Influence of Plant Model Variants for the Automatic Optimisation of Control Parameters

Patrick Bouillon and Jörg Frochte and Markus Lemmen
Dept. of Electrical Engineering and Computer Science
Hochschule Bochum
D 42579 Heiligenhaus, Germany
Email: joerg.frochte@hs-bochum.de

***Abstract*—Designing controllers in a model based engineering environment has become more important in industrial applications. As the name already suggests, the model based design process depends strongly on the plant model and its quality as well as on the accuracy of the corresponding model parameters, which sometimes are estimated or even just guessed. The goal of this paper is to investigate the automation potentials of the model based design process, concretely in this paper the parametrization of two given cascaded controllers for a plant, which might vary over time or due to certain events. Transferring controller parametrization to a computer which is in contrast to a human engineer not sensible of the real life use cases, makes the plant model even more important. In this work we want to underline this point by a case study and provide some insights into the possibilities and limitations of automated parametrization of controllers during a model based design process for two cascaded controllers.**

I. Introduction

The underlying concept of this paper is to design a workflow such that the model based automated controller parameter fitting supports changes in the considered plant of a closed loop system. The envisaged workflow is illustrated in fig. 1. The following root causes are probably the reasons for changes

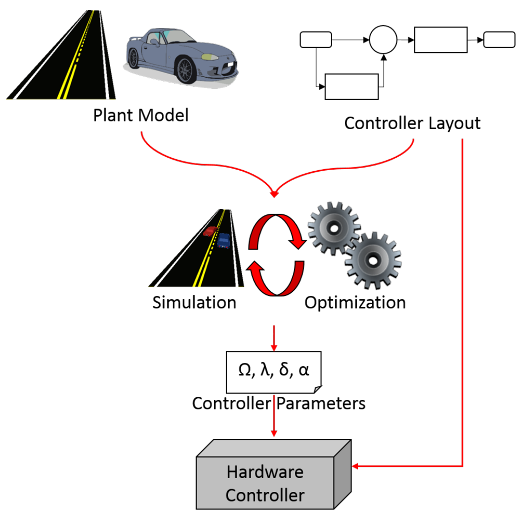

Fig. 1: Model based design with automatic optimisation

in the behavior of the plant: the plant might alter its behavior over time (ageing, different loads to be transported, different operating points etc.) or due to certain impacts F(e.g. high stresses causing deformations in bearings). Another reason why automated parameter fitting is desired is that the controller can be used for e.g. different product variants (e.g. station wagons instead of sedan cars etc.) with identical controller setups.

Whatever actual plant has been used in the model based design of the controller, the controller(s) and its parameters are quite often tested within HIL-testbeds prior to real life testing, e.g. within the V-process for mechatronic and automotive products.

In this paper we focus on the sensitivity of the (non-linear) plant model and the consequences and impacts of plant variations for a (semi-) automatic optimisation of the linear controller parameters within a fixed cascaded controller structure in the model based design environment. Most control engineers prefer to work with linear plant models due to their simplicity and well understood theoretical framework. An experienced control engineer can take uncertainties and inaccurate plant behavior into account by adequate adaption of the initially linearized designed controller. For a machine doing the parameter fitting of the controller(s) by optimisation techniques, it is assumed that this linearized design would lead to a very suboptimal solution. Hence, the hypothesis is twofold: On the one hand, it is hypothesized that such an automated process requires more detailed and accurate models and on the other hand, the process enables the reduction or even elimination of the hand fitting of the controller parameters, if they are transferred to the real life problem set.

Regarding the optimisation one may choose from a wide range of methods. One possible choice is the use of traditional numerical optimisation methods, e.g. Newton-like approaches [1], gradient descent, etc. or techniques from machine learning like reinforcement learning or data mining techniques to support the users [2].

For the case being studied here, we will consider traditional numerical optimisation methods and a plant model, in which we discuss the effect of different common friction models and their effects on the optimisation results. A friction model is a suitable model aspect to consider, since it is one of the first aspects modelers tend to eliminate or at least to simplify—and friction usually changes over life and accordingly operation time.

Thus, the paper is organized as follows: In section II we give a description of the studied test case. Different ways to model the investigated problem in particular regarding the friction as an important part of a nonlinear plant are presented in III. The controller layout is included in this section as well. Section IV deals with the mathematical optimisation of the linear controller parameters for nonlinear plant with varying behavior within the closed loop control system. We present the results of our investigation in section V and draw conclusions in section VI.

