An Environment for Verified Modeling and Simulation of Solid Oxide Fuel Cells

Stefan Kiel, Ekaterina Auer and Andreas Rauh

SCAN 2012

Table of Contents

Modeling, Simulation and Control of Solid Oxide Fuel Cells

SOFCs: devices converting chemical energy in electricity

+ high efficiency, flexibility wrt. fuel
– high operating temperature

Our goals:

- Models better suitable for control
- Verified methods for robustness
- Modeling/simulation/control in `VeriCell`

VerIPC-SOFC Project

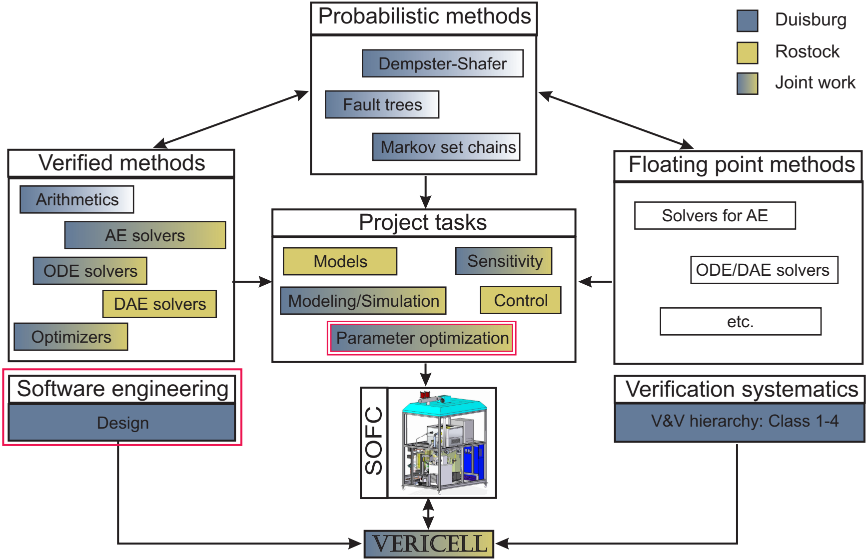
A joint project between the Universities of Rostock and Duisburg-Essen

Development of the flexible software `VeriCell`

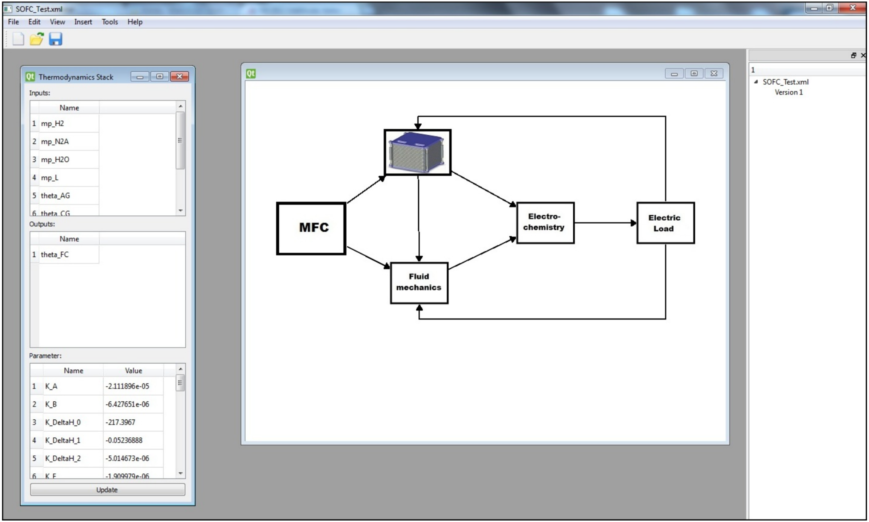
Task	**Method**
Use of different SOFC models Interfacing of existing solvers	Model abstraction, Plugins
Parameter Optimization	Global optimization
SOFC simulation and control	ODE solver, DAE solver

