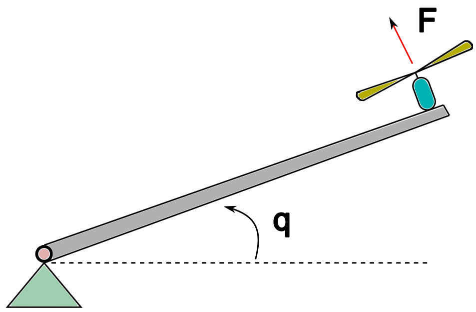

Fig. 2. Simplified model of a pendulum

In order to use the controller synthesis from section 3.5, the linearized and time discrete model of the system is needed. For the linearization the very well-known method of Taylor (Lunze (2013)) is used. The non-linear system in the state space

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$
$$\dot{\boldsymbol{x}} = \begin{bmatrix} x_2 \\ -\frac{m\,g\,l_{\mathrm{COM}}}{J}\cos x_1 - \frac{b_q}{J}x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} l\,F \tag{3}$$

can be linearized at $q = q^*$ in the following form:

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\frac{m\,g\,l_{\mathrm{COM}}}{J}\cos q^* + \frac{m\,g\,l_{\mathrm{COM}}}{J}\sin q^*\,x_1 - \frac{b_q}{J}x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} l\,F \tag{4}$$

or

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ \frac{m\,g\,l_{\mathrm{COM}}}{J}\sin q^* & -\frac{b_q}{J} \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} l\,F + \begin{bmatrix} 0 \\ -\frac{m\,g\,l_{\mathrm{COM}}}{J}\cos q^* \end{bmatrix} \tag{5}$$

To obtain a regular linear system, the new input is defined:

$$\tilde{u} := l\,F - l_{\mathrm{COM}}\,m\,g\,\cos q^* \tag{6}$$

The system would be then described by the following equations:

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\tilde{u} \tag{7}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ \frac{m\,g\,l_{\mathrm{COM}}}{J}\sin q^* & -\frac{b_q}{J} \end{bmatrix};\ \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix}$$

One should note the following points:

(1) The structure of the system (7), especially the matrix $\mathbf{A}$ is constant for a given angle q^*. However, it depends on the equilibrium and the system matrices can vary depending on the given angle to stabilize the system at. Knowing this, an adoption law can be developed to make the linearized model reliable at every given angle. An adaptive controller can be considered using this relation.

(2) The input of the system (7) $\tilde{u}$ has a constant offset for every given angle q^* which is caused by the gravity at a point which is not the physical equilibrium of the plant (in this case $q^* = \pm 90°$). Applying definition (6) for the input, the calculated force of the controller will consider this offset. The required force consists of a part to compensate the gravity effect and a part to stabilize the system or follow given trajectories.

$$F(t) = \frac{l_{\mathrm{COM}}}{l}\,m\,g\,\cos q^* + \frac{\tilde{u}(t)}{l} \tag{8}$$

(3) It is obvious that the maximum thrust to consider as the offset is limited to $m\,g\cos 0° = m\,g$. This means, the angle to linearize the system at is limited to $q \in [-180°, 0°]$ and because the interesting area for the demonstrator is the right half circle, the angle is limited to $q \in [-90°, 0°]$.

(4) For the implementation, we define a non-constant virtual input which linearizes the system at every given point. For this purpose, we replace q^* by the current value of q in equation (8):

$$F(t) = \frac{l_{\mathrm{COM}}}{l}\,m\,g\,\cos q + \frac{\tilde{u}(t)}{l} \tag{9}$$

This input can be generated separately on the micro-controller to adjust the offset in the control input to the current angle, e.g. in case of a trajectory following control.

Further, it is considered that the thrust F generated by the brushless motor and propeller does not jump from a current value to a commanded one. To model the delay and the dynamic behaviour of the propeller system, a PT_1 unit is applied:

$$\beta\,\dot{u}(t) + u(t) = w \tag{10}$$

where w is the commanded value– and here, a virtual input. β is the time constant for the PT_1 unit. Using this input, the system is extended by a new state for the input. Assuming the primary system from (7), the extended system can be defined as:

$$\mathbf{x}_e := \begin{bmatrix} \mathbf{x} \\ \tilde{u} \end{bmatrix}$$
$$\dot{\mathbf{x}}_e = \mathbf{A}_e\,\mathbf{x}_e + \mathbf{B}_e\,w\ ;\ \ y = \mathbf{C}_e\,\mathbf{x}_e$$
where
$$\mathbf{A}_e = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ 0 & \frac{-1}{\beta} \end{bmatrix};\ \mathbf{B}_e = \begin{bmatrix} 0 \\ 1 \\ \frac{1}{\beta} \end{bmatrix}$$
$$\mathbf{C}_e = [\mathbf{C}\ \ 0] \tag{11}$$

After discretization, a third-order linear time discrete system in state space will now describe the dynamics of an arm and the propeller.

3. NETWORKED CONTROL SYSTEM

Packet dropouts are one of the most common problems in networked control systems. Because of the nature of network communications, the dropouts occur randomly and cannot be defined deterministically. In most approaches, a probabilistic process describes the dropout and the controller is designed to achieve stochastic stability of the closed loop. Here, the network is assumed to be modeled as a simple packet dropout generator with a given probability p. The states of the network are modeled by a Markovian chain. For more information, refer to Seiler (2001), Nilsson (1998) or Naghshtabrizi et al. (2007).