IEEJ Journal of Industry Applications
Vol.9 No.2 pp.109–116 DOI: 10.1541/ieejjia.9.109

Paper

LCToolbox: Facilitating Optimal Linear Feedback Controller Design

Jan Swevers[*a)] Non-member, Laurens Jacobs[*] Non-member
Taranjitsingh Singh[*] Non-member, Dora Turk[*] Non-member
Maarten Verbandt[*] Non-member, Goele Pipeleers[*] Non-member

(Manuscript received April 5, 2019, revised July 24, 2019)

Optimal linear feedback controller synthesis methodologies have the potential to become a worthy alternative to standard design methodologies, particularly for high-end complex systems. Although these design methodologies have been around for more than two decades, the majority of industrial control engineers are still reluctant to adopt this new design paradigm. Motivated by these observations, the MECO Research Team at KU Leuven is developing a Linear Control Toolbox (LCToolbox). LCToolbox is an open-source MATLAB toolbox providing a high level of abstraction to apply linear optimal controller design methods. Its key features are the signal-based modeling framework that allows the user to construct the control configuration, the identification module that interfaces several routines in a uniform manner, and the control module that allows the user to specify an optimal feedback controller design problem in an intuitive way and solve it as such. Moreover, LCToolbox supports state-of-the-art B-spline-based linear parameter-varying (LPV) modeling and controller design techniques, unlocking the true potential of advanced feedback controller synthesis methodologies. The design of an LPV controller for an overhead crane is used as a running example throughout the paper to illustrate all features of LCToolbox.

Keywords: robust control, $\mathcal{H}_\infty$ loop shaping, control design software, CACSD, MATLAB

1. Introduction

In order to make the $\mathcal{H}_\infty$ controller design method more accessible to control engineers in industry, a linear control toolbox for MATLAB, 'LCToolbox', is being developed. Although The Mathworks' MATLAB software includes plenty of routines for linear system identification, controller design, and simulation, in many cases the end user is still left with tedious pre- and postprocessing steps. Typical examples are the design of weighting functions, the construction of the generalized plant, closed-loop simulations with various model types, etc. Furthermore, the support for LPV systems is currently limited to simple parameter dependencies. Since parameter-dependent identification and control is capable of improving the control performance for nonlinear plants, software tools supporting more complex parameter dependencies [(1)] are appealing.

The open-source 'LCToolbox' focuses on integrating and interfacing existing software packages for these purposes in a consistent way for end users [(2)]. These software packages consist of the readily available MATLAB tools (e.g. Robust Control Toolbox, Signal Processing Toolbox) as well as third party software and in-house developed routines. We illustrate the design procedure within this new toolbox using a mechatronic case study: identification and feedback control design and evaluation of a 2D overhead crane.

a) Correspondence to: Jan Swevers. E-mail: jan.swevers@kuleuven.be
* MECO Research Team, Dept. of Mech. Eng., KU Leuven
DMMS lab, Flanders Make
3001 Leuven, Belgium

The structure of this paper is as follows. Section II defines the case study we consider throughout the paper to illustrate the different modules of the toolbox: the modeling, system identification and control design modules (section II-IV). Section V discusses the simulation capabilities of the toolbox and compares simulation and experimental results. Some concluding remarks are finally drawn in section VI.

2. Case Study: A 2D Overhead Crane

The considered overhead crane is depicted in Fig. 1.

The base of the system consists of a velocity-controlled trolley with time constant τ and reference u. The encoder on the trolley's track provides a measurement of x_0. The trolley is equipped with a hoisting mechanism that allows the system to vary the cable length L, which is also being measured. Moreover, L is constrained between 0.25 m and 0.8 m. Another sensor on the hoisting mechanism measures the load angle θ.

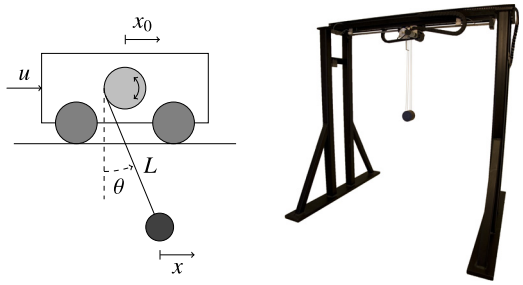

Fig. 1. Schematic and picture of the considered overhead crane. The cart is driven by the velocity reference u. x_0, θ and L are measured

The dynamics of the overhead crane are summarized in following equations:

$$\begin{cases} \ddot{x}_0 &= -\tau^{-1}(\dot{x}_0 - u) \\ \ddot{\theta} &= L^{-1}\left(\tau^{-1}(\dot{x}_0 - u)\cos\theta - g\sin\theta - c\dot{\theta}\right) \\ x &= x_0 + L\sin\theta \end{cases} \cdots\cdots\cdots (1)$$

This equation clearly shows the nonlinear coupling between the state and input variables, and the dependency on the measured variable L.

3. Modeling Module

At the core of the presented toolbox resides a general system modeling module. This module adopts a new design paradigm which makes a clear distinction between systems and models. A system reflects a physical entity, e.g. a pump, an electrical circuit or a chemical process. All of these have inputs and outputs which are intrinsically related. This relationship is never truly known but one or more models of the same system might be available. Therefore a system can be regarded as a collection of models which all refer to the same physical entity. The advantage of this approach is that different models can be consulted when most relevant. For instance, the controller design might be based on a simplified linear model whereas a time-domain simulation of a high-fidelity nonlinear model is preferable for a validation.