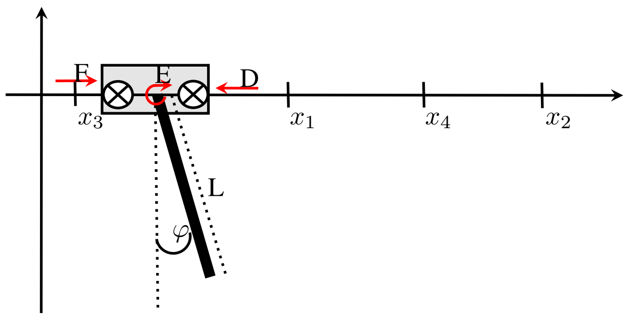

Fig. 2: Problem set

II. Application example for the case study

Fig. 2 depicts the problem set of the the application example used as case study: A crane trolley is at position x. The shaft is deflected by an angle φ. The control task of the crane trolley is to move to a given list of positions x_i and stop at each of them within a certain accuracy. The list of positions should be reached in the order of their indeces, so first x_1, second x_2 etc. While moving to a position x_i, the angular deflection φ is supposed to be kept within a certain bound. The behavior of the crane trolley and its load can be controlled by applying a couple of standard controllers from linear control theory. One common approach for this kind of problems is a cascaded controller design for angle deflection and crane trolley position. Hence, the resulting controller scheme is a cascade of two linear standard controllers. Once the linear controller type is chosen, the controller parameters have become fixed according to the given design targets for the closed loop control system: the bounds for the deflection angle and positional accuracy in combination with additional measures for the transient controlled behavior of the complete system. Thus, the design task for the controls is split into two parts: First, the choice of the control structure and controller type and second, the optimisation of the controller parameters in order to stay within the given targets for the closed loop controlled system.

In section III-A we have fixed the structure of the controller as well as the type of the linear controllers. For the controller cascade remain eight parameters to be optimized.

The control task can be interpreted as a sequence of positioning tasks of the crane trolley. For each task the trolley starts at a given position x and has to move to position x_i. In this context *reaching the position* x_i is defined as staying around position x_i and having almost zero velocity $|\dot{x}|$, all tolerances ε_x, ε_v. Thus, we consider a target point x_i to be successfully reached, if

$$|x - x_i| < \varepsilon_x \text{ and } |\dot{x}| < \varepsilon_v \ .$$

In this case study we use $\varepsilon_x = 10^{-2}\,\mathrm{m}$ and $\varepsilon_v = 10^{-4}\,\mathrm{m\,s^{-1}}$ as absolute tolerances.

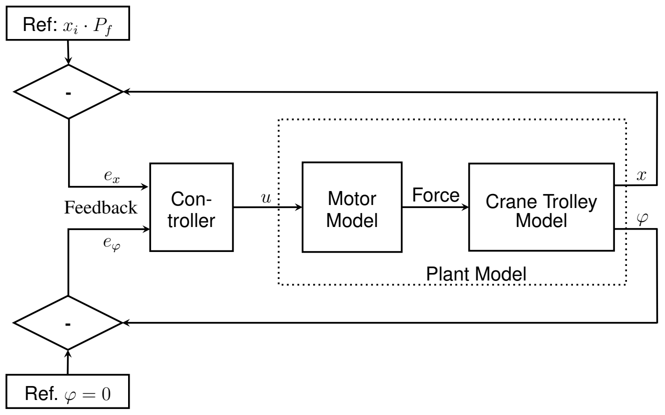

Fig. 3: Block model of the simulation

III. Crane Trolley - Models and Controllers

Fig. 3 shows a high level block diagram of the system used for our study. This kind of modular design enables the engineer to build different variants of the model simply by exchanging different parts or components of the plant model. For this paper we provide three Crane Trolley Models that differ regarding the friction model. More details of the used model are given in section III-B3.

In fig. 3 we make use of the following conventions:

- $u \in [-1, 1] \subset \mathbb{R}$ can be interpreted as a pulse-width modulation (PWM) signal and is a real valued scalar between -1 and 1. A value of 1 means full power forward, -1 backward and 0 no power.
- P_f in fig. 3 is a scalar, which corrects the changing closed loop systems gain during the optimisation process of the controller parameters. More details are given in section III-A.
- x is the position of the trolley.
- φ is the deflection angle of the crane trolley shaft or of a hoist rope, respectively.

For our study the reference value for φ is zero and the reference value for the position x is the position x_i from the list of target positions.

A. Controller Scheme

Controlling a cran trolley is technically similar to controlling an inverted pendulum, where different approaches from the literature are known (see e.g. [3], [4]). We decided for our study to follow a common and well established linear controller choice in a cascaded structure on the one hand for

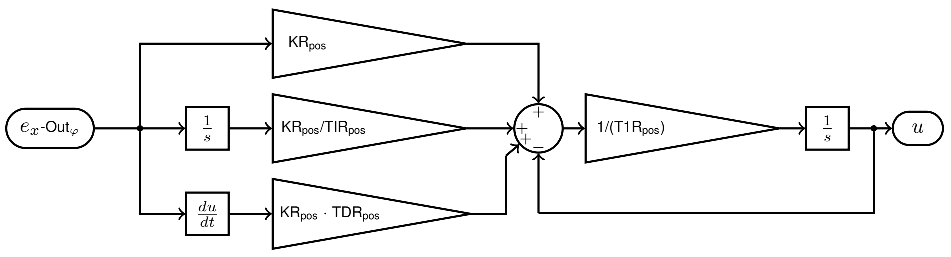

Fig. 5: Position controller layout from the subsystems

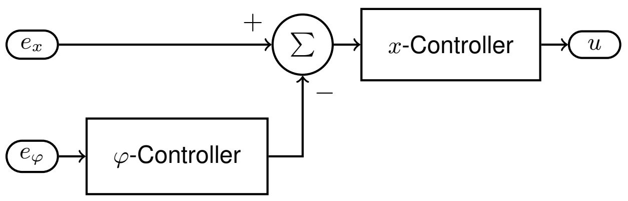

Fig. 4: Top-level controller scheme

position and on the other hand for the deflection angle. The scheme of the cascade control is shown in fig. 4. The cascade consists of an inner and an outer control loop. In a cascaded control structure, usually the inner loop is supposed to be the faster loop, while the outer loop may (slightly) lag compared to the inner one. The inner controller calculates an valuefor the actuator such that the input difference of the inner controller vanishes. The output of the inner controller is the actuator input—in our case this is the motor input, i.e. basically a PWM duty cycle ratio. The outer loop varies the input of the controller of the inner loop in order to achieve a change of the actuator variable of the plant and to decrease its outer control loop error, as well.

