Applied and Computational Mechanics 5 (2011) 231–238

Adaptive calibration method with on-line growing complexity

T. Skopec[a,*], Z. Šika[a]

[a]*Czech Technical University in Prague, Faculty of Mechanical Engineering, Department of Mechanics, Biomechanics and Mechatronics, Prague*

Received 8 June 2011; received in revised form 14 November 2011

Abstract

This paper describes a modified variant of a kinematical calibration algorithm. In the beginning, a brief review of the calibration algorithm and its simple modification are described. As the described calibration modification uses some ideas used by the Lolimot algorithm, the algorithm is described and explained. Main topic of this paper is a description of a synthesis of the Lolimot-based calibration that leads to an adaptive algorithm with an on-line growing complexity. The paper contains a comparison of simple examples results and a discussion. A note about future research topics is also included.

Keywords: redundant calibration, adaptive calibration algorithm, dynamically driven calibration complexity, Lolimot, growing calibration complexity

1. Introduction

Even despite of the very accurate manufacture of machines it is not possible (especially in case of the parallel kinematics) to use the design dimensions for the nonlinear kinematical transformations in the machine control system [5].

It is necessary to determine the actually manufactured dimensions as accurately as possible in order to improve the accuracy of the calibrated parameter values in the control algorithms and consequently to improve the final accuracy of the machine operations [3].

This paper describes improvement of the calibration algorithm. It introduces new an approach to the calibration when a core of the Lolimot algorithm in integrated into the calibration process. The approach should be efficient when applied to the redundant parallel mechanism calibration and its implementation for these mechanisms is still under development.

2. Current calibration algorithm

2.1. Description of redundant mechanisms calibration

During redundant calibration, the number of measured mechanism variables is higher than the number of mechanism's DOF. Moreover, no external device is used.

The basis of a calibration problem formulation [1,7] is a kinematical transformation between measured coordinates $\mathbf{s}$ in kinematical pairs, dimensions of the mechanism $\mathbf{d}$ and possibly the position of the machine spindle axis (generally end-effector position) $\mathbf{v}$

$$\mathbf{f}(\mathbf{d}, \mathbf{s}, \mathbf{v}) = \mathbf{0}. \tag{1}$$

[*]Corresponding author. Tel.: +420 224 357 231, e-mail: Tomas.Skopec@fs.cvut.cz.

The classical calibration algorithm uses Newton's method modified for the over-constrained system of nonlinear algebraic equations

$$\mathbf{J_d}\delta\mathbf{d} = -\mathbf{J_s}\delta\mathbf{s} - \mathbf{J_v}\delta\mathbf{v} - \mathbf{f}(\mathbf{d},\mathbf{s},\mathbf{v}) = \delta\mathbf{r}. \tag{2}$$

Within the *i-th* iteration of the Newton's method the following dimension corrections $\delta\mathbf{d}$ for all machine positions are computed

$$\delta\mathbf{d}_i = (\mathbf{J}^T_{\mathbf{d}_i}\mathbf{J}_{\mathbf{d}_i})^{-1}\mathbf{J}^T_{\mathbf{d}_i}\delta\mathbf{r}_i\,, \tag{3}$$

where $\mathbf{J}_{\mathbf{d}_i}$ is the Jacobi matrix of partial derivatives of kinematical transformation with respect to the calibrated dimensions $\mathbf{d}$ and $\delta\mathbf{r}_i$ is the vector of deviations computed from measured quantities and calibrated quantities $\mathbf{d}_i$ from the previous step. The new values of dimensions are then computed as

$$\mathbf{d}_{i+1} = \mathbf{d}_i + \delta\mathbf{d}_i\,. \tag{4}$$

Based on results, new values of $\delta\mathbf{d}_{i+1}$ and $\mathbf{J}_{\mathbf{d}_{i+1}}$ are computed. The iteration process continues until the deviations are being decreased.

2.2. *Complexity vs. accuracy of calibration models*

As described in [4], kinematical model that is used for the calibration of redundant mechanisms might gradually become more and more complex.