The following paragraphs shed a brighter light on the toolbox's understanding of systems and signals, models and parameters, and their mutual relation.

3.1 Systems and Signals Within the Linear Control Toolbox, systems and signals serve the purpose of describing a configuration, i.e. how different components interact with each other. Systems are created via the function call

```
>> crn = IOSystem(ni,no);
>> K = IOSystem(no,ni);
```

where `ni` and `no` denote the number of inputs and outputs of the overhead crane `crn`, 1 resp. 2. Apart from the system to be controlled, we also create a system `K`, representing the controller that is sought for. Inputs and outputs are internally stored as a custom `Signal` object and can be accessed via:

```
>> crn.in, crn.out
```

The user is also encouraged to use signal aliases which make the code easier to read:

```
>> u = crn.in; x0 = crn.out(1); th = crn.out(2);
```

where `x0` and `th` represent the cart position x_0 and swing angle θ respectively. Moreover, new signals can be constructed either via the `Signal()` call or as a linear combination of other signals, for example:

```
>> r = Signal(n);
>> e = r - x0;
```

where the optional argument `n` indicates the dimension of the signal, in our case 1. Signals serve the purpose of connecting systems together. A connection is established by simply equating the signals with the `==` operator and creating a new system from the subsystems and the connection list:

```
>> c1 = (K.in == [e;th]);
>> c2 = (K.out == u);
```

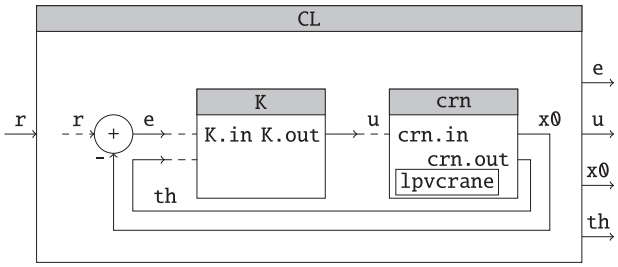

Fig. 2. Schematic of the plant constructed throughout section 3. Systems `crn` and `K` are connected to form system `CL` with input `r` and outputs `e`, `u`, `x0` and `th`. `lpvcrane` is the model added to the system `crn` in section 4.3. The system `K` remains empty as long as no control law is available

```
>> CL = IOSystem(crn,K,[c1;c2]);
```

In this case, the new system `CL` is constructed from `crn` and `K`, arranged in a standard error feedback configuration. A schematic representation is found in Fig. 2.

3.2 Models Multiple mathematical models of different complexities and with various representations may exist for such a system. The toolbox manages to make different models interact with each other by using its own model types. If a standard MATLAB counterpart exists, however, these are bi-directionally compatible for LTI models. Examples are `SSmod` (`ss`), `TFmod` (`tf`), `ZPKmod` (`zpk`). Nevertheless, the toolbox versions also allow these models to contain scheduling parameters, so that a standard MATLAB representation is not available. A scheduling parameter is defined using the `SchedulingParameter` object:

```
>> p = SchedulingParameter(name,range,rate);
```

This object has three attributes: a name, a range indicating the bounds of parameter values and a range for the rate of variation, keeping track of the minimum and maximum change of the parameter value over time. Under the hood, a `SchedulingParameter` is a B-spline based identity function. As such, the user can explicitly construct parameter-dependent state-space matrices through standard operations, as long as the result can again be represented by a B-spline (e.g. sum, product, concatenation). The resulting matrices are then fed into the earlier commands (e.g. `SSmod`) to obtain an LPV model. For the overhead crane the scheduling parameter is defined as:

```
>> L = SchedulingParameter{'L', [0.25 0.8],[0 0]);
```

Two other relevant model types in the context of this work are `ODEmod`, which can handle semi-explicit differential algebraic equations of index 1 and `FRDmod`, the counterpart of the standard `frd` model, containing nonparametric frequency response data. The former, although not directly suitable for LTI or LPV control design, allows time domain simulations of closed-loop systems, which are particularly interesting for validation of the closed-loop performance of the designed controller(s). The latter is relevant to verify nonparametric frequency responses. Finally, an uncertain model class `Umod` (in some sense related to MATLAB's `umat` and `uss`) is introduced to define an uncertain model exhibiting unstructured uncertainty, i.e. the model could be expressed as the linear

fractional transformation (LFT) of a certain part and a normalized uncertain part.

4. System Identification

LCToolbox is interfaced with frequency identification methods that are elaborated in detail in (3), accompagnied by their MATLAB implementations (4). Linear parameter-varying identification is possible through the so-called local LPV identification method SMILE [5]. The regularized SMILE and regularized nonlinear least squares approaches [6][7] will be integrated in the toolbox in the near future. Also routines from the MATLAB System Identification Toolbox are planned to be interfaced.