→ Allow for using verified and non-verified methods

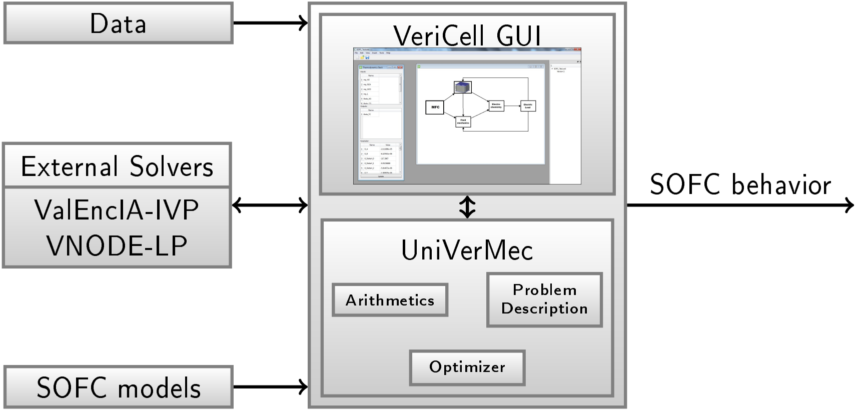
Project Overview



VeriCell GUI



Software Architecture



Ingredients of an SOFC Model Component

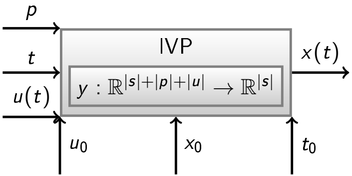
Model equation (IVP):

$$\dot{x}(p, u(t), t) = \underbrace{f\,(x(t), p, u(t))}_{y:\mathbb{R}^{|s|+|p|+|u|}\rightarrow\mathbb{R}^{|s|}}$$

depends on

t	Time
p	Parameters
$u(t)$	Dependent parameters
$s := x(t)$	Model states

Abstract IVP class



Values for

Starting time t_0

$u_0 = u(t_0), p_0 = p$

$x_0 = x(t_0)$

Acts as a basis for

- simulation
- parameter optimization

Solving the IVPs (Simulation)

Information needed by an IVP solver

IVP

End time t_{end}

Solver's specific options S_o

Possible solvers

Euler's method

ValEncIA-IVP

Needs derivatives of y

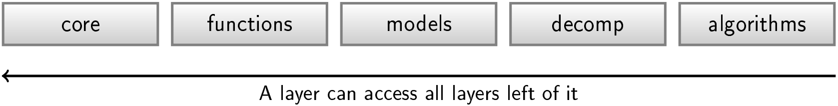
VNODE-LP

Needs Taylor coefficients of y and its Jacobian

How to represent an IVP (and y) for use with different solvers?

UniVerMeC

Unified Framework for Verified GeoMetric Computations



Uniform handling of verified techniques through a relaxed layer structure

- core: *Adapter for underlying arithmetic libraries*
- functions: *Uniform representation for functions*
- models: *IVP models*, Implicit surfaces, CSG models
- decomp: *Multisection schemes*, Spatial decomposition
- algorithms: *Global optimization*, *IVP solver interfaces*

Application in VeriCell

algorithms — Algorithms built upon the framework
→ Global optimization; VNODE-LP interface; ValEncIA-IVP interface

trees — Uniform interval tree representation

section — Multisection schemes and box management
→ Naive/Ratz multisection; Coordinate direction weights | Standard interval trees; Contracting trees

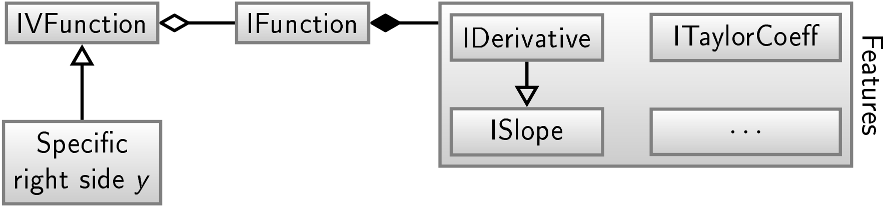
objects — Uniform geometric object representation

ivp_models — IVP model representation
→ SOFC models | Implicit objects/CSG-operations; Polyhedrons/parametric objects; Deformations/transformations

functions — Uniform function representation
→ Model right hand side; Derivatives; Taylor coefficients

core — Uniform interfaces for arithmetics
→ Real/interval/affine arithmetic; Taylor models; Matrices and vectors

Representation of the Model's Right Hand Side



Right side $y : \mathbb{R}^n \to \mathbb{R}^{|s|}$ mapped to `IVFunction`

$|s|$ member functions $f : \mathbb{R}^n \to \mathbb{R}$ offer optional *features*

The interfaces

- *hide* how y is really computed
- *hide* how derivatives are computed
- *allow* evaluation with different arithmetics

Interfacing the Solvers

Euler's Method ("Verified Approximation")

$$\mathbf{y}_k := \mathbf{y}_{k-1} + h \cdot f(\mathbf{y}_{k-1}, p)$$

→ Directly implemented in `UniVerMeC`

VNODE-LP (Verified)

1. Adapter for arithmetic compatibility to `UniVerMeC`
2. Implement VNODE's abstract `AD` interface

→ Both steps are possible thanks to VNODE's architecture

ValEnclA-IVP

ValEnclA-IVP
is coupled with libraries:

