Parameter Estimation based on FMI

Rüdiger Kampfmann Danny Mösch Nils Menager

Bosch Rexroth AG, Lohr am Main, Germany
{ruediger.kampfmann, fixed-term.danny.moesch, nils.menager}@boschrexroth.de

Abstract

In order to stay competitive the requirements on machinery in the producing industry have enormously increased. Within the automation industry these demands, like higher throughput or better energy efficiency, result in increasing complexity of the installed plants. Additionally, Industry 4.0 and the Internet of Things continuously increase the amount of software. Using model-based development methods is one approach to deal with this complexity. But model-based methods can also be utilized during the operational phase of a plant in order to generate additional value for the plant operator. Introducing smart services based on the usage of physical models enables new control and diagnosis features, e.g. the utilization of inverse plant models for feedforward control or comparing the output of a model with measurements of the plant in order to prove for correct behavior. For all these services the accuracy of the considered models is crucial. With an inexact model neither the future behavior can be foreseen nor the control quality can be improved. The used models don't have to be built up from scratch, existing models already created for sizing can be reused. However, these models cannot be used directly. First a reparametrization is necessary, because effects like friction or manufacturing tolerances cannot be taken into account correctly during sizing. For this special kind of problem dedicated optimization algorithms are available for parameter estimation, which take randomly distributed measurement errors and the special structure of this problem class into account.
In this paper a work flow for parameter estimation based on open source tools is presented, in which the considered models are provided as Functional Mock-up Unit. Afterwards the performance of this work flow is demonstrated on a real industrial problem: A three arm Delta Robot.

Keywords: Parameter Estimation, Levenberg-Marquardt Algorithm, FMI, Least Squares Optimization, Log-likelihood Method

1 Outline

The paper is structured as follows. First the considered optimization problem is derived from an approach based on probability theory. Afterwards suitable algorithms for this problem class are discussed with a special focus on the Levenberg-Marquardt algorithm, which is used in this contribution. Then the used software tools are presented: The Functional Mock-up Interface for the description of the dynamic systems and the software library Ceres for the solution of the underlying optimization problem. Afterwards the whole architecture of the used toolchain is presented. Finally this toolchain is applied to a real problem.

2 Mathematical Background

In this contribution it is assumed that the simulation model of a real plant is described in the following way:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \tag{1}$$

$$\mathbf{y}(t, \mathbf{p}) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{p}) \tag{2}$$

$$\mathbf{x}(t_{\text{start}}) = \mathbf{x}_0 \tag{3}$$

The dynamic system consists of a set of ordinary differential equations (1), a set of algebraic equations (2) and an initial condition (3). The time interval $T = [t_{\text{start}}, t_{\text{end}}]$ is considered. The functions $\mathbf{x} : T \to \mathbb{R}^{n_\mathbf{x}}$, $\mathbf{y} : T \times \mathbb{R}^{n_\mathbf{p}} \to \mathbb{R}^{n_\mathbf{y}}$, $\mathbf{u} : T \to \mathbb{R}^{n_\mathbf{u}}$ denote the states, the outputs and the inputs, respectively. The vector $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$ represents the parameters. Additionally,

$$\begin{aligned} &\mathbf{f} : T \times \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{p}} \to \mathbb{R}^{n_\mathbf{x}}, \\ &\mathbf{g} : T \times \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{p}} \to \mathbb{R}^{n_\mathbf{y}}, \\ &n_\mathbf{x}, n_\mathbf{u}, n_\mathbf{p}, n_\mathbf{y} \in \mathbb{N}. \end{aligned}$$

The input $\mathbf{u}$ from the real plant is assumed to be known exactly over the whole interval, whereas $\boldsymbol{\eta}_i$ denotes the measured output vector of the real plant at time t_i for $i = 1, \ldots, n_t$. The most obvious approach for parameter estimation is just to minimize some norm $\|\cdot\|_q$ with $q \in [1, \infty)$ or $q = \infty$ of the deviation between the measured and the simulated outputs by varying the parameters $\mathbf{p}$, i. e.

$$\underset{\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}}{\arg\min} \sum_{i=1}^{n_t} \|\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p})\|_q^q$$

or

$$\underset{\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}}{\arg\min} \sum_{i=1}^{n_t} \|\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p})\|_\infty,$$

respectively. That $q = 2$ is a reasonable choice is going to be the result of the following subsections. Therefore a little bit of probability theory has to be consulted.
Ensuing from (Krengel, 1988) the maximum-likelihood approach is introduced first. In the next subsection the idea is used to formulate the underlying optimization problem which is the foundation of the presented process of parameter estimation.