As the complexity of the model rises, calibration's conditionality usually degrades. On the other hand, better sum of all discrepancies for constraint equations might be obtained. There raises a question when to stop this gradual addition of complexity to the kinematical model.

Therefore we want to develop some kind of a calibration algorithm that artificially increases the complexity of the used kinematical model during calibration execution. We must also add some conditions to decide when the algorithm should stop with enhancing of model's complexity. The described algorithm uses ideas from the online identification to realize growing complexity in the calibrated model.

3. Lolimot algorithm

In order to model (and calibrate) nonlinear parts of the calibrated mechanism, online identification is used. Identification of nonlinear dynamic parts is a challenging task, because modeled processes are unique, therefore capability of describing a wide class of structurally different systems is required [6].

Interesting approach to the identification problem is usage of artificial neural networks.

Alternatively we can use fuzzy logic and fuzzy rules based systems, which contain some elements from the human reasoning. The fuzzy relations are fluid and approximate rather than fixed and exact.

Combination of these two approaches leads to the neuro-fuzzy algorithms and models. They try to combine advantages of the neural networks (flexibility and learning) with advantages of the fuzzy systems (fluid approximations, interpretability).

At the end of last century, O. Nelles developed new nonlinear system identification scheme that tries to avoid various limitations of the currently used neuro and fuzzy approaches [2]. The Local Linear Model Tree (called Lolimot) algorithm described in his thesis offers an important advantage; a very efficient structure optimization that automatically adapts the model in order to handle nonlinearity. It also contains a section scheme which allows us to optimize dynamic model properties. The algorithm is particularly well suited for identification of nonlinear dynamic processes and systems.

3.1. Lolimot basic idea

Each neuron in the network structure contains a local linear model (LLM) and has associated a validity function Φ that determines a region of the validity of the LLM.

The outputs $\hat{y}_i$ of the LLMs for inputs u are

$$\hat{y}_i = w_{io} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p \,, \tag{5}$$

where w_{ij} denote LLM parameters for the neuron i for the input $j$and u is input.

The validity functions of the local linear models are normalized such that

$$\sum_{i=1}^{M} \Phi_i(\underline{u}) = 1 \,. \tag{6}$$

This property is necessary and required, because it ensures that contributions of all local linear models sums up to 100 %.

The final network output is then calculated as a weighted sum of the outputs $\hat{y}_i$ of all local linear models. The algorithm interpolates between different LLMs with the validity functions as follows

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \Phi_i(\underline{u}) = \sum_{i=1}^{M} (w_{io} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p)\Phi_i(\underline{u}) \,. \tag{7}$$

Used validity functions have been chosen as axis-orthogonal normalized Gaussians as

$$\Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{8}$$

with

$$\mu_i(\underline{u}) = \exp\left(-\frac{1}{2}\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} + \frac{(u_2 - c_{i2})^2}{\sigma_{i2}^2} + \ldots + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right) . \tag{9}$$

The normalized Gaussian validity functions $\Phi_i(\cdot)$ are interpreted as weighting factors that depend on the center coordinates c_{ij} and the individual standard deviations σ_{ij}. These parameters are non-linear and represent hidden layer parameters of the neural network. The parameters σ_{ij} are usually selected as equal constants for the whole calculation. These parameters are chosen and they influence size of the standard deviations of the validity functions Φ_i.

3.2. Lolimot algorithm steps

The classical Lolimot algorithm consist of two loops; an outer loop in which a rule premise structure is determined and a nested inner loop in which the rule consequent parameters are optimized by local estimation. Following tasks are performed:

1. *Start with initial model*: Construct the validity functions for given space partitioning, estimate Local Linear Models (LLM) parameters.

2. *Find worst LLM*: Calculate a local loss function for each LLM. This can be done with usage of the weighting squared model errors with degree of validity of the corresponding local model.

3. *Check all divisions of the worst LLM*: Selected LLM's hyper rectangle is split into two halves with an axis-orthogonal split. All divisions for each dimension are tried. New parameters and new validity functions for both parts are constructed.

4. *Select best division*: The best division from the alternatives considered in step 3 is selected. The number of LLMs is increased by one (one LLM is split into two new LLMs).

5. *Test for convergence*: Go to Step 2 unless termination condition is satisfied.