4.1 Excitation Signals and Measurement Data For the overhead crane, we want to identify a continuous-time B-spline based LPV model with the cable length L as the scheduling parameter. The identification data set consists of 9 frequency response functions (FRFs) corresponding to the following fixed values of the cable length:

```
>> lgrid = [0.2500, 0.2988, 0.3298, 0.3660, 0.4084,
            0.4587, 0.5000, 0.5918, 0.8000]
```

Each FRF estimate is based on 7 realizations of a random-phase multisine excitation, and for each realization 10 periods of the steady-state system response were measured. The multisines have a frequency resolution $df = 0.0244$ Hz and a frequency range $f \in [0.0244, 5]$ Hz.

One period of a random-phase multisine signal with the aforementioned specifications is generated using following command:

```
>> u = Multisine('label', 'experiment1', 'fs', 100,
   'fwindow', [0.0244 5], 'freqres', 0.0244);
```

`u` is the so-called `TimeSignal` object, containing all information that is characteristic for these specific signals. Additional name-value pairs allow the user to specify more options, e.g. the phase distribution of the signal, its amplitude spectrum, etc. `TimeSignal` objects feature useful methods as well. The user can for example plot their spectrum by

```
>> plotSpectrum(u, 'dft');
```

The actual signal content (one multisine period in this case) is obtained by

```
>> [s,t] = signal(u);
```

Time domain measurements are imported in the toolbox by wrapping them into `TDMeasurementData` objects. The constructor for such an object is called as follows:

```
>> real1 = TDMeasurementData('label',
   'realization1', 'excitation', u,
   'data', [x0,th,u], 'datalabels',
   {'x0', 'theta', 'u'}, 'periodic', true);
```

Similar to `TimeSignal`, this way of storing measurement data makes it possible for the control engineer to call relevant methods on the object. For example, to avoid inclusion of transient behavior in the measurement, we only keep the last 10 periods of the measurements in our case study:

```
>> real1 = clip(real1, 'lastnper', 10);
```

Complementary to `TDMeasurementData`, an `FDMeasurementData` class that can handle nonparametric FRFs is provided as well. This class is useful in case the user prefers to perform a nonparametric identification in another software environment.

4.2 Nonparametric Frequency Response Function Estimation By performing several experiments –each with a different realization of a random-phase multisine– and by considering several periods during each experiment, it is possible to estimate an FRF that is the Best Linear Approximation (BLA) of the nonlinear system response, and the sample total variance of the FRF estimate resulting from stochastic nonlinear distortions and output noise [3]. To this end, two methods that are elaborated in (3) are interfaced: a robust detection algorithm for nonlinearities (`Robust_NL_Anal`) and an algorithm based on the robust local polynomial (RLP) method (`RobustLocalPolyAnal`).

Returning to the overhead crane case, we calculate for a value of the cable length the average FRF, referred to as the best linear approximation (BLA) [3], based on the 7 measurements as follows:

```
>> IOlabels.input = {'u'};
>> IOlabels.output = {'x0', 'th'};
>> [FRF, info] = nonpar_ident(real1,real2,real3,
   real4,real5,real6,real7,
   IOlabels,'Robust_NL');
```

All 9 frequency response frequency response functions are gathered into a grid of non-parametric models `gfrf`:

```
>> gfrf = Gridmod({FRF1,FRF2,...,FRF8,FRF9},
          {'L',lgrid});
```

4.3 Parametric LPV Model Identification The total 1-input, 2-output model is identified in two steps. First the relation between input u and cart position x_0 is identified. This relation does not depend on the scheduling parameter and hence a second order LTI model is fitted to the FRF obtained by averaging the FRFs estimated for the 9 different cable lengths. The resulting model contains one integrator and a real pole at $s = -\sigma = -63.4475$ rad/s. In order to ensure that exactly the same poles are present in the model relating input u and swing angle θ, the effect of these poles is removed from the estimated FRFs relating u to θ by multiplying them with $j\omega(j\omega + \sigma)$. The second identification step comprises of identifying a second order LPV model to these compensated FRFs. Due to page limitations, only this second identification step is discussed. It consists of three steps. First, an LTI model G_i is fitted to each of the 9 compensated FRFs following the guidelines of maximum likelihood frequency domain identification [3], i.e., using the inverse of the estimated sample total variance of the corresponding FRF to build the weighting matrix. Considering the shape of the estimated FRFs, continuous-time models of the second order and with two zeros turned out to be most suitable. Note that fitting LTI models to local snapshots of a nonlinear system introduces a certain modeling error and an LPV model based on these LTI models will –inevitably– inherit this error.

```
>> FRF_W1 = 1./FRF1.G.stdNL;
>> config = struct('denh', 2, 'denl', 0, 'numh', 2,
   'numl', 0, 'FRFW', FRF_W1);
```

```
>> [FRF, info] = param_ident('data', FRF1, 'method',
   'nllsfdi', 'config', config);
```

Second, an LPV model is fitted through these LTI models using the SMILE technique and a B-spline basis for the scheduling parameter [6][7]. The B-spline basis is constructed using the Cox-de Boor recursive formula [8]. This requires a choice of the spline degree g and a set of knots k. In this particular case we select $g = 2$ and take following spline knot sequence spanning the entire range of the scheduling parameter to form `kgrid`:

```
>> kgrid = [0.2500, 0.2500, 0.2500, 0.2988,
            0.3298, 0.3660, 0.4084, 0.4587,
            0.5000, 0.8000, 0.8000, 0.8000];
```

