59th ILMENAU SCIENTIFIC COLLOQUIUM
Technische Universität Ilmenau, 11 – 15 September 2017
URN: urn:nbn:de:gbv:ilm1-2017iwk-064:3

CONTROL OF A TOWER CRANE BY MEANS OF ALGORITHMIC DIFFERENTIATION

Mirko Franke[1], *Klaus Röbenack*[1], *Robert Weiß*[1] *and Stefan Palis*[2]

[1]Technische Universität Dresden, Fakultät Elektrotechnik und Informationstechnik,
Institut für Regelungs- und Steuerungstheorie, D-01062 Dresden, Germany

[2]Otto-von-Guericke-Universität Magdeburg, Institut für Automatisierungstechnik,
D-39106 Magdeburg, Germany

ABSTRACT

Deriving Euler-Lagrangian equations of motion for sophisticated mechanical systems symbolically often results in complex and large expressions. Based on algorithmic differentiation we present an alternative to the direct implementation of this equations. The Lagrangian provides the starting point for deriving the equations of motion and not necessarily has to be given explicitly. Algorithms containing loops or other control structures will work as well. We will demonstrate the usage of this alternative differentiation method on the system of a tower crane. Experimental results will complete this proposal.

Index Terms— Algorithmic differentiation, automatic differentiation, tower crane, partial linearization

1. INTRODUCTION

As like for other systems in mechatronics and robotics, providing minimal coordinates are found, modeling the system dynamics of a tower crane can be written as a system of first order nonlinear differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}), \tag{1}$$

where $\boldsymbol{F} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear map and $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$ denote state vector and input of the system, respectively. This system easily can be simulated using tools like Matlab, Scilab or Python.

Usually the equations of motion are derived symbolically based on the Lagrangian algorithm. Very often this results in highly complicated nonlinear expressions. As an alternative to implementation of the symbolic equations using algorithmic differentiation (AD) has been suggested in [3, 11, 12]. In simulation this approach turned out as an easy to use method since only the Lagrangian of the system has to be given [3, 11, 12]. In this paper we show the practicality of AD using the example of a tower crane. Note that AD can also be utilized for the simulation of mechanical systems described by Hamilton's equations [8].

The paper is structured as follows. In Section 2 the model of the tower crane is derived based on the Lagrangian algorithm. Section 3 gives a short introduction in computing derivatives using AD. It is shown which derivatives are required in Euler-Lagrangian equations of motion and how their values are computed with AD. In Section 4 a controller design based on partial linearization is described. Application of this controller with an AD based implementation is provided in Section 5. Finally in Section 6 a summary and an outlook is given.

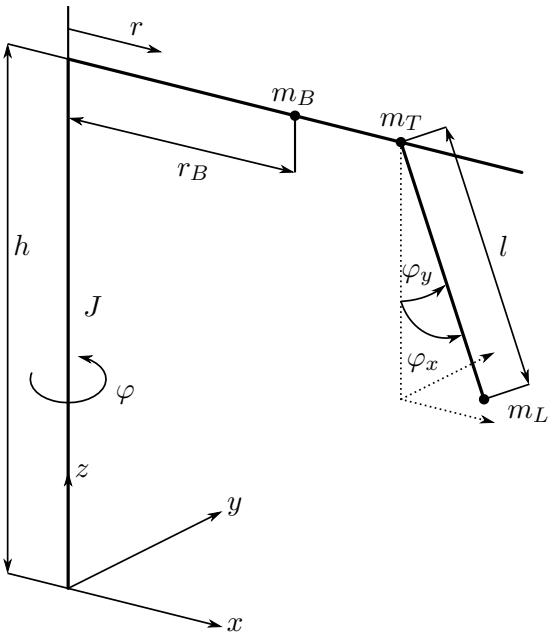

Fig. 1. Schematic figure of the tower crane

2. MODELING

As depicted in Fig. 1 a typical tower crane consists of the rotating tower with a boom. Along the boom the trolley with the load is operated. The length of the rope can be controlled by the hoisting gear. From a theoretical point of view, a tower crane is an underactuated mechanical system [14] with three actuated degrees of freedom: rotation of the tower φ, position of the trolley r and rope length l and the two non actuated degrees of freedom associated with the load φ_x and φ_y.

