EChem++ - An Object-Oriented Problem Solving Environment for Electrochemistry. Part 4. Adaptive Multilevel Finite Elements Applied to Electrochemical Models

Algorithm and Benchmark Calculations[⋆]

Kai Ludwig and Bernd Speiser [*]

Universität Tübingen, Institut für Organische Chemie, Auf der Morgenstelle 18, D-72076 Tübingen, Germany

Abstract

We present an approach for the modeling and simulation of electrochemical processes based on a general mathematical formulation, for which we expect future extensibility. For the numerical solution we propose the Rothe method with adaptive time integration performed by Rosenbrock schemes. For spatial integration we choose an adaptive finite element algorithm, controlled by a hierarchical *a posteriori* error estimator. The algorithm is designed to simulate models of potential-controlled transient experiments in 1D cell geometries and diffusional transport. It is tested for cyclic voltammetry with a fast electron transfer under semi-infinite diffusion conditions. Results of selected Rosenbrock methods are compared. We discuss the software design of the algorithm, as well as its coupling to the electrochemical compiler Ecco within an object-oriented framework. The software developed within this study is part of EChem++ and can be obtained freely as open-source code.

Key words: Adaptive finite element method, Rosenbrock schemes, computational electrochemistry, kinetic compiler, cyclic voltammetry, modeling, simulation

⋆ Part 3 see[1]

* author to whom correspondence should be addressed
Email address: bernd.speiser@uni-tuebingen.de (Bernd Speiser).

1 Introduction

The modeling and simulation of electrochemical processes is a complex task: an almost infinite number of phenomena and effects can be combined in a real system. Various electrochemical experiments, usually controlled by the potential or current at one or more working electrodes, may be performed in different cell geometries. Transport phenomena, adsorption effects as well as a non-constrained variety of coupled homogeneous and surface chemical reactions may occur[2–4]. This complexity hampers a closed theoretical treatment of electrode processes with algebraic expressions. Even nominally general modeling and simulation software, for example the well-known DigiSim[5], is not capable of dealing with more than a small class of such situations.
Thus, it has become a legitimate praxis to consider only certain limited aspects of the problems above, introduce specific assumptions, define the according governing equations and write computer code that solves the resulting specialized mathematical model by means of a particular numerical method. Although such a strategy is justified to obtain simulation results after a relatively short development period, it leads to computer code that can hardly be applied to model problems derived under more general assumptions.
Attempts to handle part of this complexity have been made[5, 6]. However, they suffer from a restriction of the available phenomena[5] and/or use programming methodologies that may complicate future extensions[5, 6] (see also the discussion in[7, 8]). With EChem++, we recently started out to provide C++ software components that are designed to minimize the efforts for reuse and feature extensions. The building blocks form the basis of an extensive software package[9]. Examples of such components are the ExcitationFunction template library[7] and the electrochemical compiler Ecco[8].
While the latter tackles only *modeling* aspects of electrochemical reaction kinetics, e.g. the translation of reaction mechanisms and related information into model equations, in the present work we proceed with a software component that addresses *simulation* tasks. We define simulation as the process that numerically solves the model equations, and results in the calculation of a physical quantity which is characteristic for the system[8].
In general, the mathematical formulation of electrochemical models forms a so-called *parabolic* partial differential equation (PDE) system. Often, these equations describe the evolution of concentrations in space and time. The concentrations in turn form the basis for the calculation of a quantity of interest, e.g. the electric current, which is also available from experiments.
For the numerical solution we propose here an algorithm recently analyzed by Lang[10], which is a derivative of the *Rothe* method: the PDE system is discretized first in time by means of *Rosenbrock* schemes. The resulting stationary problems are then solved with an adaptive multilevel finite element method.
In the context of electrochemical simulations, an adaptive algorithm similar

to the Rothe-Rosenbrock strategy was proposed by Bieniasz for 1D models, using finite difference formulae for the discretization of the stationary problems[11, 12]. However, the extension of finite difference schemes to higher-dimensional model geometries is regarded as difficult[13]. On the other hand, the finite element (FE) method is a commonly-accepted flexible alternative [13–17]. Moreover, during the last decades, a profound mathematical framework considering FE error analysis has been developed, which can be used to control adaptive solution processes[10, 18–37]. In particular, automatic grid adaption is indispensable for the accurate simulation of transient electrochemical reaction kinetics where complicating features occur, such as steep concentration fronts propagating with time.

In the context of electrochemistry, the first adaptive FE algorithm was described by Nann and Heinze[38] and applied to 2D models[39, 40]. Mesh refinement was controlled by a residual error estimator that compares the gradients of the computed solution on neighbouring elements. In regions with large differences in the gradients the mesh was locally improved by means of h-refinements (subdivision of elements). The approach is straightforward and generally applicable, but does not give any information about the accuracy of the simulation results.

To overcome this drawback, Harriman et al. developed an approach for stationary 2D models[41, 42], which is based on the estimation of the error in the quantity of interest, e.g. the electric current for potential-controlled experiments. In addition to the solution of the underlying PDEs, this estimator demands for the numerical approximation of a so-called dual (or adjoint) problem. Thus, in general the reliability of the estimator cannot be guaranteed ([42], see also the discussion in[43]). For transient problems the additional solution of a time-dependent dual problem was regarded far too expensive and further approximations had to be made. Thus, "the analysis is not completely rigorous and the bound achieved should be thought of as a refinement criterion rather than an upper bound on the error in the computed current"[44]. The mathematical expressions for the error estimators are complicated and apparently rely on problem-specific information[45, 46]. The use of this estimator in general modeling and simulation software therefore seems to be at least questionable. A similar but maybe more general approach was analyzed by Abercrombie and Denuault but applied to steady-state problems only[43]. While both these approaches concentrate on the estimation of the error within the electric current (or another linear functional of the concentrations) and in general rely on the rather expensive computation of a dual problem, the multilevel process proposed in the present work is controlled by an error estimator which ensures that the approximated error *in the concentrations* does not exceed a prescribed tolerance. If necessary, the mesh is locally improved by means of h-refinements. The estimator is based on a hierarchical enhancement of the linear FE approximation spaces by quadratic bubble functions[31, 47] and thus can be efficiently computed by small *local* residual problems, even for transient models. Exact robustness (lower and upper error bounds uni-

form in the time step and the mesh size) was proven for one-stage Rosenbrock methods[10]. For higher stage numbers, a nonlinear spatial error transport is possible. However, for practical computation, it turned out that the perturbation terms are negligible[10].
The algorithm mentioned above has been applied to a wide range of non-electrochemical real-life models in earlier works[10]. It is therefore regarded as a natural basis for a computational and software framework designed for a general electrochemical simulator. Applicability in electrochemical work will be shown here for a simple test example.
In addition to such a test, however, an electrochemical simulation framework must attain "generality" in three important aspects: (1) the basic mathematical formulation has to account for a sufficiently broad variety of processes, (2) the numerical algorithm has to provide correct solutions in these situations, and (3) the programming implementation has to minimize the effort to expand the actual code. While it will be necessary to show applicability in a large variety of complex situations in future papers, the present work concentrates on 1D transient reaction-diffusion models for potential-controlled experiments, in order to explain the algorithmic principles.
The simulated concentration profiles are directly related to other, non-electrochemical quantities, for example absorbance in spectroelectrochemical or the light intensity in electrogenerated luminescence experiments[2]. Thus, control of concentration errors seems to be a more general approach than control of current errors. Moreover, reliable approximations of the concentration profiles may yield acceptable approximations of the electric current for potential-controlled or for the potential in current-controlled experiments. However, there is no guarantee that this is indeed true. Within this context, a second objective of this paper is to show that the current as a concentration-derived quantity is reliably calculated with the proposed algorithm, at least for the test system.
Finally, as a third purpose of this manuscript, aspect (3) of the desirable "generality" will be considered. We will show that object-oriented programming methodology is advantageously used in our work to provide the program framework with mechanisms for extension.

2 Mathematical Model

A general framework of electrochemical model equations is adapted from a formulation of vector valued initial boundary value problems given by Lang[10]

$$\begin{aligned} H(x,t,u(x,t))\partial_t u(x,t) &= F(x,t,u(x,t)) \text{ in } \Omega \text{ (or on } \Gamma) \times (0,t_{\max}] \\ B(x,t,u(x,t))u(x,t) &= g(x,t,u(x,t)) \text{ on } \Gamma \times (0,t_{\max}] \\ u(x,0) &= u_0(x) \qquad\qquad \text{on } \bar{\Omega} \end{aligned} \tag{1}$$

where $\Omega \subset \mathbb{R}^d$, $d = 1, 2$, or 3, represents the cell geometry with boundary Γ. The symbol $\bar{\Omega}$ denotes the closure of Ω (Ω together with Γ). The solution vector $u = (v^{(0)}, v^{(1)}, ..., v^{(N_{\text{ads}}-1)}, u^{(0)}, u^{(1)}, ..., u^{(N_{\text{diss}}-1)})$ includes concentrations of adsorbed ($v^{(\cdot)}$) and dissolved ($u^{(\cdot)}$) species. The former are localized on a boundary. Furthermore, u could contain other quantities for instance the electric potential in the simulation of migration processes. The matrix $H(x,t,u)$ may introduce algebraic constraints to solution components. The right hand side $F(x,t,u)$ incorporates transport and reaction processes. The fact that $F(x,t,u)$ may not only be defined in Ω but also on Γ accounts for the treatment of surface reactions. Heterogeneous surface processes are modeled by matrix $B(x,t,u)$ and are treated together with right hand side $g(x,t,u)$ as boundary conditions. Symbols x and t denote the space and time variables. Time $t_{\max}$ is the experiment duration and vector u_0 determines initial conditions (at $t = 0$).
In the following, all quantities shall be suitably normalized as mentioned. We concentrate on transient models including linear diffusional transport combined with chemical reactions and heterogeneous electron transfers. Then, vector u represents the (normalized) concentrations of all involved dissolved species, $H(x,t,u)$ reduces to the identity matrix, I, and $F(x,t,u)$ is splitted into diffusional and reaction terms. Furthermore, we assume all diffusion coefficients and rate constants to be independent of temporal and spatial coordinates. Then, the first line of problem (1) reads

$$\begin{aligned} \partial_t u(x,t) &= F_{\text{diffusion}}(u(x,t)) + F_{\text{reaction}}(u(x,t)) \\ &= \mathbf{D} \cdot \partial_{xx} u(x,t) + R(u(x,t)) \qquad\qquad \text{in } \Omega \times (0,1] \end{aligned} \tag{2}$$

