Verified Parameter Identification for Solid Oxide Fuel Cells

Ekaterina Auer[1], Stefan Kiel[1] and Andreas Rauh[2]
[1]*University of Duisburg-Essen, INKO, {auer,kiel}@inf.uni-due.de,*
[2]*University of Rostock, Chair of Mechatronics, andreas.rauh@uni-rostock.de*

Abstract.
In the last decades, a lot of research in the area of decentralized energy supply systems has been focused on design and development of solid oxide fuel cells (SOFC). These devices convert chemical energy directly into electricity and represent an environmentally friendly alternative for the use as auxiliary power supply units. The advantages of SOFCs include high efficiency and flexibility with respect to the kind of fuel, whereas the main disadvantages are the complicated production process and the necessity for advanced control procedures to deal with instationary operating points. Contrary to this demand, most state-of-the-art control strategies for fuel cells cover stationary operating points only. Another difficulty is that many system parameters are influenced by significant uncertainty.

An important goal of a current joint project between the Universities of Rostock and Duisburg-Essen is to develop dynamic system models which accurately describe the instationary behavior of SOFCs. Here, one possibility to deal with parameter and model uncertainty is the use of interval analysis. Aside from providing a natural representation of bounded uncertainties, interval and similar methods guarantee the correctness of simulation results. We apply a verified global optimization algorithm based on that from (Hansen and Walster(2004)) to identify uncertain parameters of a dynamic SOFC model by (Rauh et al.(2011)). The model covers the effects of preheated air and fuel gas supply along with the corresponding reaction enthalpies on the thermal behavior. The parameters of interest describe the thermal resistances of the stack materials, the dependency of heat capacities on temperature, and the heat produced during the electrochemical reactions on the surface of each individual fuel cell. Because of the complex structure of the goal function, the optimization software has to be adjusted to the problem, which was one of the reasons we chose the solver UNIVERMEC by (Dyllong and Kiel(2010)) allowing for additional flexibility.

Keywords: Interval analysis, verified optimization, SOFC systems, software design

1. Introduction

Solid oxide fuel cells (SOFCs) convert chemical energy directly into electricity and represent an environmentally friendly alternative for the use as auxiliary power supply units in, for example, different types of vehicles or stationary industrial or domestic systems. These devices are currently in the focus of research on decentralized energy supply systems due to their high efficiency and flexibility with respect to the kind of fuel. However, SOFCs are difficult to produce and in need of procedures for dealing with instationary operating points. Control strategies for SOFCs are mostly designed for constant operating conditions and are based on simplifying assumptions which are not valid for wide operation ranges (Bove and Ubertini(2008); Pukrushpan et al.(2005)). This makes development of approaches taking into account

5th International Conference on Reliable Engineering Computing (REC 2012)
Edited by M. Vořechovský, V. Sadílek, S. Seitl, V. Veselý, R. L. Muhanna and R. L. Mullen

ISBN 978-80-214-4507-9. Published by Ing. Vladislav Pokorný − LITERA

also instationary points necessary, which is the major goal of a current joint project between the Universities of Rostock and Duisburg-Essen (Rauh et al.(2011); Rauh et al.(2012a); Dötschel et al.(2012); Rauh et al.(2012b)).

Control-oriented mathematical models for SOFCs should be both accurate and applicable in real time for engineers to be able to develop robust controllers and state estimators. The thermal behavior of SOFC systems is usually described by partial differential equations. To obtain a model of the fuel cell stack temperature suitable for control design, the stack is semi-discretized into $L \times M \times N$ finite volume elements (cf. Figure 1). This leads to a set of $L \times M \times N$ nonlinear ordinary differential equations (ODEs). They are generated by the use of the first law of thermodynamics for each finite volume element to express the spatial temperature distribution in the interior of the SOFC stack. The ODEs are derived in such a way as to be valid in a wide operating range including not only the neighborhood of the desired operating point, but also the SOFC system's heating and cooling phases. The influence of varying electrical load conditions can be included in the thermal system model by means of a disturbance input. At the moment, the models are developed under the assumption that the temperature is homogenous in each volume element. However, using finite element approaches which approximate the actual situation of temperature inhomogeneities is also possible and a topic for our future work.

In this paper, we discuss possibilities to parameterize such control-oriented mathematical models in an accurate and robust way. The parameters of interest describe the thermal resistances of the stack materials, the dependency of heat capacities on the temperature, and the heat produced during the electrochemical reactions on the surface of each individual fuel cell. The parametrization is performed on the basis of measured data for the SOFC test rig available at the Chair of Mechatronics at the University of Rostock. The basics for parameter identification in our case are shown in (Rauh et al.(2011)). Note that parameters of the models developed for SOFCs are influenced by a considerable uncertainty. Aside from the inevitable model simplification, its sources are temporal and spatial discretizations as well as measurement and rounding errors.

One possibility to deal with the uncertainty (which can be accepted as bounded for the purpose of this first study) is the use of verified methods and, in particular, interval analysis (Moore et al.(2009)). They provide

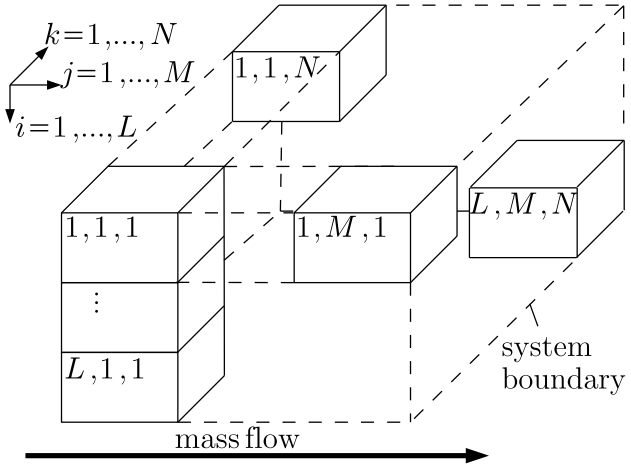

Figure 1. Semi-discretization of the fuel cell stack module into finite volume elements.

a natural representation of such kind of incertitude and guarantee the correctness of results. A successful attempt at using verified techniques for parameter identification has been made in (Rauh et al.(2012a)). By employing a basic interval optimization routine for three possible system orders (corresponding to spatial resolutions $1 \times 1 \times 1$, $1 \times 3 \times 1$, and $3 \times 3 \times 1$, cf. Figure 1), the authors managed to reduce the estimation error in comparison to the non-verified procedure which used Nelder-Mead simplex algorithm provided by MATLAB (`fminsearch`).