For modeling the dynamical behavior of the tower crane the Lagrange algorithm can be applied. Here, the equations of motion for each degree of freedom q_i can be given as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i, \tag{2}$$

where Q_i are the generalized forces and L is the Lagrangian, consisting of the kinetic and potential system energies T and V, respectively:

$$L = T - V. \tag{3}$$

For the given tower crane the Lagrangian consists of the following kinetic and potentials energies of each component:

$$T_{\text{tower}} = \frac{1}{2} J \dot{\varphi}^2, \tag{4a}$$

$$T_{\text{boom}} = \frac{1}{2} m_{\text{B}} r_{\text{B}}^2 \dot{\varphi}^2, \tag{4b}$$

$$T_{\text{trolley}} = \frac{1}{2} m_{\text{T}} (r^2 \dot{\varphi}^2 + \dot{r}^2), \tag{4c}$$

$$T_{\text{load}} = \frac{1}{2} m_{\text{L}} \dot{\boldsymbol{q}}_{\text{L}}{}^T \dot{\boldsymbol{q}}_{\text{L}}, \tag{4d}$$

$$V_{\text{load}} = m_{\text{L}} g l (1 - \cos(\varphi_x)\cos(\varphi_y)), \tag{4e}$$

where $\boldsymbol{q}_\mathrm{L}$ is the position of the load

$$\boldsymbol{q}_\mathrm{L} = \begin{bmatrix} r\cos(\varphi) - l(\sin(\varphi)\sin(\varphi_y) + \cos(\varphi)\sin(\varphi_x)\cos(\varphi_y)) \\ r\sin(\varphi) + l(\cos(\varphi)\sin(\varphi_y) + \sin(\varphi)\sin(\varphi_x)\cos(\varphi_y)) \\ h - l\cos(\varphi_x)\cos(\varphi_y) \end{bmatrix}. \tag{5}$$

Therefore, the tower crane Lagrangian consists of the aforementioned system energies (4):

$$L = T_\text{tower} + T_\text{boom} + T_\text{trolley} + T_\text{load} - V_\text{load}. \tag{6}$$

To derive the Euler-Lagrangian equations of motion we first rewrite (2) in a vectorial form

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial \dot{\boldsymbol{q}}} L(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \frac{\partial}{\partial \boldsymbol{q}} L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q}, \tag{7}$$

where $\boldsymbol{q} = (r, \varphi, l, \varphi_x, \varphi_y)^T$ is the vector of generalized coordinates and $\boldsymbol{Q} = (F_\mathrm{T}, M, F_\mathrm{L}, 0, 0)^T$ are the corresponding generalized forces. Here, F_T denotes the force to the trolley, M the torque the drive of the tower generates and F_L is the force to the cable of the load. Next, the total time derivative in (7) will be replaced by partial derivatives using the chain rule:

$$\frac{\partial^2}{\partial \dot{\boldsymbol{q}}^2} L(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \frac{\partial^2}{\partial \boldsymbol{q}\partial \dot{\boldsymbol{q}}} L(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \frac{\partial}{\partial \boldsymbol{q}} L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q}. \tag{8}$$

Using matrix notation this yields

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q} \tag{9}$$

where $\boldsymbol{M}$ denotes the mass of inertia and $\boldsymbol{c}$ includes Coriolis and centrifugal terms as well as terms derived from potential energy. Assuming that the Lagrangian L is regular, i. e.,

$$\det \frac{\partial^2}{\partial \dot{\boldsymbol{q}}^2} L(\boldsymbol{q}, \dot{\boldsymbol{q}}) \neq 0 \tag{10}$$

we can transform (8) into a system of $2n$ first order differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q} = \dot{\boldsymbol{q}} \tag{11a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\dot{\boldsymbol{q}} = \left(\frac{\partial^2}{\partial \dot{\boldsymbol{q}}^2} L(\boldsymbol{q}, \dot{\boldsymbol{q}})\right)^{-1} \cdot \left(\frac{\partial}{\partial \boldsymbol{q}} L(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \frac{\partial^2}{\partial \boldsymbol{q}\partial \dot{\boldsymbol{q}}} L(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{Q}\right). \tag{11b}$$

Defining the state vector $\boldsymbol{x} = (\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T)^T$ we can rewrite (11) in the form of a general nonlinear state space system (1).

It should be mentioned that the derived Lagrangian, mainly due to the coordinate transformation, is already a difficult to read and interpret nonlinear function. Obviously, the situation is getting even more difficult when calculating the derivatives needed for the Euler-Lagrange equations (2). Therefore, it should be clear that modeling of tower cranes as multi-body systems generally results in large nonlinear equations of motion, being hard to handle by hand and providing little physical insight due to their complexity. Thus, in previous contributions, model simplifications have been introduced based on small angle assumptions, fast/slow decomposition and controlled electric drives [7]. This approach of course introduces additional model uncertainties and hence performance and robustness issues. As an alternative to the described symbolic derivation of the equation of motions in this contribution an automatic differentiation based solution strategy will be presented.