In our application study we assume the position of the trolley to be prioritized over the deflection angle. Thus, we choose the x-controller for the inner control loop and the φ-controller for the outer. As controller type we use a standard linear $PIDT_1$-system as x-controller (c.f. fig. 5) and a standard linear PDT_1-system as φ-controller (c.f. fig. 6). In addition to the $PIDT_1$-system as x-controller, we add a linear gain prefilter P_f (c.f. Fig. 3), since we have to take into account that generally speaking the closed loop system may have a different overall static gain compared to the plant model and that the closed loop controlled system may vary depending on the choice of the actual parameters of the controller undergoing the later optimisation. Hence, we utilize the prefilter $P_{f,\text{ideal}} = 1/K_{\text{cl}}$ to compensate for varying static gains of the closed loop system K_{cl}. This static gain can be determined by the limit value of the step response $K_{\text{cl}} = \lim_{t\to\infty} h(t)$ or from the closed loop transfer function $K_{\text{cl}} = G(s)|_{s=0}$ for proportional-type linear systems. Please note, that the actual value of the prefilter gain P_f will be calculated during the optimisation and will not be obtained by measuring or estimating the overall closed loop gain K_{cl}. Please note further, that P_f does vary strongly depending on the operating point of the plant, since the plant in our case is usually nonlinear, while the optimized controllers are linear systems—and due to the plants nonlinear behavior, also K_{cl} will vary and is strongly depending on the actual operating point of the plant, which is unknown to the optimisation procedure.

Therefore, we now have fixed the controller type and structure but leave the controller parameters of each controller open for automatic optimisation. Consequentially, we have 8 degrees of freedom for optimisation, the 8 controller parameters KR_{pos}, TIR_{pos}, TDR_{pos}, $T1R_{pos}$ for the $x-$controller and KR_φ, TDR_φ, $T1R_\varphi$ for the φ-controller as well as and P_f. Coherently, the optimisation task is to find a controller parameter vector

$$\begin{aligned} c_c = \quad & [\ KR_{pos}, TIR_{pos}, TDR_{pos}, \dots \\ & T1R_{pos}, KR_\varphi, TDR_\varphi, T1R_\varphi, P_f\ ] \in \mathbb{R}^8 \end{aligned}$$

resulting in a better, maybe even optimal behavior of the closed loop controlled system. The properties of the plant model to be controlled are discussed in the following section III-B.

B. Plant Models

As depicted in fig. 3, the plant model consists of a motor and a crane trolley model. The crane trolley model is compound of a mechanical model of the crane itself and the friction models for the two motions: the angular motion of the rotatinal shaft and the linear motion of the translational movement of the crane trolley as well. In the next sections, we study three different friction variations for the linear motion and two for the angular motion.

1) Motor Model: While the controller provides an output $u \in [-1, 1]$ as actuator variable, this signal is transformed by the motor model into a force. This force F is used as the single input in the crane trolley model (see fig. 2).

The used motor model represents a direct current motor with permanent magnets as stator. The dynamic behavior can be approximated as

$$\frac{L}{K_1}\dot{M}(t) + \frac{R}{K_1}M(t) = u(t) - \frac{K_2}{r_p}\dot{x}(t) \quad ; \qquad (1)$$

with the inductance of the rotor L_a, the resistance of the rotor R_a, the torque constant K_1 , the electromotive force constant K_2, the radius of the pulley r_p and the resulting motor torque M. The rotor voltage U is modelled as $U(t) = \hat{u} \cdot u$ and

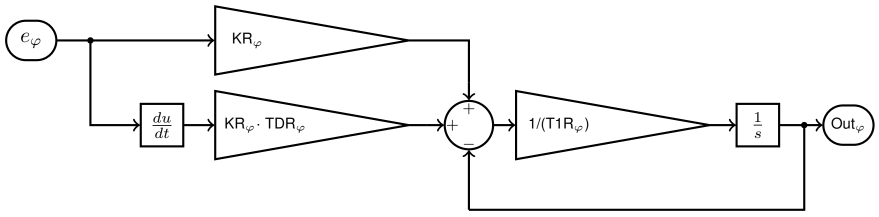

Fig. 6: φ controller layout from the subsystems

limited by the max. motor voltage $\hat{u}$. With the radius of the pulley r_p we calculate the force $F = \frac{M(t)}{r_p}$.

2) Crane Trolley Model: For the simulation of the plant we need to derive the equations of motion of the trolley and the shaft. In order to support an easy exchange of the friction models, we model the plant using Newton mechanics instead of the usually used Lagrange mechanics for the model.