Optimization tasks for practical applications often have to deal with a complex goal function structure. It might include, for example, solutions to initial value problems and many summands, as is the case for the identification problem considered in this paper. Because of the complex structure of the goal function, the optimization software has to be adjusted to the problem at hand. These two reasons (complexity and the need for adjustments) usually render the use of such well known verified optimizers as GLOBSOL (Kearfott(1996)) difficult. In this paper, we consider the simplest situation of the $1 \times 1 \times 1$ spatial discretization in detail and show how the results obtained using the basic procedure from (Rauh et al.(2012a)) can be improved by exploiting the flexibility of the newly developed solver UNIVERMEC (Dyllong and Kiel(2010)). Besides implementing the usual global optimization algorithm from (Hansen and Walster(2004)) in C++, this solver allows users to choose the underlying data type for the evaluation of the goal function (e.g. naive interval or Taylor model (Berz(1995))) or the optimization strategy (e.g. with/ without differentiation, cf. Section 4.1) freely. Additionally, it is implemented in such a way as to allow for parallelization. Note that although the issue of real-time applicability is not important for the considered case of the offline parametrization, the complexity of the problem makes parallelization necessary if we want to obtain results in an acceptable time, especially if models for fluidic and electrochemical SOFC subsystems are to be included in the simulation process along with the thermal one (Rauh et al.(2011)).

The use of UNIVERMEC has one more reason. In the context of the already mentioned joint project, we plan to develop a framework for modeling, simulation, and control of SOFC systems based on the strategies developed in (Rauh et al.(2011); Dötschel et al.(2012)). This framework, supplemented by a graphical interface, should allow users to perform both verified and usual floating point computations with SOFC models of their choice easily, making flexibility of the underlying routines with respect to basic data types and algorithms unavoidable. Obviously, optimization software should also comply with this requirement, which UNIVERMEC does.

The paper is structured as follows. In Section 2, the considered problem is described in detail. A basic identification procedure for this task is summarized in Section 3. The new optimizer, parameter identification results and a comparison between the basic procedure and UNIVERMEC are reported on in Section 4. Finally, conclusions are in Section 5.

2. Problem Formulation

To describe the dynamics of fuel cell systems such as SOFCs, it is necessary to subdivide the overall model into three parts characterizing its fluidic, electrochemical and thermal behavior. Each of these subsystems is modeled by nonlinear ODEs expressing the corresponding dynamics in terms of the dominant physical phenomena such as temperature dependent heat capacities, current density dependent partial pressures of the gas mixtures, and internal Ohmic losses combined with electrical storage effects. For the modeling of

the thermal subsystem, the electrochemical reactions

$$2\,H_2 + 2O^{2-} \rightarrow 2\,H_2O + 4e^- \quad \text{and} \quad O_2 + 4e^- \rightarrow 2O^{2-} \tag{1}$$

at the anode and the cathode of the fuel cell are considered. A finite dimensional model is generated by using integral balances of the internal energy to characterize the temperature distribution. The modeling procedure described in detail in (Rauh et al.(2011)) leads to the following ODE for the discretization with $L = M = N = 1$:

$$\begin{aligned}
\dot{\theta}_{FC} \;=\; & \dot{m}_{H_2} \cdot \Big(p_{\Delta H,2} \cdot \theta_{FC}^2 + p_{\Delta H,1} \cdot \theta_{FC} + p_{\Delta H,0}\Big) + 6 \cdot p_A \cdot (\theta_A - \theta_{FC}) + (\theta_{AG} - \theta_{FC}) \\
& \cdot \; (\dot{m}_{H_2} \cdot (p_{H_2,2} \cdot \theta_{FC}^2 + p_{H_2,1} \cdot \theta_{FC} + p_{H_2,0}) + \dot{m}_{H_2O} \cdot (p_{H_2O,2} \cdot \theta_{FC}^2 + p_{H_2O,1} \cdot \theta_{FC} + p_{H_2O,0}) \\
& + \; \dot{m}_{N_2} \cdot (p_{N_2,A,2} \cdot \theta_{FC}^2 + p_{N_2,A,1} \cdot \theta_{FC} + p_{N_2,A,0})) + I_{FC} \cdot p_{el} - \dot{m}_A \cdot (\theta_{FC} - \theta_{CG}) \\
& \cdot \; (77 \cdot p_{N_2,C,0}/100 + 11 \cdot p_{O_2,0}/50 + 77 \cdot p_{N_2,C,1} \cdot \theta_{FC}/100 \\
& + \; 11 \cdot p_{O_2,1} \cdot \theta_{FC}/50 + 77 \cdot p_{N_2,C,2} \cdot \theta_{FC}^2/100 + 11 \cdot p_{O_2,2} \cdot \theta_{FC}^2/50)
\end{aligned} \tag{2}$$

with the initial condition $\theta_{FC}(0) = 299.7053$ K. Initial guesses for the time invariant parameters along with their meanings are shown in Table I. Here, the temperature dependent heat capacities c_{H_2} of hydrogen, c_{H_2O} of water vapor, $c_{N_2,A}$ of nitrogen at the anode, $c_{N_2,C}$ of nitrogen at the cathode, and c_{O_2} of air as well as the reaction enthalpy $\Delta_r H$ are replaced by their second-order polynomial approximations with certain coefficients $\alpha_{g,i}$ for $g \in \{H_2, H_2O, N_2, O_2\}$ and $i = 0, 1, 2$. After some expression manipulations, the corresponding coefficients $p_{g,i}$ from Table I appear, which are proportional to $\alpha_{g,i}$ (Rauh et al.(2011); Rauh et al.(2012a)).