3. ALGORITHMIC DIFFERENTIATION

3.1. Motivation

We consider a smooth map $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m$ of which certain derivatives have to be computed. In case $\boldsymbol{F}$ is given as an explicit function, usually symbolic differentiation will be the first choice. For low order derivatives mostly it is very efficient. However, symbolic differentiation will have some limitations as for higher order derivatives the size of symbolically computed derivatives may increase exponentially. Furthermore symbolic differentiation is not applicable if the considered function is not given explicitly but by an algorithm that may contain structures as loops or branches.

When only experimental data are given, numerical differentiation may be the only choice. For explicitly given functions, replacing differential quotient by difference quotient in general results in lack of precision due to truncation and cancellation errors. The computation of derivatives will always be of lower accuracy then the associated function value evaluation [5]. Also numerical differentiation is not very convenient to get higher order derivatives.

Often symbolic expressions for derivatives are not necessarily required. This motivates for automatization of the known rules to compute derivatives. The function under consideration will be split up into a series of elementary functions (multiplication, summation, $\sin, \ln, \ldots$) whose derivatives are known. Evaluating each subexpression numerically will make the main difference to symbolic differentiation and leads to the differentiation method called *algorithmic* or *automatic differentiation* (AD), see [5]. As like the symbolic case, the chain rule will be used to get the entire derivative that now is represented by a set of numerical values instead of symbolic expressions. Thus, the computed derivatives have the same level of precision as the evaluated function. Furthermore exponential increasing memory effort for higher order derivatives will be avoided [9]. Another interesting point is that the function itself does not have to be given in form of a mathematical equation but also can be given as an algorithm.

3.2. Algorithmic Differentiation Principles

AD separates two modes of operation. While the *forward mode* is used to calculate directional derivatives, the *reverse mode* provides weighted gradients. Both modes will be introduced in the following.

Forward Mode

The function $\boldsymbol{F}$ maps a curve $\boldsymbol{x} \in \mathbb{R}^n$ given by a Taylor series

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \boldsymbol{x}_1 t + \boldsymbol{x}_2 t^2 + \ldots + \boldsymbol{x}_d t^d + \mathcal{O}(t^{d+1}) \tag{12}$$

where $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_d \in \mathbb{R}^n$ are vector-valued Taylor coefficients, into a curve $\boldsymbol{y} \in \mathbb{R}^m$, given by

$$\boldsymbol{y}(t) = \boldsymbol{F}(\boldsymbol{x}(t)) = \boldsymbol{y}_0 + \boldsymbol{y}_1 t + \boldsymbol{y}_2 t^2 + \ldots + \boldsymbol{x}_d t^d + \mathcal{O}(t^{d+1}) \tag{13}$$

with vector-valued Taylor coefficients $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_d \in \mathbb{R}^m$. Clearly the coefficient $\boldsymbol{y}_0$ can be obtained by function evaluation at time $t = 0$, i. e.,

$$\boldsymbol{y}_0 = \boldsymbol{F}(x(0)) = \boldsymbol{F}(\boldsymbol{x}_0). \tag{14}$$

Evaluating the time derivative of (13) at $t = 0$, with application of the chain rule one gets the next Taylor coefficient

$$\boldsymbol{y}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}(\boldsymbol{x}(t)) = \boldsymbol{F}'(\boldsymbol{x}_0)\boldsymbol{x}_1, \tag{15}$$

that is the directional derivative of $\boldsymbol{F}$ in direction of $\boldsymbol{x}_1$ at the point $\boldsymbol{x}_0$.

Reverse Mode

The Jacobian matrix $\boldsymbol{F}'(\boldsymbol{x}_0)$ characterizing the push forward $\boldsymbol{F}_* : T_{\boldsymbol{x}_0}\mathbb{R}^n \to T_{\boldsymbol{F}(\boldsymbol{x}_0)}\mathbb{R}^m$, maps the tangent space in $\boldsymbol{x}_0$ into the tangent space in $\boldsymbol{F}(\boldsymbol{x}_0)$. In contrast, the pull back $\boldsymbol{F}^* : T^*_{\boldsymbol{F}(\boldsymbol{x}_0)}\mathbb{R}^m \to T^*_{\boldsymbol{x}_0}\mathbb{R}^n$ is a linear map between the associated cotangent spaces whose elements are covectors. The dual spaces of $\mathbb{R}^n$ and $\mathbb{R}^m$ are denoted by $\mathbb{R}^{n*}$ and $\mathbb{R}^{m*}$ respectively. The weighted gradients we obtain by