This knot sequence, with repeated knots at the boundaries of the range of the scheduling parameter, ensures that the first derivative of the polynomial spline is continuous on the open interval defined by the scheduling parameter range. With these choices, the number of basis functions is equal to the number of LTI models and hence an exact interpolation of the LTI models is realized. With these settings, the SMILE method is applied resulting in the so-called `lpvsmile` model. The SMILE interpolation is readily carried out in LCToolbox by means of the following commands:

```
>> gmod = Gridmod({G1,G2,...,G8,G9}, {'L',lgrid});
>> basis = BSplineBasis(kgrid,2);
>> lpvsmile = SSmod(gmod,basis,L);
```

First a `Gridmod` that acts as an interpolated LPV model is created based on the identified LTI models and their respective parameter values. Provided a B-spline basis `basis`, `gmod` is readily converted to an `SSmod` via a SMILE interpolation.

In the last step we apply the regularized nonlinear least squares approach, using the SMILE model as initial guess. This regularized nonlinear least squares approach uses the inverse of the estimated sample total variance of the corresponding FRF to weight the squared error between measured and model FRFs, similarly as during the LTI identification. This regularized approach aims to find a compromise between model accuracy according to this weighted model error and model complexity with respect to the dependency on the scheduling parameter [6][7]. Depending on a so-called regularization parameter, more or less emphasis is put on model complexity. In this specific case, the three spline knots at {0.2988, 0.3298, 0.3660} m could be removed, reducing the number of model parameters with 33% (270 to 180 parameters) while model accuracy measured according to the above mentioned weighted squared error criterion increase with 20%. This model is referred to as `lpvregnls`. Since this regularized approach has only recently been developed, this functionality has not been integrated in LCToolbox yet, and tailored code was used to obtain `lpvregnls`. The comparison of the frequency response of the two models with that of the measured FRFs can be seen in Fig. 3 and Fig. 4. The full model is shown in Fig. 5.

Straightforward combination of the identified LTI model relating u to x_0 and LPV model `lpvregnls` yields the total 1-input, 2-output LPV crane model denote as `lpvcrane`. This model is added to the `crn` system using following command:

```
>> crn.add(lpvcrane);
```

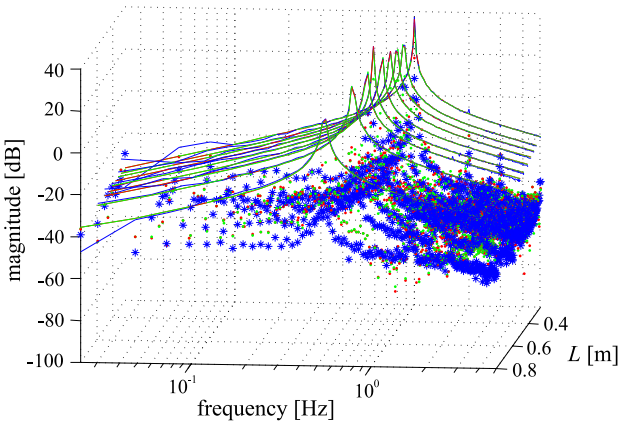

Fig. 3. Magnitude of the FRF of `lpvsmile` (green) and of `lpvregnls` (red), evaluated for the values of `lgrid` and compared to the estimated FRFs (blue). Solid lines represent the magnitude of the corresponding FRFs, while the dots denote the values of the model error compared to the standard deviation of the estimated FRF data

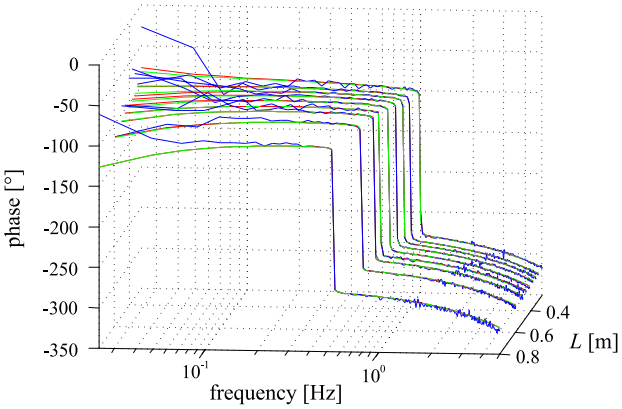

Fig. 4. Phase of the FRF of `lpvsmile` (green) and of `lpvregnls` (red), evaluated for the values of `lgrid` and compared to the estimated FRFs (blue)

5. Controller Design

Since one of the goals of the presented toolbox is to facilitate the design of $\mathcal{H}_\infty/\mathcal{H}_2$ feedback controllers, the controller design module is another key module of the LCToolbox. The following paragraphs provide more details about how design specifications are formulated based on the control configuration and how the resulting optimal feedback controller design problem is being solved.

5.1 Design Specifications The control of an overhead crane boils down to a trade-off between accurate tracking of the cart's reference position and preventing the load from oscillating. The proposed controller design immediately reflects this observation. On the one hand, accurate position tracking with a bandwidth of 0.3 Hz and a maximum peak sensitivity of 4.5 dB is considered to be the first requirement of the closed loop. The second requirement is to provide a minimum amount of damping on the oscillation of the load. The remaining freedom is exploited to minimize the input's high-frequency content.