The algorithm usage can be demonstrated on a simple problem.

3.3. Lolimot algorithm example

The concept of described algorithm can be easily explained on the following example. Let as assume that we want to approximate the given function $\delta = \delta(u)$ (10) for input u by a network with three neurons (i.e. 3 LLMs)

$$\delta = \frac{a}{e^{bu}}, \quad a = 0.03,\ b = 2,\ u \in \langle 0, 1 \rangle m. \tag{10}$$

This function is quite non-linear in regions close to $u = 0$ and becomes more linear with increasing u (Fig. 4). The Lolimot algorithm will automatically divide left-sided half of proposed interval to model this non-linearity.

During the first step, only one LLM exists, therefore its validity function is equal to one. During the second step, the only one available interval is split into two halves. Then two new validity functions are constructed. During the third step, as the left LLM is worse than the right one, it is split into two halves. The same situation occurs during two following steps.

As stated in [2], it is very appropriate and beneficial that during each step all LLMs are considered for further refinement. Therefore the algorithm's complexity is adaptive; the Lolimot algorithm always constructs new LLMs where they are actually needed.

As Lolimot local models are linear, their parameters w_{ij} might be represented as lines. Because values of LLMs parameters w_{ij} in each algorithm computation step will be used in the next example (see section 5), numerical values are included — they represent absolute and linear members in the line equations

$$\text{Step 1:} \quad w_{ij}^{I} = [\ -0.0470 \quad 0.0295\ ], \quad c_{ji}^{I} = [0.5], \tag{11a}$$

$$\text{Step 2:} \quad w_{ij}^{II} = \begin{bmatrix} -0.0470 & 0.0295 \\ -0.0285 & 0.0250 \end{bmatrix}, \quad c_{ij}^{II} = [0.25\ 0.75], \tag{11b}$$

$$\text{Step 3:} \quad w_{ij}^{III} = \begin{bmatrix} -0.0470 & 0.0295 \\ -0.0285 & 0.0250 \\ -0.0137 & 0.0173 \end{bmatrix}, \quad c_{ij}^{III} = [0.125\ 0.375\ 0.75]. \tag{11c}$$

When we look at the numerical results, we can see, that between steps 2 and 3, only the first LLM has been recomputed — the second LLMs remains the same.