2.1 Maximum Likelihood Approach

Let $\mathbf{X} \in \mathbb{R}^n$ be a random vector with independent and identically distributed components and concrete realizations $\mathbf{x} \in \mathbb{R}^n$ of $\mathbf{X}$. Each component $\mathbf{X}_i$ has the density function $\nu(\cdot|\mathbf{p}) = \nu(\mathbf{x}_i|\mathbf{p})$, which depends on a parameter set $\mathbf{p}$. It describes the probability of $\mathbf{x}_i$ given the parameters $\mathbf{p}$. Since they are independent, the joint density can be written as

$$\nu(\mathbf{x}|\mathbf{p}) = \prod_{i=1}^{n} \nu(\mathbf{x}_i|\mathbf{p}).$$

To formulate the optimization problem the *likelihood function* $\mathrm{L}(\cdot|\mathbf{x}) = \mathrm{L}(\mathbf{p}|\mathbf{x})$ is defined as

$$\mathrm{L}(\mathbf{p}|\mathbf{x}) := \nu(\mathbf{x}|\mathbf{p}),$$

which is now a function of the parameters $\mathbf{p}$ given the data $\mathbf{x}$. $\mathrm{L}(\cdot|\mathbf{x})$ is not a proper probability density function, since its integral over all parameters is not necessarily equal to 1. Therefore, it also should not be considered a conditional probability density function, which might be supposed by the vertical bar.
Thus, it is obvious to choose the optimization problem

$$\underset{\mathbf{p}\in\mathbb{R}^{n_\mathbf{p}}}{\arg\max}\ \mathrm{L}(\mathbf{p}|\mathbf{x})$$

to get the *maximum likelihood estimator* $\mathbf{p}_{\mathrm{ML}}$ which makes the sample data $\mathbf{x}$ most likely. In some cases it will simplify the optimization process if the *Log-likelihood function* $\ln \mathrm{L}(\mathbf{p}|\mathbf{x})$ is chosen instead of $\mathrm{L}(\mathbf{p}|\mathbf{x})$ as will be seen later. In fact, it does not make a difference whether choosing $\mathrm{L}(\mathbf{p}|\mathbf{x})$ or $\ln \mathrm{L}(\mathbf{p}|\mathbf{x})$, since the logarithm is a monotonic function that does not influence the maximum.

2.2 The Underlying Optimization Problem

Each measured data vector $\boldsymbol{\eta}_i$ at time t_i can be expressed as the real output $\mathbf{y}(t_i, \mathbf{p}^*)$ with the exact but naturally unknown parameter set $\mathbf{p}^*$ plus a measurement error $\boldsymbol{\varepsilon}_i$, i. e.

$$\boldsymbol{\eta}_i = \mathbf{y}(t_i, \mathbf{p}^*) + \boldsymbol{\varepsilon}_i. \tag{4}$$

It is assumed that the measurement errors $\boldsymbol{\varepsilon}_i$ are statistically independent and underly a certain distribution. The most common assumption is to choose a normal distributed error vector $\boldsymbol{\varepsilon}_i$ with statistically independent components, known (diagonal) covariance matrix $\boldsymbol{\Sigma}_i$ and expectation $\mathrm{E}(\boldsymbol{\varepsilon}_i) = \mathbf{0}$, i. e.

$$\boldsymbol{\varepsilon}_i \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_i) \quad \text{with} \quad \boldsymbol{\Sigma}_i = \mathrm{diag}\left(\boldsymbol{\sigma}_{i,1}^2, \dots, \boldsymbol{\sigma}_{i,n}^2\right). \tag{5}$$