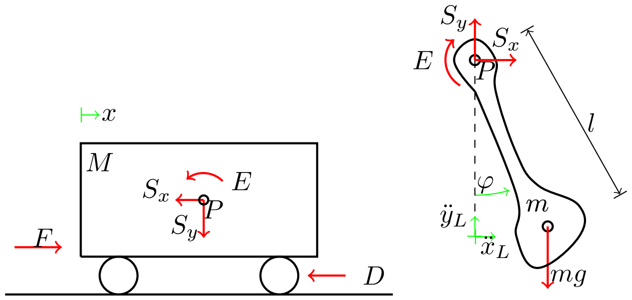

Fig. 7: Force and moment balances of trolley and shaft

First, we balance the forces and moments according to the sketch in fig. 7

$$M\ddot{x} = -S_x + F - D, \tag{2}$$
$$m\ddot{x}_L = S_x, \tag{3}$$
$$m\ddot{y}_L = -mg + S_y, \tag{4}$$

where M is the mass of the trolley, m the mass of the shaft, F the force provided from the motor, D the friction of the trolley and E the friction moment of the shaft.

The shaft motion x_L can be calculated from the motion x by

$$x_L = x + l\sin(\varphi) \tag{5}$$
$$\Rightarrow \ddot{x}_L = \ddot{x} - \dot{\varphi}^2 l \sin(\varphi) + \ddot{\varphi} l \cos(\varphi) \tag{6}$$

accordingly, we deduce in y_L-direction

$$y_L = l - l\cos(\varphi) \tag{7}$$
$$\Rightarrow \ddot{y}_L = \dot{\varphi}^2 l \cos(\varphi) + \ddot{\varphi} l \sin(\varphi) \tag{8}$$

Equations (2), (3) in (6) yields the differential equation of the trolley motion:

$$(M+m)\ddot{x} = \dot{\varphi}^2 ml\sin(\varphi) - \ddot{\varphi} ml\cos(\varphi) + F - D. \tag{9}$$

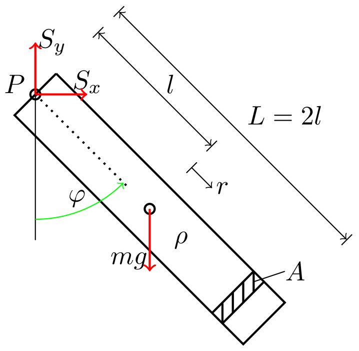

Fig. 8: Homogeneous shaft

The torque balance around the shaft axis using (3) and (4) is described by

$$\theta_S\ddot{\varphi} = -S_x l\cos(\varphi) - S_y l\sin(\varphi) - E,$$
$$\theta_S\ddot{\varphi} = -m\ddot{x}_L l\cos(\varphi) - m\ddot{y}_L l\sin(\varphi) - mgl\sin(\varphi) - E.$$

Using (6) and (8) to eliminate $\ddot{x}_\mathrm{L}$ and $\ddot{y}_\mathrm{L}$ leads to the differential equation of the shaft

$$\left(\theta_S + ml^2\right)\ddot{\varphi} = -\ddot{x}ml\cos(\varphi) - mgl\sin(\varphi) - E \tag{10}$$

The moment of inertia θ_S is calculated for a homogeneous shaft (see Fig. 8) with the cross-sectionial area A, the overall length L and the distance between axis and the center of mass l. As the shaft is homogeneous and A is constant, we have the relation of l and L as follows:

$$L = 2l \tag{11}$$

The moment of inertia θ_S to the center of mass is

$$\theta_S = \int_V \left(\vec{r}_\perp\right)^2 \rho(\vec{r})\mathrm{d}V = \int_{-l}^{l} r^2\rho A\,\mathrm{d}r = \rho A\int_{-l}^{l} r^2\mathrm{d}r$$
$$\theta_S = \frac{2}{3}\rho A l^3 = \frac{2}{3}\rho A\left(\frac{L}{2}\right)^3 = \frac{1}{12}\rho A L^3 \tag{12}$$

With the mass of the shaft $m = \rho AL = 2\rho Al$ where A and ρ are constant with respect to r, we get for the moment of inertia

$$\theta_S = \frac{1}{12}mL^2 = \frac{1}{3}ml^2 \tag{13}$$

3) Friction model of the crane trolley: The friction of the trolley $D(\dot{x})$ is generally described by

$$D(\dot{x}) = F_{NT} \cdot h(\dot{x})$$

with the normal force F_{NT} of the trolley and the velocity-dependent friction coefficient $h(\dot{x})$.

In total we distinguish between the three following different friction models for the crane trolley, where a) is the most elaborate model, b) is a simplified one and c) is a friction-free model.

a) In the most detailed model, also known as Stribeck friction model, we assume a friction decrease with increasing velocity for a certain velocity region near standstill, while we observe an increase in friction with speed for higher velocities (c.f. Fig. 9). Therefore the model reflects static friction as well as a speed dependent dynamic friction which is shown in Fig. 9 and is also referred to as linear viscous damping. Thus, this friction model is a superposition of an exponential and a linear function which changes its coefficient at a certain velocity. The friction coefficient $h(\dot{x})$ is modelled for $\alpha > \beta$ as