where $\partial_{xx} u(x,t)$ denotes the Laplace operator applied componentwise to $u(x,t)$, $\mathbf{D}$ is the diagonal matrix,

$$\mathbf{D} = \frac{D_{\text{ref}} \cdot t_{\max}}{L_{\text{ref}}^2} \cdot \text{diag}(\beta^{(0)}, ..., \beta^{(N-1)}) \tag{3}$$

where $\beta^{(m)} = D^{(m)}/D_{\text{ref}}$ is the dimensionless and normalized diffusion coefficient of the species with index m, D_{ref} and L_{ref} are reference diffusion coefficient and a reference length respectively (see below). The term $R(u(x,t))$ represents all kinetic rate laws describing homogeneous chemical reactions.
At electrode surfaces, the boundary conditions of dissolved, electroactive species with concentration $u^{(m)} = u^{(m)}(x,t)$, are assumed for the purpose of this work to obey classical Butler-Volmer kinetics[2]. Thus, in equation (1), $B(x,t,u^{(m)})$ reads $\beta^{(m)}\partial_x$ and $g(x,t,u^{(m)}) = g_{\text{BV}}(p(t),u)$, where $g_{\text{BV}}(p(t),u)$ incorporates the rates of all electron transfers of the m-th species. In the simplest case, for a single electron transfer step linking species m and j, it is

$$\beta^{(m)} \cdot \partial_x u^{(m)} = \pm\psi[\exp(-\alpha p(t)) \cdot u^{(m)} - \exp((1-\alpha)p(t)) \cdot u^{(j)}] \tag{4}$$

on $\Gamma \times (0,1]$. The sign in front of the right hand side depends on whether $u^{(m)}$ is oxidized $(-)$ or reduced $(+)$. For the meanings of the non-explained symbols, we refer to ref.[8]. For all species at inert walls and at the bulk end of diffusion layers, as well as for electroinactive species at electrode surfaces we apply homogeneous Neumann boundary conditions, $B(x,t,u^{(m)}) = \partial_x$ and $g(x,t,u^{(m)}) = 0$.
Furthermore, in the present paper we restrict the discussion to 1D models, such that domain $\Omega \subset \mathbb{R}^1$ is an open interval $\mathcal{I} = (0,1)$, normalized by reference length L_{ref} which represents the maximal diffusion layer thickness (possibly multiplied by a safety factor) for semi-infinite diffusion. For finite diffusion models, L_{ref} stands for the distance between electrode/electrode or electrode/inert wall arrangements or the thickness of a thin film[48]. The reference diffusion coefficient D_{ref} is taken as the maximum value of all diffusion coefficients within the problem.
In compact form, problem (1) then reads: find solution $u(x,t) = (u^{(0)}, ..., u^{(N-1)})$ that fulfills

$$\begin{aligned}
\partial_t u &= \mathbf{D} \cdot \partial_{xx} u + R(u) && \text{in} \quad \mathcal{I} \times (0,1] \\
\beta^{(m)}\partial_x u^{(m)} &= g_{\text{BV}}(p(t),u) && \text{on } \Gamma_{\text{e}} \times (0,1] \ , \ m \text{ electroactive} \\
\partial_x u^{(m)} &= 0 && \text{on } \Gamma_{\text{e}} \times (0,1] \ , \ \text{otherwise} \\
\partial_x u^{(m)} &= 0 && \text{on} \quad \Gamma \times (0,1] \\
u &= u_0 && \text{on} \qquad \bar{\mathcal{I}} \ , \ \text{at } t = 0
\end{aligned} \tag{5}$$

where Γ_{e} denotes an electrode surface, Γ all other boundaries and $\bar{\mathcal{I}} = [0,1]$.

In potential-controlled experiments, e.g. cyclic voltammetry, the quantity of interest is usually the electric current, i, through a working electrode, deter-

mined by the fluxes of electroactive species at the electrode surface,

$$i = \mathrm{F} A D_{\mathrm{ref}} c_{\mathrm{ref}} L_{\mathrm{ref}}^{-1} \sum_{j=0}^{M-1} n_j \sum_{k=0}^{N_j-1} \beta^{(\sigma_j(k))} \partial_x u^{(\sigma_j(k))} \tag{6}$$

where M denotes the total number of electron transfer steps. The number N_j includes the educt of step j, namely $\sigma_j(0)$, and its precursors (educts $\sigma_j(k+1)$ of those electron transfer steps in which $\sigma_j(k)$ is produced[49]). The mapping σ_j assigns k to the according species indices (entries in vector u). The number of transferred electrons in step j is n_j, F is the Faraday constant, A the area of the electrode surface, and c_{ref} the reference concentration. For c_{ref} we use the maximum value of all initial concentrations.
For a single one electron transfer step, expression (6) reduces to

$$\tilde{\chi} = i L_{\mathrm{ref}} (\mathrm{F} A D_{\mathrm{ref}} c_{\mathrm{ref}})^{-1} = \beta^{(m)} \partial_x u^{(m)} \tag{7}$$

which can be compared to the dimensionless current function[50],

$$\sqrt{\pi}\chi = \tilde{\chi} \cdot \sqrt{D_{\mathrm{ref}}} (L_{\mathrm{ref}} \sqrt{a})^{-1} \tag{8}$$

The values of $\sqrt{\pi}\chi$ will be used in section 5 to prove the convergence of the algorithm.

3 The Algorithm

The algorithm proposed here belongs to the Rothe methods that discretize the parabolic PDE systems first in time. For this part of the integration we will apply Rosenbrock methods (section 3.1). The time discretization leads to so-called *elliptic* differential equation systems with respect to the spatial variable that need to be solved at each time step.[10, 26] An adaptive finite element algorithm is then employed for the solution of such systems (section 3.2.1). The following derivations are based on the work of Lang[10, 51–54], and take into account the specific requirements characteristic for electrochemical simulations.

3.1 Time Integration

Rothe methods regard the time-dependent PDEs as a system of ordinary differential equations (ODEs)

$$\partial_t u(t) = F(t, u(t)) \quad t > 0 \quad u(0) = u_0 \tag{9}$$

and then apply well-known numerical ODE integrators. In the context of electrochemical simulation we frequently encounter stiff ODE systems, with largely different time behaviour of different unknowns, which are notoriously difficult to solve. In general, for stability reasons, the solution of stiff ODE systems demands implicit integration algorithms, e.g. the backward Euler method,

$$u_{n+1} = u_n + \tau_n F(t_{n+1}, u_{n+1}) \tag{10}$$

where u_n is the solution at time t_n, $\tau_n = t_{n+1} - t_n$ is the step size and u_{n+1} the desired solution at t_{n+1}. In formula (10), u_{n+1} is implicitly defined. Therefore, usually an iterative Newton method is applied to obtain the solution.

3.1.1 Rosenbrock Methods

Rosenbrock schemes are derived by incorporating Newton iteration steps with the exact Jacobian of the right-hand side $F(t, u(t))$ directly into integration formula (10)[55–57]. Thus, Rosenbrock methods offer the advantage that an explicit control of Newton iterations is not necessary. They allow a rapid change in step size τ and are rather simple to implement[10]. In particular, the relationship between different Rosenbrock schemes enables an elegant and efficient object-oriented implementation by the *Strategy* pattern (section 4.1). The standard form of the s-stage Rosenbrock methods, which is adequate for problems of type (5) reads[10]

$$\begin{aligned} u_{n+1} &= u_n + \textstyle\sum_{i=1}^{s} m_i U_{ni} \\ \hat{u}_{n+1} &= u_n + \textstyle\sum_{i=1}^{s} \hat{m}_i U_{ni} \end{aligned} \tag{11}$$

with

$$\left(\frac{1}{\tau_n \gamma} I - F_u(t_n, u_n)\right) U_{ni} = F(t_i, U_i) - I \sum_{j=1}^{i-1} \frac{c_{ij}}{\tau_n} U_{nj} + \tau_n \gamma_i F_t(t_n, u_n) \tag{12}$$

and internal values

$$t_i = t_n + \alpha_i \tau_n, \qquad U_i = u_n + \sum_{j=1}^{i-1} a_{ij} U_{nj} \tag{13}$$

Note that each time step includes the calculation of s internal (13) and stage values (12). In formulas (11)-(13) the coefficients m_i, $\hat{m}_i$, γ, c_{ij}, γ_i, α_i and a_{ij} characterize a particular Rosenbrock method. Together with the number of stages, s, the coefficients determine the order of convergence, the stability, as well as the efficiency of the formula. The values of the coefficients are tabulated in the literature[10, 52, 53, 55, 56]. Symbol $F_u(t_n, u_n)$ denotes the Jacobian of $F(x, t, u)$ (note that x has been omitted for simplicity) evaluated at time step t_n and with corresponding solution u_n, $F(t_i, U_i)$ is the right hand side $F(x, t, u)$ evaluated at stage time t_i with internal value U_i. The latter values are calculated by equations (13). $F_t(t_n, u_n)$ is the time derivative of $F(x, t, u)$ evaluated at t_n and with u_n, τ_n is the predicted step size (section 3.1.2). At each stage, i, the stage values U_{ni} are computed by solving the linear system of equations (12). Solutions u_{n+1} and $\hat{u}_{n+1}$ are then obtained by equations (11). The values of $\hat{u}_{n+1}$ are used to estimate the time errors (section 3.1.2).
Note that for the solution of PDE systems, also schemes for the boundary conditions need to be considered (for details, see[10]),

$$-D_n U_{ni} = g(t_i, U_i) - B(t_i, U_i) U_i + \tau_n \gamma_i E_n \quad \text{ on } \Gamma \tag{14}$$

with

$$\begin{aligned} D_n &= \partial_u (g(t, u) - B(t, u)u), \text{ at } u = u_n,\ t = t_n \\ E_n &= \partial_t (g(t, u) - B(t, u)u), \text{ at } u = u_n,\ t = t_n \end{aligned} \tag{15}$$

Next, we apply algorithm (11)-(15) to problem (5). Since $u = (u^{(0)}, ..., u^{(N-1)})$, all solution, stage and internal values (u_n, $\hat{u}_n$, U_{ni}, U_i) are also vectors of space dependent concentrations. Equation (12) becomes

$$\left(\frac{1}{\tau_n \gamma} I - [\mathbf{D} \cdot \partial_{xx} + R_u(u_n)] \right) U_{ni} = \mathbf{D} \cdot \partial_{xx} U_i + R(U_i) - \sum_{j=1}^{i-1} \frac{c_{ij}}{\tau_n} U_{nj} \tag{16}$$

where $R_u(u_n)$ denotes the Jacobian of $R(u)$ at time t_n. Note that diffusion and reaction terms do not depend explicitly on time owing to the fact that we assume time-independent diffusion coefficients and rate constants. Thus, the last term of equation (12) vanishes in equation (16).