These specifications result in optimization problem Eq. (2), where $\mathcal{U}$, $\mathcal{S}$ and $\mathcal{T}$ denote the transfer functions $r \to u$, $r \to e$ and $r \to \theta$ respectively. The different weights are deduced from the list of specifications.

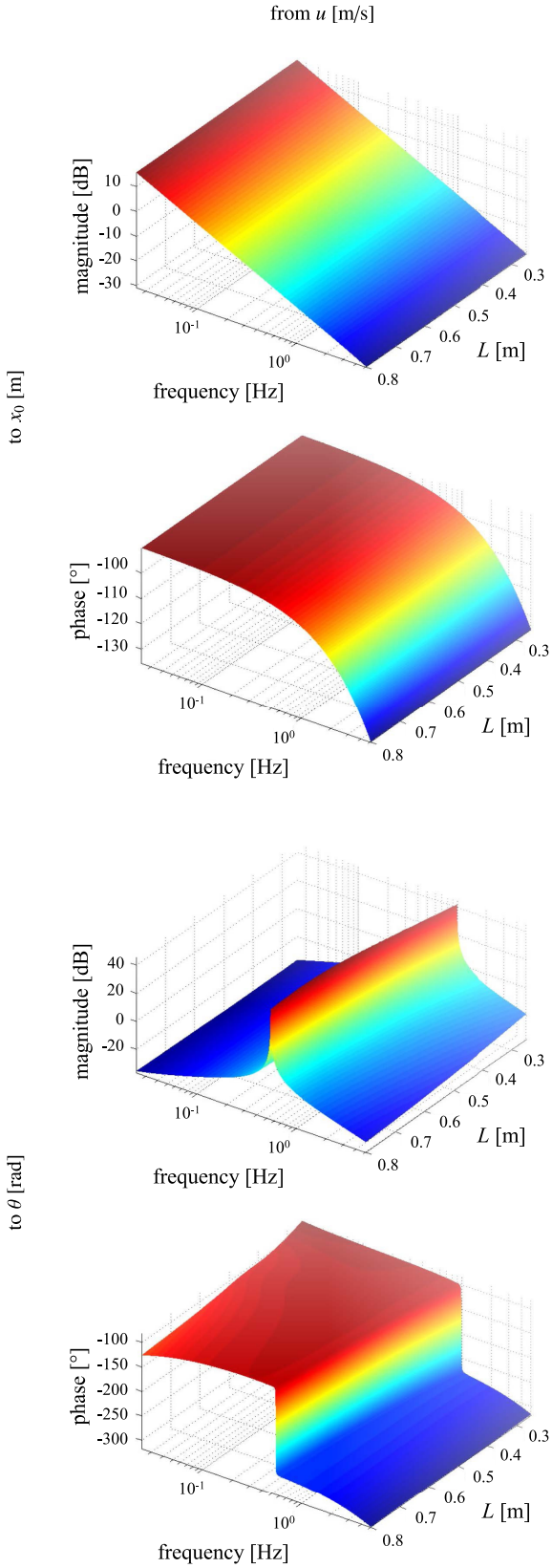

Fig. 5. Frequency responses of the regularized identified parametric model `lpvregnls` as a function of L

$$\begin{aligned} \text{minimize} \quad & \|\mathcal{W}_u \mathcal{U}\|_\infty \\ \text{subject to} \quad & \|\mathcal{W}_s \mathcal{S}\|_\infty \leqslant 1 \\ & \|\mathcal{M}_s \mathcal{S}\|_\infty \leqslant 1 \\ & \|\mathcal{W}_t \mathcal{T}\|_\infty \leqslant 1 \end{aligned} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

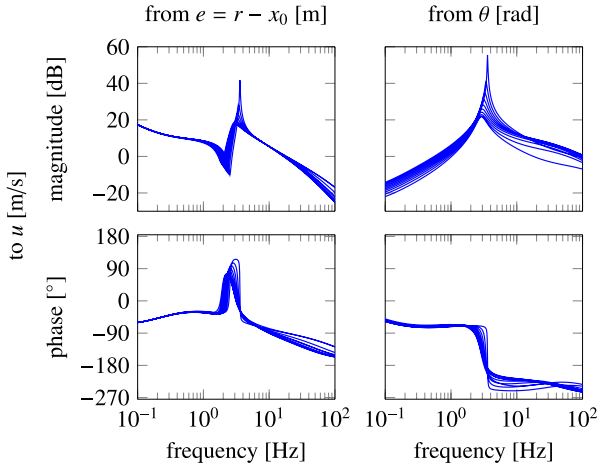

Fig. 6. LTI samples of the LPV solution `K` resulting from the design specifications for different values of L. This controller transforms the tracking error e of the cart and the measured angle θ into the velocity control signal u

Ideally, stating this optimization should be the only burden for a control engineer. In practice however, the control engineer is confronted with a considerable amount of pre- and postprocessing, mainly related to translating the problem to a suitable input format for the selected solver. To overcome these issues, LCToolbox provides a natural way of formulating these specifications which are internally converted to the format expected by the selected solver.

