Journal of Computational Physics 229 (2010) 7260–7276

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Efficient algebraic multigrid for migration–diffusion–convection–reaction systems arising in electrochemical simulations

P. Thum [a,*], T. Clees [a], G. Weyns [b], G. Nelissen [b], J. Deconinck [b]

[a] Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
[b] Vrije Universiteit Brussel, Faculty of Engineering, Department of Electrotechnical Engineering, Building Z, Pleinlaan 2, B-1050 Brussels, Belgium

ARTICLE INFO

Article history:
Received 30 October 2009
Received in revised form 14 May 2010
Accepted 8 June 2010
Available online 15 June 2010

Keywords:
Point-based algebraic multigrid
Systems of partial differential equations
ILU-type smoothing
Electrochemical simulation
MITReM

ABSTRACT

The article discusses components and performance of an algebraic multigrid (AMG) preconditioner for the fully coupled multi-ion transport and reaction model (MITReM) with nonlinear boundary conditions, important for electrochemical modeling. The governing partial differential equations (PDEs) are discretized in space by a combined finite element and residual distribution method. Solution of the discrete system is obtained by means of a Newton-based nonlinear solver, and an AMG-preconditioned BICGSTAB Krylov linear solver. The presented AMG preconditioner is based on so-called point-based classical AMG. The linear solver is compared to a standard direct and several one-level iterative solvers for a range of geometries and chemical systems with scientific and industrial relevance. The results indicate that point-based AMG methods, carefully designed, are an attractive alternative to more commonly employed numerical methods for the simulation of complex electrochemical processes.

1. Introduction

Electrochemical processes play an increasingly important role in industry, for instance, for plating a large variety of parts, or for controlled etching by means of electrochemical corrosion when mechanical stress should not be applied to a part. The numerical solution of a model for electrochemical plating/corrosion processes involving complex physical, chemical and electrical phenomena is considered. In particular, we are interested in developing appropriate linear solvers since solving the linear systems is one of the most time-consuming parts of an electrochemical simulation.

In general, a complex interplay of the following phenomena takes place: electrochemical electrode kinetics, electrolyte hydrodynamics (movement of the electrolyte due to forced convection), ionic mass transport, gas evolution at the electrodes, and heat generation in the bulk of the electrolyte and at the electrode–electrolyte interfaces (see [12]).

From an engineering point of view, the main focus is on the current density and the layer thickness distribution of the deposed metal, which principally depend upon the following phenomena:

- Ohmic drop in the electrolyte solution
- concentrations of the species in the electrolyte
- cathodic polarization describing the deposition reactions at the electrodes and the plating efficiency
- plating tank configuration (geometry) including anode/cathode positioning, screens and current thieves

* Corresponding author.
E-mail address: peter.thum@scai.fraunhofer.de (P. Thum).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.06.011

- electric boundary conditions: imposed potentials and/or currents. It is clear that the current density in the electrolyte is depending on the potentials and currents imposed at the electrodes.
- chemical boundary conditions: imposed ion concentrations in the bulk electrolyte and chemical reactions.

The modeling approach that takes into account these phenomena is commonly denoted as the *multi-ion transport and reaction model* (MITReM), which consists of the transport equation for each ion, the electro-neutrality constraint or the Poisson equation, and the nonlinear Butler–Volmer boundary conditions describing the electrochemical processes at the electrodes. Typically, the total number of equations of the resulting system of partial differential equations (PDE system) is between 5 and 10.

Discretization in time and space as well as linearization produces a series of linear systems to be solved. Due to the high computational complexity of classical linear solvers, optimal solvers which provide (nearly) linear complexity are necessary in order to run realistic simulations in appropriate time. Algebraic multigrid methods (AMG) for systems, carefully designed, promise optimality and robustness and can easily be integrated into existing numerical simulation software. For the MITReM PDE system considered here, AMG methods have not been investigated so far though.

However, AMG methods have already proven to be efficient in related areas. For example, AMG has successfully been applied in the areas of semiconductor process simulation [4] and battery simulation [20] where diffusion–reaction systems have to be solved. Moreover, the MITReM system strongly resembles the systems to be solved in semiconductor physics for simulating the electrostatic behavior of devices such as diodes or transistors (semiconductor device simulation). For such equations, algebraic multigrid methods have also been applied with success (see [1,2,7,10]).

The main aim of this paper is to describe a robust and fast AMG approach for MITReM. The approach makes use of a point-based algebraic multigrid (PAMG) framework for systems of partial differential equations (see [1] for details on PAMG). We discuss how PAMG can be tailored to and further enhanced for solving electrochemical systems. A comparison of the performance of the corresponding solver, as integrated in the SAMG library [3], to the direct solver PARDISO [16] as well as to standard one-level iterative solvers is performed. Furthermore, we investigate several geometries and chemical setups with scientific and industrial relevance and show that the use of PAMG already results in a considerable reduction of the overall simulation time along with an advantageous memory requirement compared with the other solvers.

Section 2 gives a detailed description of the underlying mathematical problem. Section 3 introduces the solution approach employed. Section 4 describes the geometries for the numerical experiments, results of which are reported in Section 5. The paper closes in Section 6 with a summary and an outlook on potential improvements.

2. Mathematical model

In this section, we summarize the mathematical model considered, a nonlinear system of partial differential equations. In addition, we briefly indicate how existence and uniqueness of solutions can be analyzed.

An electrochemical system usually consists of ionic solutions bounded by electrodes, membranes and insulators. At the electrodes, electrochemical reactions take place. The flow of electrons in the outer circuit is converted into a flow of ions in the electrolyte. This means there will be a transfer of material and charge in the electrolyte. A mathematical model of the mass transfer of ions from the solution to the surface of the electrodes requires a description of the mobile ionic species, material balances, energy balances, current flow, electro-neutrality and fluid flow.

In the most general case, it is necessary to combine all these phenomena in order to calculate the mass and charge transfer in electrochemical reactors. As this is hardly possible in practice, simplifications to the model are needed. When dealing with dilute solutions, the cross diffusion and a different velocity for each species is not considered in the transport equations. Furthermore, it is assumed that the concentration of the species does not affect the velocity field. Hence, fluid mechanics can be studied separately from electrochemical analysis. Nevertheless, an involved system of nonlinear partial differential equations remains to be solved, as will be discussed below and is described in detail in [12].

Ionic mass transport and mass conservation are determined by the derivative of the ionic concentration c_k [mol m^{-3}] to the time t [s], which equals the divergence of the flux $\overline{N_k}$ of the ion [mol m^{-2} s^{-1}] and the production rate R_k [mol m^{-3} s^{-1}] of the ion

$$\frac{\partial c_k}{\partial t} = -\overline{\nabla} \cdot \overline{N}_k + R_k \tag{1}$$

with k being the index of the species.

The flux density of each ion in a solution is expressed by

$$\overline{N}_k = \bar{v}c_k - D_k\overline{\nabla}c_k - z_k u_k F c_k \overline{\nabla}U \tag{2}$$

with $\bar{v}$ being the velocity of the solution [m s^{-1}], D_k the diffusion coefficients of species k [m^2 s^{-1}], z_k the charge of species k, u_k the mobility of species k [m^2 mol J^{-1} s^{-1}], U the potential [V] and F the Faraday constant (=96485.3399 C mol^{-1}). The three terms in the expression are the results of three different processes: convection, diffusion and migration.

The phenomenon of particles carried away in a solution with velocity $\bar{v}$ is called convection. Convection depends on the velocity of the solution and the concentration of the ion k,

$$\overline{N}_{k,conv} = \bar{v}c_k. \tag{3}$$

Diffusion occurs when the concentration of an ion in the solution varies. The particles move from zones with a relatively high concentration to zones with a lower concentration. Diffusion is proportional to the gradient of the concentration,

$$\overline{N}_{k,diff} = -D_k \overline{\nabla} c_k. \tag{4}$$

Migration is caused by the electric field working on the charged ions,

$$\overline{N}_{k,migr} = -z_k u_k F c_k \overline{\nabla} U. \tag{5}$$

Migration is proportional to $z_k F c_k$, the gradient of the potential $\overline{\nabla} U$, and the mobility constant u_k. Note that the negative gradient of the electric potential at a point is equal to the electric field there.

Combining all the equations presented above leads to the system of equations describing mass transport in electrochemical systems:

$$\frac{\partial c_k}{\partial t} = -\bar{v}\overline{\nabla} \cdot c_k + \overline{\nabla} \cdot (D_k \overline{\nabla} c_k) + z_k u_k F \overline{\nabla} \cdot (c_k \overline{\nabla} U). \tag{6}$$

This equation expresses mass conservation concerning convection, diffusion and migration, also chemical reactions and electrode reactions are taken into account.