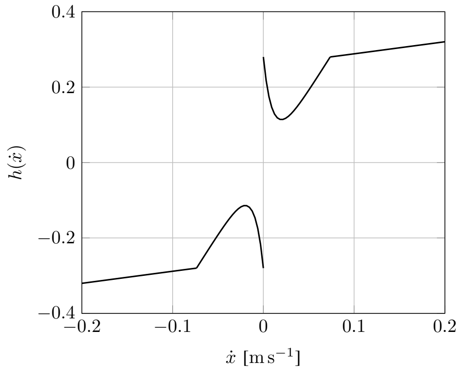

Fig. 9: Coefficient for stribeck model friction

$$h(\dot{x}) = \mu_s \operatorname{sgn}(\dot{x}) \exp\left(-\frac{|\dot{x}|}{T_m}\right) + \begin{cases} \alpha\dot{x} & ;\text{if } \sim\mathbf{A} \text{ OR } \sim\mathbf{B} \\ \beta\dot{x} + \gamma & ;\text{if } \mathbf{A} \text{ AND } \mathbf{B} \end{cases}$$

with the conditions $\mathbf{A}$ and $\mathbf{B}$ as

$$\mathbf{A} := |\mu_s \operatorname{sgn}(\dot{x}) \exp\left(-\frac{|\dot{x}|}{T_m}\right)| < |\alpha\dot{x}|$$

$$\mathbf{B} := C_{rr} < \left|\mu_s \operatorname{sgn}(\dot{x}) \exp\left(-\frac{|\dot{x}|}{T_m}\right) + \alpha\dot{x}\right|$$

and

- Static friction coefficient $\mu_s \approx 0.27$ (Steel on Steel)
- Coefficient $C_{rr} \approx 0.28$ (where C_{rr} can be interpreted similar to the rolling resistance of hardened steel ball bearings on steel)
- Here, we use $\alpha = 3.8$, $\beta = 0.32$, $T_m = 0.01$ which lead to $\gamma = 0.2563$.

The calculation of the static friction at standstill is obtained with the maximum static force F_s by

$$F_s = (m + M) \cdot g \cdot \mu_s$$

at standstill.

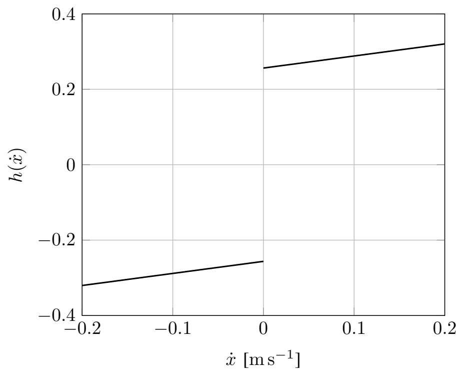

Fig. 10: Simplified friction model coefficient $h(\dot{x})$

b) Model b) is a simplified friction model of the elaborate type model a) for the crane trolley. In this model the friction coefficient depends linearly on the speed $\dot{x}$ of the trolley. It is defined as

$$h(\dot{x}) = \dot{x}\beta + 0.2563,$$

which is depicted in Fig. 10. Therefore, in model type b) the crane trolley friction is basically modelled as a combination of Coulomb friction and linear viscous damping.

c) The last model is the negation/omission of friction. Hence, so there is no friction at all modelled. That implies a friction coefficient

$$h(\dot{x}) = 0.$$

In general, the discontinuity at zero velocity in the first two model variations yields difficulties for the simulation, which already have been issued in some publications, e.g. [5] and [6, chapter 9].

4) Friction Model of the shaft: The friction model for the shaft must be chosen accordingly to the friction model for the trolley from the following two variants:

- With no friction at the trolley, there is also no friction at the shaft.
- In case of modelled trolley friction a) or b), we assume for the rotation a friction moment E as a Coulomb friction. The friction moment with proper orientation is described as

$$E = F_{NS}\mu_b r_b \cdot \frac{200\dot{\varphi}}{\pi} \tag{14}$$

with the radius of the bearings r_b, the specific friction coefficient μ_b and the normal force of the shaft F_{NS}. For the normal force, we consider both the weight force and the centripetal force, when the shaft is in motion. Hence, we obtain the following highly nonlinear normal force multiplier for the shaft friction

$$F_{NS} = ml\dot{\varphi}^2 + mg\cos(\varphi). \tag{15}$$

IV. Mathematical Optimization

A. Optimization Function

In order to enable (semi-)automatic optimisation we simulate the complete closed loop controlled model for an actual set of control parameters c_c

$$[I_{ITAE}, \varphi, x, t_{\mathrm{end}}] = \mathrm{simulation}(c_c).$$

The return values of the simulation are used to evaluate the performance of the control task. Among other control performance evaluation criteria the ITAE criterion

$$I_{ITAE} = \int_0^{t_{\mathrm{end}}} \|error\|_2 \cdot t \, \mathrm{d}t \quad \text{with}$$
$$error = \begin{pmatrix} x(t) - x_i \\ \varphi(t) - 0 \end{pmatrix}$$

is used. The choice for this criterion is driven by the fact that weighting over time t is used. The weighting should compensate for the initial error, which is always large. Secondly, this choice rewards fast controllers one would prefer for this task, as well.

The upper limit for the simulation time to reach the three points of the list of target positions is set to 10 seconds. For each control parameter set we require reaching the target within this time and as well $e(t) = 0$ for $t > t_{\mathrm{end}}$. Consequently, we stop the integration of the ITAE performance criteria at that point in time.