The identification is performed with respect to parameters for which interval bounds are given in the table, namely, the zero-order terms of the polynomial approximations of the temperature-dependent specific heat capacities and the reaction enthalpy. Time variant inputs are shown in Table II. Their values along with the temperature θ_{FC} are measured each second ($h = 1$) for the time period of $T = 19963$ seconds. The measurement device is known to cause the uncertainty of $\Delta y_m \in [\Delta y_m]$. Aside from including the information about the measurement error of the device, the interval $[\Delta y_m]$ should enclose the effects of not being able to measure the temperature at exactly the same point as defined by the output variable of a certain finite volume element in the model.

Since we plan to use verified techniques, the task is to minimize the upper bound $\overline{J}$ of the goal function

$$J = \sum_{k=1}^{T} \left(y(t_k, p) - y_m(t_k)\right)^2 \tag{3}$$

with respect to the six parameters $p = [p_{H_2,0}\;\; p_{H_2O,0}\;\; p_{N_2,A,0}\;\; p_{N_2,C,0}\;\; p_{O_2,0}\;\; p_{\Delta H,0}]$, where $y(t_k, p) = \theta_{FC}(t_k, p)$ is the simulated temperature of the fuel cell at the time $t_k = 1, \ldots, T$ obtained from Eq. (2) and $y_m(t_k)$ the measured temperature at the same point. Note that here and in the following, the goal function is written down for our situation of $h = 1$, $t_k = h \cdot k$, $k = 1, \ldots, T$. The function J quantifies deviations between the measured output vector and the simulated temperature vector acquired with T samples and a

Table I. Meanings and values of the constant parameters in Eq. (2)

Name	Value	Physical meaning (proportional to)
p_A	$-2.111896 \cdot 10^{-5}$	inverse of the thermal insulation resistance
p_{el}	$1.645381 \cdot 10^{-3}$	resistance of SOFC materials
$p_{H_2,0}$	-7.614159 ± 1	heat capacity of hydrogen (order 0)
$p_{H_2,1}$	$-6.023259 \cdot 10^{-5}$	heat capacity of hydrogen (order 1)
$p_{H_2,2}$	$9.513841 \cdot 10^{-8}$	heat capacity of hydrogen (order 2)
$p_{H_2O,0}$	0.6273529 ± 1	heat capacity of water vapor (order 0)
$p_{H_2O,1}$	$-6.479947 \cdot 10^{-4}$	heat capacity of water vapor (order 1)
$p_{H_2O,2}$	$-1.583060 \cdot 10^{-7}$	heat capacity of water vapor (order 2)
$p_{N_2,A,0}$	-0.8367768 ± 1	heat capacity of nitrogen at the anode (0)
$p_{N_2,A,1}$	$6.250080 \cdot 10^{-4}$	heat capacity of nitrogen at the anode (1)
$p_{N_2,A,2}$	$-6.366022 \cdot 10^{-9}$	heat capacity of nitrogen at the anode (2)
$p_{N_2,C,0}$	0.4525605 ± 1	heat capacity of nitrogen at the cathode (0)
$p_{N_2,C,1}$	$-1.453636 \cdot 10^{-4}$	heat capacity of nitrogen at the cathode (1)
$p_{N_2,C,2}$	$1.974873 \cdot 10^{-8}$	heat capacity of nitrogen at the cathode (2)
$p_{O_2,0}$	-0.5931353 ± 1	heat capacity of air (0)
$p_{O_2,1}$	$-8.060370 \cdot 10^{-6}$	heat capacity of air (1)
$p_{O_2,2}$	$-2.049129 \cdot 10^{-9}$	heat capacity of air (2)
$p_{\Delta H,0}$	-217.3967 ± 1	reaction enthalpy (0)
$p_{\Delta H,1}$	$-5.236888 \cdot 10^{-2}$	reaction enthalpy (1)
$p_{\Delta H,2}$	$-5.014673 \cdot 10^{-6}$	reaction enthalpy (2)

constant sampling time h. An additional condition, which is induced by the accuracy of the measurements $[\Delta y_m]$, is

$$y(t_k) \subseteq [y_m(t_k) - 15, y_m(t_k) + 15] =: [\Delta y_m(t_k)] \quad \text{for} \quad t_k = 1, \ldots, T \ . \tag{4}$$

Note that since we have only one temperature to model the thermal behavior of the whole stack in the $1 \times 1 \times 1$ case, the uncertainty of ± 15 also includes the incertitude arising from spacial discretization.

The global optimization problem (3) can be solved in the following two general ways, if there is no possibility to find an analytical solution to Eq. (2):

1. Verified approximation. The true solution $\theta_{FC}(t) = y(t)$ of the Eq. (2) is approximated by the explicit Euler method as

$$[y_k] := [y_{k-1}] + h \cdot f([y_{k-1}], [p]) \ , \tag{5}$$

where f denotes the right side of the Eq. (2). The approximation $[y_k]$ is substituted for the exact solution $y(t_k)$ in the goal function (3) and the discretization error ignored. In our setup, the sampling time h of 1 second is by at least two orders of magnitude smaller than the dominant time constants of the thermal process (2). For this reason, $[y_k]$ is an acceptable approximation of the true solution $y(t_k)$. Although we cannot verify the whole process by applying interval procedures in this case, the results of optimization of

Table II. Time-dependent inputs in Eq. (2)

Name	Meaning
$\dot{m}_{H_2}$	mass flow of hydrogen
$\dot{m}_{N_2}$	mass flow of nitrogen at the anode
$\dot{m}_{H_2O}$	mass flow of water vapor
$\dot{m}_A$	mass flow of air at the catode
θ_{AG}	initial temperature of the anode gas in K
θ_{CG}	initial temperature of the cathode gas in K
θ_A	temperature of the environment in K
I_{FC}	electric current in A

the approximated goal function

$$J_{app} = \sum_{k=1}^{T} \left(y_{k-1} - y_m(t_k) + h \cdot f(y_{k-1}, p)\right)^2 \tag{6}$$

are verified. It is easy (although costly) to find the first (and second) derivatives of J_{app} with the help of algorithmic differentiation using, for example, FADBAD++ (Stauning and Bendtsen(2006)), should the chosen optimization algorithm (Kearfott(1996); Hansen and Walster(2004); Rauh et al.(2012b)) require that.