To close this system of equations, charge conservation has to be added,

$$\Delta U = -\frac{F}{\epsilon_0} \sum_{\text{ions } i} z_i c_i, \tag{7}$$

with ϵ_0 being the vacuum permittivity (=electric constant = $8.8541878 \times 10^{-12}$ m^{-3} kg^{-1} s^4 A^2).

The following boundary conditions are applied:

- at insulators and outlets

$$\overline{\nabla} U \cdot \bar{n} = 0 \quad \text{and} \quad \overline{\nabla} \cdot c_k = 0 \tag{8}$$

hold, with $\bar{n}$ being the direction normal to the boundary
- at the inlet

$$\overline{\nabla} U \cdot \bar{n} = 0 \quad \text{and} \quad c_k = c_{k,bulk} \tag{9}$$

are valid, and
- at the electrodes the Butler–Volmer boundary condition

$$i_n = i_0 \left[exp\left(\frac{\alpha_a F \eta}{RT}\right) - exp\left(\frac{-\alpha_c F \eta}{RT}\right) \right] \tag{10}$$

is imposed, with i_n being the current density [A m^{-2}], i_0 the exchange current density [A m^{-2}] depending on the composition of the solution close to the electrode, α_a and α_c the anodic and cathodic transfer coefficients, η the surface overpotential [V], R the universal gas constant (=8.3143 J mol^{-1} K^{-1}) and T the absolute temperature [K].

The resulting set of equations is a nonlinear system of partial differential equations, consisting of migration, diffusion, convection and reaction terms, and not trivially solvable – if at all. The existence and uniqueness of solutions, depending in particular on the boundary conditions and reactions involved, has to be checked and/or ensured first which is not trivial.

However, the system strongly resembles the so-called drift–diffusion(–convection–reaction) systems to be solved in semiconductor physics. Theoretical work on existence and (local) uniqueness of solutions of drift–diffusion systems has been investigated by several authors, see for instance [5,8,9,17] and references given therein. Lyapunov functions, maximum principle and compactness arguments are of importance in the corresponding proofs. An application to electrochemical systems shall be presented later.

3. Solution approach

The model is discretized using a combined finite element and residual distribution approach and linearized by means of a Newton–Raphson method. For the solution of the arising linear systems, the SAMG library [3] is used, choosing two appropriate point-based algebraic multigrid (PAMG) methods from its point-based framework. The results are compared to the one-level iterative solver BICGSTAB-BILU (0), the direct solver PARDISO [16] as well as the one-level iterative GMRES-ILU (10) solver (previously implemented in the simulation software used).

Before we describe the investigated linear solvers in detail, we explain the expressions points, unknowns and variables (from [1]) which will be used in the following. Variables are the entries of the solution vector v of the linear system $Av = b$ to be solved. In the example provided by Fig. 1, these are the small colored[1] circles. Points are depicted by big gray

[1] For interpretation of color in Fig. 1, the reader is referred to the web version of this article.

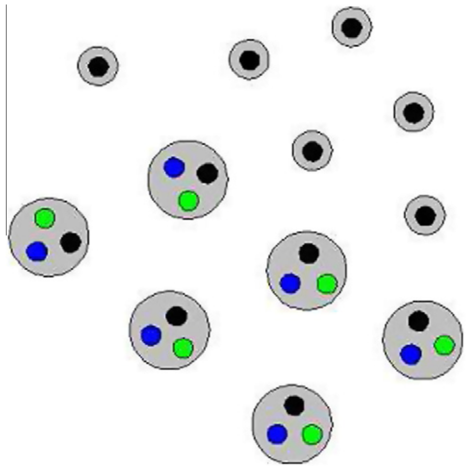


Fig. 1. Illustration of points, unknowns and variables.

circles. Points are small disjoint subsets of variables, the union of which gives the set of all variables. Here, points are identified with the grid points, at which the values of the solution vectors of the linear systems are computed. Unknown functions (in the following referred to as unknowns) represent physical functions to be solved for; in the case of MITReM the unknowns are the ion concentrations c_i and the potential U. In Fig. 1, variables belonging to the same unknown have got the same color.

In the following, we describe the discretization (Section 3.1) and introduce the linear solvers employed. In Section 3.2, we briefly introduce PAMG; Section 3.3 describes the direct solver PARDISO, and Section 3.4 the one-level iterative solvers BICGSTAB-BILU (0) and GMRES-ILU (10). Finally, in Section 3.5 we point out the usefulness of a Krylov method (here, BICGSTAB) to accelerate PAMG and also one-level methods such as BILU (0).

3.1. Discretization

In order to discretize the PDE system, a combined residual distribution and finite element method is used. The convection terms in the mass transport equations are discretized using the N-scheme of the residual distribution method [11,13]. The N-scheme ensures the positivity of the concentrations. Standard finite element discretization using linear basis functions is applied to the diffusion, migration and homogeneous reaction terms. All the numerical schemes provide at least second-order accuracy. The system is linearized via Newton–Raphson iteration with explicit calculation of the Jacobian matrices.

3.2. Point-based algebraic multigrid solver

Algebraic multigrid methods have their roots in geometric multigrid. Hence, we often speak of grids rather than levels, which would be more adequate due to the algebraic context. AMG methods commonly do not rely on the grids of the underlying discretized system. They only work with the resulting linear system and typically do not use extra information, besides a classification of unknowns and points.

The employed point-based AMG has been developed in [1] and is based on the variable-based AMG (VAMG) described in [18]. In the following sections, we only introduce the main components of the employed PAMG methods and their relation to components of VAMG.

Point-based algebraic multigrid was, in particular, developed for strongly coupled systems of PDEs which, among other things, also arise when solving MITReM. It is based on the observation, that for many PDE systems the different unknowns are discretized on (principally) the same grid, so that it appears quite natural to create the same hierarchy for all unknowns. PAMG only makes use of matrix entries and the variable-unknown and variable-point mappings.

3.2.1. Overview on components and cycling

Three main components define an algebraic multigrid method: smoothing, restriction/interpolation and a coarsest-level solver. First, so-called smoothing steps are applied. In this first phase, a relaxation method, an iterative one-level solver such as Gauss–Seidel or of ILU-type, is applied to the linear system. Relaxation methods shall smooth the high error frequencies. Then the residual r of our linear system can be restricted to the next coarser level. The residual is defined by

$$r_l = A_l\hat{u}_l - f_l \tag{11}$$

with l being the current level, $Au = f$ the system to be solved, and $\hat{u}_l$ the smoothed approximation on level l. The restricted residual r_{l+1} is given by

$$r_{l+1} = R_l r_l, \tag{12}$$

where R_l is the restriction operator. The points on the coarser level are chosen by means of purely algebraic criteria which take the smoothing behavior of the employed one-level iterative solver into account. Principally, the coarsening is (or shall be) only done in the direction of smooth error. Restriction and smoothing are repeated until we end up with a small linear system on the coarsest level which we can efficiently solve with a direct solver. The system to be solved within the coarse-grid correction step is given by

$$A_{l+1} v_{l+1} = r_{l+1} \tag{13}$$

with $A_{l+1} = R_l A_l P_l$ and v_{l+1} the correction to be solved for. The direct solver corrects the remaining error frequencies. In the next step, we interpolate the correction back,

$$v_l = P_l v_{l+1}, \tag{14}$$

and update the solution,

$$\overline{u_l} = \hat{u}_l + v_l. \tag{15}$$

The interpolation induces some error frequencies. Thus, some smoothing steps are applied. The interpolation, correction and smoothing is iterated until we reach the finest level again. This procedure is called a multigrid V-cycle. In order to reach a predefined accuracy, several of these V-cycles are performed.

Finally, the multigrid method is accelerated with BICGSTAB to obtain an overall converging PAMG approach which is also efficient in terms of memory requirement and CPU time (see also Section 3.5). On the coarsest level, we use the direct solver PARDISO.

3.2.2. Smoothing

At least formally any relaxation method could be used as a smoother in a point-based approach. However, a distinctly point-oriented smoothing often is a prerequisite for the success of a point-based approach. Such a smoother is often most appropriate for handling strong unknown cross-couplings and for producing algebraically smooth error which allows for a point-coarsening because it treats variables belonging to the same point simultaneously. This should especially be true for applications where the decisive unknown cross-couplings are located mostly on the block-diagonal of A (if considering point-wise ordering). Note that the relaxation does not have to converge, but it has to smooth the error. Examples for block-smoothers (point-based smoothers) are Block–Gauss–Seidel and BILU (0) (see below).

For the electrochemical systems considered here, BILU (0) has proven to be a good choice, whereas Block–Gauss–Seidel is not robust enough due to convergence problems.