At an electrode surface Γ_e and for an electroactive species with index m, boundary condition (14) becomes

$$-\partial_u g_{\mathrm{BV,n}} \cdot U_{ni}^{(m)} + \beta^{(m)} \partial_x U_{ni}^{(m)} = g_{\mathrm{BV,i}} - \beta^{(m)} \partial_x U_i^{(m)} + \tau_n \gamma_i \partial_t g_{\mathrm{BV,n}} \tag{17}$$

with $g_{\mathrm{BV,n}} = g_{\mathrm{BV}}(p(t_n), u_n)$ and $g_{\mathrm{BV,i}} = g_{\mathrm{BV}}(p(t_i), U_i)$. On diffusion layer limits or inert walls, Γ, or on Γ_e, if species m is not electroactive, it follows from (14)

$$\partial_x U_{ni}^{(m)} = -\partial_x U_i^{(m)} \tag{18}$$

Problem (16), together with (17) and (18) defines a system of elliptic differential equations that are solved in section 3.2.1 for the spatial concentration profiles at each time stage. The initial conditions u_0 are obtained from the concentration distribution of the chemical system at equilibrium, before the electrochemical experiment starts.

3.1.2 Control of Step Size

An efficient time integration is often achieved by variation of step size τ during simulation. If the solution exhibits strong variations in time, τ must be sufficiently small to preserve reliable approximations. However, in time periods where the solution gradients (with respect to time) are small, pessimistically chosen time steps waste computation time and should be increased.
In the tradition of Runge-Kutta methods, Rosenbrock schemes offer an efficient way to control this adaptive process. For each Rosenbrock method of order p, an embedded formula is defined by coefficients $\hat{m}_i$. These coefficients produce solution $\hat{u}_{n+1}$ (equations (11)) which is of lower order, usually $p-1$, compared to u_{n+1} (for details, see[57]).
Therefore, only minimal additional computational cost is necessary to estimate the local error of a Rosenbrock step by the difference $r_{n+1} = u_{n+1} - \hat{u}_{n+1}$. Moreover, instead of computing $\hat{u}_{n+1}$ by equations (11), one directly calculates r_{n+1} by

$$r_{n+1} = \sum_{i=1}^{s} (m_i - \hat{m}_i) U_{ni} \tag{19}$$

where coefficients $m_i - \hat{m}_i$ are calculated prior to simulation. Measured in an appropriate norm, $|||\cdot|||$, the predicted local error at t_{n+1} is

$$\epsilon_{n+1}^t = |||r_{n+1}||| \tag{20}$$

For electrochemical models, e.g. (5), $r_{n+1} = (r_{n+1}^{(0)}, ..., r_{n+1}^{(N-1)})$ must be controlled for each species and depends on the spatial coordinate, $x \in \mathcal{I}$. Thus,

for norm $|||\cdot|||$ we use a weighted root mean square norm[10, 54–56, 58],

$$|||r||| = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\frac{||r^{(i)}||_{L^2(\mathcal{I})}}{\mathrm{ATOL} + \mathrm{RTOL}||u^{(i)}||_{L^2(\mathcal{I})}} \right)^2 \right)^{1/2} \tag{21}$$

where $||\cdot||_{L^2(\mathcal{I})}$ is the L^2-norm over interval $\mathcal{I}$. For discretized solutions on a spatial grid, the L^2-norm is equivalent to the well-known euclidean norm[13], $||x||_2 = \sqrt{x^T x}$. This is used in our computations. ATOL and RTOL are absolute and relative error tolerances. Setting ATOL = 0 results in a relative error measurement. On the other hand, if RTOL = 0, we obtain the absolute error. In the present simulations, we set

$$\mathrm{ATOL} = \mathrm{RTOL} = 1/2 \cdot \mathrm{TOL} \tag{22}$$

which enables a relative error measure for high concentrations and a reliable absolute error value for those concentrations that are close to zero. TOL denotes the accuracy to be achieved by the overall algorithm, including both time and space tolerances (factor 1/2 in equation (22)).
The ϵ^t values are used at each time step to predict a new optimal step size τ. If $\epsilon^t \leq 1$, the tolerance is achieved and τ may be increased for the next time step. On the other hand, if ϵ^t exceeds unity, the step is repeated with reduced τ. The prediction of optimal τ values is controlled by an approach developed by Gustafsson[59] and slightly modified by Lang[10, 51], where the predicted step size is

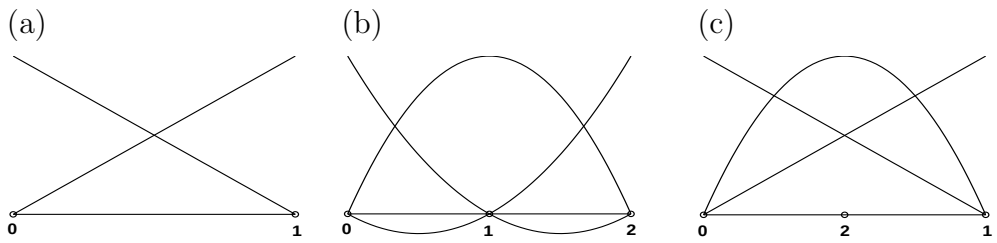
$$\tau_{\mathrm{pred}} = \min \left(\frac{\tau_n}{\tau_{n-1}} \left(\frac{\epsilon_n^t}{\epsilon_{n+1}^t \epsilon_{n+1}^t} \right)^{1/p}, \left(\frac{1}{\epsilon_n^t} \right)^{1/p} \right) \cdot \tau_n \tag{23}$$

An outline of the code of the complete *predictive controller* is given in ref.[59]. In the present formulation, in order to achieve a smooth τ variation, we supply the controller with heuristic safety factors

$$\begin{aligned} \tau_{n+1} &= \min(1.5 \cdot \tau_n, 0.9 \cdot \tau_{\mathrm{pred}}) \quad \text{for } \tau_n \text{ accepted} \\ \tau_n &= \max(0.5 \cdot \tau_{\mathrm{n}}, 0.9 \cdot \tau_{\mathrm{pred}}) \quad \text{for } \tau_n \text{ rejected} \end{aligned} \tag{24}$$

Moreover, for the simulations discussed here, we introduce the additional constraint

$$\tau \leq 5 \cdot 10^{-3} \tag{25}$$



$\phi_0(x) = 1 - x$ $\quad$ $\phi_0(x) = (1-x)(1-2x)$ $\quad$ $\phi_0(x) = 1 - x$

$\phi_1(x) = x$ $\quad$ $\phi_1(x) = 4x(1-x)$ $\quad$ $\phi_2(x) = 4x(1-x)$

$\phi_2(x) = -x(1-2x)$ $\quad$ $\phi_1(x) = x$

Figure 1. 1D finite elements: Linear (a) and quadratic interpolation (b), as well as hierarchical element with quadratic bump function ϕ_2 (c); the lengths of the elements are equal to unity; local indices of the nodal points are shown.

This value restricts the growth of the (dimensionless) step size τ, in order to adequately react to rapid changes within the boundary variations at electrode surfaces.

3.2 Space Integration

The spatial discretization of the elliptic stage problems (16) with boundary conditions (17) and (18) is performed here by a multilevel FE approach.

3.2.1 Adaptive Multilevel Finite Elements

For the calculation of a discrete Rosenbrock solution $u_{n+1,k}$, we seek the corresponding stage values, $U_{ni,k}$, within a sequence of (linear) FE subspaces,

$$\mathcal{V}_{n+1,1} \subset \mathcal{V}_{n+1,2} \subset ... \subset \mathcal{V}_{n+1,k} \tag{26}$$

generated by locally refined grids

$$\mathcal{I}_{n+1,1} \subset \mathcal{I}_{n+1,2} \subset ... \subset \mathcal{I}_{n+1,k} \tag{27}$$

which are supplied with piecewise linear interpolation polynomials ϕ (Figure 1(a)). Index k denotes the refinement level counted from the coarsest grid, $\mathcal{I}_{n+1,1}$ at time step t_{n+1}.
Thus, starting from the weak formulation of equation (16) (see Appendix A), we seek $U_{ni,k} \in \mathcal{V}_{n+1,k}$ that fulfill

$$(L_{n,k}U_{ni,k}, v) = (r_{ni,k}, v) \quad \forall v \in \mathcal{V}_{n+1,k} \quad i = 1, ..., s \tag{28}$$

with system matrix

$$L_{n,k} = \left(\frac{1}{\tau_n \gamma} I - [\mathbf{D} \cdot \partial_{xx} + R_u(u_{n,k})] \right) \tag{29}$$

and right hand side

$$r_{ni,k} = \mathbf{D} \cdot \partial_{xx} U_{i,k} + R(U_{i,k}) - \sum_{j=1}^{i-1} \frac{c_{ij}}{\tau_n} U_{nj,k} \tag{30}$$

The FE solution at time t_{n+1} and level k is computed by equation (11)

$$u_{n+1,k} = u_{n,k} + \sum_{i=1}^{s} m_i U_{ni,k} \tag{31}$$

Therefore, the previous solution $u_{n,k}$ is interpolated on the refined grids.