In an applied sense that means that the measurements $\boldsymbol{\eta}_i$ do not influence each other and the errors $\boldsymbol{\varepsilon}_i$ do not contain a systematic error.
With these requirements the density function

$$\nu(\boldsymbol{\varepsilon}_{i,j}) = \frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}_{i,j}} \exp\left(-\frac{\boldsymbol{\varepsilon}_{i,j}^2}{2\boldsymbol{\sigma}_{i,j}^2}\right)$$

for each measurement error $\boldsymbol{\varepsilon}_{i,j}$ is obtained. Since the $\boldsymbol{\varepsilon}_{i,j}$ are statistically independent for all j and for all i, too, it holds

$$\begin{aligned}\nu(\boldsymbol{\varepsilon}) &= \prod_{i=1}^{n_t} \nu(\boldsymbol{\varepsilon}_i)\\ &= \prod_{i=1}^{n_t}\prod_{j=1}^{n_\mathbf{y}} \nu(\boldsymbol{\varepsilon}_{i,j})\\ &= \prod_{i=1}^{n_t}\prod_{j=1}^{n_\mathbf{y}} \frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}_{i,j}} \exp\left(-\sum_{k=1}^{n_t}\sum_{l=1}^{n_\mathbf{y}} \frac{\boldsymbol{\varepsilon}_{k,l}^2}{2\boldsymbol{\sigma}_{k,l}^2}\right),\end{aligned} \tag{6}$$

where $\boldsymbol{\varepsilon} = \begin{pmatrix}\boldsymbol{\varepsilon}_1 & \dots & \boldsymbol{\varepsilon}_n\end{pmatrix}$. Because of (4) we also get $\boldsymbol{\eta}_i \sim \mathrm{N}(\mathbf{y}(t_i, \mathbf{p}^*), \boldsymbol{\Sigma}_i)$ and thus

$$\nu(\boldsymbol{\eta}_{i,j}) = \frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}_{i,j}} \exp\left(-\frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}^*))^2}{2\boldsymbol{\sigma}_{i,j}^2}\right).$$

For the same reason as in (6) and with $\boldsymbol{\eta} = \begin{pmatrix}\boldsymbol{\eta}_1 & \dots & \boldsymbol{\eta}_n\end{pmatrix}$ this leads to the conditional density function

$$\nu(\boldsymbol{\eta}|\mathbf{p}) = \prod_{i=1}^{n_t}\prod_{j=1}^{n_\mathbf{y}} \frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}_{i,j}} \exp\left(-\frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}))^2}{2\boldsymbol{\sigma}_{i,j}^2}\right).$$

The Log-likelihood function is then defined by

$$\begin{aligned}\ln \mathrm{L}(\mathbf{p}|\boldsymbol{\eta}) = &-\frac{n_t n_\mathbf{y}}{2}\ln 2\pi - \sum_{i=1}^{n_t}\sum_{j=1}^{n_\mathbf{y}} \ln \boldsymbol{\sigma}_{i,j}\\ &- \frac{1}{2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_\mathbf{y}} \frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}))^2}{\boldsymbol{\sigma}_{i,j}^2}.\end{aligned}$$

Since the first and the second term are constant they can be omitted from the optimization.
Finally, the whole constrained nonlinear optimization problem can be formulated:

$$\underset{\mathbf{p}\in\mathbb{R}^{n_\mathbf{p}}}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p}))\|_2^2 \tag{7a}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t), \mathbf{p}), \tag{7b}$$

$$\mathbf{y}(t, \mathbf{p}) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{p}) \tag{7c}$$

$$\mathbf{x}(t_{\text{start}}) = \mathbf{x}_0 \tag{7d}$$

observing the box constraints

$$\mathbf{p}_{\text{lower}} \leq \mathbf{p} \leq \mathbf{p}_{\text{upper}} \tag{7e}$$

given

$$\mathbf{u}(t_i) \quad \text{with} \quad t_i \in [t_{\text{start}}, t_{\text{end}}] \quad \text{for} \quad i = 1, \dots, n_t$$

and related measured outputs $\boldsymbol{\eta}_i$. The result is the parameter set $\mathbf{p}_{\mathrm{ML}}$ which is the most likely for the given measured output values. It should be noted again that it is important for the chosen approach to have measurement errors following (5). The method is not reasonable for different distributions, although others can be established.