From the optimisation problem's point of view we want to find the minimum of a function:

$$f: \begin{array}{l} \mathbb{R}^8 \to \mathbb{R}^+ \\ x \mapsto w_1 I^x_{ITAE} + w_2 t_{\mathrm{end}} + w_3 I^{\varphi}_{ITAE} + w_4 \max\limits_{0<t\leq t_{\mathrm{end}}} |\varphi| \end{array}$$

The choice of the weights w_i depend on the requirements of the task. Weights w_1 and w_2 are associated with the requirements for the task to reach the target position points as fast as possible, while w_3 and w_4 refer to the bounds of the angular deflections of the shaft during the control motions. Additionally, these weights are used for normalization purposes. We perform our test with $w_1 = w_2 = 0.1$, $w_3 = \frac{1}{10\pi}$, $w_4 = \frac{2}{\pi}$.

B. Evaluation of Optimization Techniques

When it comes to applying common techniques from optimisation to our problem we have to keep in mind that each function call of f means a simulation. Thus, each function call requires significantly more time than a usual common call on a problem set given by some algebraic equations. Therefore, the goal is to use a technique that leads to fast convergence with as less as possible function calls for f.

If we consider second order newton-like methods, see. e.g. [1], or first order gradient descent approaches, see e.g. [7], most function calls occur, while building the gradients or the Hesse matrix.

Let us consider a function $f : \mathbb{R}^n \to \mathbb{R}$. With

$$a_i = f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_n)$$

and

$$b_i = f(x_1, \ldots, x_{i-1}, x_i - h, x_{i+1}, \ldots, x_n).$$

Thus, we achieve

$$\frac{\partial f}{\partial x_i} f(\vec{x}) \approx \frac{a_i - b_i}{2h}$$

To minimize the calculation time for the function calls, one can re-use these values a_i und b_i again in order to calculate the second order partial derivatives. Additionally, one has to compute

$$c = f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$$
$$\frac{\partial^2 f}{\partial x_i^2} f(\vec{x}) \approx \frac{a_i - 2c + b_i}{h^2}$$

In addition, we have the mixed derivatives:

$$d_{ij} = f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_{j-1}, x_j + h, x_{j+1}, \ldots, x_n)$$
$$e_{ij} = f(x_1, \ldots, x_{i-1}, x_i - h, x_{i+1}, \ldots, x_{j-1}, x_j - h, x_{j+1}, \ldots, x_n)$$

$$\frac{\partial^2 f}{\partial x_i x_j} f(\vec{x}) \approx \frac{d_{ij} - a_i - a_j + 2c - b_i - b_j + e_{ij}}{2h^2}$$

For $n = 8$ this results in 28 simulation calls for d and 28 for e, provided the assumption we may assume that this Hesse matrix will fulfill the symmetry of second derivatives.

Consequently, computing the gradient and Hesse Matrix results in 73 function calls.

$$73 = \underbrace{1}_{c} + \underbrace{8}_{a_i} + \underbrace{8}_{b_i} + \underbrace{28}_{d_{ij}} + \underbrace{28}_{e_{ij}}$$

In contrast to this method the first order approach using the gradient requires 16 function calls. Therefore, the choice of the optimisation technique has a most significant influence on the computation costs and simulation time required.

Assuming that a simulation is being calculated five times faster than real time, a single simulation may take 2 seconds for our given example as worst case. Thus, a newton approach would result in calculation times of 146 seconds, while a gradient method would be calculated within 32 seconds.

Please note, that these function calls are completely independent from each other and thus can be straight forward parallelized in computation.

Today, a quad-core processor is standard state-of-the-art, while for workstations an octa-core is often used. Therefore, we can perform—in our case using the MATLAB Parallel Toolbox on a workstation—a gradient being calculated in about 4 sec. and a Hesse matrix built in about 19 seconds.

Another approach, which also is not dependent on a calculation of derivatives is commonly known as *hill climbing* methods, see e.g. [8]. These methods can be used to find an extremum by starting with an initial guess of the solution and then sampling local area around the solution by changing the current solution slightly in order to find in our case a $f(x_{new}) < f(x_{current})$. The computation costs per step depend on how many samples are needed before an update may be performed. Again, one should use the parallel architectures in today's workstations because the samples to calculate an optimisation direction do not depend on each other. For our study, we use the following adapted version of a hill climbing method for our problem set:

Algorithm 1 Parallel Cartesian/Stochastic Hill Descent

```
Require: $x_{start}$, $\Delta x[]$, $h[]$, $m$
 1: $x_{new} := x_{start}$
 2: compute $f(x_{start})$ by simulation
 3: repeat
 4:     $x_{old} = x_{new}$
 5:     for all $\Delta x[]$ do
 6:         parallel pick $2 \cdot 8$ samples with distance $\Delta x[j]$ from
            $x_{old}$ in the direction of the Cartesian coordinate $j$
 7:         add samples to sample list $S$
 8:     end for
 9:     for $i = 1$ to $m$ (parallel) do
10:         for all $h[]$ do
11:             $change$ = compute random vector $r$ with
                $r(i) \in [-0.5, 0.5]$
12:             compute $r := r \cdot x_{old} \cdot h[j]$
13:             pick sample $f(x_{new} + change)$
14:             add samples to sample list $S$
15:         end for
16:     end for
17:     selected sample $x_{\text{new}} = \text{argmin}_{\tilde{x} \in S} f(\tilde{x})$
18: until $|f(x_{new})| < |f(x_{old})|$
19: return $x_{new}$
```

In our tests it turned out that the second order Newton-like approach is not reasonable concerning the computational cost. Therefore, we tested the plant models with the gradient descent method and an adapted version of the Hill climbing method given in algorithm 1.