3.2.3. Coarsening

In PAMG, the coarsening is done with the help of a so-called "primary" matrix P. There are many possibilities for its construction. The aim is to set up a primary matrix which reasonably represents the point-coupling structure of the original matrix. For ease of notation, we assume the matrix entries in the original matrix to be ordered point-wise,

$$A = \begin{pmatrix} A_{(1,1)} & A_{(1,2)} & \cdots & A_{(1,n-1)} & A_{(1,n)} \\ A_{(2,1)} & A_{(2,2)} & \cdots & A_{(2,n-1)} & A_{(2,n)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{(n-1,1)} & A_{(n-1,2)} & \cdots & A_{(n-1,n-1)} & A_{(n-1,n)} \\ A_{(n,1)} & A_{(n,2)} & \cdots & A_{(n,n-1)} & A_{(n,n)} \end{pmatrix}, \tag{16}$$

where n denotes the number of (grid-) points and $A(k,l)$ the so-called point-coupling matrices of dimension "number of unknowns in point k" $\times$ "number of unknowns in point l". The aim is now to assign a scalar value p_{kl} to each point-coupling matrix $A(k,l)$, which represents the coupling strength of point k to l, and thereby obtain the primary matrix P of dimension $n \times n$.

The amount of diffusion, migration and reaction in the electrochemical PDE system considered and thus in the respective linear systems differ strongly for different ion systems and over process time. Hence, basing the entries p_{kl} of the primary matrix on norms of the point-coupling matrices $A_{(k,l)}$ seems to be a reasonable first attempt. We define

$$\text{for } l \neq k \quad p_{kl} = -\|A_{(k,l)}\| \quad \text{and} \quad p_{kk} = -\sum_{l \neq k} p_{kl}, \tag{17}$$

where $\|\cdot\|$ denotes a suitable norm. This approach has indeed proven of value for the models considered. In order to save computational effort, we choose the maximum norm which works quite well for our numerical experiments (see also [1] for detailed discussions of suitable norms). However, depending on the application considered, it might be more suitable to choose the primary matrix based on the couplings of a specific unknown to itself. For instancen, if it shall be based on the first unknown, we define $p_{kl} = (A_{(k,l)})_{1,1}$. We will also take a brief look at this possibility in Section 5.

On the basis of a given criterion for the strength of couplings, C-points which exist at the coarser level, and F-points which only exist at the finer level are set up. We define the point P_k to be **strongly (negatively) coupled** to the point P_l ($k \neq l$) if

$$-p_{kl} \geqslant \epsilon_{str} max_{j\neq k} \|p_{kj}^{-}\| \quad \text{with} \quad p_{kj}^{-} = \begin{cases} p_{kj} & \text{if } p_{kj} < 0 \\ 0 & \text{else} \end{cases} \tag{18}$$

with some $\epsilon_{str} > 0$, a typical value being 0.25 (and never touched for our numerical experiments!). The coarsening information is then transferred to the variables of the matrix equation at hand, i.e. "copied" to each physical unknown.

3.2.4. Interpolation

A PAMG interpolation can be performed block-wise, based on a single (primary) unknown (SU) or based on multiple unknowns (MU), see [1]. As discussed there, the apparently natural block-wise interpolation has two major disadvantages. First, a straightforward extension of classical variable-based interpolation formulas to the block-wise case is mathematically not guaranteed to be defined and has to be adapted carefully. Second, due to necessary inversions of small matrices, it often needs more computational time and also more memory compared to the other variants, if it works at all.

As has been discussed in [1], the SU-interpolation is advantageous for drift–diffusion systems arising in semiconductor device simulation. Our numerical experiments clearly indicate the same to be true for MITReM systems. The SU-interpolation formulas are computed variable-wise, but are the "same" for the variables belonging to the same point, so that unknown cross-interpolation is avoided. The procedure is similar to the coarsening process. Namely, VAMG's setup of interpolation is applied to the primary matrix, and then the interpolation scheme is transferred, i.e. copied, to the unknowns.

Note that the primary matrix for interpolation and the one for coarsening are not necessarily identical. In some applications from semiconductor process simulation, for instance, it is beneficial to combine a coordinates-based coarsening with (either) SU-interpolation with a norm-based primary matrix (or MU-interpolation with weights modified according to distances). For our MITReM systems, we take the same primary matrix for both SU-interpolation and coarsening.

For simplicity, we only describe weights for the so-called "direct" interpolation scheme (see also [1,18]). This scheme uses interpolatory points $\mathscr{P}_k^p$:

$$\mathscr{P}_k^p \subseteq \mathscr{N}_k^p \cap \mathscr{C}^p, \tag{19}$$

where $\mathscr{N}_k^p$ denotes the neighbors of point k and $\mathscr{C}^p$ the set of coarse grid points. The superscript p denotes that we are working with points (sets of variables) rather than variables. Here, we define sets

$$\mathscr{N}_k^p := \{l \in \mathscr{V}^p | l \neq k \wedge p_{kl} \neq 0\} \quad \text{and} \quad \mathscr{P}_k^p := \mathscr{N}_k^p \cap C^p, \tag{20}$$

where $\mathscr{V}^p$ denotes the set of all points and p_{kl} the entries of the primary matrix P. The interpolation weights are then defined by

$$\forall l \in \mathscr{P}_k^p : \quad w_{kl}^p := -\frac{p_{kl}}{p_{kk}} \alpha_k^p \quad \text{with} \quad \alpha_k^p := \frac{\sum_{j\in\mathscr{N}_k^p} p_{kj}}{\sum_{j\in\mathscr{P}_k^p} p_{kj}}. \tag{21}$$

Note that all off-diagonal entries of P are non-positive. Hence, we have $w_{kl}^p > 0$.

In the case considered here, each point k and all $l \in \mathscr{N}_k^p$ are attached to the same set of unknowns (and not different subsets of the set of all unknowns). Then, each submatrix $W_{(k,l)}$ with $l \in \mathscr{N}_k^p$ is square and contains a positive interpolation weight,

$$W_{(k,l)} = \begin{bmatrix} w_{kl}^p & & 0 \\ & \ddots & \\ 0 & & w_{kl}^p \end{bmatrix}. \tag{22}$$

De facto, our PAMG approaches make use of the so-called "standard" interpolation, i.e. they also resolve, in particular, all strong F-to-F couplings in the construction of the interpolation operator. For a more general definition of SU-interpolation, see [1].

3.3. PARDISO

PARDISO [6] is a direct solver which is known to be an at least moderately stable and still quite robust solver (for instance, for semiconductor device simulation). PARDISO is also known to be quite efficient in terms of memory requirement and run time, as long as the problem is of moderate size and moderate numerical complexity. Its performance can suffer, for instance, in case of three-dimensional applications (not considered here). The basics of the solver are described in [15]. Comparisons of PARDISO to other direct solvers, like WSMP or MUMPS as well as SuperLU, can be found in [16]. We also use PARDISO due to the fact that it is easily available for commercial and academic use; it is implemented in the widespread Intel MKL library.

PARDISO was especially developed for the parallel usage (shared memory). However, it can also compete with other direct solvers, if its single-thread variant is used. This is the case here since our focus is convergence and robustness analysis, not parallelization topics. More comments on parallelization are made in the following sections.

3.4. One-level iterative solvers

The standard one-Level iterative solver employed for comparisons, BICGSTAB-BILU (0), is a BICGSTAB solver with Block-ILU (0) preconditioning which is taken from the SAMG library. The difference of BILU (0) to an ordinary ILU (0) is that each point-coupling matrix is considered as one element. ILU abbreviates an incomplete LU factorization, the "(0)" means without fill-in. For details, see e.g. [14].

The originally build-in solver of the considered simulator is a GMRES-ILU (10). The "(10)" denotes that an absolute fill-in of 10 was used. This solver is not taken from the SAMG library.

3.5. Acceleration of Iterative Solvers

A possibility to accelerate multigrid methods and one-level iterative solvers as, for instance, ILU-type methods is to use them as preconditioners for Krylov methods like CG (symmetric matrices), BICGSTAB or GMRES (nonsymmetric matrices), see e.g. [14].

Krylov methods are well known to accelerate iterative solvers in the case that the convergence of the solver is bounded by some eigenvalues which are not clustered with the remainder of the eigenvalues. That is, Krylov methods are able to "capture" the error frequencies belonging to these eigenvalues very well. The resulting convergence of the overall preconditioned Krylov method is usually by far better than the convergence of the stand-alone accelerators or one-level iterative solvers for the problem at hand. Quite often, accelerator or preconditioner alone do not converge, only their combination.

In the case of accelerated AMG methods, memory requirements are frequently lowered and CPU time is usually saved. In addition, more linear systems can be solved without adjusting the AMG method. Hence, with the help of acceleration, one usually also gains efficiency and robustness of the employed approach. Additionally, linear systems arising in many PDE systems or "tough" scalar problems cannot be solved by stand-alone AMG methods, see also Section 5.6. Therefore, we make use of Krylov acceleration for the multigrid as well as for the one-level iterative solvers employed.