$$\bar{\boldsymbol{x}}^T = \bar{\boldsymbol{y}}^T \boldsymbol{F}'(\boldsymbol{x}_0) \tag{16}$$

with $\bar{\boldsymbol{x}}^T \in \mathbb{R}^{n*}$, $\bar{\boldsymbol{y}}^T \in \mathbb{R}^{m*}$. By combination of forward mode and reverse mode also second order derivatives can be computed effectively [5]. Therefore, we apply the reverse mode for both the function evaluation (14) and directional derivative calculation (15). We obtain

$$\bar{\boldsymbol{z}}^T = \bar{\boldsymbol{y}}^T \boldsymbol{F}''(\boldsymbol{x}_0)\bar{\boldsymbol{v}}, \tag{17}$$

where the vector $\bar{\boldsymbol{v}} \in \mathbb{R}^n$ and the covector $\bar{\boldsymbol{y}}^T \in \mathbb{R}^{m*}$ reduce the Hessian matrix $\boldsymbol{F}''(\boldsymbol{x}_0)$ to a covector $\bar{\boldsymbol{z}}^T \in \mathbb{R}^{n*}$.

3.3. ADOL-C

We use the open source AD package ADOL-C (Automatic Differentiation by OverLoading in C++) [15, 16] that is one of the leading software tools for AD of vector valued functions. Using the concept of operator overloading it provides an efficient implementation of AD. The user can choose whether the information that is necessary for the computation of derivatives is stored in a separate structure (trace/tape) or not. The latter will result in the tapeless forward mode where only first order derivatives can be computed. Thus we will focus on the more capable method of using tapes. Generating such tapes requires the following procedure [15, 16]:

- The header file `adolc.h` has to be included.
- The active section containing the function whose derivatives will be required has to be marked. Therefore the statements `trace_on(tag, keep)` and `trace_off(file)` have to be used, where `tag` is an integer, giving the tape a distinct ID. The optional flag `keep` will prepare the tape for an immediate call of the reverse mode. Setting the optional flag `file` forces the tape to be written on hard drive. Otherwise a file for the tape will only be written if the tape exceeds a certain length.
- Variables that may be considered as differentiable quantities, i. e., dependent, independent and intermediate variables of the considered function, have to be declared as active variables. This is done by changing their type from `double` or `float` to `adouble`.
- All independent and dependent variables have to be selected. For independent variables this is done with <<= assignments and for dependent variables with >>= assignments respectively.

For detailed information see the ADOL-C manual [15]. Fig. 2 provides an example listing how to generate the tape of a Lagrangian function. To make the functionality of ADOL-C available for Matlab and to automatize the tape generation process for simulation purpose also an ADOL-C interface for Matlab based on MEX functions has been developed [3].

3.4. Derivatives in Euler-Lagrange Equations of Motion

Simulating a mechanical System using the Euler-Lagrange equations of motion, we have seen that certain derivatives of the Lagrangian are required. We define the state vector $\boldsymbol{x} = (\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T)^T$. The gradient

$$\mathrm{d}L(\boldsymbol{x}) = \begin{pmatrix} \dfrac{\partial L}{\partial \boldsymbol{q}} & \dfrac{\partial L}{\partial \dot{\boldsymbol{q}}} \end{pmatrix} \tag{18}$$

```
#include <adolc/adolc.h>

void lagrangian(int n, double *pq, double pL)
{
    short int tag = 0;                  // tape specifier

    adouble *q = new adouble[n];        // declare independent active variables
    adouble *L = new adouble[1];        // declare dependent active variable

    trace_on(tag);                      // start tracing

    for (int i = 0; i < n; i++)
        q[i] <<= pq[i];                 // select independent variables

    L[0] = ...;                         // Lagrangian

    L[0] >>= pL;                        // select dependent variable

    trace_off();                        // stop tracing
}
```

Fig. 2. Minimal example code for tape generation of a Lagrangian

can be obtained by application of (16). ADOL-C provides the function `gradient` for this purpose. Besides the gradient in term of $\boldsymbol{q}$, also the generalize impulse $\mathrm{d}L/\mathrm{d}\dot{\boldsymbol{q}}$ is returned without any additional computational effort. The second order derivatives, included in the Hessian matrix

$$\frac{\partial^2 L(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^2} = \begin{pmatrix} \dfrac{\partial^2 L}{\partial \boldsymbol{q}^2} & \dfrac{\partial^2 L}{\partial \boldsymbol{q} \partial \dot{\boldsymbol{q}}} \\ \dfrac{\partial^2 L}{\partial \dot{\boldsymbol{q}} \partial \boldsymbol{q}} & \dfrac{\partial^2 L}{\partial \dot{\boldsymbol{q}}^2} \end{pmatrix} \tag{19}$$

can be computed using (17). The required ADOL-C function is `hessian`.

