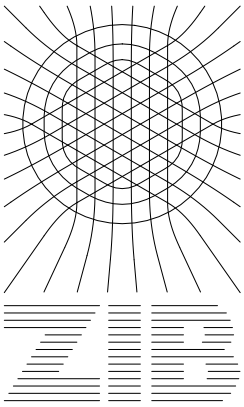
Konrad-Zuse-Zentrum für Informationstechnik Berlin



Ulrich Nowak

A Fully Adaptive MOL–Treatment of Parabolic 1D–Problems with Extrapolation Techniques

Preprint SC 95–25 (December 1995)

Ulrich Nowak

A Fully Adaptive MOL–Treatment of Parabolic 1D–Problems with Extrapolation Techniques

Abstract. A fully adaptive method is presented for the numerical solution of highly nonlinear, coupled systems of parabolic differential equations in one space dimension. Time discretization is by means of the linearly–implicit Euler discretization. Space discretization is by finite differences on non–uniform grids. Both basic discretizations are combined with extrapolation. Based on local error estimates for both the time and the space discretization error, the accuracy of the numerical approximation is controlled and the discretization stepsizes are adapted automatically and simultaneously. The algorithm is implemented in a user friendly software package, PDEX1M. To be a powerful tool for users coming from applications the package has been equipped with some additional useful devices.

1 Introduction

Consider a system of $n_{\rm pde}$ partial differential equations (PDEs) of reaction–diffusion type with possibly mild convection,

$$B(x,t,u,u_x)u_t = f(x,t,u,u_x,(D(x,t,u)u_x)_x), \tag{1}$$
$$u(x,t_0) = u_0(x), \tag{2}$$
$$\alpha(x)u + \beta(x,t,u)u_x = \gamma(x,t,u) \text{ or } u_t = \delta(x,t,u), \tag{3}$$
$$x \in [x_{\rm L}, x_{\rm R}]\ ,\ \ t \in [t_0, t_{\rm end}]. \tag{4}$$

The matrix B may be singular but we assume that, after space discretization, the resulting system of differential–algebraic equations (DAEs) is of index ≤ 1. A popular approach to solving (1)–(4) is a method–of–lines (MOL) treatment. In this approach, space discretization is applied first and the resulting large semi–discrete system of DAEs is then solved by an adaptive time integrator of BDF, Runge–Kutta or extrapolation type. In a classical MOL treatment, the spatial mesh is fixed throughout the whole integration. However, except for very simple problems, an efficient and robust solution procedure requires frequent adaptation of the space discretization during the time integration. There are three major strategies for spatial adaptation – mesh refinement and coarsening (static regridding), mesh motion (moving grids) and order variation. A large number of methods have been developed which use one or more of these techniques, see, for example, [1, 5, 7, 10, 11,

14, 16]. However, only a few methods use an integrated adaptation strategy which is based on comparable a posteriori error estimates in space and time.

In this paper, we employ extrapolation techniques in space and time. The basic space discretization is done by means of standard finite differences, see, for example, [18]. For time discretization, we apply the linearly–implicit Euler method [3, 4]. Because of its linearly–implicit one–step structure, this method avoids the solution of nonlinear systems and allows an easy change of the computational grids after each time step.

This paper is intended to give an overview of the current state of the method proposed in [15] (Sections 2, 3.1), to present some new features which make the associated software package, PDEX1M, an interesting tool for users (Section 4), and to demonstrate the algorithmic behavior by means of some numerical experiments (Section 5).

2 Basic Discretizations

2.1 Time Discretization

For ease of presentation, we first consider the time discretization method applied to a system of DAEs of the form $B(t,y)y' = f(t,y)$, $y(t_0)$ given. To determine an approximation y_k to $y(t_{\mathrm{R}})$ at $t_{\mathrm{R}} = t_{\mathrm{L}} + k\Delta t$, k steps of the basic linearly–implicit Euler discretization are applied:

$$(B_l - \Delta t A)(y_{l+1} - y_l) = \Delta t f(t_{l+1}, y_l)\ , \quad l = 0, 1, \ldots, k-1, \tag{5}$$

with $y_0 = y(t_{\mathrm{L}})$, where $A = (f - By')_y$, the Jacobian matrix of the residual at $t = t_{\mathrm{L}}$, or a sufficiently good approximation to it. Under the assumption that there exists an asymptotic Δt–expansion for the approximation error,

$$y_k - y(t_{\mathrm{R}}) = e_1(t_{\mathrm{R}})\Delta t + e_2(t_{\mathrm{R}})\Delta t^2 + e_3(t_{\mathrm{R}})\Delta t^3 + \ldots + O(\Delta t^{N+1}), \tag{6}$$