4. Geometries

We compare the three linear solvers described above for a range of industrially relevant test cases. Section 4.1 describes a simple channel geometry which we investigate to gain insight in the basic solver behavior. Section 4.2 describes a crevice geometry which is particularly interesting for investigating corrosion processes. Finally, Section 4.3 describes a backward facing step geometry which is the most difficult to solve within the considered ones due to recirculating areas.

4.1. Channel

It is clear that new strategies should be tested on easy systems first. However, although the channel, see Fig. 2, is a very easy geometry, a lot of insight can be gained by investigating the solver behavior for this case. The convection and migration contributions are varying slightly in the geometry and are easy to manipulate by adapting the inlet velocity of the velocity field and the electrode potential. Hence, also the dominance of these phenomena can be varied very easily.

Moreover, in industrial applications, comparable reactors and geometries are used in, for example, plating applications. Typically, the electrolyte is then refreshed by the forced convection and has a high refresh rate. Hence, the channel test case is even of practical importance.

4.2. Crevice

The crevice geometry, see Fig. 3, is particularly interesting for simulations of electrochemical corrosion processes in materials where for example mechanical fatigue occurs. Inside the crack, a different chemical composition of the electrolyte can locally introduce corrosion.

The challenge in performing an electrochemical calculation of a crevice is that inside the crevice convection is very small or even negligible whereas migration can play an important role in this region, but outside the crevice the opposite is true.

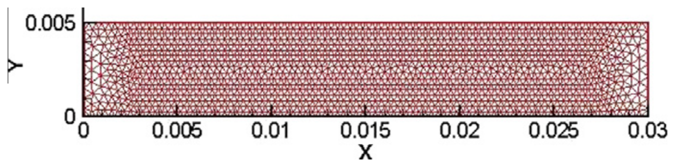


Fig. 2. Channel geometry.

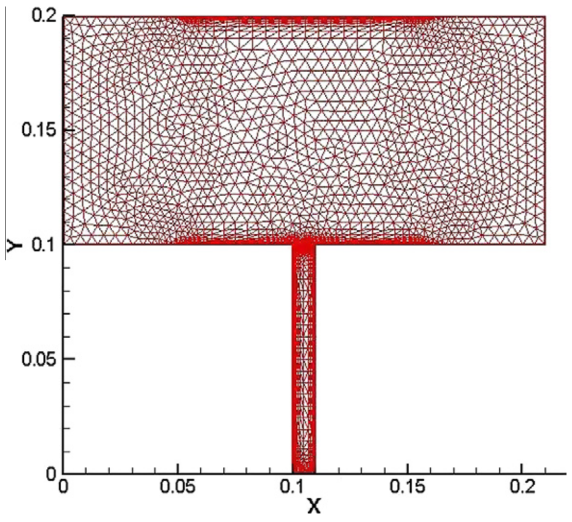


Fig. 3. Crevice geometry.

4.3. Backward facing step

As a final test case, a backward facing step is considered, see Fig. 4. The electrodes in this geometry are localized at the reattachment point of the fluid flow. The difficulty with this geometry is the recirculation region featuring cyclic convection.

5. Numerical experiments

In the following, we analyze the results of our experiments and examine the usefulness of the PAMG components chosen. We investigate two PAMG approaches which only differ in the choice of the primary matrix. The first approach makes use of a norm-based primary matrix. This approach will be denoted with PAMGn. The other approach uses a specified unknown coupling which represents the primary matrix. We denote this approach by PAMGp.

In order to analyze the efficiency of the components chosen, we compare numerical results to one-level iterative and direct solvers (see Section 5.2). The comparison will be performed for a range of industrially relevant model problems. The comparison to one-level iterative solvers is performed in order to investigate if the usually higher memory requirement of the multigrid approach will pay off in terms of a shorter run time. Additionally, a comparison to the one-level iterative solver corresponding to the smoother of the AMG method gives information whether the multigrid hierarchy pays off or not. Direct solvers are known to be robust and still reasonably fast. However, direct solvers usually have an enormous memory requirement and do not scale linearly in CPU time, especially for large linear systems. Hence, the comparison to the direct solver gives additional insight into the robustness and efficiency of the AMG approach.

Section 5.1 describes the model problems with its geometries, grid sizes and ion-models. The models have different physical properties and thus also different numerical properties. In Section 5.3, we perform a multi-level analysis in order to determine possible problems with the hierarchy, for instance, scaling problems on the coarser grids. We conduct a smoothing analysis in Section 5.4 and investigate the scalability in Section 5.5. We conclude with an efficiency analysis of the acceleration in Section 5.6.

Remark. In the following, BICGSTAB-BILU (0) is usually abbreviated with BILU, GMRES-ILU (10) with ILU, and BICGSTAB-PAMGn/PAMGp-BILU (0) with PAMGn/PAMGp, unless specific aspects concerning the components of the solvers (smoothing, acceleration) are discussed.

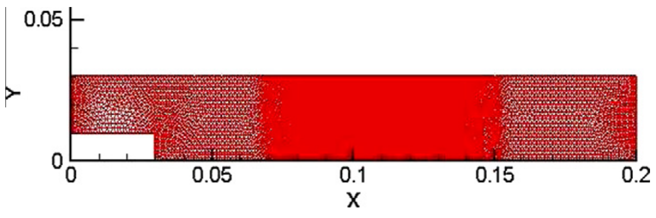


Fig. 4. Backward facing step geometry.

5.1. Model configuration

We use two grid sizes for each geometry in our numerical experiments. The geometries and associated grid sizes are listed in Table 1; see Section 4 for a detailed description of the geometries. The size of the big grids is the maximal possible size for which the direct solver PARDISO on one processor works for most ion systems. However, considering system 3 (see Table 2) which consists of seven unknowns (six ion concentrations plus the potential), the model size of BFS-BIG and CREVICE-BIG along with the numerical complexity exceeds PARDISO's capabilities. Note that the big grids are (more than) fine enough to resolve all important MITReM characteristics of the electrochemical systems considered here. Note that the bid grids are (still considerably) more resolved than grids typically used for industrial applications.

In Table 2, the ion systems considered are shown. The numbering of the ion systems represents the rank of difficulty of each system. Namely, system 1 is the easiest to solve. Note that systems 1 and 2 do not make use of homogeneous reactions, resulting in the absence of reactive terms $R_i \neq 0$ in Eq. (1) (see Section 5.1). System 3 is a realistic silver-ion model; the other systems are standard model cases for electrochemical simulators designed to reveal several difficulties, not restricted to, but particularly for linear solvers.

All models were simulated at 80% of the limiting current. The limiting current is approached as the rate of the charge-transfer process is increased by varying the potential. It is independent of the applied potential over a finite range and is usually evaluated by subtracting the appropriate residual current from the measured total current.

In Section 3.2, PAMGn and PAMGp have been introduced. Table 3 shows the share which each unknown-to-unknown coupling contributes to the entries of the primary matrix of PAMGn. More precisely, for system 1 the table shows a 83% share of the $(c(NO_3^-)$, potential) coupling on average. This means that the entries of the primary matrix in 83% of the cases of PAMGn represent the $(c(NO_3^-)$, potential) coupling. Such a dominating concentration-to-potential coupling is typical for ion systems without homogeneous reactions. If a system features very dominating homogeneous reactions, this is not likely the case anymore. In our cases, the couplings of the reacting ions dominate in more than 70% then.

5.2. Comparison to the direct and one-level iterative solvers

In this section, we compare the performance of the algebraic multigrid solver PAMGn to PAMGn (5), PAMGp, the one-level iterative solver BILU, the direct solver PARDISO, as well as the ILU one-level iterative solver originally implemented in the simulation software. PAMGn (5) denotes the same approach as PAMGn, except that the number of levels is restricted to 5 levels. PAMGp uses the concentration couplings of Ag^+ for systems 1 and 2 and $NaS_2O_3^-$ for system 3, respectively. The performance of PAMGp is nearly independent of the specific coupling used, as long as the respective ion is not involved in homogeneous reactions. The comparison between the methods is drawn in terms of CPU time, average convergence rate and memory requirement.

The average convergence rate R_{av} is defined by

$$R_{av} = \frac{1}{N}\sum_{i=1}^{N} R_i, \tag{23}$$

where N is the number of nonlinear iterations, accumulated over all Newton–Raphson steps, and R_i the convergence rate of the linear solver in the i-th nonlinear iteration. The convergence rate R_i is defined by

$$R_i = (r^i_{out}/r^i_{in})^{(1/cycle_i)}, \tag{24}$$

where $cycle_i$ denotes the number of linear iterations in the i-th nonlinear iteration, r^i_{in} is the residual of the initial approximation in the i-th nonlinear iteration, and r^i_{out} respectively of the final residual returned by the linear iterative solver.