4. CONTROLLER DESIGN

4.1. Partial Linearization

We can see that the right hand side of (9) does not depend on the generalized coordinates $\boldsymbol{q}$ and thus it is easy to split into a fully actuated subsystem and a non actuated subsystem

$$\boldsymbol{M}_{11}\ddot{\boldsymbol{q}}_1 + \boldsymbol{M}_{12}\ddot{\boldsymbol{q}}_2 + \boldsymbol{c}_1 = \tilde{\boldsymbol{Q}}, \tag{20a}$$

$$\boldsymbol{M}_{21}\ddot{\boldsymbol{q}}_1 + \boldsymbol{M}_{22}\ddot{\boldsymbol{q}}_2 + \boldsymbol{c}_2 = \boldsymbol{0}, \tag{20b}$$

where both subsystems are coupled. We denote $\boldsymbol{q}_1 = (r, \varphi, l)^T$ and $\boldsymbol{q}_2 = (\varphi_x, \varphi_y)^T$ the actuated and non actuated coordinates respectively. The input for the actuated subsystem is $\tilde{\boldsymbol{Q}} = (F_\mathrm{K}, M, F_\mathrm{L})^T$. Considering (20b) it can be resolved for the acceleration of the non actuated coordinates

$$\ddot{\boldsymbol{q}}_2 = -\boldsymbol{M}_{22}^{-1}(\boldsymbol{M}_{21}\ddot{\boldsymbol{q}}_1 + \boldsymbol{c}_2). \tag{21}$$

Next, the acceleration $\ddot{\boldsymbol{q}}_2$ in (20a) can be eliminated such that we get

$$\underbrace{\left(\boldsymbol{M}_{11} - \boldsymbol{M}_{12}\boldsymbol{M}_{22}^{-1}\boldsymbol{M}_{21}\right)}_{\tilde{\boldsymbol{M}}}\ddot{\boldsymbol{q}}_1 + \underbrace{\boldsymbol{c}_1 - \boldsymbol{M}_{12}\boldsymbol{M}_{22}^{-1}\boldsymbol{c}_2}_{\tilde{\boldsymbol{c}}} = \tilde{\boldsymbol{Q}}. \tag{22}$$

Model Component	Value
boom length	$760\,\mathrm{mm}$
tower height	$1580\,\mathrm{mm}$
maximal rope length	$1300\,\mathrm{mm}$
load mass	$3.2\,\mathrm{kg}$
trolley mass	$1.2\,\mathrm{kg}$

Table 1. Laboratory model parameters

We choose $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{M}}\boldsymbol{v} + \boldsymbol{c}$, where $\boldsymbol{v} \in \mathbb{R}^3$ is a new virtual input that is equivalent to acceleration of the fully actuated subsystem

$$\ddot{\boldsymbol{q}}_1 = \boldsymbol{v}, \tag{23a}$$

$$\ddot{\boldsymbol{q}}_2 = -\boldsymbol{M}_{22}^{-1}\left(\boldsymbol{M}_{21}\boldsymbol{v} + \boldsymbol{c}_2\right). \tag{23b}$$

This is equivalent to an input-output linearization where actuated coordinates $\boldsymbol{q}_1$ are treated as output. In particular, this corresponds to a collocated partial linearization [13]. We obtain a linear system (23a) for the actuated subsystem while (23b) represents an internal dynamics [10].

We can see, for partial linearization mass matrix $\boldsymbol{M}$ and vector $\boldsymbol{c}$ are required. According to Section 3.4 Jacobian and Hessian matrix of the Lagrangian L are required and can be computed using AD.