3.2.2 Hierarchical A Posteriori Error Estimator

The refinement process (27) is controlled by an approximation of the spatial stage errors,

$$E_{ni,k} = U_{ni} - U_{ni,k} \qquad i = 1, ..., s \tag{32}$$

resulting from the difference between the exact solution, U_{ni}, and the FE solution, $U_{ni,k}$. However, since the exact solution is *a priori* unknown, a common approach is to obtain a so-called *a posteriori* error estimate for $E_{ni,k}$. Let $\mathcal{W}_{n+1,k}$ be the (quadratic) FE space created on grid $\mathcal{I}_{n+1,k}$ by piecewise quadratic interpolation polynomials (Figure 1(b)). Then, stage errors (32) may be approximated by

$$E_{ni,k} \approx \bar{U}_{ni,k} - U_{ni,k} \quad \text{for } \bar{U}_{ni,k} \in \mathcal{W}_{n+1,k} \tag{33}$$

Recalling equations (28) and (32), $E_{ni,k}$ is given by the global residual problem

$$(L_{n,k} E_{ni,k}, w) = (\hat{r}_{ni,k}, w) \quad \forall w \in \mathcal{W}_{n+1,k} \;\; i = 1, ..., s \tag{34}$$

with

$$\hat{r}_{ni,k} = r_{ni,k}(E_{n1,k} + U_{n1,k}, ..., E_{ni-1,k} + U_{ni-1,k}) - L_{n,k} U_{ni,k} \tag{35}$$

The predicted spatial error of solution $u_{n+1,k} \in \mathcal{V}_{n+1,k}$ results from the $E_{ni,k}$ values

$$e_{n+1,k} = u_{n,k} - \tilde{u}_{n,1} + \sum_{i=1}^{s} m_i E_{ni,k} \tag{36}$$

In equation (36), $u_{n,k} - \tilde{u}_{n,1}$ measures the error that appears in regions where grid nodes have been removed from the optimal grid $\mathcal{I}_{n,k}$ at t_n, in order to obtain the starting grid $\mathcal{I}_{n+1,1}$ of time step $n+1$ (see section 3.2.3).
Unfortunately, the computation of the $E_{ni,k}$ values by equations (34) would be rather expensive. At each Rosenbrock stage, both $U_{ni,k}$ and $E_{ni,k}$ must be calculated by solving possibly large linear systems of equations resulting from the weak forms (28) and (34).
A more efficient approach is the use of so-called *hierarchical bases*[34] for the approximation of $E_{ni,k}$. Instead of employing full space $\mathcal{W}_{n+1,k}$, we enrich $\mathcal{V}_{n+1,k}$ by quadratic bump functions associated with the midpoint of each element and vanishing at the elements' boundaries (Figure 1(c)). The resulting enriched FE space can be regarded as decomposition

$$\bar{\mathcal{V}}_{n+1,k} = \mathcal{V}_{n+1,k} \oplus \mathcal{Z}_{n+1,k} \tag{37}$$

with subspace $\mathcal{Z}_{n+1,k}$, spanned by the quadratic bump functions (symbol $\oplus$ denotes the *direct sum* of vector spaces[60]). Then, for a given grid $\mathcal{I}_{n+1,k}$ which is composed of $N_{n+1,k}$ finite elements S^e (in the following, the superscript e denotes the index of an element),

$$\mathcal{I}_{n+1,k} = \bigcup_{e=0}^{N_{n+1,k}-1} S^e \tag{38}$$

local stage errors $E^e_{ni,k}$ result from the solution of small *local* residual problems[51]

$$\begin{aligned} (L^e_{n,k} E^e_{ni,k}, z) &= (\hat{r}^e_{ni,k}, z) \quad \text{in } S^e \quad \forall z \in \mathcal{Z}_{n+1,k} \\ E^e_{ni,k} &= 0 \qquad\qquad \text{on } \Gamma^e \quad i = 1, ..., s \end{aligned} \tag{39}$$

for each element. $L^e_{n,k}$ and $\hat{r}^e_{ni,k}$ denote the local system matrix and right hand side, respectively. Homogeneous Dirichlet conditions are applied at the elements' boundaries, Γ^e. Finally, local error quantities are defined by

$$\eta^e_{n+1,k} = ||u^e_{n,k} - \tilde{u}^e_{n,1} + \sum_{i=1}^{s} m_i E^e_{ni,k}||_2 \tag{40}$$

```
1   n = 0, t_n = 0, τ_{n+1} = τ_init
2   I_{n+1,1} = I_{n,k}
3   while t_n < 1
4     k = 1
5     compute u_{n+1,k}, r_{n+1,k}, η^e_{n+1,k}, ϵ^x_{n+1,k}
6       while ϵ^x_{n+1,k} > maxTOL_x and k < k_max
7         refine all elements with η^e_{n+1,k} > η^barr_{n+1,k}
8         k = k + 1
9         compute u_{n+1,k}, r_{n+1,k}, η^e_{n+1,k}, ϵ^x_{n+1,k}
10    compute ϵ^t_{n+1}
11      if ϵ^t_{n+1} ≤ 1.0
12        accept solution
13          if ϵ^x_{n+1,k} < minTOL_x
14            coarsen all elements with 2 · η^e_{n+1,k} < η^barr_{n+1,k}
15        t_{n+1} = t_n + τ_{n+1}, n = n + 1
16        I_{n+1,1} = I_{n,k}
17        compute τ_{n+1}
18      else
19        reject solution
20        restore I_{n+1,1}
21        reduce τ_{n+1}
```

Figure 2. Outline of the MFEM algorithm

and the global spatial discretization error, ϵ^x, is

$$\epsilon^x_{n+1,k} = \left(\sum_{e=1}^{N_{n+1,k}} (\eta^e_{n+1,k})^2 \right)^{1/2} \tag{41}$$

3.2.3 Grid Evolution

Let maxTOL_x be the spatial tolerance to be achieved and $\eta^{\mathrm{barr}}_{n+1,k} = \gamma \cdot \max_e \eta^e_{n+1,k}$, a local error barrier, adjustable by $0 < \gamma < 1$[10, 37]. On this basis, both, grid refinement and coarsening are controlled by the local quantities $\eta^e_{n+1,k}$, given by equation (40) (Figure 2).
Starting from the computation of solution $u_{n+1,1}$ on a coarse grid $\mathcal{I}_{n+1,1}$ together with error estimates $\eta^e_{n+1,1}$ and $\epsilon^x_{n+1,1}$, the grid is successively refined, until $\epsilon^x_{n+1,k}$ is lower than maxTOL_x or a maximum number of refinement levels $k_{\max}$ is reached (lines *6-9*). Only elements with an error value greater than

$\eta^{\text{barr}}_{n+1,k}$ are refined (line *7*). In the present work we use $\gamma = 0.7$.
If an optimal spatial solution is determined, in line *10*, the time error ϵ^t_{n+1} is estimated according to equations (20) and (21). The solution is accepted if the time tolerance is fulfilled, i.e. $\epsilon^t_{n+1} \leq 1.0$ (line *11*).
Note that situations may occur in which after $k_{\max}$ refinement levels, the spatial error tolerance is not achieved, although condition $\epsilon^t_{n+1} \leq 1.0$ is fulfilled. In this case, we will still accept the solution. We have observed that the use of this "*in dubio pro* time" strategy is seldom necessary and that the effect on the current calculation is rather small (section 5).
At each accepted time step, the next grid is computed by coarsening of elements with $2 \cdot \eta^e_{n+1,k} < \eta^{\text{barr}}_{n+1,k}$. This condition shall ensure that a coarsened element will not be directly refined again in the following refinement procedure[10].
If the time tolerance is not fulfilled, the initial grid of the actual time step is restored and a new refinement process is started with reduced τ (lines *18-21*).
At each refinement or coarsening step *mass balancing* is performed on the grid[18]

$$\frac{1}{\mu} \leq \frac{h_{e-1}}{h_e} \leq \mu \quad \text{for a fixed } \mu \geq 1 \tag{42}$$

where h_e denotes the length of an element within partition (38). In the present simulations, we use $\mu = 2$.

4 Object-Oriented Design

For the implementation of the algorithm discussed in section 3, we apply advanced programming methodologies based on object-orientation in the C++ programming language[61]. C++ combines efficient scientific programming with object-oriented software design that enhances the readability and the reusability of code[61–65].

4.1 Rosenbrock Policy — Strategy Pattern

In order to compare several different Rosenbrock methods combined with the multilevel FE approach, the *Strategy* pattern[62] was employed to create the class hierarchy shown in Figure 3. Each specific Rosenbrock scheme is represented by its own subclass. Each subclass defines a step() method, that performs the according Rosenbrock time step, defined by equations (11), (28), (13) and (17)–(19), as well as the a posteriori error estimation (section 3.2.2).

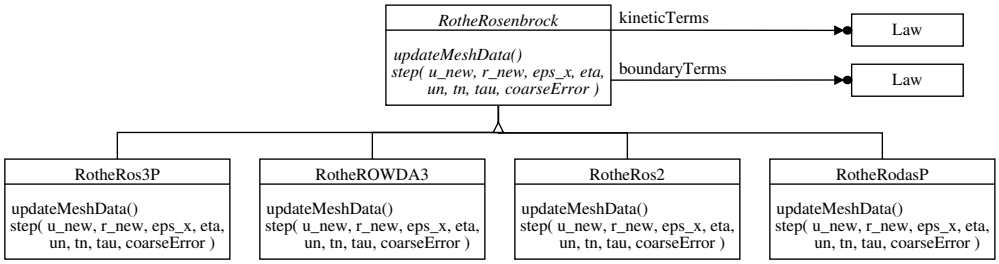


Figure 3. Rosenbrock policy classes (*Strategy* pattern); only selected methods are shown; each subclass represents a specific type of Rosenbrock scheme.

The step() method calculates the discrete solution, u_new ($u_{n+1,k}$), the predicted time residual, r_new ($r_{n+1,k}$), the global spatial error estimate, eps_x ($\epsilon^x_{n+1,k}$), as well as the local error quantities, eta ($\eta^e_{n+1,k}$). Input data are the solution, un, at time tn, the step size tau, and the error due to previous coarsenings (equation (36)). Clearly, the details of this process depend on the FE discretization. Thus, each time the grid structure changes (due to refinements or coarsenings), all relevant data, e.g. FE matrices, need to be updated. Therefore, each class defines an updateMeshData() method. On the other hand, the classes are largely decoupled from the specific electrochemical conditions, since problem dependent terms like reaction rates or boundary conditions are obtained from Ecco's Law classes.
The MFEMSolver template class (Figure 4) implements the algorithm of Figure 2 through its solve() method. Amongst others, the class is parameterized by a Rosenbrock strategy as template parameter. Therefore, the specific Rosenbrock method employed within the algorithm (Figure 2, lines *5* and *9*) can easily be exchanged without changing the overall algorithm.
Figure 5 shows an interaction diagram of an algorithm that is configured with the RotheRos3P solver. Moreover, several time controllers[10, 59, 64] are implemented and can be used as time stepping strategies in an analogous way as the Rosenbrock policy. For the simulations discussed in this work, we have used the controller of Lang (equation (23)). The FE mesh is represented by a mesh class. All information concerning the mesh structure can be obtained by calling an appropriate member function (e.g. nbElements() for the number of finite elements within the mesh).