5.2.1. CPU times

Analyzing Table 4, we see that PAMGn and PAMGp outperform ILU considerably. Hence, the PAMG methods are candidates for replacing the original solver. PAMGn is faster than PAMGp for the easy CHANNEL geometries. However, PAMGn does not converge for all ion systems and geometries, whereas PAMGn (5) and PAMGp do always converge. Additionally,

Table 1
Description of the grids, electrode positions and grid sizes.

Name	Points	Geometry
CHANNEL-SMALL	30123	Channel with length 300 and height 30
CHANNEL-BIG	49693	
BFS-SMALL	31868	Backward facing step with length 300, inlet height 10 and outlet height 30,
BFS-BIG	54624	step after 50, electrodes behind the recirculation area at 110
CREVICE-SMALL	36611	Channel with length 300 and height 30 and a crack at the bottom after 145
CREVICE-BIG	54011	with width 10 and height 100

Table 2
Description of the ion systems.

Ion system	Homogeneous reactions	Ions
1	–	$Ag^+,\ NO_3^-$
2	–	$Ag^+,\ NO_3^-,\ K+$
3	×	$NaS_2O_3^-,\ Na^+,\ S_2O_3^{2-},\ NO_3^-,\ AgS_2O_3^-,\ Ag(S_2O_3)_2^{3-}$

Table 3
Dominating couplings in the primary matrix on average for all matrices per ion system. (x,y) denotes coupling from x to y.

Ion system	Coupling	Amount of dominance (%)
1	$(c(NO_3^-)$, potential)	83
2	$(c(NO_3^-)$, potential)	91
3	$(c(S_2O_3^{2-}),\ c(AgS_2O_3^-))$	72

PAMGn (5) is faster than PAMGn in nearly all cases. PAMGp is very robust, but in several cases much too slow. These results indicate problems with the hierarchy for both PAMG approaches. We will examine this further in Sections 5.3 and 5.4.

Solving ion system 3 on the big geometries is not possible with the originally build-in ILU solver because of division-by-zero events. Due to this behavior and the non-competitive CPU times, we do not show any further results of ILU in the following sections. Instead, we use the one-level iterative solver BILU for comparisons to PAMG. In contrast to ILU, BILU exploits the naturally given structure of the PDE, namely the point couplings. Therefore, it is able to considerably outperform ILU. Another advantage of the BICGSTAB-BILU (0) solver is that the BILU (0) method corresponds to the smoother of the (BICGSTAB-) PAMG (-BILU (0)) methods used. Hence, a comparison additionally shows the benefit of the hierarchy of levels (grids). This is investigated in Section 5.4.

The direct solver PARDISO shows very competitive CPU times for ion systems 1 and 2. However, considering ion system 3, PARDISO loses its supremacy. Whereas ion systems 1 and 2 do not feature homogeneous reactions, system 3 incorporates them. This results in a different coupling structure (cf. Section 5.1). The different numerical properties do not affect the convergence of PAMGn (see Table 5) but they seem to affect the direct solver severely. The relatively good performance of PARDISO for the still small systems considered here might additionally be caused by the problem size. The setup of the multi-level hierarchy, which is performed in a setup phase prior to the cycling phase, is costly then.

5.2.2. Memory requirements

The memory requirements (and convergence rates) shown in Table 5 are typical for the employed methods. Memory requirements of the direct solver considerably grow with problem size – and "physical complexity". This restricts its appli-

Table 4
CPU times in seconds at 80% of the limiting current. PAR = PARDISO. div = the Newton method diverged. mem = solver refused to work (considerably more than 4 GBytes memory necessary). Numbers in colons denote that the Newton method does not reach its stopping criterion after 50 nonlinear iterations. No. = ion system number.

Geometry	No.	PAR.	BILU	PAMGn (5)	PAMGn	PAMGp	ILU
CHANNEL-SMALL	1	177	344	182	**176**	295	2005
	2	159	429	**141**	169	430	2211
	3	531	1572	**284**	324	824	11292
CHANNEL-BIG	1	**321**	625	375	389	642	3677
	2	**274**	738	308	319	871	3560
	3	830	3055	**582**	663	1691	mem
BFS SMALL	1	**169**	980	383	412	1750	1971
	2	**174**	1159	437	770	661	2489
	3	**467**	5514	676	960	504	10662
BFS BIG	1	**320**	1825	1011	div	2951	3645
	2	**348**	2135	898	1974	1077	4665
	3	mem	'5102'	2509	div	**770**	mem
CREVICE-SMALL	1	**281**	'1124'	501	607	361	div
	2	**215**	752	228	275	490	4290
	3	460	'2066'	**331**	934	417	17856
CREVICE BIG	1	**435**	1631	780	div	716	div
	2	356	1127	**331**	538	1036	6311
	3	mem	'6366'	**498**	div	1414	mem

cability for big and/or complicated problems considerably. In fact, due to the high memory requirement, the direct solver does not work for ion system 3 on the BFS-BIG and CREVICE-BIG geometries on a standard PC. PAMG's memory requirement is typically slightly more than twice as high as for the one-level iterative solver BILU, thus the results in the table are as expected. Note that, for analysis purposes, only "standard" AMG coarsening is used, not "aggressive" variants which usually reduce memory requirements considerably (from a factor of about 2.3 to about 1.3 compared to BILU) without sacrificing overall run time.

Table 5 also shows the convergence rates for the one-level iterative and the multi-level methods. The convergence rates of BILU are far worse than the ones of the multigrid methods, which is usually expected for one-level iterative solvers and which also demonstrates the complexity of the problem to be solved. Note that the convergence rates of PAMGp are worse than the ones of PAMGn. The method is nonetheless more robust since it works for all considered geometries and ion systems.

5.2.3. Appropriate compromises for an industrial environment

As long as PAMG's convergence rate is satisfactory (around or below 0.3, as a guideline), it outperforms the other methods considerably, especially for realistic ion systems with homogeneous reactions and/or larger systems/geometries. To be more specific, compared to the one-level methods BILU and especially ILU, PAMGn and PAMGn (5) are much faster while requiring only a bit more than twice the memory of the one-level method (just using "standard" AMG coarsening!). Additionally, except for the BFS geometry, PAMGn (5)'s run time is as good as the one of PARDISO for systems 1 and 2. For the realistic ion system 3, PAMGn (5) clearly outperforms PARDISO. PAMGp should be seen as an alternative approach if PAMGn (5) does not work. It is the most robust method, however not very competitive with respect to run time.

The results indicate the efficiency of the PAMG methods employed, but also some weaknesses, which will be analyzed in subsequent sections, providing ideas for further enhancement. Currently, if only CPU times are considered, the results indicate that a solver switching mechanism should be implemented, with the fast PAMGn (5) and PARDISO and the robust fall-back option PAMGp.

Note that using parallel versions of PAMG or PARDISO does not change the situation considerably. Both PARDISO and PAMG can benefit from (OpenMP) threading (shared memory), PAMG can additionally benefit from MPI parallelization or even hybrid MPI/OpenMP parallelization. Corresponding versions are available. In addition, MPI-parallel direct solvers exist. However, both multigrid methods with ILU-type smoothers and direct solvers suffer from the fact that (incomplete) LU factorizations are highly sequential. Compromises in terms of numerical stability, load balancing and communication, overall run time and memory requirements have to be found.

However, CPU time is by far not the only criterion, particularly if not just one simulation run is considered. In practice, simulation shall help in enhancing the performance of electrochemical machining processes. For the purpose of parameter optimization, for instance, an ensemble of simulation runs is needed. Depending on the optimization strategy, a collection of (several or many) such runs can be performed completely in parallel. This allows for a "trivial parallelization" and hence a much better use of a machine (or cluster) with several/many cores, provided that the accumulated memory requirements of the collection does not exceed the capabilities of the machine. At this point, memory requirements of the solvers come into discussion again. The PAMG solvers require a reasonable amount of memory and far less than the direct solver, the memory requirement of which rises considerably with problem size. Therefore, a good compromise for the "optimization scenario" described is a solver switching mechanism including PAMGn (5) or even m-adaptive PAMGn (m) and PAMGp, all with PARDISO as a coarsest-level solver only.

5.3. Multi-level analysis

In this section, we analyze the performance of the PAMG approaches with respect to the number of levels employed. The aim is to identify problems concerning smoothing, coarsening and interpolation which may be the reason for PAMGn's rather poor performance on complicated geometries like the backward facing step, and PAMGp's unsatisfactory convergence rates.

Table 5
Results for CHANNEL-SMALL and BFS-BIG at 80% of the limiting current. R_{av} = average convergence rate. div = the Newton method diverged. mem = solver refused to work (considerably more than 4 GBytes memory necessary).