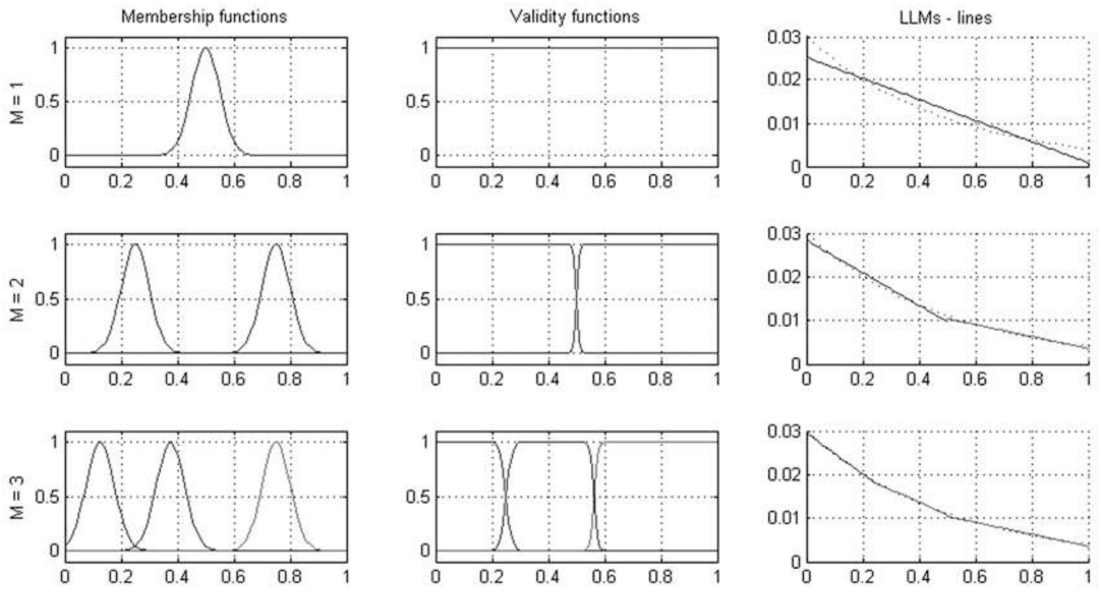

Fig. 1. Lolimot example

4. Modification of current calibration algorithm

As suggested in the previous chapters, the proposed modification uses injection of the Lolimot algorithm into the described calibration algorithm. The idea is simple. Let us replace linear transitions (and other kinematical pairs of the mechanism model if needed) with some nonlinear shapes [4]. These shapes will be constructed and modified during the calibration process. Because the calibration modifies the kinematical model (new variables are added and new model is used) new Jacobi matrix structure must be constructed in each step and for each considered division.

Similarly to the Lolimot algorithm, two loops take place. The outer loop where next calibrated model modification is selected and the inner loop where modified calibration takes place and new Lolimot parameters are computed.

A simple pseudo-code implementation follows.

1. *Start with initial model*: Construct the validity functions for given space partitioning.

2. *Generate collection of possible space divisions*: Calibrate each possible division and calculate weight function for each of the divisions.

    (a) Add new calibration parameters for new LLMs.

    (b) Generate new Jacobi matrix equations.

    (c) Calibrate newly constructed model and rank calibration.

3. *Select best possible division.* Select best possible division from the step 2. Modify the calibration model according to the selected division.

4. *Test for convergence.* Go to step 2 unless end conditions are met.

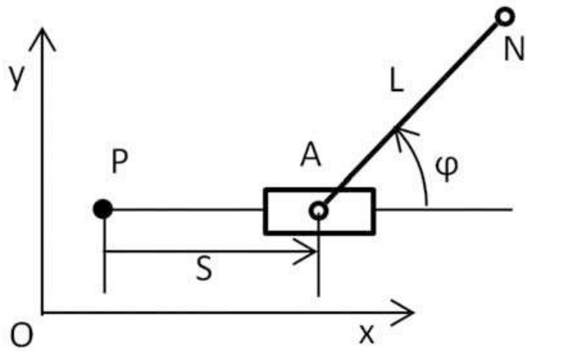

Fig. 2. Simple mechanim to be calibrated

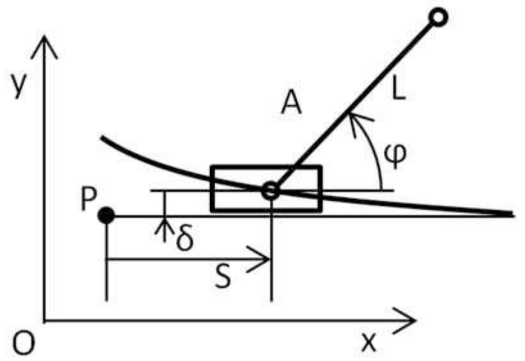

Fig. 3. Modified simple mechanim to be calibrated

4.1. Variable model complexity during calibration loops

As mentioned above, the complexity of the model is growing during the calibration process. For example, when we replace prismatic pairs (that are traditionally represented as straight lines) with a set of local linear models, we must be able to add new LLMs during calibration.

Each new LLM adds a pair of new variables w_{ij} into the set of the calibrated parameters. Moreover each LLM requires modification of the Jacobi matrix. All the partial derivatives must be reconstructed as new rows and columns are added to the Jacobi matrix. That might be quite demanding to implement as it affects the whole calibration algorithm.

It is important to specify conditions when additional complexity is redundant and inappropriate. In the following example, the weight function e into account the calibrability [7], the number of LLMs and the sum of all constraint equation discrepancies. The sum of all discrepancies is computed as

$$e = \sqrt{\sum_n f(d_n, s_n, v_n)^2}. \tag{12}$$

5. Calibration algorithm example I

In order to demonstrate the algorithm application, a simple calibration example might be used. Let us assume that we want to calibrate mechanism displayed on the Fig. 2.