The first ingredient is the so-called `Channel` object. A channel is an abstract representation of the dynamics as seen from some input signal to some output signal. The channels that appear in optimization problem Eq. (2) can for instance be declared as

```
>> S = Channel(r,e,'Sensitivity')
>> U = Channel(r,u,'Input Sensitivity')
>> T = Channel(r,th,'Angle Sensitivity')
```

where the first and second argument are the input and output signals respectively and the last argument is an optional human readable name.

The second ingredient is the weight. Although weights can in principle be any transfer function, they usually boil down to standard low- or high-pass filters which are easily designed using LCToolbox. Because the inverse of a weight can, in a SISO setting, be interpreted as a frequency dependent constraint, LCToolbox employs an inverse definition. The simplest possible weight is a DC gain, which is used to limit the peak value of some SISO transfer function. Such a weight is being created by calling

```
>> Ms = Weight.DC(4.5)
>> Mt = Weight.DC(15)
```

where the values being interpreted in dB. In order to suppress low frequencies, `Weight.LF` is being used:

```
>> Ws = Weight.LF(0.3,2,-60)
```

This constructs a second order Butterworth filter with a DC-gain of 60 dB and a cut-off frequency so that the cross-over frequency equals 0.3 Hz. Its high-frequency counterpart is declared in a similar fashion:

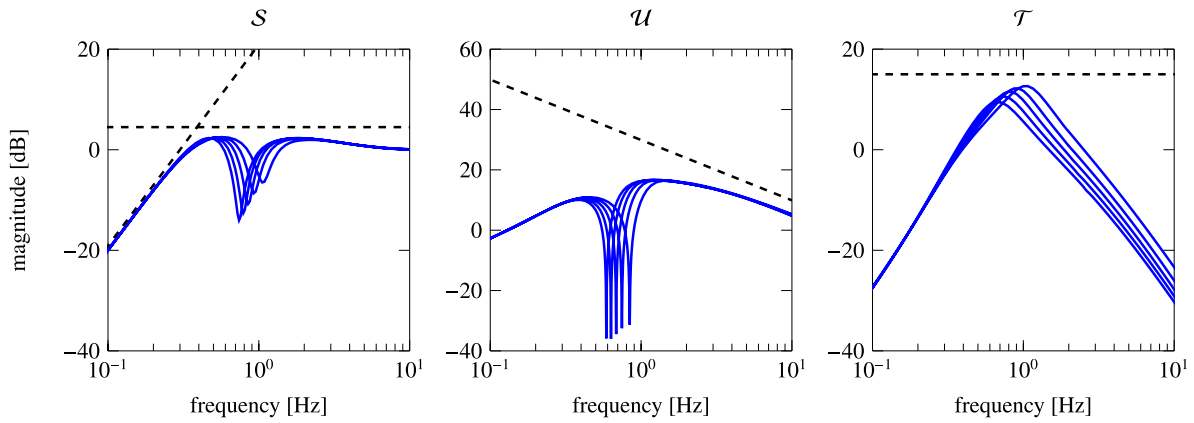

Fig. 7. Closed-loop transfer functions for the overhead crane

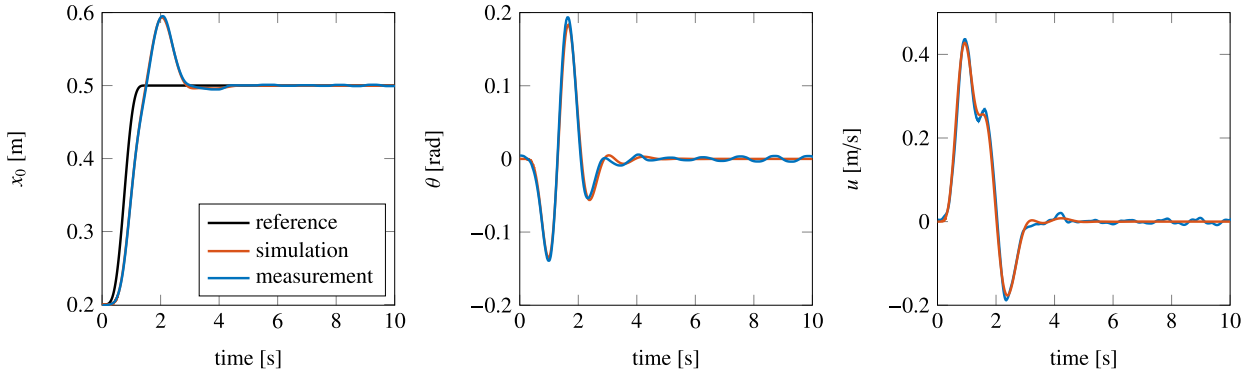

Fig. 8. Response of the overhead crane when tracking a smooth step reference. Both simulation and experimental results are displayed

```
>> Wu = Weight.HF(200,1,-20)
```

The resulting filter is a first order Butterworth filter with a cross-over frequency of 200 Hz and a high-frequency gain of 20 dB.