2. Verified solution. The true solution $\theta_{FC}(t) = y(t)$ of the Eq. (2) is enclosed numerically by a verified IVP solver such as VNODE-LP (Nedialkov(2011)). It is more difficult to compute the derivatives $\partial J/\partial p$ of the cost function J, because it requires solving an additional IVP of the form

$$\dot{s}_i = \frac{\partial f}{\partial \theta_{FC}} \cdot s_i + \frac{\partial f}{\partial p_i} \quad \text{with} \quad s_i = \frac{\partial \theta_{FC}}{\partial p_i} \tag{7}$$

in our one-dimensional case for each parameter p_i in each iteration step since

$$\frac{\partial J}{\partial p_i} = 2 \cdot \sum_{k=1}^{T} (y(t_k, p_i) - y_m(t_k)) \cdot s_i \ . \tag{8}$$

This procedure can be very expensive, especially if the sensitivity equations cannot be derived analytically and must be obtained by algorithmic differentiation. Therefore, it is advisable to prefer derivative-free optimization techniques in this case, although the additional information provided by derivatives usually improves the performance of a method.

For the purpose of an initial verified study, we consider only the first situation in this paper. Note that even the approximated situation (2), (6), (4) is not a simple one. Although the summands from (6) are easier to compute, the approximated problem is affected by the same difficulties as (3) from the point of view of verification in addition to suffering from discretization errors. For example, it involves at least T occurrences of the same (interval) parameters leading to the dependency problem and overestimation (Kieffer et al.(2011)). Since T is equal to 19963 in our case, overestimation reduction is not trivial for the considered optimization task.

3. Basic Approach to Verified Optimization

In this section, we describe the basic procedure shown in Figure 2, which was applied to the problem (2), (6), (4) in (Rauh et al.(2012a)). It is implemented as a C++ routine using PROFIL/BIAS (Keil(2008)) for basic interval evaluations and FADBAD++ for algorithmic differentiation. The first step is to define the interval vector bounding the search space. To find the globally optimal solution, this vector is subdivided further in the direction μ, chosen in each iteration step in such a way as to maximize the sensitivity measure

$$\Delta J^{<l>} = \sum_{\nu=1}^{T} \left[\frac{\partial [y^{<l>}(t_\nu)]}{\partial [p^{<l>}]} \cdot \frac{\mathrm{w}([p^{<l>}])^2}{\mathrm{w}([p^{<0>}])} \right] \tag{9}$$

for each parameter p, where L is the current list length, $l = 0, \ldots, L$, and $\mathrm{w}(\cdot)$ the width of the box. Intervals from the list which produce a small upper bound for $\overline{J}^{<l>}$ are preferred for subdivision. For the resulting subintervals, a validity criterion based on the condition in (4) is applied additionally. The following three cases are distinguished:

Consistent parameter vectors are no longer subdivided. They are characterized by

$$[y(t,p)] \subseteq y_m(t) + [\Delta y_m] \tag{10}$$

for each $t \in [0,T]$ with the worst-case measurement error $[\Delta y_m] = [-15, 15]$.

Inconsistent parameter vectors are excluded. They are identified by

$$[y(t,p)] \cap (y_m(t) + [\Delta y_m]) = \emptyset \tag{11}$$

for at least one point of time $t \in [0,T]$.

Undecided parameter vectors are subdivided further either until they fall below a minimum diameter or until a maximum number of subdivisions is reached. They are characterized by

$$[y(t,p)] \cap (y_m(t) + [\Delta y_m]) \neq \emptyset \tag{12}$$

for each $t \in [0,T]$ and

$$[y(t,p)] \not\subseteq y_m(t) + [\Delta y_m] \tag{13}$$

for at least one point of time.

Note that this procedure is not an all-encompassing one. The basic branch-and-bound algorithm is tailored for the problem at hand by exploiting the condition in Eq. (4) as its constituent part (cf. the validity test in Figure 2). In return, it can handle higher system orders ($1 \times 3 \times 1$ and $3 \times 3 \times 1$ finite volume elements along with the $1 \times 1 \times 1$ problem considered here) and more parameters (e.g. all from Table I), a difficult task for a general-purpose solver. The results produced by this procedure for the problem described in Section 2 in comparison to those obtained with UNIVERMEC (which is a highly adjustable general-purpose solver) will be shown in Section 4.2.

Initialization of the parameter range $[p^{<0>}] = [p^{<L>}]$, $L = 1$

while (stopping criterion is not reached)

- **Determine the parameter vector** $[p^{<l>}]$ to be subdivided from the complete list $[p^{<1>}], \ldots, [p^{<L>}]$
- **Determine the component** μ of the parameter vector $[p^{<l>}]$ to be subdivided
- **Splitting procedure:** $[p^{<L+1>}] := [p^{<l>}]$

$$[p_\mu^{<l>}] := \left[\inf\left([p_\mu^{<l>}]\right), \text{mid}\left([p_\mu^{<l>}]\right)\right]$$
$$[p_\mu^{<L+1>}] := \left[\text{mid}\left([p_\mu^{<L+1>}]\right), \sup\left([p_\mu^{<L+1>}]\right)\right]$$

- **Evaluate state equations** for $[p^{<l>}]$
- **Validity test:** Delete $[p^{<l>}]$ from the parameter list if guaranteed to be inconsistent
- **Evaluate state equations** for $[p^{<L+1>}]$
- **Validity test:** Delete $[p^{<L+1>}]$ from the parameter list if guaranteed to be inconsistent
- **Recount** the length L of the list

Figure 2. The basic routine for verified parameter identification.