4.2. Stabilizing Control

We will design a control law such that actuated coordinates $\boldsymbol{q}_1$ are following a sufficiently smooth desired trajectory $\boldsymbol{q}_{1,\mathrm{d}} : [0, \infty) \to \mathbb{R}^3$. The consequential tracking error is defined by $\tilde{\boldsymbol{q}}_1 := \boldsymbol{q}_1 - \boldsymbol{q}_{1,\mathrm{d}}$ for which we propose the dynamics

$$\boldsymbol{0} = \ddot{\tilde{\boldsymbol{q}}}_1 + K_\mathrm{D}\dot{\tilde{\boldsymbol{q}}}_1 + K_\mathrm{P}\tilde{\boldsymbol{q}}_1, \tag{24}$$

where $K_\mathrm{P}, K_\mathrm{D}$ are diagonal matrices of dimension 3×3 with positive entries. In order to damp internal dynamics (23b), that can be interpreted as oscillation of the load, the classical PD approach (24) will be extended by feedback of the load angles φ_x and φ_y. This results in the control law

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_{1,\mathrm{d}} - K_\mathrm{D}\dot{\tilde{\boldsymbol{q}}}_1 - K_\mathrm{P}\tilde{\boldsymbol{q}}_1 - \begin{pmatrix} k_x\varphi_x \\ k_y r^{-1}\varphi_y \\ 0 \end{pmatrix} \tag{25}$$

where $k_x, k_y > 0$.

5. EXPERIMENTAL RESULTS

5.1. Experimental Platform

The laboratory model of the tower crane used in experiment is shown in figure 3. The associate parameters are given in Table 1. The rotation of the tower is driven by a worm drive at the bottom. For movement of the trolley the jib is provided with a chain drive. A steel wire representing the hoist is driven by a winch system. All motors are equipped with incremental encoders for measuring position and speed of tower rotation, trolley and load. For real time operation purpose a DSpace controller board DS1103 is used. An inertial motion unit (IMU) including magnetometer, accelerometer and gyroscope is mounted on the hook to measure deflection of the load. It communicates to the DSpace controller via the serial protocol using Bluetooth.

For the AD based control task we use a Raspberry Pi (RPI) running Raspbian Jessie (kernel version 4.9) that is communicating with the DSpace system via a Waveshare High-Precision AD/DA Board [17]. This structure avoids necessity of compiling ADOL-C for the DSpace Power PC.

Fig. 3. Test station

5.2. Friction Compensation

In section 2 we deduced a point mass model for the tower crane were friction was completely neglected. In practice this is a very strong restriction that would make the model inapplicable for the test station. To overcome the influence of friction we follow the approach suggested in [6]. For demonstration we will give a short sketch for friction identification of the drive of the tower only. For the trolley the identification process is analogous. We consider the dynamics of the tower without any influence of the trolley and the load

$$J\ddot{\varphi} = M_{\mathrm{T}} - d_{\mathrm{T}}, \tag{26}$$

where d_{T} denotes friction of the tower drive. For this friction term we assume it depends on angular velocity $\dot{\varphi}$ and the momentum M_{T} of the drive. Taking a simple Taylor series approximation leads to

$$d_{\mathrm{T}}(M_{\mathrm{T}}, \varphi) \approx d_{\mathrm{T},0} + d_{\mathrm{T},1} M_{\mathrm{T}} + d_{\mathrm{T},2}\dot{\varphi} + d_{\mathrm{T},3} M_{\mathrm{T}}\dot{\varphi} \tag{27}$$

where $d_{T,0}, \ldots, d_{T,3} \in \mathbb{R}$ are coefficients to be identified. Introducing

$$\begin{aligned} n(M_{\mathrm{T}}) &:= -d_{\mathrm{T},0} - d_{\mathrm{T},1} M_{\mathrm{T}} + M_{\mathrm{T}}, \\ m(M_{\mathrm{T}}) &:= -d_{\mathrm{T},2} - d_{\mathrm{T},3} M_{\mathrm{T}}, \end{aligned} \tag{28}$$

for constant M_{T} results in linear dependence between $\ddot{\varphi}$ and $\dot{\varphi}$

$$J\ddot{\varphi} = m(M_{\mathrm{T}})\dot{\varphi} + n(M_{\mathrm{T}}). \tag{29}$$

Taking step responses for various momentums M_{T} approximately results in straight lines in the $\dot{\varphi}$,$\ddot{\varphi}$ -plane. For this step responses $m(M_{\mathrm{T}})$ and $n(M_{\mathrm{T}})$ can be fitted and thus the coefficients $d_{T,0}, \ldots, d_{T,3}$