Having all weights and channels defined, the different specifications can be declared using the `Norm` object. This object represents the ∞- or 2-norm of a weighted transfer function and is readily created via

```
>> constr = Norm(W,C,p)
```

where `W` represents the weight that acts on channel `C`. The argument `p` denotes the norm type, taking `inf` and `2` as values. A shorthand notation for the ∞-norm is also provided, so that problem Eq. (2) is stated as follows:

```
>> obj = Wu*U
>> constr = [Ws*S < = 1; Ms*S < = 1; Mt*T < = 1]
```

By means of the `Channel`, `Weight` and `Norm` objects, the objective and constraints are thus readily defined by the control engineer.

5.2 Obtaining a Solution Many solvers to synthesize optimal feedback controllers exist. This wealth makes picking the right solver a complicated task that requires expert knowledge. Moreover, each solver expects different input arguments which causes additional difficulties to perform a closed-loop controller design. To overcome these issues, LCToolbox is able to automatically select an appropriate solver given a certain objective and constraints. Solvers that have been interfaced include i) `mixedHinfsyn`, a highly optimized implementation of (9) towards SIMO problems, ii) `mixedFixedOrder`, a direct implementation of (10) to solve general $\mathcal{H}_\infty/\mathcal{H}_2$ problem, iii) a B-spline implementation of (11) to synthesize LPV controllers and iv) MATLAB's `systune` for reduced order controller design. Based on the design problem, the most specialized solver is chosen from the list of applicable solvers as it allows the highest degree of numerical optimization and is therefore expected to give the best result. Moreover, LCToolbox converts the objective and constraints to the format that is expected by the selected solver so that a solution is easily obtained.

This process is packed within the command `solve`, which is called on the closed-loop system:

```
>> [CL,Klpv,info] = CL.solve(obj,constr,K)
```

Except for the objective `obj` and list of constraints `constr`, the argument `K` is provided to indicate the free variable of the optimization process. Once a feasible controller has been found, the control law `Klpv` is added to the corresponding system `K`. `Klpv` is depicted in Fig. 6.

The additional structure `info` can be consulted for more details on the optimization process. A powerful command that applies to `info` is `bodemag`. This allows the user to display the closed-loop transfer functions along the corresponding constraints, as is depicted by Fig. 7. In this case, multiple closed-loop responses are shown, each of them corresponding to the evaluation of the closed loop for a particular parameter value.

5.3 Time-domain Validation In order to support the

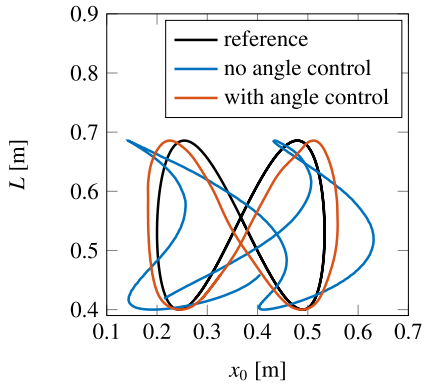

Fig. 9. Response of the overhead crane when tracking an "infinity" trajectory in 2D. Two responses are shown, one without and one with feedback of the angle

control engineer throughout the whole design process, the LCToolbox also provides an interface to perform simulations. Simulations can be carried out on models as well as systems. The latter has two advantages. First, it is possible to directly simulate the closed-loop behavior without the need to manually compute the closed-loop models. Second, it allows the user to readily compare the predictions of different models for the same system.

Apart from standard MATLAB commands such as `step` or `impulse`, LCToolbox also provides the command `lctsim` which allows the user to perform a general simulation.

```
>> rt = @(t) polytraj(t)
>> lt = @(t) 0.5
>> [y,t] = lctsim(CL([x0;th],r),rt,lt,20)
```

This command requires a function handle that represents a time-varying input and in case a `SchedulingParameter` is present, an additional time-varying function to describe the parameter trajectory. The optional last argument sets the simulation time. Note that the cable length remains fixed throughout the motion. The predicted response is depicted in Fig. 8. The cart executes a swift movement to track the reference and dampens the oscillation on θ upon its arrival. The same experiment was carried out on the actual overhead crane and the measured response is shown in Fig. 8 along the simulations. The simulations and experimental data show great resemblance, both qualitatively and quantitatively. This once again proves the validity of the identified LPV model and the practical value of the LPV controller design.

In order to show the true capabilities of the LPV controller, a second experiment was carried out. During this experiment, the cable length varies sinusoidally while the cart moves back and forth tracking a sinusoidal reference. The resulting 2D "infinity" trajectory described by the load x (1) with and without feedback of the angle is depicted in Fig. 9. Control with angle feedback is realized with `Klpv`, control without angle feedback is realized by removing the second channel of `Klpv` (see Fig. 6). The experimental data show how the controller with feedback of the angle succeeds in tracking the reference and at the same time keep the load from swinging excessively which does happen when the angle feedback is disabled.

6. Conclusion

This paper presented LCToolbox, a linear control toolbox for MATLAB. Its main purpose is to serve as an integrated toolchain for linear controller design while shielding the end user as much as possible from tedious pre- and postprocessing steps. It therefore has an intuitive modeling layer to represent and inspect measurements, to construct and inspect models and systems, to define control configurations, and to specify a control problem. System identification methods for linear time-invariant systems and linear parameter-varying systems, controller design methods based on $\mathcal{H}_\infty$ optimization for LTI and LPV models, and time domain simulation tools are interfaced. The typical workflow of the software package has been demonstrated by a study of a lab scale overhead crane setup.