The carnage with point A moves horizontally from the point P (prismatic kinematic pair). The rod of length L is connected to the carriage and might rotate around the point A. We measure the translation s, the angle ϕ and the position of the point N.

Our task is to calibrate the proposed mechanism; i.e. to compute coordinates of the point P and length of the rod L.

During creation of the calibration model, we apply the described approach to model the prismatic kinematical pair with transition s. We suppose that the prismatic pair is not ideally straight and that some curvature might be present. We don't know the exact shape of the curvature and moreover, thanks to the Lolimot approach, we don't even need this piece of knowledge.

On the Fig. 3, there is a model that is used as a source of calibration data. This model represents real mechanism with possible manufacture inaccuracies. Data acquired from an inverse kinematics of the model simulate real mechanism measurements.

In the example, we use a Lolimot representation (7) of the function (10) that consists of the three local linear models. The purpose of this example is to experimentally validate the proposed algorithm; the calibrated LLMs parameters w_{ij} should be exactly the same as the used Lolimot parameters w_{ij} (Eq. 11c) for the prismatic pair trajectory.

With a given set of coordinates of the point P, a simple direct kinematics task gives us simulated positions of the point N. The direct kinematic algorithm contains a model of the carriage with the modified trajectory

$$\begin{aligned} x_N &= x_P + s + l\,\cos(\phi)\,, \\ y_N &= y_P + \delta + l\,\sin(\phi)\,, \end{aligned} \tag{13}$$

where the member δ is constructed according to the equations (5–10) and parameters are taken from the 3rd step of the previous example's results (11c).

As mentioned above, the complexity of the calibration model grows with each addition of a new LLM. During the first outer loop step the set of calibrated dimensions d^I contains only the x-coordinate of the point $\mathbf{P}$, the length of rod $\mathbf{l}$ and the parameters for the LLM 1.

As the Jacobi matrix $\mathbf{J_d}$ consists of partial derivatives of the kinematical transformation with respect to the calibrated dimensions $\mathbf{d}$, its complexity also grows

$$\begin{aligned} d^I &= [x_P, l, w_{11}, w_{12}]\,, \\ d^{II} &= [x_P, l, w_{11}, w_{12}, w_{21}, w_{22}]\,, \\ d^n &= [x_P, l, w_{11}, w_{12}, w_{21}, w_{22},\ \ldots, w_{n1}, w_{n2}]\,. \end{aligned} \tag{14}$$

A dynamical analytical construction of the Jacobi matrix $\mathbf{J_d}$ might be a difficult task to implement. With usage of the Matlab Symbolic Toolbox, the construction of the matrix members is possible and straightforward. For each step, new symbolic variables are declared, a new kinematical transformation is constructed and its partial derivatives are computed. The result is then exported as a classical Matlab function that is later called from the calibration routines. When we define termination conditions for cases when more complex mode does not significantly improve accuracy of the calibrated variables, we get the following results

$$\text{3LLMs:} \quad w_{ij}^{III} = \begin{bmatrix} -0.047\,0 & 0.029\,5, \\ -0.028\,5 & 0.025\,0, \\ -0.013\,7 & 0.017\,3 \end{bmatrix}, \quad c^{III} = [0.125\ 0.375\ 0.75], \quad l_1 = 1m, \tag{15a}$$

$$e^I = 7.84 \cdot 10^{-2}, \quad e^{II} = 4.05 \cdot 10^{-4}, \quad e^{III} = 1.53 \cdot 10^{-7}. \tag{15b}$$

These experimental results (15a) are as expected equal to the Eq. (11c) and validate proposed calibration modification. The results (15b) are values of the Eq. (12) during calibration.

6. Calibration algorithm example II

This example is an application of the modified algorithm to a kinematical model where carriage trajectories have other (generally unknown) shapes. Let us construct a model where the carriage trajectory is represented by the Eq. 10 (Fig. 4). We intentionally choose the same parameter values. Therefore the direct kinematic algorithm contains a model of carriage with the pre-calculated modified trajectory (Eq. 13) where the member δ is constructed according to (10). A random difference generator was added to the simulated measurement to generate real measurement errors.