3 Optimization Algorithms

Eliminating the ODE constraints (7b)-(7d) through numerical integration from (7a) yields a common nonlinear optimization problem. For this kind of problem a couple of different algorithms for the efficient solution exist. Some algorithms exploit derivative information and some do not. The derivative-free optimization algorithms have the advantage that they obviously do not require derivatives with respect to the varied optimization variables, which may be costly to compute. Another advantage is that under certain circumstances global convergence is achieved, neglecting limited computational time and rounding errors. In (Gedda et al., 2012) a tool chain for parameter estimation with FMI and derivative-free methods already has been presented. Nevertheless, derivative-free optimization algorithms show very poor convergence speed.
However, optimization algorithms, which use derivatives of the objective function, have also distinct advantages. The main benefit is the fast convergence rate. These methods compute iteratively starting from an initial guess a descent direction and thus reduce the objective function in every step until a certain stop criterion is reached. The better the initial guess the faster the convergence speed. The disadvantages of these methods are on the one hand that the problem has to be sufficiently smooth and that derivatives have to be computed and on the other hand that these methods may get stuck in a local minimum, if the initial guess is too bad. The last disadvantage can be overcome with multi start-ups, i.e. run several optimization from different initial guesses (see (Raue et al., 2013) for more information). For the purposes of this contribution, whereas existing models from the sizing should be reused, good initial guesses are known, because rough parameter sets are already needed for correct dimensioning. Thus sticking in local minimum is not a problem at all. The considered models are also sufficiently smooth with respect to the parameters. Also in (Raue et al., 2013) a benchmark of different algorithms was conducted to an estimation problem from systems biology, demonstrating the slow convergence speed of derivative-free optimization algorithms compared to the ones which rely on derivatives. Since good initial guesses are known and the fast convergence speed the demonstrated toolchain is based on derivative based optimization algorithms.

3.1 Levenberg-Marquardt Algorithm

The investigated optimization problem (7a) has a special structure. It is a nonlinear least squares problem. This kind of problem is widely spread in scientific and engineering areas. Thus structure exploiting optimization algorithms have been developed, which solve this problems efficiently. The Levenberg-Marquardt algorithm is one of these algorithms and is used within this contribution. It can be seen as a conjunction of the Gauss-Newton method together with the idea of Trust-Region approaches.

To give a short overview (Björck, 1996), some abbreviations are introduced first:

$$\mathbf{h}_i(\mathbf{p}) := \mathbf{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i,\mathbf{p}))$$
$$H(\mathbf{p}) := \sum_{i=1}^{n_t} \|\mathbf{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i,\mathbf{p})\|_2^2$$
$$\mathbf{y}_i'(\mathbf{p}) := \frac{\partial \mathbf{y}(t_i,\mathbf{p})}{\partial \mathbf{p}}$$

Therein denotes $\mathbf{y}_i'$ the Jacobian matrix of $\mathbf{y}$ at time t_i with respect to $\mathbf{p}$. The Gauss-Newton method solves the linearized least squares problem

$$\underset{\Delta\mathbf{p}\in\mathbb{R}^{n_\mathbf{p}}}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{n_t}\|\mathbf{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}_i'(\mathbf{p}_k)\Delta\mathbf{p}_k)\|_2^2$$

in each iteration step k to get a new approximation

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \Delta\mathbf{p}_k$$

of the exact parameter set $\mathbf{p}^*$.
It is the idea of the Levenberg-Marquardt algorithm to add a regularization term to the linearized objective function,

$$\underset{\Delta\mathbf{p}\in\mathbb{R}^{n_\mathbf{p}}}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{n_t}\|\mathbf{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}_i'(\mathbf{p}_k)\Delta\mathbf{p}_k)\|_2^2 + \frac{\lambda_k}{2}\|\Delta\mathbf{p}_k\|_2^2.$$