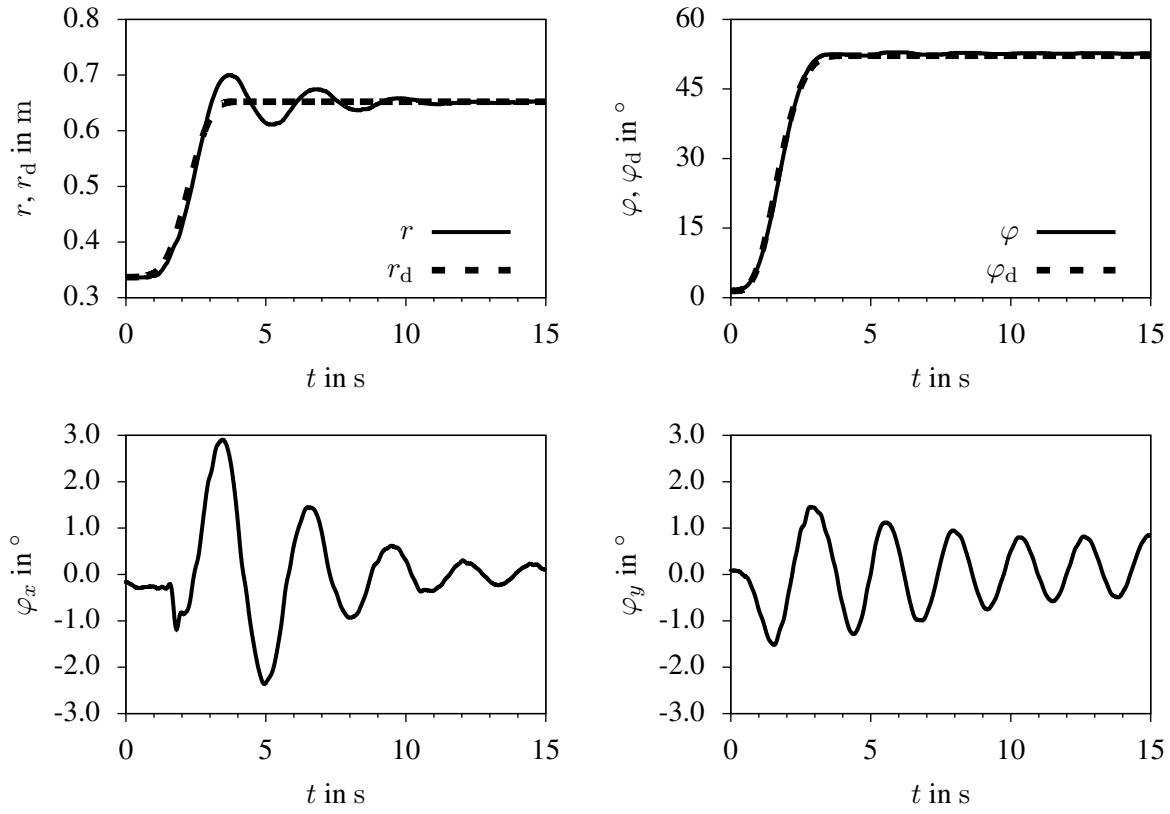

Fig. 4. Experimental results, transition between rest positions

can be determined. Finally, the momentum M_T to be set to achieve a desired acceleration $\ddot{\varphi}_\mathrm{d}$ at some point with an actual angular velocity $\dot{\varphi}$ is

$$M_\mathrm{T} = \frac{J\ddot{\varphi}_\mathrm{d} + d_{\mathrm{T},0} + d_{\mathrm{T},2}\dot{\varphi}}{1 - d_{\mathrm{T},1} - d_{\mathrm{T},3}\dot{\varphi}}. \tag{30}$$

It has to be noted that step responses have to be taken for $M_\mathrm{T} < 0$ and $M_\mathrm{T} > 0$ that will result in negative and positive angular velocities respectively. When friction depends on $\mathrm{sign}(\dot{\varphi})$ this results in different coefficients depending on the direction the tower is rotating and for (30) one also have to switch between these two sets of coefficients.

5.3. Experiment

To validate practicability of the proposed AD based controller design a simple polynomial trajectory to convey between rest positions has been chosen. This trajectory was designed in Cartesian coordinates to achieve a straight path between rest positions. We explicitly do not wanted to solve a boundary value problem as described in [4] to generate a trajectory that already ensures that the load is in rest after transition. This was done to verify load oscillation damping.

Figure 4 visualizes the trajectories of the position of the trolley and the angular position of the tower. The length of the cable was kept constant with $l \approx 1.3\,\mathrm{m}$. The transition starts at $(r_\mathrm{d}, \varphi_\mathrm{d})(0) \approx (0.34\,\mathrm{m}, 1.5°)$ and ends at $(r_\mathrm{d}, \varphi_\mathrm{d})(T) \approx (0.65\,\mathrm{m}, 52°)$ where transition time was set to $T = 4\,\mathrm{s}$. It can be seen that the transition causes load oscillations. In direction of the jib, this oscillation were damped by the controller in a greater extend than the oscillation in the other direction. This behavior is caused by friction in tower drive that could not be compensated completely. This also shows up in position trajectories of the jib and the tower. While the jib is moving more actively to damp oscillations, the tower keeps rather steady.