Acknowledgment

This work has been carried out within the framework of Flanders Make SBO ROCSIS project (Robust and Optimal Control of Systems of Interacting Subsystems), FWO G.0915.14 project of the Research Foundation - Flanders (FWO - Flanders), and KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC).

References

(1) G. Hilhorst, E. Lambrechts, and G. Pipeleers: "Control of linear parameter-varying systems using B-splines", In 2016 IEEE 55th Conference on Decision and Control (CDC), pp.3246–3251 (2016)
(2) M. Verbandt, L. Jacobs, T. Singh, D. Turk, J. Swevers, and G. Pipeleers: "LC Toolbox - An open-source linear control toolbox for MATLAB", https://github.com/meco-group/lc_toolbox
(3) R. Pintelon and J. Schoukens. "System identification: a frequency domain approach, Second Edition", John Wiley & Sons (2012)
(4) R. Pintelon and J. Schoukens: http://wiley.mpstechnologies.com/wiley/BOBContent/downloadBobProjectFile.do?bpfId=1029&bpfFileType=.zip&bpfFileName=FreqDomBox.zip, 2012. Accessed April 3, 2019
(5) J.D. Caigny, J.F. Camino, and J. Swevers: "Interpolation-based modeling of MIMO LPV systems", *IEEE Trans. Contr. Sys. Techn.*, Vol.19, No.1, pp.46–63 (2011)
(6) D. Turk: "Regularized Identification of Linear Parameter-Varying Systems: Methods and Mechatronic Applications", Department of Mechanical Engineering, KU Leuven, Belgium, PhD thesis (2018)
(7) D. Turk and J. Swevers: "Identification of linear parameter-varying systems using B-splines", In 17th European Control Conference (ECC) (2019)
(8) C. de Boor: "A practical guide to splines, revised", Springer (2001)
(9) P. Gahinet and P. Apkarian: "A linear matrix inequality approach to H_∞ control", *International Journal of Robust and Nonlinear Control*, Vol.4, No.4, pp.421–448 (1994)
(10) C. Scherer, P. Gahinet, and M. Chilali: "Multiobjective output-feedback control via LMI optimization", *IEEE Transactions on Automatic Control*, Vol.42, No.7, pp.896–911 (1997)
(11) P. Apkarian and R.J. Adams: "Advanced gain-scheduling techniques for uncertain systems", *IEEE Transactions on Control Systems Technology*, Vol.6, No.1, pp.21–32 (1998)

Jan Swevers (Non-member) received the M.Sc. degree in electrical engineering and the Ph.D. degree in mechanical engineering from KU Leuven, Belgium, in 1986 and 1992, respectively. He is full professor at the Department of Mechanical Engineering of KU Leuven, member of the DMMS core lab of Flanders Make, and coordinating the research team MECO: Motion Estimation, Control and Optimization. His research focuses on motion control and optimization: robust and iterative learning control design methodologies for (non-)linear multi-variable systems, identification and control of robot manipulators, dynamic and embedded optimization for motion control systems.

Laurens Jacobs (Non-member) received the B.Sc. (2014) and M.Sc. (2016) degrees in mechanical engineering, both from KU Leuven. He is a member of the Motion Estimation, Control and Optimization (MECO) research team, Department of Mechanical Engineering, KU Leuven, where he is currently pursuing the Ph.D. degree. His research interests include linear control techniques, with a main focus on facilitating the practical applicability of $\mathcal{H}_\infty/\mathcal{H}_2$ methods for mechatronic control problems.

Taranjitsingh Singh (Non-member) received his B.E. degree in mechatronics engineering from Gujarat Technological University, India. In 2016, he received his M.Sc. degree in mechatronics from Technical University of Hamburg-Harburg, Germany. He is currently working towards his Ph.D. at the Department of Mechanical Engineering, KU Leuven, where he is a member of the Motion Estimation, Control and Optimization (MECO) research team. He is interested in linear feedback control, with a focus on simultaneous sensor and actuator selection and $\mathcal{H}_\infty$ control design and its applications.

Dora Turk (Non-member) was born in 1988, in Croatia. She received the B.Sc. and M.Sc. degree in electrical engineering and information technology from the Faculty of Electrical Engineering and Computing, University of Zagreb. In 2018, she received the Ph.D. degree in mechanical engineering from the Faculty of Engineering Science, KU Leuven, and is currently working as a Test and Characterisation Engineer at Xenics - Infrared Solutions.

Maarten Verbandt (Non-member) received his M.Sc. degree in 2014 and his Ph.D. degree in 2019, both at the KU Leuven Department of Mechanical Engineering. He is interested in optimal motion control of mechatronic systems, mainly focussing on optimal feedback controller design.

Goele Pipeleers (Non-member) is an assistant professor at the KU Leuven Department of Mechanical Engineering. She received her M.Sc. degree in mechanical engineering and her Ph.D. degree in mechanical engineering from the KU Leuven, in 2004 and 2009, respectively. She has been a Post-doctoral Fellow of the Research Foundation – Flanders, and a visiting scholar at the Colorado School of Mines and at the University of California - Los Angeles. Her research interests include convex optimization, optimal and robust control, and their applications in mechatronics.