Ion system	PARDISO	BILU		PAMGn		PAMGp	
	Memory (MB)	R_{av}	Memory (MB)	R_{av}	Memory (MB)	R_{av}	Memory (MB)
CHANNEL-SMALL							
1	214	0.85	65	0.20	156	0.69	147
2	735	0.91	92	0.22	216	0.76	209
3	1721	0.99	207	0.23	462	0.86	472
BFS-BIG							
1	397	0.89	194	div	283	0.69	270
2	638	0.94	275	0.59	393	0.81	382
3	mem	0.98	379	div	844	0.91	865

PT Convergence rates of PAMGn are shown in Table 6. The solver on the respective coarsest level of the level-restricted PAMG is PARDISO, if it works, else BILU (0)-BICGSTAB. Note that, for the matrices of the two-level methods, PARDISO fails due to pivoting and/or stability problems.

We see a significant increase of the convergence rates after level 4 in case of systems 1 and 2 for the BFS geometry. In case of system 3, this increase already appears after level 3. BFS-SMALL and the crevice geometries behave similarly. PAMGp instead always shows a huge increase between levels 2 and 3. These significant increases in the convergence rates point to problem with the hierarchy on coarser levels. Note that for smaller grid sizes and geometries with a simple convection field the impact of the increase in the convergence rate is rather bounded for PAMGn. However, even for the channel geometry, convergence rates suffer.

Table 7 shows the two most dominating couplings of the coarse-level matrices for PAMGn's coarsening. Considering systems 1 and 2, one of the concentration-to-potential couplings is dominating on the first levels, its amount in the primary matrix tends to be constant. The c_i-to-c_i couplings, however, increase on the coarser levels. Significant increases occur at level 6 for both systems. This might also explain the bad convergence rate for the PAMGn (6) approach, as shown in Table 6. Considering system 3, the dominant $(c(S_2O_3^{2-}),\ c(AgS_2O_3^-))$ coupling caused by homogeneous reactions relatively weakens and the $(c(AgS_2O_3^-),\ c(AgS_2O_3^-))$ coupling becomes more dominant. This effect develops uniformly for increasingly coarser grids.

We also observe that PAMGn's BILU (0) smoothing steps "visually" diverge (that is, increase the residuals) for systems 1 and 2 on level 5 in case of the BFS and CREVICE geometries. Technically, BILU (0) diverges in every case. Solving the coarse level matrices with BILU (0)-BICGSTAB, instead, works for all coarse level matrices for systems 1 and 2, despite of the one for level 5 on BFS-BIG. The poor solving performance of BILU (0)-BICGSTAB as well as the apparently insufficient smoothing behavior of BILU (0) might be caused by the change in the matrix structure from level 5 to 6, as shown in Table 7. Considering system 3, the BILU (0)-BICGSTAB solver even stagnates starting from level 5 which might also be caused by a strong c_i-to-c_i coupling.

Considering the coarse level grids shown in Fig. 5, PAMGp tends to produce a coarsening which is highly related to the convection, see Fig. 6. To be more specific, many points on the coarser levels are chosen in the y-direction while less points are chosen in the x-direction. PAMGn instead coarsens the mesh very regularly. Neither seems to be the best solution.

Table 6
Average convergence rates (R_{av}) for BFS-BIG with different number of levels at 80% of the limiting current.

Method	Ion system	Level							
		2	3	4	5	6	7	8	9
PAMGn	1	0.12	0.13	0.16	0.27	0.56[a]	0.74[a]	–	–
	2	0.09	0.12	0.18	0.27	0.36	0.62	0.69	–
	3	0.14	0.27	0.49	0.55	0.80	0.89[a]	–	–
PAMGp	1	0.11	0.37	0.46	0.55	0.62	0.67	0.69	0.69
	2	0.10	0.50	0.58	0.67	0.73	0.79	0.80	0.81
	3	0.15	0.43	0.59	0.72	0.84	0.90	0.90	0.91

[a] For these methods, the Newton process diverged.

Table 7
Dominant couplings, grid sizes for BFS-BIG with increasing number of levels of PAMGn for the 5th linear system at 80% of the limiting current. P = potential. $a = NO_3^-,\ b = Ag^+,\ c = S_2O_3^{2-},\ d = AgS_2O_3^-$ for the concentrations c_i.

System		Level					
		1	2	3	4	5	6
1	Coup.	(c_a,P)	(c_a,P)	(c_a,P)	(c_a,P)	(c_a,P)	(c_a,P)
	%	88	87	82	81	84	50
	Coup.	(c_a,c_a)	(c_a,c_a)	(c_a,c_a)	(c_a,c_a)	(c_a,c_a)	(c_a,c_a)
	%	7	9	10	9	7	19
	Size	163874	56675	17507	5522	1730	257
2	Coup.	(c_a,P)	(c_a,P)	2,4	(c_a,P)	(c_a,P)	(c_a,P)
	%	95	97	95	92	93	79
	Coup.	(P,c_b)	(P,c_b)	(c_b,c_b)	(c_b,c_b)	(c_b,c_b)	(c_a,c_a)
	%	3	2	1	3	3	7
	Size	218498	76040	24840	8132	2848	488
3	Coup.	(c_c,c_d)	(c_c,c_d)	(c_c,c_d)	(c_c,c_d)	(c_c,c_d)	(c_d,c_d)
	%	77	61	56	51	45	50
	Coup.	(c_d,c_d)	(c_d,c_d)	(c_d,c_d)	(c_d,c_d)	(c_d,c_d)	(c_c,c_d)
	%	14	33	36	41	45	37
	Size	382370	129983	41552	13713	4669	1001

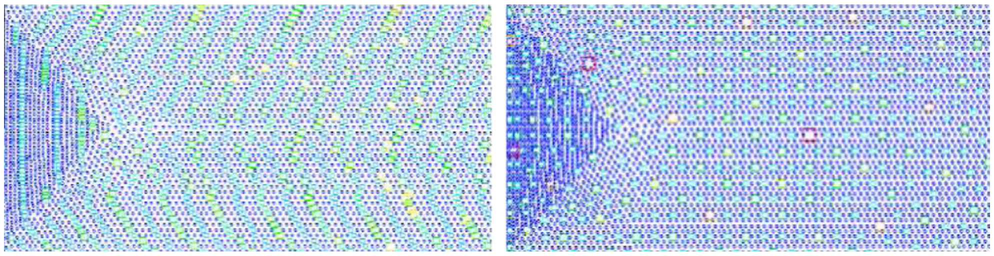


Fig. 5. Coarse levels of PAMGp (left) and PAMGn (right). The bigger the box, the coarser the level.

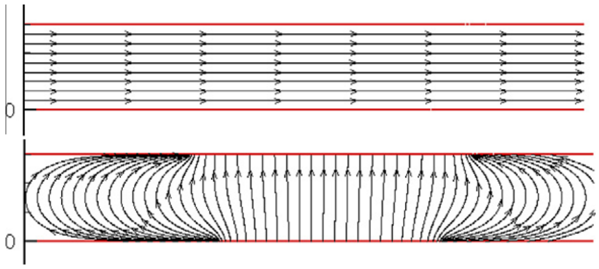


Fig. 6. Convection field (top), migration field (bottom) for the CHANNEL-SMALL after the simulation run.

PAMGp neglects the migration direction, which is not present in its primary matrix. PAMGn's coarsening apparently concentrates on the potential-to-concentration couplings, which do not resolve the convection direction.

By means of the above analysis, the observed problems within the hierarchy cannot be distinctly ascribed to either the smoother or the coarse-level correction. Their interplay is obviously insufficient. Section 5.2 shows that PAMG can principally work for electrochemical systems. However, due to the combination of diffusion, convection, migration and reaction terms, special attention should first be paid to the smoother since a strong dependence on the ordering of the variables (or points) has to be expected here. This dependence might be seen as a disadvantage. Still, it can be analyzed, and results might indicate appropriate corrections more easily than corrections of the coarse-level correction. It might hence turn out to be of advantage.

5.4. Smoothing analysis continued

The choice of a good smoother is a prerequisite for an efficient multigrid method. The smoother should reduce the "high" error frequencies and, in particular, be cheap in terms of memory requirement.