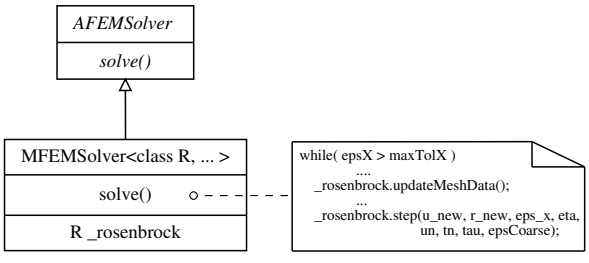


Figure 4. Multilevel FE solver class (MFEMSolver); template parameter R represents different types of Rosenbrock solvers (Figure 3). The pseudo lines of code show an excerpt of the algorithm given in Figure 2, lines *6-9*.

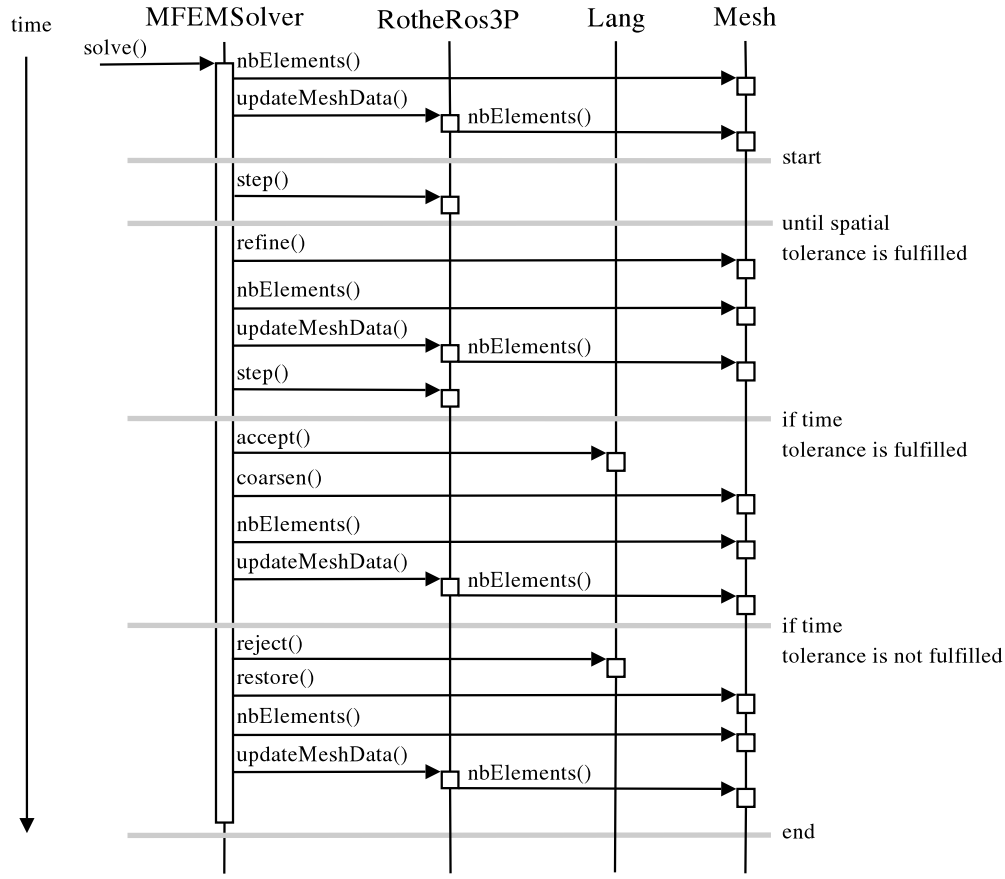


Figure 5. Interaction diagram of the MFEM algorithm. Only selected function calls are shown. For the meaning of the symbols, see ref.[62]

As soon as the solve() method is called, the MFEMSolver object initializes all relevant data and starts the simulation. At the beginning of each time step, the step() method of the Rosenbrock object is executed (see also Figure 2, line *5*). If necessary, the FE grid is refined until the spatial tolerance is fulfilled. After that, the solver accepts the solution if the time tolerance is reached. The grid is coarsened and a new time step is performed. Otherwise, the solver rejects the computed solution, restores the former, coarse grid and restarts the time step with a reduced step size. Due to the reconstructions of former grid levels the mesh class has to memorise information about the mesh evolution. Therefore, internally, a tree container is used[66].
Note that the use of a Rosenbrock class as template parameter enhances the efficiency of the algorithm, since all necessary adjustments are done at compile time and expensive virtual function calls at runtime are avoided. A slight drawback of this approach is that the MFEMSolver must be configured with a Rosenbrock strategy at compile time. However, the introduction of the AFEMSolver base class enables a flexible handling of variably configured MFEMSolver objects also at run time (Figure 6), while the efficient function calls on the Rosenbrock objects are preserved.

```
AFEMSolver* solver;
       ...
// if Ros3P shall be used
solver = new MFEMSolver<RotheRos3P,...>(...);

// else if ROWDA3 shall be used
solver = new MFEMSolver<RotheROWDA3,...>(...);
       ...
solver->solve();
```

Figure 6. Client code excerpt for the binding of various MFEMSolver configurations at runtime.

4.2 The Coupling to Ecco

The electrochemical compiler Ecco[8] translates any user defined electrochemical reaction mechanism within its scope into C++ objects. In particular, Ecco creates objects that represent the reaction and boundary terms of problem (5) for each species in the mechanism. Within the Ecco framework, these terms are represented by *Law* classes (Figure 7). Thus, by the coupling of the law classes to the present algorithm (Figure 3), we are able to simulate all electrochemical reaction mechanisms supported by the kinetic compiler, without changing the code of the algorithm. This should be even true for possible future extensions of Ecco.
For example, reaction term, $R(u)$, of the $\underline{\underline{\text{EC}}}$ mechanism with an irreversible follow-up reaction

$$\begin{aligned} &\text{A} \rightleftharpoons \text{B} + \text{e}^- \\ &\text{B} \rightarrow \text{C} \end{aligned} \tag{43}$$

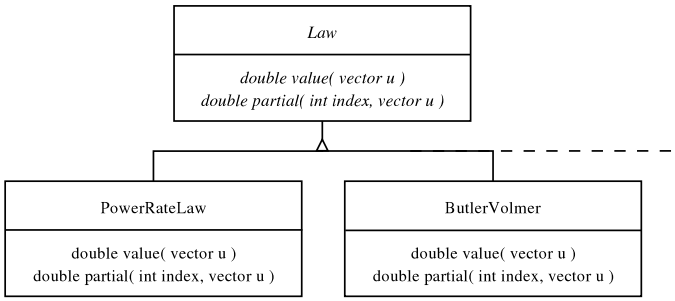


Figure 7. Ecco's law class hierarchy; the derived classes represent two examples from those generated by Ecco; selected methods for the use within a numerical algorithm are shown; details, see[8]

becomes

$$R(u) = \begin{pmatrix} R_0(u) \\ R_1(u) \\ R_2(u) \end{pmatrix} = \begin{pmatrix} 0 \\ -\ k_{\mathrm{f1}} \cdot u^{(1)} \\ k_{\mathrm{f1}} \cdot u^{(1)} \end{pmatrix} \equiv \begin{bmatrix} \texttt{Law*} \\ \texttt{Law*} \\ \texttt{Law*} \end{bmatrix} = \texttt{kineticTerms} \qquad (44)$$

where $u = (u^{(0)}, u^{(1)}, u^{(2)})$ is the concentration vector of species A, B and C. Symbol k_{f1} denotes the forward rate constant of the follow-up reaction. Within C++ code, each row of equation (44), and therefore the reaction term of each species, is represented by a Law object. These are in turn administered as a vector of pointers (kineticTerms).
Thus, in order to proceed a Rosenbrock step, the function values $R(U_i)$ in equation (16) (see also (55) in Appendix A) are obtained from the kineticTerms vector,

```
kineticTerms[0]->value(Ui)
kineticTerms[1]->value(Ui)
kineticTerms[2]->value(Ui)
```

where Ui denotes the concentration vector at stage i. Besides function values, the evaluation of the Jacobian matrix

$$R_u(u) = \begin{pmatrix} \frac{\partial R_0}{\partial u^{(0)}} & \frac{\partial R_0}{\partial u^{(1)}} & \frac{\partial R_0}{\partial u^{(2)}} \\ \frac{\partial R_1}{\partial u^{(0)}} & \frac{\partial R_1}{\partial u^{(1)}} & \frac{\partial R_1}{\partial u^{(2)}} \\ \frac{\partial R_2}{\partial u^{(0)}} & \frac{\partial R_2}{\partial u^{(1)}} & \frac{\partial R_2}{\partial u^{(2)}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ -k_{\mathrm{f1}} & 0 & 0 \\ k_{\mathrm{f1}} & 0 & 0 \end{pmatrix} \qquad (45)$$

is executed by calling

```
kineticTerms[0]->partial(0,un)
kineticTerms[0]->partial(1,un)
kineticTerms[0]->partial(2,un)

kineticTerms[1]->partial(0,un)
kineticTerms[1]->partial(1,un)
kineticTerms[1]->partial(2,un) ...
```

for each individual entry of the matrix with u_n. The partial() method is used during numerical integration of terms (54) (see Appendix A).
It is important to note that usually every numerical solver for problems of type (5) demands for the evaluation of function values and Jacobian matrix. Thus, Ecco could be used also within algorithms based on other discretization schemes, like BDF methods or the popular Crank-Nicolson time integration

as well as spatial discretizations based on finite differences or orthogonal collocation.