The function f seems to have many local minima, in which especially the gradient descent method tends to be stuck. Thus, we include a local minimum check before the gradient descent method finishes: Prior to finishing the algorithm in each Cartesian direction a sample check is performed, if there is a better solution available than the actual one. In case a better solution is found, the gradient descent starts again from this improved starting point.

V. Results

We parametrized our plant model based on an educational experimental bench in our controls education lab. Coherently, we use the values given in table I for the crane trolley and the values in table II for the motor model.

TABLE I: Trolley characteristics

trolley mass	0.36 kg
shaft mass	0.16 kg
shaft length	0.3 m
friction coefficient μ_s	0.1

As already discussed in section IV-B, each of the optimisation methods needs a starting vector. Its choice strongly influences the convergence speed and sometimes even the stability. Depending on that starting vector in combination with the randomized vector r at line 11 of Alg. 1, the computation time for the optimisation differs between a few minutes and up to three hours for our study. Of course, the chosen numerical method influences the results regarding performance and accuracy. A null vector as initial guess makes no sense considering the discussed controller layout—e.g. this would result in no control activity at all or one would divide by zero etc. Thus, a reasonable initial guess to start the computation is e.g.

TABLE II: Motor characteristics

rotor inductance L	1 mH
rotor resistance R	2 Ω
torque constant K_1	0.02 N m A^{-1}
electromotive constant K_2	0.02 N m A^{-1}
Maximum Voltage $\hat{u}$	12 V
pulley radius r_p	0.005 m

$$c_c = [1, 1, 0.1, 0.1, 0.1, 0.1, 0.1, 1] \; .$$

Both test cases we used require from the trolley to move 1.3 m. Table III shows the results for a quite homogeneous use case where P_f can be set nearly optimal, while IV shows the results for a use case, where the distances between the way points tend to differ more.

The values seem similar at first sight. However, one has to keep in mind that small changes might cause very different behaviors. Therefore, fig. 11 and fig. 12 show the big differences in the transient behavior, when using the hill climbing parameter set

$$\hat{c}_c = [1.0419, 1.2771, 0.1602, 0.0069, 0.2083, 0.0946, 0.0453, 0.9947]$$

from table IV of the non-friction model for the other two friction models. Both models suffer due to the additional

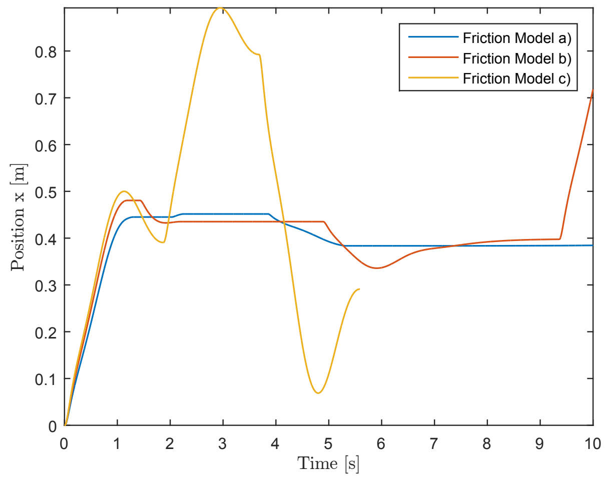

Fig. 11: x for different friction models with $\hat{c}_c$

friction. In case of friction model b), one of the position

TABLE III: Calculated parameters for the way-points [0.4, 0.8, 0.3]

Model	Methods	KR_{pos}	TIR_{pos}	TDR_{pos}	$T1R_{pos}$	KR_{φ}	TDR_{φ}	$T1R_{\varphi}$	P_f	Time Required
Stribeck friction	gradient descent	1.3544	0.8222	0.2019	0.0072	0.1317	0.1198	0.0223	0.9459	5.2997
& Shaft friction	Hill climbing	1.3542	0.7304	0.1991	0.0403	0.1238	0.1183	0.0295	0.9603	5.7528
Linear friction	gradient descent	1.3410	1.1865	0.1225	0.0623	0.1780	0.1091	0.0849	0.9833	5.8738
& Shaft friction	Hill Climbing	1.3418	1.1860	0.1229	0.0627	0.1763	0.1088	0.0954	0.9879	5.8722
No Trolley Friction	Gradient Descent	0.9966	1.0020	0.1301	0.0914	0.1691	0.1018	0.0795	0.9850	6.2771
& No Shaft friction	Hill Climbing	1.0419	1.2771	0.1602	0.0069	0.2083	0.0946	0.0453	0.9947	5.5890

TABLE IV: Calculated parameters for the way-points [0.7, 0.3, 0.5]