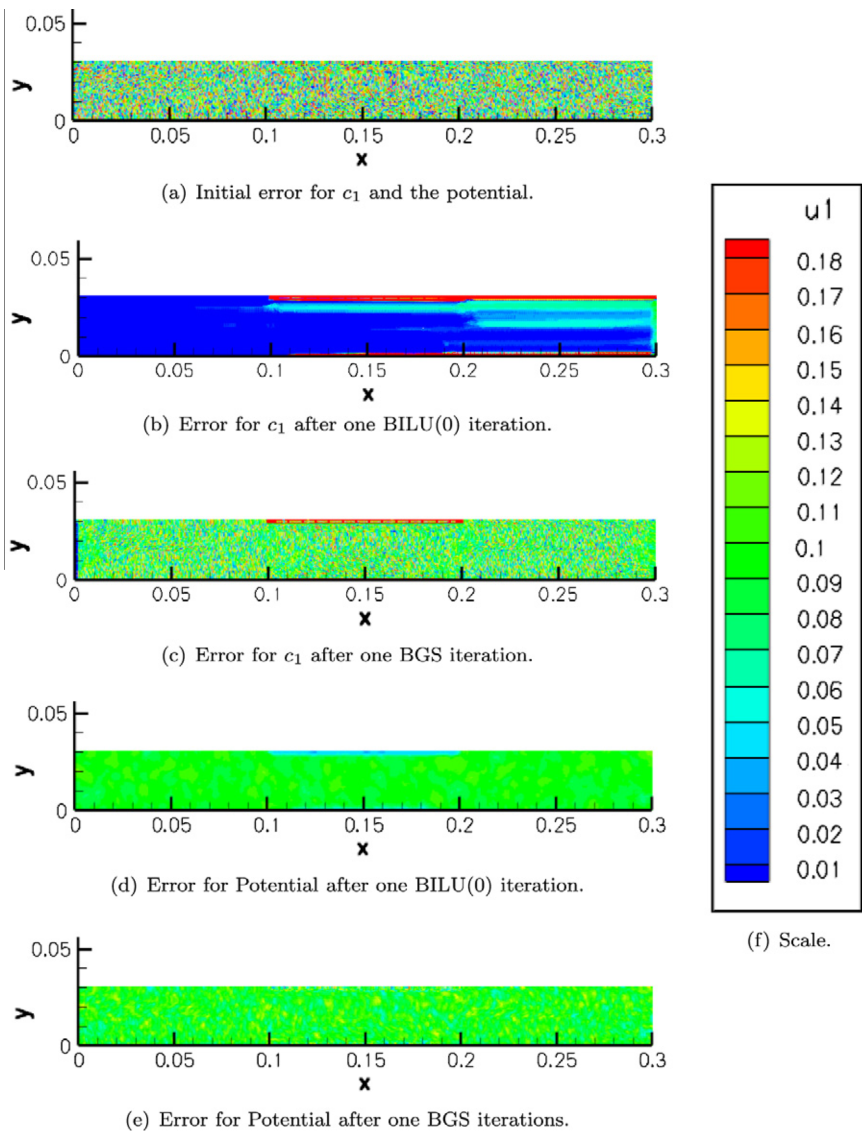
It is known that, considering "pure" convection–diffusion systems, the smoother has to capture the direction of convection appropriately. This is still a nontrivial task in practice, and particularly in the case of MITReM not that obvious. A reason being that, in addition to the convective term in our PDE system, we have to deal with migration which can be larger than, as large as or smaller than convection depending on the concrete electrochemical system (ions etc.), the geometry, and also the stage (time step) of the electrochemical process.

Migration resembles a combination of convection and anisotropic diffusion, but in different directions compared to the convective term itself. An "optimal" smoother shall thus handle all these at least partly conflicting terms and induced directions appropriately. Furthermore, for many geometries of practical relevance like the geometries presented here, the migration field can even be (more or less) orthogonal to the direction of convection (see Fig. 6). Additionally, the migration field develops during the simulation process. Therefore, the numerical behavior of the matrices changes with time step due to the evolving migration field which especially makes it difficult to detect the migration field correctly during the first time steps.

We make use of the BILU (0) smoother, due to the fact that it is the only smoother found so far leading to a converging approach along with a reasonable CPU time. Note again that the smoothers should smooth the error, and do not have to converge. Usually, stand-alone ILU-type methods, that is, without any accelerator, diverge. Luckily, in our case, the BILU (0) does not diverge for the fine-level matrix. Hence, we are able to plot the error after applying one smoothing step (see Fig. 7).

Considering the smoothing behavior for the concentration coupling (c_1 depicted exemplarily) shown in Fig. 7, the BILU (0) smoother works very well compared to the smoothing behavior of the Block–Gauss–Seidel (BGS) smoother for the fine-level matrix. We hardly see any smoothing behavior for the BGS smoother, the error maintains more or less its unsmooth shape. However, the error is reduced by the BGS method as well. Note, in the case of BILU, the error reduction proceeds in the direction of convection.

Fig. 7 shows the impact of the BILU (0) smoother on the potential. In particular, we see a much better smoothing behavior than for the concentrations. Note that the potential equation is very similar to diffusion and thus much easier to smooth than the transport equations for the concentrations, which include the convective, reactive, migrative and diffusive terms directly.



(a) Initial error for c_1 and the potential.

(b) Error for c_1 after one BILU(0) iteration.

(c) Error for c_1 after one BGS iteration.

(d) Error for Potential after one BILU(0) iteration.

(e) Error for Potential after one BGS iterations.

(f) Scale.

Fig. 7. Smoothing behavior of the BILU (0) and BGS smoother for c_1 and the potential of system 1 at 80% of the limiting current on the CHANNEL-SMALL geometry.

On the finest level, convection is the dominating term – with respect to magnitude of the contribution to the matrix entries – compared to migration and diffusion in the absence of homogeneous reactions. The ordering of the points, as produced by the grid generator employed, is determined by means of a reverse Cuthill-McKee algorithm, which seems to be advantageous for at least the fine-level matrix here. However, in Section 5.3, we already mentioned that, starting at some coarse level (4 or 5), the BILU (0) smoother faces problems for PAMGn. The above statements do not hold anymore then. In particular, the dominating direction seems not to be determined by the convection field anymore. Table 7 gives a first indication that not only dominating terms, but also directions might be level-dependent. Moreover, several equally large couplings alogn with different directions might be present.

It is tempting now to create orderings of the points/variables that follow the respective dominating direction(s). In case of more than one dominating direction (migration and convection similarly dominating, but in different directions), alternating smoothing approaches might help. In fact, current work is concerned with physically inspired algebraic orderings and alternating smoothing.

5.5. Scalability

Multigrid methods are famous for their *h*-independence in many applications. However, this is often difficult to prove theoretically. We will now investigate practically whether the multigrid approach maintains its convergence factor for increasing grid sizes, and whether the run time will only grow proportionally to the problem size, which means that the solver is scalable. We define the scaling factor σ for a fixed geometry and ion system (and, hence, also a fixed number of unknowns) as follows:

$$\sigma := \frac{\text{cpu(BIG)}}{\text{cpu(SMALL)}} \frac{n(\text{SMALL})}{n(\text{BIG})}, \tag{25}$$

where BIG and SMALL indicate the two grid sizes, cpu($*$) the corresponding CPU times from Table 4 and $n(*)$ the number of points displayed in Table 1.

Considering the scaling factors of PARDISO shown in Table 8, we see the typical nonlinear behavior of direct solvers. The direct solver is not able to solve system 3 on CREVICE-BIG since it runs out of memory on a machine equipped with 4 GB of memory.

PAMGn shows also a nonlinear scaling factor. Additionally, the Newton process diverges for this method in case of complicated geometries with big grid sizes. Here, the nonlinear scaling is most likely caused by problems with the hierarchy on coarser levels, already discussed in Section 5.3. These problems also seem to be the reason for the divergence of the method in the case of the CREVICE-BIG geometry. Hence, problems on coarser levels lead to increasing convergence rates for increasing grid sizes which then lead to unsatisfactory scaling factors. PAMGp only shows a linear scaling for system 3. The nonlinear scaling for the other systems is also caused by the worse convergence for the bigger grid.

Usually, we would expect only slight changes in the convergence rate if refining the grid and letting all other input data unchanged. Therefore, we assume that if we are able to remedy the problems due to the multigrid hierarchy, we will end up with a method of linear complexity. As mentioned in Section 5.3, improving the smoother – more specifically, ordering(s) per level plus alternating smoothing approaches – should help in improving robustness and scalability.

5.6. *The efficiency of acceleration*

A representative spectrum of the eigenvalues of the iteration matrices is shown in Fig. 8. We see that there are some eigenvalues without imaginary part which are very large. All other eigenvalues are clustered in the positive hemisphere around zero. Krylov methods like CG (only for symmetric matrices), BICGSTAB or GMRES are able to handle such large real

Table 8
Scaling factors σ and convergence rates for the CREVICE geometries. ($\times$): Method did not converge/work for the big geometry.