5 Simulation of Cyclic Voltammetry - Benchmarks

In the present work, we concentrate on the quality of the simulation results by means of a simple test system. This is intended to demonstrate the general features of the algorithm and implementation presented here. Simulations of a range of electrochemical model geometries, electroanalytical techniques, together with various reaction mechanisms will be presented separately.
Let us consider semi-infinite diffusion conditions with a single electron transfer,

$$\mathrm{A} \rightleftharpoons \mathrm{B} + \mathrm{e}^- \tag{46}$$

We use the Butler-Volmer law object generated by the Ecco compiler and set the parameters as follows: formal potential $E^0 = 0.25$ V, transfer coefficient $\alpha = 0.5$, heterogeneous rate constant $k_\mathrm{h} = 10^4$ cm s^{-1}. The large value of k_h approximates reversibility of the electron transfer step under the experimental conditions envisaged (see below). The initial concentration of species A is $1 \cdot 10^{-4}$ mol l^{-1}, which is used as reference concentration for the normalization. The diffusion coefficients are set to $D_\mathrm{A} = D_\mathrm{B} = 1 \cdot 10^{-5}$ cm^2 s^{-1} and $D_\mathrm{ref} = \max(D_\mathrm{A}, D_\mathrm{B}) = 1 \cdot 10^{-5}$ cm^2 s^{-1}.
For the potential cycle, we use a CVExcitationFunction object provided by the ExcitationFunction template library[7] with the values $E_\mathrm{start} = 0.0$ V and $E_\lambda = 0.5$ V as well as an experiment duration of $t_\mathrm{max} = 2$ s, which is equivalent to a scan rate of $v = 0.5$ V s^{-1}. The reference length is $L_\mathrm{ref} = 3 \cdot \sqrt{D_\mathrm{ref} t_\mathrm{max}}$. All simulations are performed for a temperature of $T = 293.15$ K with dimensionless quantities.
For the calculation of the electric current (equation (7)), the gradient at the electrode surface is computed from the (linear) FE solution, which is equivalent to the first order finite difference approximation,

$$\partial_x u|_{x=0} \approx \frac{u_1 - u_0}{h_0} \tag{47}$$

where u_0 and u_1 denote discrete solution values at the first and second grid nodes and h_0 is the length of the first finite element. The initial grid is divided into 5 segments and locally refined 5 times towards the electrode boundary. Both refinements and coarsenings on the first element are prohibited, to avoid interferences in the gradient approximation (47).
Four different Rosenbrock schemes were tested in combination with the adap-

tive FE approach: the second order ROS2, the third order ROWDA3 and ROS3P as well as the fourth order RODASP schemes (Table 1). We use equation (8) to compare the current values to those tabulated by Nicholson and Shain[50]. Although the latter were calculated for Nernstian boundary conditions, the comparison within the present work, where a large heterogeneous rate constant is used, yields at least an indication about the reliability of the algorithm. Unfortunately, closed form solutions for cyclic voltammetric currents are not available.

Figure 8 shows the simulated cyclic voltammograms for the four schemes using four descending relative error tolerances from 10% to 1%. Although we control the error with respect to the concentration profiles, the results clearly indicate a parallel improvement in the electric current curves with decreasing tolerances. The current curves of the low order method ROS2 never reach satisfying approximations, but ROWDA3 only slightly exceeds the reference line (circles) at the lowest tolerance value. The ROS3P method, which was especially designed for parabolic PDEs[53], shows very good agreement for a tolerance lower than 5%. This cannot be improved further by the higher order method RODASP.

Figure 9(a) compares the relative errors in the peak currents,

$$\frac{|\chi_{\mathrm{p}}^{\mathrm{MFEM}} - \chi_{\mathrm{p}}|}{\chi_{\mathrm{p}}} = \frac{|\sqrt{\pi}\chi_{\mathrm{p}}^{\mathrm{MFEM}} - 0.4463|}{0.4463} \tag{48}$$

of the four Rosenbrock methods ($\sqrt{\pi}\chi_{\mathrm{p}} = 0.4463$ is the Nicholson and Shain[50] reference value). Again, the ROS2 and ROWDA3 solvers perform rather weakly, while ROS3P and RODASP show good results. For ROS3P, the tolerances applied for the approximations of the concentration profiles are preserved also for the peak currents.

ROS3P also reveals better performance in terms of efficiency compared to the other methods (Figure 9(b)). It is as fast as the lower order method ROS2, although one more stage needs to be computed. RODASP shows an unusually high CPU time of approximately 2 seconds for the lowest tolerance. This artefact is due to the fixing of the first element of the initial grid, with respect

Table 1
Selected Rosenbrock methods taken from[10] and references therein. The numbers in parentheses denote the order of the embedded formula (section 3.1.2).

Method	stages	$R(u)$ evaluations	order p
ROS2	2	2	2(1)
ROWDA3	3	2	3(2)
ROS3P	3	2	3(2)
RODASP	6	6	4(3)

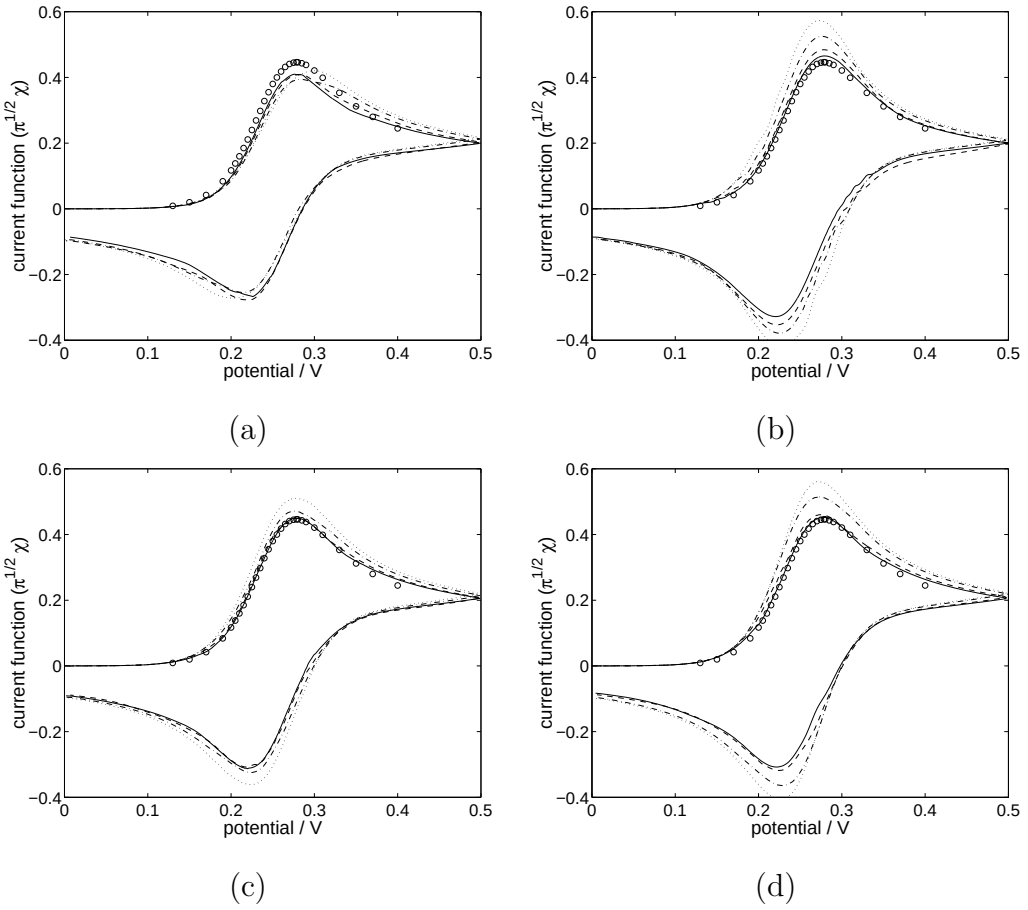
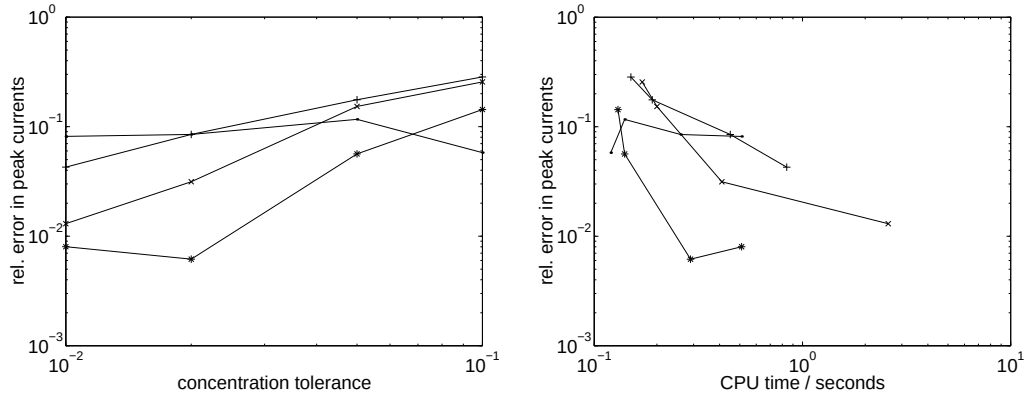


Figure 8. Simulated cyclic voltammograms for the Rosenbrock schemes ROS2 (a), ROWDA3 (b), ROS3P (c) and RODASP (d) combined with the multilevel FE approach. Relative error tolerances TOL: 10% (dotted), 5% (dash-dotted), 2% (dashed) and 1% (solid). The reference values (circles) are taken from[50].

to further refinements or coarsenings. Thus, a dominating error fraction that appears on the first element cannot be reduced by the refinement process. This leads to a maximum number of refinement cycles at certain time points and, therefore, high computational costs. Even though, we observe satisfying current approximations for RODASP.

Some more characteristics of the algorithmic behaviour during simulation are given in Figure 10. Figures 10(a) and (b) clearly show the correlation between estimated time error (a) and step size variation (b). High error estimates result in step size reduction, while low errors lead to increases in τ. In the case of RODASP and ROS3P larger τ values are chosen than for ROS2 and ROWDA3. If we compare the third order methods, ROS3P and ROWDA3, which obey the same number of stages as well as function evaluations (Table 1), we see that in particular the choice of a larger step size is crucial with respect to the overall efficiency of the solver (see also Figure 9(b)).

Spatial error estimates and the corresponding grid evolution are shown in Figures 10(c) and (d). Except for RODASP, all solvers achieve the prescribed



(a) (b)

Figure 9. Relative errors in peak current approximations (a) and efficiencies (b) of the four Rosenbrock solvers, ROS2 ($\cdot$), ROWDA3 ($+$), ROS3P ($*$) and RODASP ($\times$); data for different values of TOL (10, 5, 2, 1 %).

spatial error tolerance at all times. Note that the number of degrees of freedom of ROS3P exceeds those of ROWDA3. This demonstrates on the one hand that the temporal and spatial error estimates are coupled (larger step sizes result in a higher number of grid nodes and vice versa). On the other hand, it seems to be more efficient if the solver introduces more grid nodes, if this results in larger time step sizes.