Model	Methods	KR_{pos}	TIR_{pos}	TDR_{pos}	$T1R_{pos}$	KR_{φ}	TDR_{φ}	$T1R_{\varphi}$	P_f	Time Required
Stribeck friction	gradient descent	1.3393	0.9046	0.2049	0.0406	0.1347	0.1233	0.0310	0.9790	5.7488
& Shaft friction	Hill climbing	1.3408	0.9047	0.2050	0.0421	0.1330	0.1240	0.0292	0.9818	5.7554
Linear friction	gradient descent	1.3277	1.2237	0.1698	0.0048	0.2148	0.1068	0.0977	0.9763	6.9054
& Shaft friction	Hill climbing	1.3304	1.2208	0.1648	0.0080	0.1988	0.1100	0.0950	0.9896	5.7178
No trolley friction	gradient descent	0.9940	1.0147	0.1445	0.0737	0.1911	0.1026	0.0605	0.9818	6.1760
& No Shaft friction	Hill climbing	1.0306	1.2758	0.1603	0.0019	0.2248	0.0841	0.0571	1.0079	5.7739

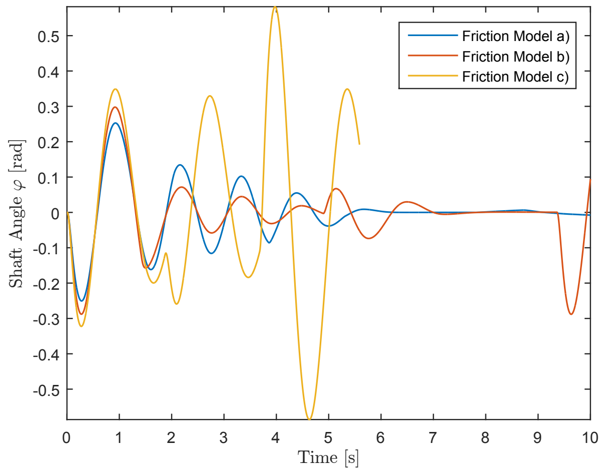

Fig. 12: $\varphi(t)$ for different friction models with $\hat{c}_c$

targets is met, while none at all is met with model a). Fig. 12 illustrates the benefit of the shaft friction regarding controlling φ. An examination of the two benchmark values t_{end} and $\varphi_{\max}$ in table IV and III shows, that both criteria are taken into account. In case weights $w_3 = w_4 = 0$ are chosen, the result would be some rollover of the shaft. In contrast, a choice of $w_1 = w_2 = 0$ or just some too small numbers, might result in a controller setting that does not lead to any motion at all.

VI. Conclusion

In this paper, we propose an approach for a (semi-) automatic fitting of the parameters of linear cascaded controllers for varying nonlinear plants and evaluate different optimisation techniques for this purpose. For the investigated problem, gradient descendent and an adapted hill climbing variant have been used. Both methods result in well-working parameter sets. It is shown that quite detailed plant models are required, since otherwise the computer tends to compute suboptimal solutions.

Future work is needed to extend the verification to more general problems and different controller layouts. Beyond the results presented here, further development is required. Indeed, the combination of a hill climbing approach with some machine learning for the optimisation of picking samples might lead to higher robustness and performances for the optimisation. Alternatively, one could use the trajectory of x and phi as a feature for parameter changes based on machine learning. To automatically choose proper start values Data Mining in the sense of [2], [9] or [10] seems to be a promising starting point for further developments as well.

Acknowledgment

The authors would like to thank J. Weber from the controls and automotive systems research lab for providing measurement results on a crane trolley for the Stribeck friction fitting.

References

[1] P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, ser. Springer Series in Computational Mathematics. Springer, 2004. 1, 6

[2] S. Burrows, B. Stein, J. Frochte, D. Wiesner, and K. Müller, "Simulation data mining for supporting bridge design," in *Proceedings of the 9th Australasian Data Mining Conference, CRPIT*, vol. 121, 2011, pp. 163–170. 1, 8

[3] O. Föllinger, *Regelungstechnik: Einführung in die Methoden und ihre Anwendung*. VDE, 2008. 2

[4] R. C. Dorf and R. Bishop, *Modern Control Systems*. Pearson Education, 2011. 2

[5] M. Otter, H. Elmqvist, and S. E. Mattsson, "Hybrid modeling in modelica based on the synchronous data flow principle," in *Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on*. IEEE, 1999, pp. 151–157. 5

[6] F. E. Cellier and E. Kofman, *Continuous system simulation*. Springer New York, 2006, vol. 1. 5

[7] S. Koziel and X. Yang, *Computational Optimization, Methods and Algorithms*, ser. Studies in Computational Intelligence. Springer, 2011. 6

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010. 6

[9] I. Bernst, P. Bouillon, J. Frochte, and C. Kaufmann, "An approach for load balancing for simulation in heterogeneous distributed systems using simulation data mining," in *Proceedings of the 11th International Conference Applied Computing 2014*, H. Weghorn, Ed. Porto, Portugal: IADIS, 2014, pp. 254–258. 8

[10] S. Burrows, J. Frochte, M. Völske, A. B. Martnez Torres, and B. Stein, "Learning overlap optimization for domain decomposition methods," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Springer Berlin Heidelberg, 2013, vol. 7818, pp. 438–449. 8