Ion system	PARDISO	PAMGn			PAMGp		
			R_{av}			R_{av}	
	Scaling	Scaling	Small	Big	Scaling	Small	Big
1	1.05	$\times$	0.30	$\times$	1.14	0.67	0.72
2	1.12	1.33	0.31	0.44	1.10	0.76	0.79
3	$\times$	$\times$	0.70	$\times$	1.04	0.83	0.82

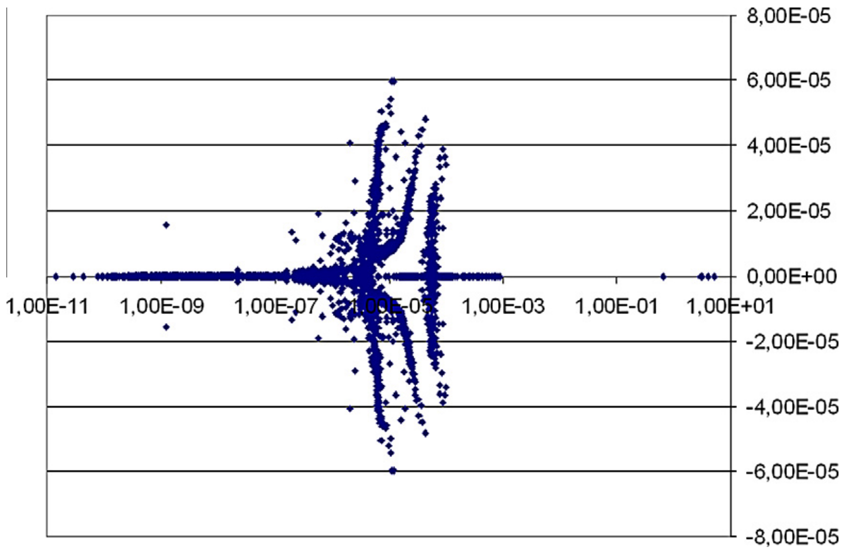


Fig. 8. Typical eigenvalue distribution of an iteration matrix for the discretized electrochemical system.

Table 9
PAMGn results for BFS-SMALL. div = the Newton method diverged. T = CPU time in seconds.

Ion system	Without accel.		GMRES accel.		BICGSTAB accel.	
	R_{av}	T	R_{av}	T	R_{av}	T
1	div	–	0.50	437	0.23	412
2	div	–	0.81	11201	0.45	770
3	div	–	0.78	1678	0.43	960

eigenvalues. Hence, preconditioned Krylov methods help to accelerate the preconditioner used and can, therefore, be called accelerators.

Table 9 shows that the non-accelerated multigrid method PAMGn-BILU (0) does not converge for BFS-SMALL. Analyzing this method on the simple channel geometry, we observe very bad convergence factors around 0.9. Therefore, it is obvious that the higher memory requirement and additional computational effort for the acceleration pays off in terms of robustness, overall run time and convergence, as we expect from the eigenvalue spectrum.

We decide to use BICGSTAB rather than GMRES acceleration due to the better overall convergence behavior, lower memory requirements and better run time, cf. Table 9. Note, however, that average convergence rates do neither directly show problems arising with several matrices needing a considerable part of the overall run time – as is the case for GMRES for ion system 2 – nor directly measure computational cost; GMRES performs one cycle of the preconditioner, but BICGSTAB two.

6. Conclusion and outlook

We investigated two PAMG preconditioners to solve linear systems arising in electrochemical applications modeled by means of MITReM. This is, to our knowledge, the first time that an AMG method has (successfully) been applied in this context. We motivated the use of the AMG components chosen and compared the performance to a direct and one-level iterative solvers for a range of geometries and ion systems with high importance to industry.

In comparison to the one-level iterative methods, the best out of the PAMG methods always turns out to be far more efficient. PAMG's efficiency originates from its better convergence rates, lower CPU time, and only slightly higher memory requirements than the one-level iterative solvers. The run time compared to the direct solver PARDISO is, depending on the geometry and ion system, better or very similar. However, PAMG is far more memory efficient than PARDISO, due to PARDISO's high memory requirement. The high memory requirement of direct solvers is often prohibitive for today's industrial applications and computer equipment. In fact, we demonstrated that the direct solver PARDISO could not solve linear systems with more than 350 k variables due to insufficient memory on a standard 4 GB machine. In addition, PARDISO faces problems in solving second-level matrices produced by PAMG due to pivoting and/or stability issues.

The two PAMG approaches, PAMGn and PAMGp, differ only in their coarsening strategy. PAMGn/PAMGn (5) shows fast convergence and good CPU times for many cases. However, this method has a lack of stability when it comes to complicated geometries and big grid sizes. PAMGp, instead, shows a very robust behavior. In fact is is the only solver which works for all test cases considered. However, PAMGp is not competitive with respect to CPU time. Both methods suffer from problems within their hierarchy.

Future work will concentrate on finding a PAMG approach which is robust and fast for all geometries. Therefore, one focus will be developing appropriate smoothing procedures capturing, per level, the changing dominating "characteristic" directions reflected by the matrix on that level. Reordering strategies, inspired by physics, but performed algebraically on each level before applying the smoother, is one promising way. In addition, alternating smoothing in order to cope with more than one dominating direction might be of help. Alternating smoothers are applied in geometric multigrid methods in case of convection–diffusion equations, for instance. The transfer of corresponding ideas from geometric multigrid to algebraic multigrid is not straightforward. First steps, however, seem to be promising. Related work shall be published soon.

Another focus is to find better coarsening and interpolation strategies. The currently employed coarsening and interpolation are generalizations of the interpolation for scalar applications. However, there are many parameters and possibilities to vary them. A first idea might be to employ knowledge about dominating "characteristic" directions for both smoothing and coarsening.

Two different, but also important topics of future work will be the parallelization of all components of the simulation software and its extension to three-dimensional applications.

References

[1] T. Clees, AMG Strategies for PDE Systems with Applications in Industrial Semiconductor Simulation, PhD Thesis, Dissertation, University of Cologne, Germany, November 2004, ISBN: 3-8322-4497-2.
[2] T. Clees, K. Stüben, Algebraic multigrid for industrial semiconductor device simulation, in: E. Bänsch (Ed.), Challenges in Scientific Computing – CISC 2002, Proceedings of the Conference "Challenges in Scientific Computing", Berlin, October 2–5, 2002, Lecture Notes in Computational Science and Engineering, vol. 35, Springer, Berlin, 2003, pp. 110–130.
[3] Fraunhofer Institute SCAI, SAMG User's Manual. Version 22c, 2005 (For the tests, an extended SAMG v.23a (2008) has been used), <www.scai.fraunhofer.de/samg.html>.

[4] T. Füllenbach, K. Stüben, Algebraic multigrid for selected PDE systems, in: Elliptic and Parabolic Problems, Rolduc and Gaeta 2001, Proceedings of the Fourth European Conference, World Scientific, London, 2002, pp. 399–410.
[5] H. Gajewski, K. Gärtner, On the discretization of van roosbroeck's equations with magnetic field, Z. Angew. Math. Mech. (ZAMM) 76 (1996) 247–264.
[6] Intel, Intel Math Kernel Library, version 10 – Reference Manual.
[7] Paul T. Lin, John N. Shadid, Marzio Sala, Raymond S. Tuminaro, Gary L. Hennigan, Robert J. Hoekstra, Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling, J. Comput. Phys. 228 (17) (2009) 6250–6267.
[8] P.A. Markowich, The Stationary Semiconductor Device Equations, Springer, Wien, 1986.
[9] P.A. Markowich, C.A. Ringhofer, C. Schmeiser, Semiconductor Equations, Springer, Wien, 1990.
[10] Juan C. Meza, Ray S. Tuminaro, A multigrid preconditioner for the semiconductor equations, SIAM J. Sci. Comput. 17 (1) (1996) 118–132.
[11] G. Nelissen, Simulation of Multi-Ion Transport in Turbulent Flow, PhD Thesis, Fakulteit Toegepaste Weteneschappen, Vrije Universiteit Brussel, 2003.
[12] J. Newman, K.E. Thomas-Alyea, Electrochemical Systems, third ed., Wiley-Interscience, 2004.
[13] H. Paillere, J. Boxho, G. Degrez, H. Deconinck, Multidimensional upwind residual distribution schemes for the convection-diffusion equation, Int. J. Numer. Methods Fluids 23 (9) (1996) 923–936.
[14] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM Society for Industrial and Applied Mathematics, 2003.
[15] O. Schenk, Scalable Parallel Sparse LU Factorization Methods on Shared Memory Multiprocessors, PhD Thesis, ETH Zurich, 2000.
[16] O. Schenk, K. Gärtner, Solving unsymmetric sparse systems of linear equations with pardiso, Future Generation Computer Systems 20 (3) (2004) 475–487.
[17] S. Selberherr, Analysis and Simulation of Semiconductor Devices, Springer, Wien, 1984.
[18] K. Stüben, An introduction to algebraic multigrid, in: [19], pp. 413–532.
[19] U. Trottenberg, C.W. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2001 (with appendices by K. Stüben, P. Oswald, and A. Brandt).
[20] J. Wu, Venkat Srinivasan, J. Xu, C.Y. Wang, Newton–Krylov-multigrid algorithms for battery simulation, J. Electrochem. Soc. 149 (10) (2002) A1342–A1348.