4. Application of a General-Purpose Optimizer and Comparison of Results

In this section, we show how to apply the general-purpose optimizer UNIVERMEC to the problem (2), (6), (4). The major difficulty here is to overcome overestimation. Besides, we need to incorporate the condition in Eq. (4) in an appropriate form. First, we describe the software-oriented foundation making our optimizer more flexible than other general-purpose ones such as GLOBSOL. After that, we show the results of its application to the problem of SOFC parameter identification. Finally, we compare the outcome to that of the basic routine from Section 3.

4.1. UNIVERMEC: DESIGN, ALGORITHMS, FEATURES

The optimization algorithm (Dyllong and Kiel(2010)) is implemented in the uniform framework UNIVERMEC (**Uni**fied Framework for **Ver**ified Geo**Me**tric **C**omputations), originally developed for geometric computations (Dyllong and Kiel(2011)). UNIVERMEC provides a conceptual and a software basis for

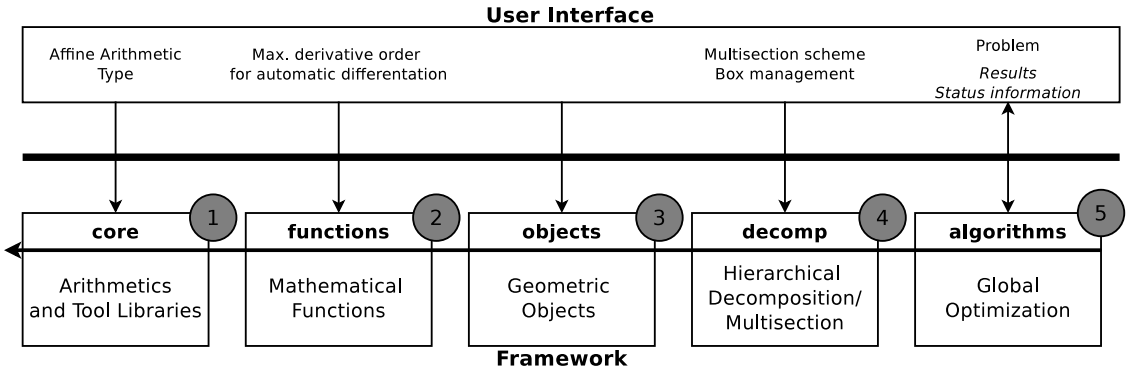

Figure 3. The basic structure of UNIVERMEC. Every layer depends only on the ones left of it. The parameters of each layer can be changed through the user interface. Words in italics denote information from the framework.

handling different kinds of (verified) arithmetics (e.g interval, affine (de Figueiredo and Stolfi(1997)), or Taylor model) and algorithms uniformly. Owning to its design, the framework allows users to evaluate their models (e.g. for geometric objects) with the arithmetic suitable for their task. If necessary, various decomposition or branching schemes can be applied to user-defined models regardless of their actual mathematical representation. Higher level algorithms, such as parameter identification or distance computation, can access these different methods easily, allowing for ready reuse and exchange. Additionally, UNIVERMEC provides a fair comparison between specific techniques inside high level algorithms because the overhead and implementations of the algorithms themselves are identical.

An overview of the framework components is given in Figure 3. The code is organized into five conceptual layers. The first layer (`Core`) implements arithmetic concepts in form of an abstract algebra and wrappers for actual libraries (e.g. C-XSC (Hofschuster and Krämer(2004)) for intervals or YALAA (Kiel(2012)) for affine arithmetic). The second layer (`Functions`) defines an interface for scalar functions formally and independently of the chosen kind of arithmetic. The layer `Objects` specifies a uniform formal representation for different models of objects, which is important for geometric computations and can be skipped in our context of parameter identification. The forth layer `Decomp` encapsulates various hierarchical decomposition and multisection strategies. Finally, the layer `Algorithms` provides a basis for flexible implementations of different kinds of high level algorithms, in particular, the optimization algorithm we use in this paper to solve the problem (2), (6), (4). Although the algorithm can be implemented without the framework, UNIVERMEC ensures the maximum flexibility necessary to solve the task.

The optimizer is developed to solve a general problem with inequality constraints according to the approach described in (Hansen and Walster(2004)):

$$\begin{aligned} &\min \phi(x) \quad \text{with } x \in \mathbb{R}^{\mathrm{d}} \\ &\text{subject to } \; g_i(x) \leq 0 \;\; \text{for } i = 1, ..., m, \end{aligned} \tag{14}$$

where the objective function $\phi : \mathbb{R}^d \mapsto \mathbb{R}$ is scalar. The algorithm calculates a verified interval enclosure of the minimum and maintains three lists: the working list $\mathcal{L}$, the temporary list $\mathcal{L}_{\mathrm{tmp}}$, and the result list $\mathcal{L}_{\mathrm{final}}$. All lists contain parts of the search space in the form of boxes. $\mathcal{L}$ accommodates the boxes to be processed further, while $\mathcal{L}_{\mathrm{final}}$ is filled with the final results. The basic structure of the algorithm is outlined in Figure 4.

To allow users to adapt the algorithm to their problems, it is subdivided into several stages (PHASE_ in the figure), the behavior of which can be changed individually. Following (Hansen and Walster(2004)), we choose contractors and strategies for box reduction in dependence on the feasibility of the current box. That is, it is necessary to distinguish the stages PHASE_POS_INFEAS, PHASE_FEAS, and PHASE_STRICT_FEAS for boxes with unknown, certain and strictly certain feasibility, respectively (cf. Figure 4). The contractors in the stages PHASE_A to PHASE_D are called independently of the feasibility of the current box in each iteration, while PHASE_SPLIT is called on boxes directly after the multisection step.