When the same termination conditions are used, described algorithm constructs 3 LLMs with the following parameters w_{ij} and results

$$\text{3LLMs:} \quad w_{ij}^{III} = \begin{bmatrix} -0.047\,3 & 0.029\,6 \\ -0.027\,2 & 0.024\,7 \\ -0.012\,9 & 0.016\,7 \end{bmatrix}, \quad c^{III} = [0.125\ 0.375\ 0.75], \quad l_1 = 1m. \tag{16}$$

When compared to the results of the Lolimot and the previous calibration examples (15), described calibration modification is able to model and calibrate generally nonlinear and unknown

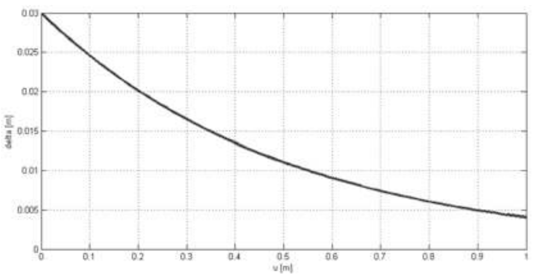
Fig. 4. Trajectory δ

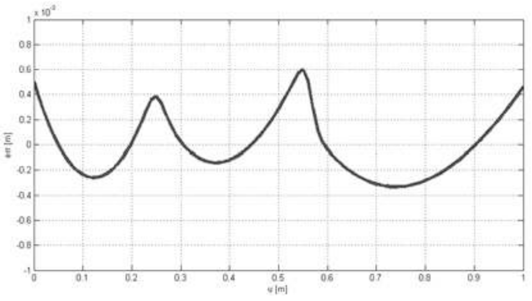
Fig. 5. Calibration error of trajectory δ

trajectories. The Fig. 5 represents error of the obtained results – a difference between the calculated trajectory δ (Fig. 4), computed from the equations (10) and (13) and the results obtained from the calibration; the Lolimot representation of the trajectory according to the equation (7) with the parameters w_{ij} from the results (16).

7. Conclusion

The described adaptive method of the kinematical calibration merges the classical calibration algorithm with the Lolimot algorithm. The main idea of the Lolimot algorithm has been reviewed and its incorporation into the calibration algorithm has been described. The new algorithm has been demonstrated on several examples.

The main advantage is its dynamical adaptability when nonlinear parts of the mechanism are used within the calibration model. This method will be applied to a calibration of complex parallel mechanisms, especially the currently developed fiber parallel kinematical structures.

Implementation for redundant parallel mechanisms is still under development.

Acknowledgements

The authors appreciate the kind support by the grant P101/11/1627 “Tilting Mechanisms Based on Fiber Parallel Kinematical Structure with Antibacklash Control” of the Czech Science Foundation.

References

[1] Mooring, B. W., Roth, Z. S., Driels, M. R., Fundamentals of Manipulator Calibration. New York : Wiley Interscience 1991.

[2] Nelles, O., Nonlinear System Identification with Local Linear Neuro-Fuzzy Models, PhD thesis, Technische Universität Darmstadt, 1998.

[3] Neugebauer, R. (ed.), Parallel Kinematic Machines in Research and Practice, Proceedings of the 4th Chemnitz Parallel Kinematics Seminar PKS 2004, IWU FhG, Chemnitz 2004.

[4] Skopec, T., Šika, Z., Valášek, M., Measurement and Improved Calibration of Parallel Machine Sliding Star, Bulletin of Applied Mechanics 6 (23) (2010) 52–56.

[5] Stejskal, V., Valášek, M., Kinematics and Dynamics of Machinery. Marcel Dekker, New York, 1996.

[6] Štefan, M., Identification of non-linear systems using neurofuzzy models, Bulletin of Applied Mechanics 2 (2005) 121–131. (in Czech)

[7] Valášek, M., Šika, Z., Hamrle, V., From Dexterity to Calibrability of Parallel Kinematical Structures. In Proceedings of 12th World Congress in Mechanism and Machine Science [CD-ROM]. Besancon: Comité Francais pour la Promotion de la Science des Mécanismes et des Machines, 2007, p. 1–6.