6 Computational Aspects

The software development of all components discussed here was conducted under the SuSE Linux 8.1 and 9.1 operating systems (SuSE Linux, Nürnberg, Germany[67]) with the g++ compiler (versions 3.2 and 3.3) from the free GNU Compiler Collection[68]. The CPU times given in Figure 9 have been measured by the STL clock() function[61] with a Pentium M 1.3 GHz processor under the SuSE Linux 8.1 system. For the solution of the linear equation systems as well as other standard matrix computations, we employed the GMM++ library[69]. The implementation of grid refinements and coarsenings was based on tree.hh[66]. Calculations with physical quantities were performed with the Quantity library[70]. The software developed within the present work is licenced under the GPL (GNU General Public Licence[71]) and can be obtained freely[9].

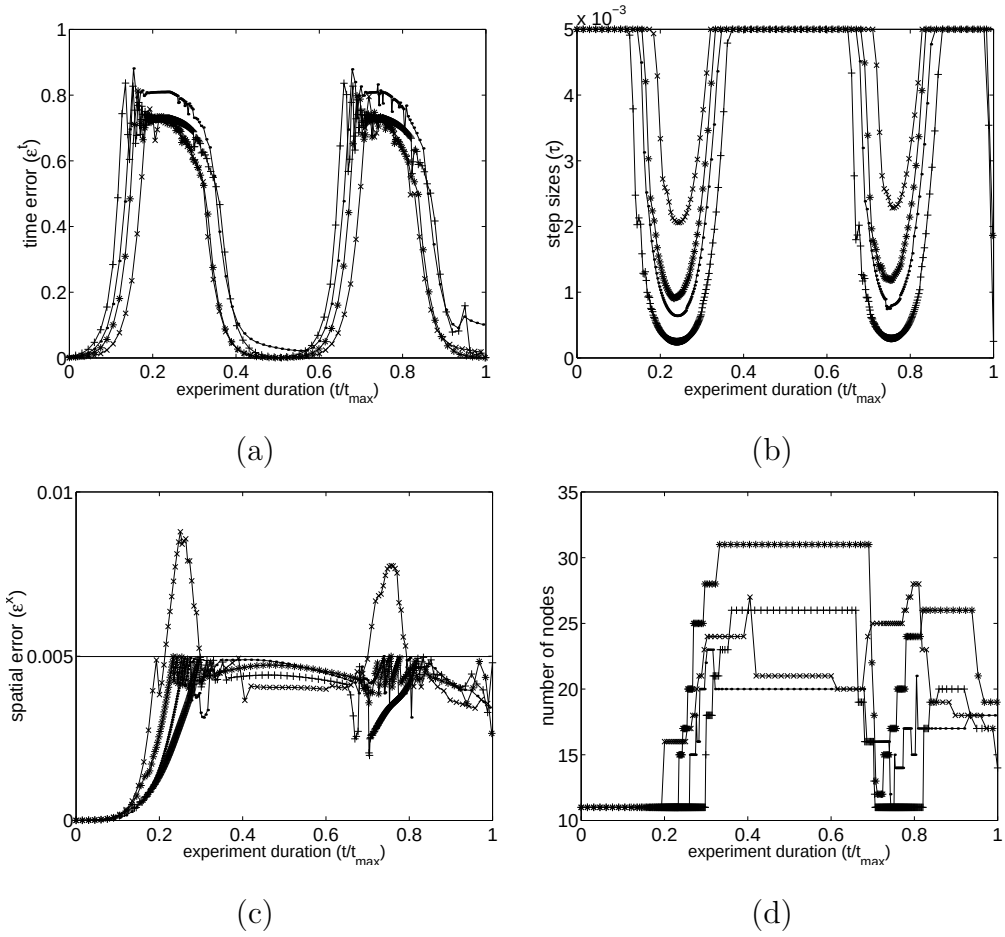


Figure 10. Characteristics of the solvers, ROS2 ($\cdot$), ROWDA3 ($+$), ROS3P ($*$) and RODASP ($\times$), at a tolerance of 1%: estimated time errors (a) and step sizes (b), a posteriori spatial error estimates (c), as well as grid evolution (d).

7 Conclusions

In the present work, the application of an adaptive multilevel FE algorithm based on the work of Lang[10] to general electrochemical simulation models is described. The fully adaptive solution process is controlled by Rosenbrock solvers in time and a hierarchical error estimator in space. Thus, in contrast to non-adaptive integrators, an error estimation of concentrations within the diffusion layer is possible. The behaviour of four different Rosenbrock schemes is analyzed for a simple electrode process test system (1D semi-infinite diffusion model, reversible electron transfer, cyclic voltammetry).
The intuitive assumption that an accurate approximation of concentration profiles during simulation also yields acceptable current calculation results is confirmed. The tolerance prescribed for the temporal and spatial error estimators working on the concentration profiles is achieved also within the quantity of interest, i.e. the current.

In particular, the Rosenbrock solver ROS3P yields excellent results. Compared to the reference system[50], very good agreement can be found for a tolerance lower than 5%. Except for ROS2, the other tested Rosenbrock methods, ROWDA3 and RODASP, yield satisfying results, while being less efficient.
The algorithm has been coupled to the electrochemical compiler Ecco. Thus, the simulation of a variety of electrochemical processes, under semi-infinite and finite 1D diffusion conditions should be easily possible. The behaviour of the algorithm under such conditions will be explored in further work.
Although the underlying C++ code is fully object-oriented, results for the test system can be obtained even for the lowest error tolerance of 1% within approximately 0.5 seconds on a contemporary computer.
The results of this paper for 1D models indicate that the described fully adaptive solver provides an alternative to the FE algorithm relying on the error estimation for the current only[42]. However, in general, the current approximation from a fixed linear finite element interpolation (equation (47)) can lead to convergence problems. Work on an alternative strategy is in progress.
The efficiency of the algorithm easily allows the simulation of transient experiments, while the object-oriented approach ensures simple coupling to all mechanistic varieties supported by the Ecco compiler[8].
In the context of electrochemical simulation, the Rothe-Rosenbrock strategy is no longer limited to adaptive finite difference calculations[11], and extensions to 2D and 3D systems may be attempted.

8 Acknowledgements

We acknowledge e-mail correspondence with Yves Renard about the GMM++ library and financial support of the Deutsche Forschungsgemeinschaft, Bonn-Bad Godesberg, Germany within the Forschergruppe *Analysis and modeling of diffusion/dispersion-limited reactions in porous media*.

Appendix A

To derive the weak, or variational form (28), elliptic stage problem (16) is multiplied by a so-called trial function, v, and integrated over interval $\bar{\mathcal{I}} =$

$[0,1]$.

$$\begin{aligned}
&\tfrac{1}{\tau_n\gamma}\int_0^1 U_{ni}\cdot v\ dx - \mathbf{D}\cdot\int_0^1 \partial_{xx}U_{ni}\cdot v\ dx - \int_0^1 R_u(u_n)U_{ni}\cdot v\ dx \\
&\qquad = \mathbf{D}\cdot\int_0^1 \partial_{xx}U_i\cdot v\ dx + \int_0^1 R(U_i)\cdot v\ dx - \textstyle\sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_n}\int_0^1 U_{nj}\cdot v\ dx
\end{aligned} \tag{49}$$

Integration by parts yields

$$\begin{aligned}
&\tfrac{1}{\tau_n\gamma}\int_0^1 U_{ni}\cdot v\ dx + \mathbf{D}\cdot\int_0^1 \partial_x U_{ni}\cdot\partial_x v\ dx - \int_0^1 R_u(u_n)U_{ni}\cdot v\ dx \\
&\underbrace{-\mathbf{D}\cdot[\partial_x U_{ni}\cdot v|_{x=1} - \partial_x U_{ni}\cdot v|_{x=0}]}_{(*)} \\
&\qquad = -\mathbf{D}\cdot\int_0^1 \partial_x U_i\cdot\partial_x v\ dx + \int_0^1 R(U_i)\cdot v\ dx - \textstyle\sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_n}\int_0^1 U_{nj}\cdot v\ dx \\
&\qquad \underbrace{+\mathbf{D}\cdot[\partial_x U_i\cdot v|_{x=1} - \partial_x U_i\cdot v|_{x=0}]}_{(\star)}
\end{aligned} \tag{50}$$

where boundary conditions (17) and (18) have to be introduced in place of terms $(*)$ and $(\star)$. For weak form (50) to be well posed, functions U_{ni}, U_i and v must fulfill certain continuity requirements, i.e. they must belong to the Sobolev space $H^1(\mathcal{I}) = \{u \in L^2(\mathcal{I}) : \partial_x u \in L^2(\mathcal{I})\}$ of *square integrable* functions, where $L^2(\mathcal{I}) = \{u : \int_0^1 u^2 dx < \infty\}$.[15, 17]
By approximating U_{ni}, U_i and v with the interpolation polynomials, ϕ, that span FE space $\mathcal{V}_{n+1,k}$ (i.e. replacing $H^1(\mathcal{I})$ by $\mathcal{V}_{n+1,k}$),

$$\begin{aligned}
U_{ni} &\approx \textstyle\sum_{l=1}^{N_k}\mu_{ni,l}\phi_l \\
U_i &\approx \textstyle\sum_{l=1}^{N_k}\mu_{i,l}\phi_l \\
v &\approx \textstyle\sum_{m=1}^{N_k}\nu_{ni,m}\phi_m
\end{aligned} \tag{51}$$

where N_k denotes the total number of grid nodes at refinement level k, equation (50) becomes

$$\underbrace{\left(\frac{1}{\tau_n\gamma}\mathbf{M} + \mathbf{D}\cdot\mathbf{S} - \mathbf{J}_n\right)}_{=\mathbf{L}_n} U_{ni,k} = -\mathbf{D}\cdot\mathbf{S}\cdot U_{i,k} + \mathbf{R}_i - \textstyle\sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_n}\mathbf{M}\cdot U_{nj,k} \tag{52}$$

with mass and stiffness matrices,

$$\mathbf{M} = \left(\int_0^1 \phi_l \phi_m \, dx \right)_{l,m=1}^{N_k} \qquad \mathbf{S} = \left(\int_0^1 \partial_x \phi_l \, \partial_x \phi_m \, dx \right)_{l,m=1}^{N_k} \tag{53}$$

and Jacobian,

$$\mathbf{J}_n = \left(\mathbf{J}_n^{(p,q)} \right)_{p,q=0}^{N-1} \qquad \mathbf{J}_n^{(p,q)} = \left(\int_0^1 \frac{\partial R_n^{(p)}}{\partial u^{(q)}} \phi_l \phi_j \, dx \right)_{l,m=1}^{N_k} \tag{54}$$

where $\mathbf{J}_n^{(p,q)}$ denotes the coupling term between species p and q. Vector

$$\mathbf{R}_i = \left(\int_0^1 R(U_i) \phi_l \, dx \right)_{l=1}^{N_k} \tag{55}$$

denotes the discretized reaction terms. Both Jacobian and reaction terms are integrated numerically by means of Newton-Cotes formulas[13]. Vectors $U_{ni,k}$ and $U_{i,k}$ stand for the discretized stage values

$$U_{ni,k} = \left(\mu_{ni,1}^{(0)}, ..., \mu_{ni,N_k}^{(0)}, \mu_{ni,1}^{(1)}, ..., \mu_{ni,N_k}^{(1)}, ..., \mu_{ni,1}^{(N-1)}, ..., \mu_{ni,N_k}^{(N-1)} \right) \tag{56}$$

and internal values, respectively. The linear systems of equations (28) and (52) respectively, are solved by means of LU decomposition[69] of the system matrix, $\mathbf{L}_n$ (since $\mathbf{L}_n$ is sparse, the use of more efficient direct solvers might be possible[72]).
In equation (28) we use the variational form (50) or its discrete form (52) supplied with appropriate boundary conditions.