Inside an `apply_contractor` call, a box may be pruned, completely discarded or moved to $\mathcal{L}_{\mathrm{final}}$ if it satisfies the termination criterion. In the latter two cases, the main loop is restarted. Unlike other global optimization algorithms, UNIVERMEC maintains not only the lists $\mathcal{L}$ and $\mathcal{L}_{\mathrm{final}}$, but also $\mathcal{L}_{\mathrm{tmp}}$ similarly to the distance computation algorithm in (Dyllong and Kiel(2011)). If a box is subdivided below a certain minimum width ϵ_t, it is temporarily deleted from $\mathcal{L}$ and moved into $\mathcal{L}_{\mathrm{tmp}}$. This strategy ensures that the problem domain is subdivided more uniformly and prevents heuristics such as best-first from causing a deep subdivision in the wrong region. When $\mathcal{L}$ becomes empty, all boxes from $\mathcal{L}_{\mathrm{tmp}}$ are moved back into $\mathcal{L}$, and the user can alter the algorithm stages again. In this way, accelerating devices such as interval Newton can be configured dynamically.

The algorithm can be parallelized efficiently, since it can work on parts of the search region independently of other parts. The search region cannot be subdivided a priori, because it is unknown where a deep search will take place. For workload sharing, a permanent synchronization among all threads is important. In our current implementation, $\mathcal{L}$ is shared among all threads, which is suitable only for shared-memory architectures. Besides, there is a possible bottleneck in the algorithm, because every access to $\mathcal{L}$ represents a critical section. However, the time spent there can be reduced by an intelligent initialization approach for the verified upper bound of the minimum $\tilde{\phi}$ (e.g. using IPOPT (Wächter and Biegler(2006))).

In practice, evaluating the goal function is expensive. While derivatives are theoretically available through FADBAD++ in UNIVERMEC, we do not use them, because their recursive evaluation at every time step slows down the computations considerably. Instead, we employ the following derivative-free strategies:

PHASE_SPLIT: Bounds the goal function with interval arithmetic and test condition (4).

PHASE_PA: Tests the feasibility.

PHASE_POS_FEAS: Tests the box consistency on constraints.

PHASE_TMP: Tries to find a verified upper bound on the minimum using the midpoint test.

PHASE_FINAL: Bounds the goal function using affine arithmetic and pruning by (15)–(16).

We use the implicit linear interval estimation (ILIE) technique (Bühler(2002)) to take the condition in Eq. (4), which cannot be written down in terms of expressions valid globally for all $t = 1, \ldots, T$, into account in our general purpose solver. Originally, this technique was developed and used for computer graphics applications. However, it also helps to incorporate such heuristic conditions as Eq. (4) into the overall optimization procedure.

Input: Search region X_0
Output: Enclosure of minimum ϕ^*

$\mathcal{L} := \{X_0\}$; $S := RUNNING$;
while $S == RUNNING$ **do**
  if $\mathcal{L} = \emptyset$ **then**
    Get configuration for next phase;
    forall $X' \in \mathcal{L}_{\text{tmp}}$ **do**
      `apply_contractors(`*PHASE_TMP,* X'`)`;
    end
    $\mathcal{L} := \mathcal{L}_{\text{tmp}}$; $\mathcal{L}_{\text{tmp}} := \emptyset$;
    if $\mathcal{L} == \emptyset$ **then** $S := FINISHED$; **continue;**
  end
  $X := head(\mathcal{L})$; $\mathcal{L} := tail(\mathcal{L})$;
  `apply_contractors(`*PHASE_A,* X`)`;
  if $\neg$`feasible(`X`)` **then**
    `apply_contractors(`*PHASE_POS_INFEAS,* X`)`;
  end
  `apply_contractors(`*PHASE_B,* X`)`;
  if `feasible(`X`)` **then** `apply_contractors(`*PHASE_FEAS,* X`)`;
  `apply_contractors(`*PHASE_C,* X`)`;
  if `strictly_feasible(`X`)` **then**
    `apply_contractors(`*PHASE_STRICT_FEAS,* X`)`;
  end
  `apply_contractors(`*PHASE_D,* X`)`;
  if $\mathrm{w}(X) < \epsilon_t$ **then** $\mathcal{L}_{\text{tmp}} \leftarrow X$;
  else
    $\mathcal{N}$:= `split(`X`)`;
    forall $X' \in \mathcal{N}$ **do**
      `apply_contractors(`*PHASE_SPLIT,* X'`)`;
    end
    $\mathcal{L} \leftarrow \mathcal{N}$;
  end
end
forall $X' \in \mathcal{L}_{\text{final}}$ **do**
  `apply_contractors(`*PHASE_FINAL,* X'`)`;
end
$\underline{\phi^*} := \min_{X \in \mathcal{L}_{\text{final}}} (\underline{\phi(X)})$; $\overline{\phi^*} := \min(\max_{X \in \mathcal{L}_{\text{final}}} (\overline{\phi(X)}), \phi^*)$;

Figure 4. Interval optimization algorithm UNIVERMEC: $\tilde{\phi}$ is a verified upper bound on the minimum.

Let $y(t_k, p)$ be the solution of the IVP (2) at the point t_k, p the parameter vector ($p \in \mathbb{R}^6$ in our case), and $L(t_k, p)$ its linearized enclosure

$$L(t_k, p) := \sum_{i=1}^{6} a_i p_i + [l] \quad \text{with} \quad [l] \in \mathbb{I}, p_i \in \mathbb{R} \ . \tag{15}$$

Further, if $[\Delta y_m(t_k)]$ is the measurement with the corresponding uncertainty at the point t_k and $[p]$ the current interval box for the (unknown) parameters p, then the interval $L(t_k, p)$ for each $p \in [p]$ should not exceed $[\Delta y_m(t_k)]$ for the condition in Eq. (4) to be valid. After factoring out a parameter p_i, $i = 1 \ldots 6$,

from this relation, we obtain the following transformed condition:

$$p_i \in \frac{[\Delta y_m(t_k)] - [l] - \sum_{j=1, j\neq i}^{6} a_j p_j}{a_i} \ , \tag{16}$$

which can be incorporated as a solution strategy into the overall algorithm. Here, the coefficients a_i and the enclosure of the nonlinearity $[l]$ are determined by using affine arithmetics and, in particular, the library YALAA, which UNIVERMEC allows us to access.