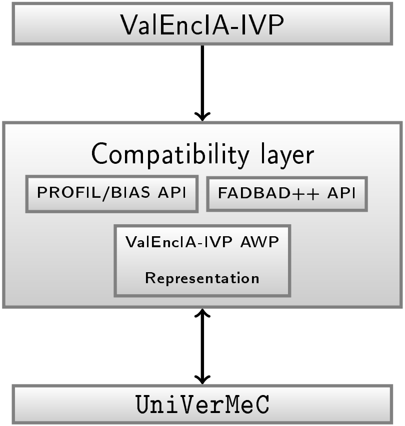
1. fadbad++
2. PROFIL/BIAS

$\rightarrow$ Compatibility layer necessary

Not thread-safe
(ValEnclA-IVP uses global variables)

In our version
no recompiling for each problem

decoupled from specific libraries

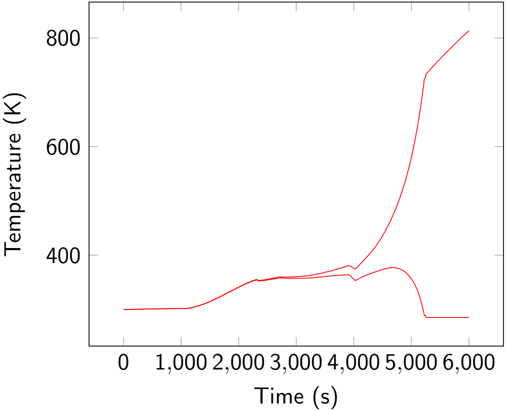


Solver Results

Problem
thermodynamics $1 \times 1 \times 1$

IC
$\theta(0) = 299.7053$
$p = (p_1, \ldots, p_{23})$
$u(t)$ from measurement file

Solver
ValEncIA (h=1)



Parameter Identification

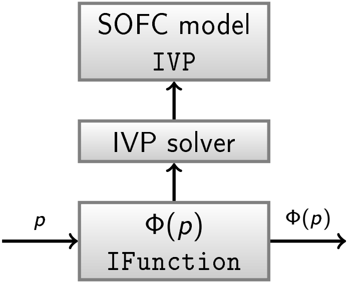
Goal

Parametrize the model for temperature in a robust and accurate way.

$$\Phi(p) = \sum_{k=1}^{T} (y(t_k, p) - y_m(t_k))^2$$

with

$p \rightarrow$ Parameters to identify
$y(t_k, p) \rightarrow$ Simulated temperature at time t
$y_m(t_k) \rightarrow$ Measured temperature at time t
$T \rightarrow$ Number of measurements (19963)
$t_{k-1} - t_k = 1s \rightarrow$ Step size 1s



→ Can be performed using different models/IVP solvers.
(Currently, we use Euler's method.)

Prolem Statement

Optimization problem

$$\min_{p \in \boldsymbol{p}_0} \Phi(p)$$

Bound constraint problem ($\boldsymbol{p}_0 \in \mathbb{IR}^6, \operatorname{wid} \boldsymbol{p}_0 = 2.0$)
Initial vector for $\boldsymbol{p}_0$ derived by floating-point methods

Difficulties

- Objective function is computationally expensive
- Calculating derivatives is really slow (even with `fadbad++`)
- Considerable overestimation

$\rightarrow$ Individual strategies for dealing with the problem (derivative free).

Consistent States

Consistent parameter vectors

A state vector $\boldsymbol{p}$ is consistent if $\forall t \in \{0, ..., T\}$:

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

with the worst-case measurement error $[\Delta y_m] = [-15, 15]$ holds.

Inconsistent parameter vectors

A state vector $\boldsymbol{p}$ is inconsistent if $\exists t \in \{0, ..., T\}$:

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

Branch & Bound

Basic pattern

1. $\boldsymbol{p} \leftarrow \mathcal{L}$
2. `Discard` $\boldsymbol{p}$ `if it is infeasible`
3. `Discard` $\boldsymbol{p}$ `if` $\underline{\Phi(\boldsymbol{p})} > \overline{D}$
4. `Contract` $\boldsymbol{p}$
5. `Update of` $\overline{D}$
6. `Add` $\boldsymbol{p}$ `to` $\mathcal{L}_{\text{final}}$ `if termination criteria are satisfied`
7. `Split` $\boldsymbol{p}$ `and add new boxes to` $\mathcal{L}$

Main data structures

- Two lists containing parts of the search space (boxes)
- Ordered working list $\mathcal{L}$
- Solution list $\mathcal{L}_{\text{final}}$