With the regularization the problem has always a solution even with rank deficient Jacobians $\mathbf{y}_i'$. The parameter λ_k also controls the step length $\|\Delta\mathbf{p}_k\|_2$ as well as the direction $\Delta\mathbf{p}_k$. It can be observed (Marquardt, 1963) that for a large λ_k the direction is almost a gradient step with only small step size, whereas a small λ_k leads to a direction close to a Gauss-Newton step. Therefore it is reasonable to choose a small λ_k near the actual minimum where the linearized problem is a rather good approximation. Hence, choosing λ_k is a significant task in each iteration to reach a preferably fast convergence rate. There are different ways to update the parameter λ_k. A central role plays the ratio between actual reduction and (by the linearized problem) predicted reduction

$$\psi_k(\Delta\mathbf{p}_k) = \frac{H(\mathbf{p}_k) - H(\mathbf{p}_k + \Delta\mathbf{p}_k)}{H(\mathbf{p}_k) - \sum_{i=1}^{n_t}\|\mathbf{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}_i'(\mathbf{p}_k)\Delta\mathbf{p}_k)\|_2^2},$$

whose value decides whether $\mathbf{p}_k$ will be updated or not and how λ_k will be changed.

4 Functional Mock-up Interface

The Functional Mock-up Interface (FMI) is a tool independent standard to support model exchange between different simulation environments (Blochwitz et al., 2011). A model which is shared via FMI is referred to as Functional Mock-up Unit (FMU). An FMU consists of a XML-File, describing the whole model variables and parameters, and a compiled library containing the model equations and additionally required functions, i.e. functions for initialization or data exchange. The models are described as hybrid

ODEs supporting state and time events. The FMI standard supports two modes to share these models. On the one hand there is Model Exchange, whose models are described in a form similar to the equations 1 and 2, and on the other hand the Co-Simulation mode, which delivers the FMU additionally with its own integrated ODE solver. In this contribution the Co-Simulation mode is used due to the fact that no extra ODE solver is required for simulating an FMU. The models considered in the estimation procedure are built up in a simulation environment and exported as FMU. In 7.2 an example is described. For the optimization derivatives with respect to the parameters are required. With the actual standard 2.0 of the FMI, only derivatives with respect to inputs and states are supported (Blochwitz et al., 2012). Hence finite differences are used. As far as the authors know, parameter sensitivities are currently considered by the FMI steering committee.

5 Ceres Solver

For the solution of the nonlinear least squares problem (7a), (7e) the Ceres Solver (Agarwal et al.) is utilized. The Ceres Solver is an open source C++ library for solving large optimization problems. Beneath general unconstrained optimization problems it was developed to solve nonlinear least squares problems with bound constraints. There are several reasons why the Ceres Solver was chosen. The solver is published under the New BSD license, so there are almost no license restrictions for commercial use. In addtion to the Levenberg-Marquardt Algorithm 3.1 this software library is equipped with other state of the art algorithms and has reached a certain maturity since it is used in commercial applications for more than four years and still has an active community.
Furthermore the library is developed in C++ and has already been migrated to Android and iOS. Hence an migration to Bosch Rexroth embedded systems should be possible with little effort. Ceres can compute the required derivatives of the objective function by finite differences or the user can provide them. Because the actual FMI version is not supporting parameter derivatives, they are computed by Ceres via finite differences. With upcoming features of the next FMI version, this can be easily adapted. Ceres is one of the few libraries which is also capable to derive covariance estimations for the solution. Hence, confidence intervals for the computed parameters can be computed directly.
Within Ceres a problem class needs to be implemented which corresponds to the desired residual function (7a). The model to be investigated and the measured data have to be provided therefor. An additional class method handles possible solver options and is responsible for the actual optimization.