4.2. RESULTS AND COMPARISON

We solved the optimization problem with UNIVERMEC on an Intel Xeon CPU x5570, 2.9 GHz, 12Gb RAM virtualized computer with four cores. The algorithm was parallelized with the help of OpenMP (`http://openmp.org`). To obtain results comparable to those from the basic routine in Section 3, we limited the number of iterations to 54000 as had been done there. Besides, we used only the midpoint test as a discarding strategy. Note that the search space is bisected inside the basic routine, whereas UNIVERMEC uses the multisection scheme according to (Ratz(1992)). To reduce the computational effort, we updated the upper bound only between stage switches (cf. Figure 4). The value for ϵ_t was initially set to $\mathrm{w}(X_0)/5$ and then reduced by half during every change of the stage.

After 54000 iterations, we obtained 36942 candidate boxes for optimal parameters p. No verified solution strictly complying with (4) could be found, as was also the case for the basic procedure from Section 3. The results are shown in Figure 5, right, in comparison to those from the basic procedure (Figure 5, left). In the figure, the difference $\delta(t_k)$ between the simulated and the measured temperature value is shown for the midpoints of the set of candidate optimal parameters $[\hat{p}]$, chosen in such a way as for the absolute error value $|\delta(t_k)|$ to be the smallest. Note that by selecting such $[\hat{p}]$, we no longer work with entirely verified results, but only with candidates. The results obtained by UNIVERMEC have less deviation from the measured values than those from the basic routine.

In Table III, we compare the outcome for the problem (2), (6), (4) obtained by a floating-point optimization in MATLAB, the basic interval routine from Section 3, and UNIVERMEC. The non-verified results are described in more detail in (Rauh et al.(2012a)). As the comparison measure e given in the table, we use the following practice-motivated expression:

$$e = \sqrt{\frac{\sum_{k=1}^{T} \left(y_{k-1} - y_m(t_k) + f(y_{k-1}, \mathrm{mid}([\hat{p}]))\right)^2}{T}} \ . \tag{17}$$

This choice leads to values of e similar to the usual root mean square error measure in the floating point case. The table shows that the interval based strategies are less accurate than the floating point one for this rudimentary model of order $1 \times 1 \times 1$, although the results obtained in UNIVERMEC are better than those from the basic routine. More accurate interval solutions can be produced by using higher order temperature models as shown in (Rauh et al.(2012a)). We plan to test these enhanced models with UNIVERMEC in the future.

In the third line of Table III, we provide CPU times for all three kinds of computations. Note that they cannot be considered absolutely, because they were obtained on different computers (Intel dual core

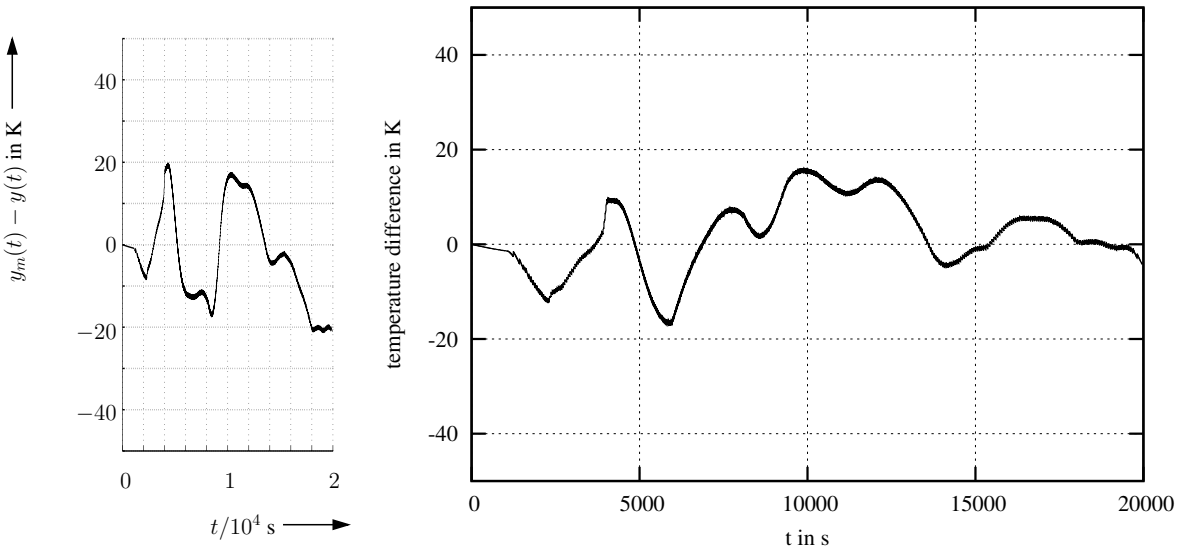

Figure 5. Difference between the simulated and measured temperatures for the model of order $1 \times 1 \times 1$ for the basic optimization routine from Section 3, left, and UNIVERMEC, right.

CPU, 2.8 GHz, 2 GB RAM for the basic interval and floating point routines) and in different environments (MATLAB for floating point vs. C++ for the others). Besides, the floating-point optimization was carried for all 20 parameters from Table I. However, the CPU times can give an idea of what the advantage of parallelization can be. For UNIVERMEC, we provide both the CPU time and the real time to show how long the computations took in reality. The results demonstrate that the usage of derivative free techniques in UNIVERMEC speeds up the computations (3.2 vs. 10 hours of CPU time) and that parallelization is effective in this case (1.33 hours of real time). Note that the CPU times include not only the ones needed to obtain candidate solutions, but also those for constructing the solutions in Figure 5 in all cases.