References

[1] E.P. Sapozhnikova, M. Bogdan, B. Speiser, W. Rosenstiel, submitted for publication.

[2] A.J. Bard, L.R. Faulkner, Electrochemical Methods. Fundamentals and Applications, 2nd Edition, Wiley, New York, 2001.

[3] F. Scholz (Ed.), Electroanalytical Methods. Guide to Experiments and Applications, Springer, Berlin, 2002.

[4] B. Speiser, Methods to Investigate Mechanisms of Electroorganic Reactions, in: A.J. Bard, M. Stratmann, H. Schäfer (Eds.), Encyclopedia of Electrochemistry, Vol. 8 Organic Electrochemistry, Wiley-VCH, 2004, Ch. 1, pp. 1 – 23.

[5] A.W. Bott, S.W. Feldberg, M. Rudolph, Curr. Sep. 13 (1995) 108 – 112.

[6] L.K. Bieniasz, Computers Chem. 21 (1997) 1 – 12.

[7] K. Ludwig, L. Rajendran, B. Speiser, J. Electroanal. Chem. 568 (2004) 203 – 214, Erratum:[73].

[8] K. Ludwig, B. Speiser, J. Chem. Inf. Comput. Sci. 44 (2004) 2051 – 2060.

[9] `http://echempp.sourceforge.net` (July 2005).

[10] J. Lang, Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems. Theory, Algorithm and Applications, in: M. Griebel, D.E. Keyes, R.M. Nieminen, D. Roose, T. Schlick (Eds.), Lecture Notes in Computational Science and Engineering, Vol. 16, Springer, Berlin, 2000.

[11] L.K. Bieniasz, J. Electroanal. Chem. 469 (1999) 97 – 115.

[12] L.K. Bieniasz, J. Electroanal. Chem. 481 (2000) 115 – 133.

[13] K.H. Huebner, D.L. Dewhirst, D.E. Smith, T.G. Byrom, The Finite Element Method for Engineers, 4th Edition, Wiley, New York, 2001.

[14] H.R. Schwarz, Methode der Finiten Elemente, Teubner, Stuttgart, 1991.

[15] D. Braess, Finite Elemente - Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie, Springer, Berlin, 1992.

[16] J.N. Reddy, D.K. Gartling, The Finite Element Method in Heat Transfer and Fluid Dynamics, CRC Press, Boca Raton, Florida, 1994.

[17] T.J.R. Hughes, The Finite Element Method, Linear Static and Dynamic Finite Element Analysis, Dover Publications, Mineola, New York, 2000.

[18] I. Babuska, W.C. Rheinboldt, SIAM J. Num. Anal. 15(4) (1978) 736 – 754.

[19] I. Babuska, A. Miller, Int. J. Num. Meth. Eng. 20 (1984) 1085 – 1109.

[20] I. Babuska, A. Miller, Int. J. Num. Meth. Eng. 20 (1984) 1111 – 1129.

[21] I. Babuska, A. Miller, Int. J. Num. Meth. Eng. 20 (1984) 2311 – 2324.

[22] R.E. Bank, A. Weiser, Math. Comp. 44(107) (1985) 283 – 301.

[23] W. Gui, I. Babuska, Numer. Math. 49 (1986) 577 – 612.

[24] W. Gui, I. Babuska, Numer. Math. 49 (1986) 613 – 657.

[25] W. Gui, I. Babuska, Numer. Math. 49 (1986) 659 – 683.

[26] F.A. Bornemann, Imp. Comp. Sci. Eng. 2 (1990) 279 – 317.

[27] P. Leinen, Ein Schneller Adaptiver Löser für Elliptische Randwertprobleme auf Seriell- und Parallelrechnern, Dissertation, Universität Dortmund, 1990.

[28] K. Eriksson, C. Johnson, SIAM J. Num. Anal. 28(1) (1991) 43 – 77.

[29] J. Lang, A. Walter, Imp. Comp. Sci. Eng. 4 (1992) 269 – 314.

[30] M. Ainsworth, J.T. Oden, Numer. Math. 65 (1993) 23 – 50.

[31] R.E. Bank, R.K. Smith, SIAM J. Num. Anal. 30(4) (1993) 921 – 935.

[32] P. Leinen, Computing 55 (1995) 325 – 354.

[33] K. Eriksson, C. Johnson, SIAM J. Num. Anal. 32(6) (1995) 1729 – 1749.

[34] R.E. Bank, Acta Numerica (1996) 1 – 43.

[35] R. Verfürth, Numer. Math. 78 (1998) 479 – 493.

[36] P.K. Moore, SIAM J. Sci. Comp. 21(4) (2000) 1567 – 1586.

[37] M. Ainsworth, J.T. Oden, A Posteriori Error Estimation in Finite Element Analysis, in: Pure and Applied Mathematics, Wiley, New York, 2000.

[38] T. Nann, J. Heinze, Electrochem. Comm. 1 (1999) 289 – 294.

[39] T. Nann, J. Heinze, Electrochim. Acta 48 (2003) 3975 – 3980.

[40] I. Henley, A. Fisher, Electroanalysis 17 (2005) 255 – 262.

[41] K. Harriman, D.J. Gavaghan, P. Houston, E. Süli, Electrochem. Commun. 2 (2000) 150 – 156.

[42] K. Harriman, D.J. Gavaghan, P. Houston, E. Süli, Electrochem. Commun. 2 (2000) 157 – 162.

[43] S.C.B. Abercrombie, G. Denuault, Electrochem. Comm. 5 (2003) 647 – 656.

[44] K. Harriman, D.J. Gavaghan, E. Süli, Electrochem. Comm. 5 (2003) 519 – 529.

[45] K. Harriman, D.J. Gavaghan, E. Süli, J. Electroanal. Chem. 569 (2004) 35 – 46.

[46] K. Harriman, D.J. Gavaghan, E. Süli, J. Electroanal. Chem. 573 (2004) 169 – 174.

[47] P. Deuflhard, P. Leinen, H. Yserentant, IMPACT Comp. Sci. Eng. 1 (1989) 3 – 35.

[48] V. Mirčeski, J. Phys. Chem. B 108 (2004) 13719 – 13725.

[49] D. Britz, Digital Simulation in Electrochemistry, 2nd Edition, Springer, Berlin, 1988.

[50] R.S. Nicholson, I. Shain, Anal. Chem. 36 (1964) 706 – 723.

[51] J. Lang, Appl. Numer. Math 26 (1998) 105 – 116.

[52] J. Lang, J.G. Verwer, ROS3P - An Accurate Third-Order Rosenbrock Solver Designed for Parabolic Problems, Tech. Rep. MAS-R0013, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands (2000).

[53] J. Lang, J. Verwer, BIT 41 (2001) 731 – 738.

[54] J. Lang, W. Cao, W. Huang, R.D. Russel, Appl. Numer. Math. 46 (2003) 75 – 94.

[55] A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, Benchmarking Stiff ODE Solvers for Atmospheric Chemistry Problems II: Rosenbrock Solvers, Tech. Rep. NM-R9614, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands (1996).

[56] A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, F.A. Potra, Atmosph. Environm. 31 (1997) 3459 – 3472.

[57] P. Deuflhard, F. Bornemann, Scientific Computing with Ordinary Differential Equations, in: J.E. Marsden, L. Sirovich, M. Golubitsky, S.S. Antman (Eds.), Texts in Applied Mathematics, Vol. 42, Springer, New York, 2002.

[58] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Elsevier, New York, 1989.

[59] K. Gustafsson, ACM Trans. Math. Software 20(4) (1994) 496 – 517.

[60] S.H. Friedberg, A.J. Insel, L.E. Spence, Linear Algebra, Prentice Hall, New Jersey, USA, 2003.

[61] B. Stroustrup, The C++ Programming Language, Addison-Wesley, Reading, 1997.

[62] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley, Boston, 1995.

[63] S. Meyers (Ed.), Effective C++: 50 Specific Ways to Improve Your Programs and Designs, Addison-Wesley, Reading, 1998.

[64] W.H. Press, S.A.Teukolsky, W.T. Vetterling, B.P. Fannery, Numerical Recipes in C++, The Art of Scientific Computing, Second Edition, Cambridge University Press, Cambridge, 2002.

[65] D. Vandevoorde, N.M. Josuttis, C++ Templates. The Complete Guide, Addison-Wesley, Boston, 2003.

[66] `http://www.aei.mpg.de/~peekas/tree` (November 2005).

[67] `http://www.novell.com/linux/suse/index.html` (March 2005).

[68] `http://www.gnu.org` (March 2005).

[69] `http://www.gmm.insa-tlse.fr/getfem/gmm_intro` (March 2005).

[70] `http://sourceforge.net/projects/quantity` (March 2005).

[71] http://opensource.org/licenses/gpl-license.php (March 2005).

[72] http://crd.lbl.gov/~xiaoye/SuperLU (June 2005).

[73] K. Ludwig, L. Rajendran, B. Speiser, J. Electroanal. Chem. 571 (2004) 119.