6 Structure of the Tool Chain

Figure 1 shows the structure of the parameter estimation toolchain. The stimulation of the real plant and the real

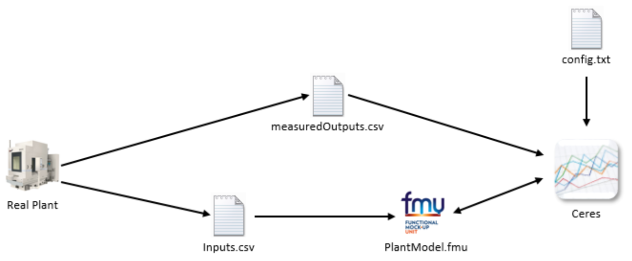

Figure 1. Structure of parameter estimation toolchain

measured outputs have to be provided as CSV-file. The whole estimating procedure is configured by a configuration file. In this file the desired model, the parameters to estimate and the paths of the CSV-files are denoted. Also an initial guess and the variances for each measurement noise have to be stated. Additionally, upper and lower bounds for the parameters can be specified. Within Ceres a problem class was defined which takes the configuration file and manages the whole parameter estimation procedure. This problem class directly interfaces the FMU. No additional library for calling the FMU functions is used. An evaluation of the residual function implies a simulation of the FMU. In every step the inputs are written to the FMU. Thereafter, the residual function is built up by comparing the measured outputs with the outputs of the FMU. Hence the whole simulation is triggered by Ceres. The derivatives of the residual function with respect to the parameters are computed directly by Ceres via finite differences which corresponds to multiple simulation runs. Ceres then conducts the chosen optimization algorithm by varying the parameters. Subsequently, an a posteriori evaluation of the covariance matrix of the estimated parameters can be conducted. Out of this matrix confidence intervals for each parameters can be derived directly. For the import of the FMU into Ceres an own light weight framework was implemented.

7 Application of the Tool Chain

In this section the capability of the toolchain is demonstrated on a Delta Robot. This type of robot was developed in the 1980s (Clavel, 1988) and is widely used for pick and place applications. It is built up out of parallel bars and has 3 degrees of freedom for translational manipulation and one for manipulating the orientation. Hence it has to be driven by 4 motors. Since the robot should move as fast as possible, knowing the exact dynamic behavior is advantageous, i. e. an accurate model can be exploited for feedforward control in order to enhance the dynamic behavior. The dynamic of the robot is mainly influenced by frictional effects and mass parameters. The mass parameters underly a certain manufacturing tolerance and the friction is hard to be known beforehand. Therefore, estimating these parameters is a good use case for the toolchain.

7.1 Real Set-up of the Robot

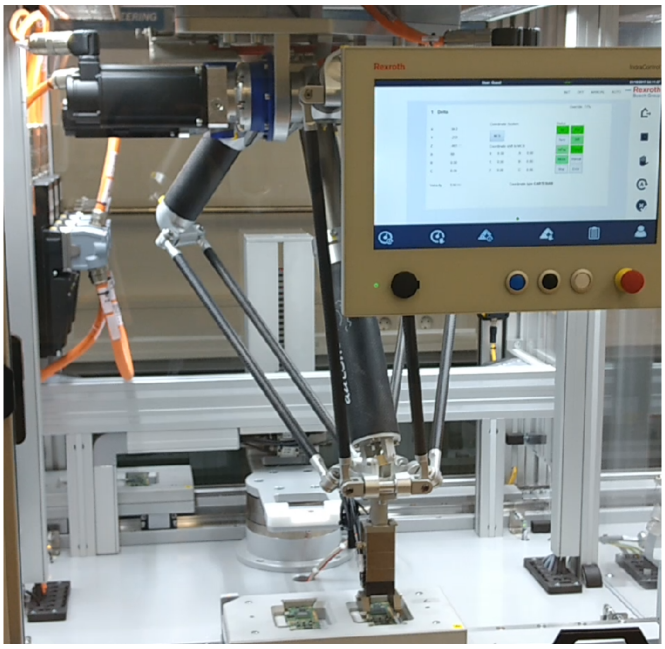

Figure 2. Real delta robot

Figure 2 shows the investigated robot. The kinematic is manufactured by Autonox24 and is driven by 4 Rexroth synchronous motors. Three MSK040B-0600 for the translational movement and one MS2N03-B0BYN for the orientation axis are used. The movement of the robot is controlled by a Rexroth IndraControl VPB 40.3 industrial PC. No special trajectories were considered. The robot just executes a usual pick and place cycle and the motor torque is measured via actual motor current. Through the recorded motor torques the parameters of the robot should be identified. Since the dynamics of the orientation axis is well known, no measurements for this axis have been taken into account.