6. SUMMARY AND OUTLOOK

AD is a powerful tool that not only can be used for a simulation of mechanical systems where just the Lagrangian has to be given [12] but also for control purpose. For a controller based on partial linearization or for Computed Torque [1, 2] the same derivatives are required as for system simulation. The advantage of using AD is that the Euler-Lagrangian equations do not have to be explicitly given as for complex mechanical systems they can become complicated with very long expressions. The partial linearization approach using AD was practically applied to a tower crane test station.

In experiment it has been shown that friction compensation is quite challenging. Due to this the experimental results lag the results that can be achieved in simulation [3]. To overcome this problem a more sophisticated friction compensation is required. It could be noticed that friction at the test station also depends on position. This property is not jet mapped by the used friction model. A completely different approach would be using nonlinear model predictive control (NMPC) instead of partial linearization.

Another point is the sampling rate. Transferring data between the DSpace system and the RPI turned out to be the bottleneck of the control structure. A direct implementation of the controller on the Power PC therefore is worth striving for. This had not been done so far as this requires compiling the ADOL-C package for the DSpace system.

References

[1] C. H. An, C. G. Atkeson, J. D. Griffiths, and J. M. Hollerbach. Experimental evaluation of feedforward and computed torque control. *IEEE Transactions on Robotics and Automation*, 5(3): 368–373, 1989.

[2] M. Franke. *Lösung regelungstechnischer Aufgabenstellungen mit Hilfe des Algorithmischen Differenzierens*. Createspace, Dresden, 2015.

[3] M. Franke, K. Röbenack, A. Wobar, and R. Weiß. Simulation und Regelung eines Drehkrans mit Hilfe des algorithmischen Differenzierens. In *Fachtagung Mechatronik*, pages 272–277, Dresden, 2017. T. Bertram, B. Corves, K. Janschek.

[4] K. Graichen and M. Zeitz. Inversionsbasierter Vorsteuerungsentwurf mit Ein- und Ausgangsbeschränkungen (Inversion-Based Feedforward Control Design under Input and Output Constraints). *Automatisierungstechnik*, 54(4):187–202, 2006.

[5] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, second edition, 2008.

[6] C. Knoll. *Regelungstheoretische Analyse- und Entwurfsansätze für unteraktuierte mechanische Systeme*. PhD Thesis, TU Dresden, 2016.

[7] F. Palis and S. Palis. *High Performance Tracking Control of Automated Slewing Cranes*. In-tech, 2008.

[8] S. Palis and F. Palis. Mechanical system simulation via automatic differentiation. In *Proc. of Advanced Problems of Mechanics*, 2010.

[9] K. Röbenack. *Regler- und Beobachterentwurf für nichtlineare Systeme mit Hilfe des Automatischen Differenzierens*. Shaker Verlag, Aachen, 2005.

[10] K. Röbenack. *Nichtlineare Regelungssysteme: Theorie und Anwendung der exakten Linearisierung*. Springer, Berlin, Heidelberg, 2017.

[11] K. Röbenack, J. Winkler, and C. Knoll. Direct simulation of mechanical control systems using algorithmic differentiation. In *Proc.* 56^{th} *International Scientific Colloquium*, Ilmenau, 2011.

[12] K. Röbenack, J. Winkler, and M. Franke. Simulation of Holonomic Mechanical Systems by Means of Automatic Differentiation. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, EOOLT'14, pages 43–46. ACM, 2014.

[13] M. W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Proc. of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS '94)*, volume 1, pages 314–321, Sept. 1994.

[14] M. W. Spong. Underactuated mechanical systems. In B. Siciliano and K. P. Valavanis, editors, *Control Problems in Robotics*, pages 135–150. Springer-Verlag, London, 1998.

[15] A. Walther and A. Griewank. *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, Version 2.6.3-stable*, 1 2017. URL `https://projects.coin-or.org/ADOL-C/export/743/stable/2.6/ADOL-C/doc/adolc-manual.pdf`. (26. July 2017).

[16] A. Walther, A. Griewank, and O. Vogel. ADOL-C: Automatic differentiation using operator overloading in C++. *Proceedings in Applied Mathematics and Mechanics*, 2(1):41–44, 2003.

[17] Waveshare. *High-Precision AD/DA Board, User Manual*, 10 2015.