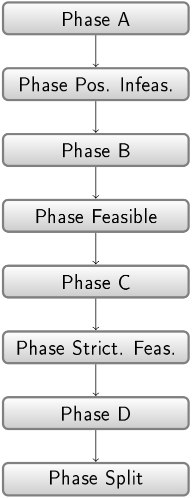
Termination criteria

- $\operatorname{wid} \boldsymbol{p} \leq \epsilon_p, \epsilon_p > 0$
- $\operatorname{wid}(\Phi(\boldsymbol{p})) \leq \epsilon_\Phi, \epsilon_\Phi > 0$

- Finds the minimum in the specified starting box
- Based on Hansen's interval optimization algorithm

Configurable Algorithm in UniVerMeC

Divided into phases



Configuration

Phase A
Midpoint Test

Phase Feasible
Update upper bound

Phase D
Linearization and pruning based on the consistency constraint

Phase Split
Calculate bound on $\Phi(\boldsymbol{p})$
Check for (in)consistent states

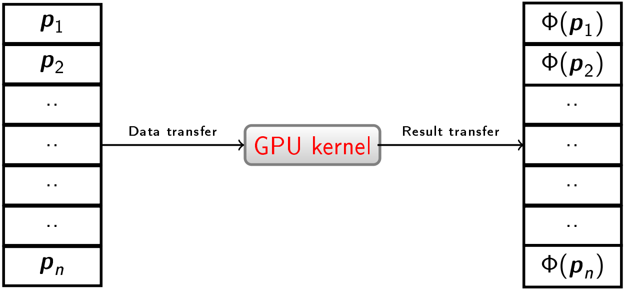
Use of GPU in Parameter Identification

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

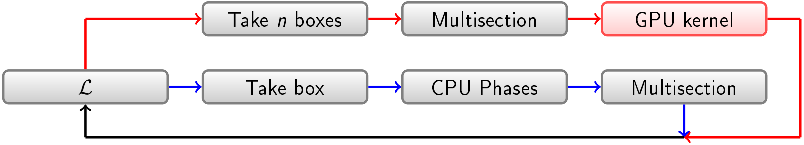
Value of $y(t_k, p)$ depends on $y(t_{k-1}, p)$

A single evaluation cannot be parallelized

But we can evaluate $\Phi(\boldsymbol{p})$ over different subdivision intervals in parallel!



Integration into the Optimization Algorithm



GPU and CPU run in parallel

Working list $\mathcal{L}$ is in host memory

One CPU thread feeds the GPU with data

Other CPU threads work normally

$\rightarrow$ Currently, only bounds on Φ are derived using the GPU

Quality Measure

Identified candidate intervals $\boldsymbol{p}$ are characterized by

$$e = \sqrt{\frac{\sum_{k=1}^{T} (y_{k-1} - y_m(t_k) + f(y_{k-1}, \mathrm{mid}(\boldsymbol{p})))^2}{T}} .$$

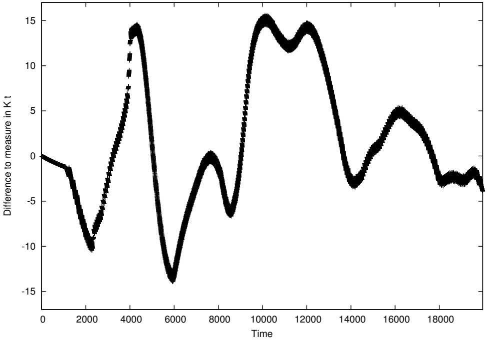
This measure is

- practice-motivated
- similar to the root mean square error measure

Candidate $\boldsymbol{p}$ with lowest e is chosen as solution, if there is no other way to proof that $\boldsymbol{p}$ is the optimum (minimum).

GPU Results ($1 \times 1 \times 1$)

Difference between measured and simulated temperature



	Error measure	Wall time
GPU	7.42944	$\approx 135s$
CPU (OpenMP)	7.68	$\approx 2491s$

Conclusions & Outlook

Conclusions

- An environment for SOFCs presented
- Flexibility wrt. different models and solvers implemented
- Model parameters identified by global optimization
- A speed up of 18 achieved for the $1 \times 1 \times 1$ model by GPU in the parameter identification algorithm (against the parallel CPU version)

Future Work

- Incorporate further solvers
- Allow easy addition of new models through a plugin based system
- Simulations for more complicated models

Thank You for Your Attention

Thank You for Your Attention!

References

E. Auer, S. Kiel, and A. Rauh.
Verified parameter identification for solid oxide fuel cells.
In *In Proc. of REC 2012*, 2012.
Accepted.

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

A. Rauh, T. Doetschel, E. Auer, and H. Aschemann.
Interval methods for control-oriented modeling of the thermal behavior of high-temperature fuel cell stacks.
In *In Proc. of SysID 2012*, 2012.
Accepted.