7.2 Delta Robot Model

The physical model of the robot was built up in the modeling language Modelica using Dymola. The mechanical model consists of Modelica Standard Library (MSL) components. Mainly joints and body components from the multi body library are used.

7.3 Real Set-up of the Robot

Figure 3 shows the animation of the MSL components within Dymola. All parallel bars were considered. No simplifications of the mechanical structure were made. Additionally the drive train of each axis was modeled in the way that motor and gear inertia, gear efficiency and Coloumb and viscous friction are considered. Therefore own Modelica components were added to standard rotational mechanics components. As input of the model the position, velocity and acceleration of each axis were used. The resulting motor torques were declared as output. It is assumed that the inertia and friction properties of all 3

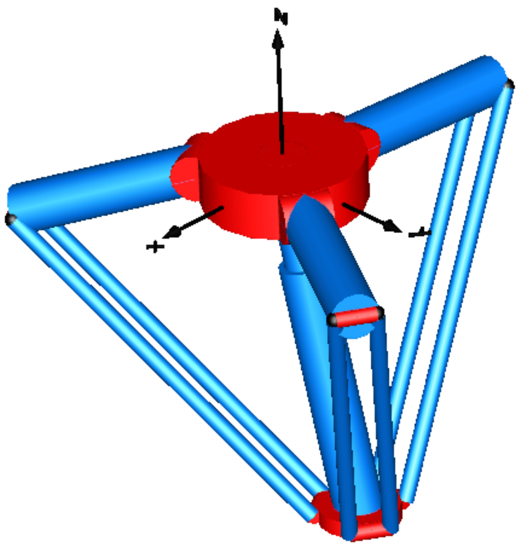

Figure 3. Animation of the multi body model

considered axes are equal. Hence, the following parameters should be identified:

- Mass of lower arm
- Mass of upper arm
- Mass of base plate
- Motor and gear inertia
- Gear efficiency
- Parameter for Coloumb friction
- Parameter for viscous friction

The model was exported as an FMU 2.0 for Co-Simulation containing the CVODE solver (Hindmarsh et al., 2005).

7.4 Estimation of Parameters

For the measurement the robot moves the usual pick and place cycle at four different speeds. 6250 time points were considered. The complete cycle lasts 62.278 seconds. For each of the three axes the position, velocity, acceleration and torque were recorded. For the identification procedure Ceres compares the measured motor torque with the one resulting from the FMU.
Table 1 shows the results of the parameter estimation procedure. The residual of the objective function (7a) was reduced significantly from 2.12×10^6 to 1.25×10^6. The computed parameter sets especially the friction and gear efficiency parameter seem reasonable. Also the computed confidence intervals, i. e. the intervals in which the real parameters are located with a probability of $\beta = 0.95$, imply that the computed estimations are reliable. Unfortunately it is not possible to validate the estimation results by scaling the components, because the robot cannot be disassambled.

Parameter	Unit	Initial Value	Estimated Value	Confidence Interval $\beta = 0.95$
Mass of lower arm	[kg]	0.1	0.08	± 0.00350
Mass of upper arm	[kg]	1.5	1.74	± 0.0211
Mass of base plate	[kg]	0.87	1.1	± 0.0212
Motor and gear inertia	[$\mathrm{kg\,m^2}$]	0.000144	0.000155	$\pm 2.65 \times 10^{-6}$
Gear Efficiency	[1]	1.0	0.907	± 0.0136
Coloumb Friction	[N m]	0.12	0.105	± 0.00158
Viscous Friction	[$\mathrm{N\,m\,s\,rad^{-1}}$]	0.001	0.00128	$\pm 1.92 \times 10^{-5}$