Table III. Empirical comparison between floating point routine, the basic interval routine, and UNIVERMEC for the SOFC model of order $1 \times 1 \times 1$

	floating point	basic interval	UNIVERMEC
error e in K	5.17	11.84	7.68
CPU time in hours	≈ 6	≈ 10	3.2 (1.33 real time)

Other experiments we performed showed that the settings we chose represented a good compromise between performance and accuracy for this rudimentary model. Increasing the number of iterations up to 270000 lead to a slightly smaller value of 7.5 for the root mean square error measure while increasing the CPU time considerably. The linearization strategy (15)–(16) does not offer much improvement in such situations because the resulting boxes are still too large. Setting the termination criterion to the box width of

10^{-1} (for which the linearization might work) is accompanied by computing times of over a week, which is not justified for this simple model. That is, if better accuracy is requested, better models should be used.

5. Conclusions

In this paper, we considered the simplest model for the temperature of a fuel cell stack in detail from the point of view of obtaining accurate and robust results for the problem of identification of its parameters. We showed how to apply a range of interval based algorithms to it. Although the results obtained using interval procedures are slightly less accurate than the floating point ones in this simplest case, the research in this direction seems promising because the accuracy of interval techniques improves if better models are used (Rauh et al.(2012a)). The presented verified optimizer UNIVERMEC helped to increase both the accuracy and the performance of interval techniques. The reasons were its flexible implementation, the use of derivative-free techniques and parallelization.

Our future work includes three general directions. The first one concerns SOFC modeling and control. The developed models will be used for a nonlinear state and disturbance observer design. Sensitivity based parameter identification routines will be implemented for a further improvement of the model quality. The second direction is to incorporate all the developed models and algorithms into a unified tool for modeling, simulation, and control of SOFCs with the help of both verified and floating point methods.

The last future work direction concerns the improvement of the quality of the obtained parameter set. For that purpose, we plan to implement the fully verified identification technique mentioned in Section 2 in UNIVERMEC. Moreover, the use of better models of orders $1 \times 3 \times 1$ and $3 \times 3 \times 1$ might help to actually verify the optimum of the problem (2), (6), (4). Besides, more thought should be given to the empirically motivated condition (4) itself. Although it helps to reduce the overestimation in the goal function (6), our tests have shown that the midpoints of the obtained parameter sets are infeasible with respect to it in spite of a high subdivision depth. Therefore, we should either prove the validity of the condition mathematically (at least, under certain assumptions), or try to reduce the overestimation in the goal function in other ways.

Acknowledgements

This work is founded by the German Research Council.

References

M. Berz. Modern Map Methods for Charged Particle Optics. *Nuclear Instruments and Methods A363*, pages 100–104, 1995.

R. Bove and S. Ubertini, editors. *Modeling Solid Oxide Fuel Cells*. Springer, Berlin, 2008.

K. Bühler. Implicit Linear Interval Estimations. In *Proceedings of the 18th spring conference on Computer graphics*, page 132. ACM, 2002.

L. de Figueiredo and J. Stolfi. *Self-Validated Numerical Methods and Applications*. IMPA, Rio de Janeiro, 1997.

T. Dötschel, A. Rauh, and H. Aschemann. Reliable Control and Disturbance Rejection for the Thermal Behavior of Solid Oxide Fuel Cell Systems. In *Proc. of Vienna Conference on Mathematical Modelling MATHMOD 2012*, Vienna, Austria, 2012. Accepted.

E. Dyllong and S. Kiel. A Comparison of Verified Distance Computation between Implicit Objects Using Different Arithmetics for Range Enclosure. *Computing*, 2011.

E. Dyllong and S. Kiel. Verified Distance Computation Between Convex Hulls of Octrees Using Interval Optimization Techniques. *PAMM Special Issue: 81st Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM)*, 10(1):651–652, 2010.

E. Hansen and G. W. Walster. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 2004.

W. Hofschuster and W. Krämer. C-XSC 2.0 — A C++ Library for Extended Scientific Computing. *LNCS 2991: Numerical Software with Result Verification*. Springer, 2004

R. Kearfott. *Rigorous Global Search: Continuous Problems*. Nonconvex optimization and its applications. Kluwer Academic Publishers, 1996.

C. Keil. PROFIL/BIAS, Version 2.0.8, 2008. `www.ti3.tu-harburg.de/keil/profil/`.

M. Kieffer, M. Csaba, H. Schichl, and E. Walter. Verified Global Optimization for Estimating the Parameters of Nonlinear Models. In *Modeling, Design and Simulation of Systems with Uncertainties*, Mathematical Engineering. Springer, 2011.

S. Kiel. YALAA – Yet Another Library for Affine Arithmetic. *Reliable Computing*. Submitted.

R. Moore, B. Kearfott, and M. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

N. S. Nedialkov. Implementing a Rigorous ODE Solver through Literate Programming. In A. Rauh and E. Auer, editors, *Modeling, Design, and Simulation of Systems with Uncertainties*, Mathematical Engineering. Springer, 2011.

J. Pukrushpan, A. Stefanopoulou, and H. Peng. *Control of Fuel Cell Power Systems: Principles, Modeling, Analysis and Feedback Design*. Springer, Berlin, 2nd edition, 2005.

D. Ratz. *Automatische Ergebnisverifikation bei Globalen Optimierungsproblemen*. Universität Karlsruhe, 1992.

A. Rauh, T. Dötschel, and H. Aschemann. Experimental Parameter Identification for a Control-Oriented Model of the Thermal Behavior of High-Temperature Fuel Cells. In *CD-Proc. of IEEE Intl. Conference MMAR 2011*, Miedzyzdroje, Poland, 2011.

A. Rauh, T. Dötschel, E. Auer, and H. Aschemann. Interval Methods for Control-Oriented Modeling of the Thermal Behavior of High-Temperature Fuel Cell Stacks. In *Proc. of SysID 2012*, 2012a. Accepted.

A. Rauh, L. Senkel, and H. Aschemann. Sensitivity-Based State and Parameter Estimation for Fuel Cell Systems. In *Proc. of 7th IFAC Symposium on Robust Control Design ROCOND 2012*, Aalborg, Denmark, 2012b. Submitted.

O. Stauning and C. Bendtsen. FADBAD++ web page. `http://www.fadbad.com/` .

A. Wächter and L. T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Math. Program.*, 106(1):25–57, 2006. ISSN 0025-5610.