Table 1. Parameter estimation results

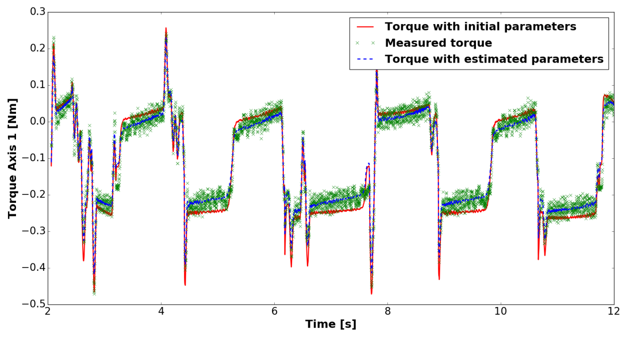

Figure 4. Results for arm 1

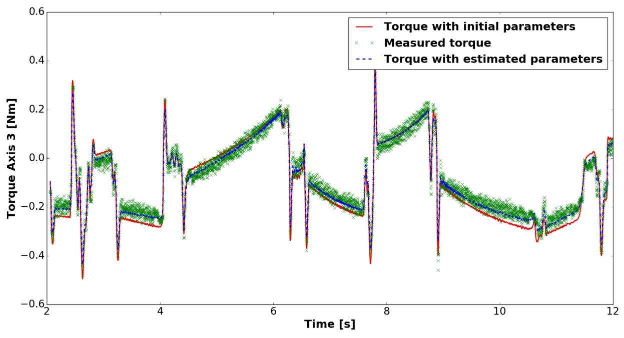

Figure 6. Results for arm 3

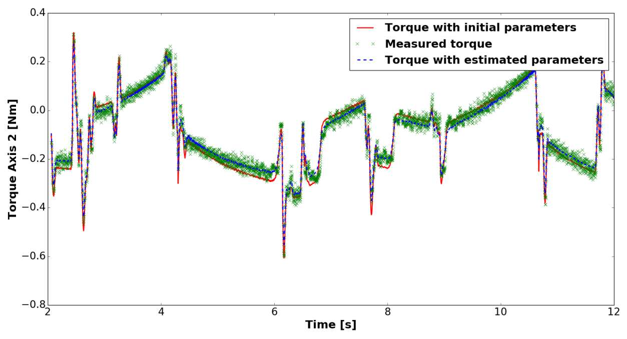

Figure 5. Results for arm 2

Figure 4 to 6 show a section of the results for each axes. With the estimated parameters the accuracy of the model has been improved significantly.

8 Summary and Outlook

A tool chain for parameter estimation with a state of the art Open Source software library and the Functional Mockup has been presented. The capabilities of this tool chain were demonstrated on a real industrial robot. The results are very promising such that this approach should be pursued. On the one hand a migration of the whole tool chain to embedded systems seems meaningful. For example the estimation procedure can be used for auto calibration of feedforward controllers using inverse models.
On the other hand with Industry 4.0 and the Internet of Things new use cases occur. The intelligent plants equipped with sensors record all their data. With these measurements and the presented tool chain parameter estimations could be conducted automatically. Upon these well known parameters and accurate models new smart services for diagnosis or control purposes can be enabled.

References

Sameer Agarwal, Keir Mierle, et al. Ceres solver. `http://ceres-solver.org`.

Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, H Elmqvist, A Junghanns, J Mauß, M Monteiro, T Neidhold, D Neumerkel, et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 063, pages 105–114. Linköping University Electronic Press, 2011.

Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

Reymond Clavel. A fast robot with parallel geometry. In *Proc. Int. Symposium on Industrial Robots*, pages 91–100, 1988.

Sofia Gedda, Christian Andersson, Johan Åkesson, and Stefan Diehl. Derivative-free parameter optimization of functional mock-up units. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 819–828. Linköping University Electronic Press, 2012.

Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

Ulrich Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*, volume 8. Springer, 1988.

Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, U. Klingmüller, and J. Timmer. Lessons learned from quantitative dynamical modeling in systems biology. *PLoS ONE*, 8(9):e74335, Sept. 2013. doi:10.1371/journal.pone.0074335.