Richardson extrapolation can be applied to construct approximations of higher order. In general, only perturbed expansions exist and this may restrict the maximum attainable (classical) order. Nevertheless, extrapolation can be applied successfully as further extrapolation still reduces the approximation error [12].

Assume that a so–called basic stepsize ΔT and an associated expected extrapolation stage number s is given. Then, one constructs approximations to $y(t_{\mathrm{L}} + \Delta T)$ using the discretization method (5) with stepsizes $\Delta t_j = \Delta T/n_j$, $j = 1, \ldots, \kappa = s+1$, where $\{n_j\}$ is the step number sequence. For the linearly–implicit Euler method, the harmonic sequence $\{n_j\} = \{1, 2, 3, \ldots\}$

may be applied. Denoting the approximation obtained at t_{R} with internal stepsize Δt_j by $T_{j,1} = y(t_{\mathrm{L}} + \Delta T; \Delta t_j)$, the t–extrapolation tableau is

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{(n_j/n_{j-k+1}) - 1} \ , \ \ 2 = 1, \ldots, \kappa \, , \ 2 = 1, \ldots, j. \tag{7}$$

A general order and stepsize control technique for extrapolation methods has been derived in [2] and is used, except for minor changes and additions, in the time discretization part of our method. Its essential features are:

- *Abstract Convergence Model:* Models the average convergence behavior.
- *Subdiagonal Error Estimate:* Componentwise error estimates are obtained from the subdiagonal difference $E_{j-1} = T_{j,j-1} - T_{j,j}$.
- *Convergence Monitor:* For $j = 2, \ldots, \kappa$, there is a check if convergence can be expected at least in stage $s + 1$.
- *Order Window:* The convergence check $\epsilon_{j-1} = ||E_{j-1}|| \leq \mathrm{tol}_t$ is performed only for $j = \kappa - 1, \kappa, \kappa + 1$.
- *Order and Stepsize Optimization:* After a successful step (say within s stages), the following stepsize estimates (predictors) are calculated:

$$\Delta T_k^{\mathrm{new}} = \sqrt[k+1]{\mathrm{tol}_t/\epsilon_k} \, \Delta T \ , \ \ k = 1, \ldots, s \ . \tag{8}$$

In case of step failure, (8) is used to compute a stepsize corrector. A stepsize estimate for the stage $s + 1$ is based on the estimate ϵ_s and the abstract convergence model. The optimal stepsize/stage combination is derived by minimizing the (estimated) amount of work per unit step. So, with proposed values $\Delta T^{\mathrm{new}}, s^{\mathrm{new}}$ and with the approximation $T_{s+1,s+1}$ as starting value, the integration is continued.

2.2 Space Discretization

To discretize the space derivative terms in (1), centered 3–point finite difference approximations are applied on non–uniform grids. Consider a grid $\mathcal{G} = \{x_{\mathrm{L}} = x_1 < \ldots < x_{n_x} = x_{\mathrm{R}}\}$ of dimension n_x with nodes x_i and subintervals $\Delta x_i = x_{i+1} - x_i$. The first derivative vector u_x is approximated (componentwise) by

$$\frac{\partial u(x_i, t)}{\partial x} \approx \frac{1}{\Delta x_{i-1} + \Delta x_i} \left[\Delta x_{i-1} \frac{u_{i+1} - u_i}{\Delta x_i} + \Delta x_i \frac{u_i - u_{i-1}}{\Delta x_{i-1}} \right] . \tag{9}$$

Except for the approximation (9), the same space discretization as presented in [18] is used. On uniform grids, this finite difference (FD) approximation is

second order accurate. On non–uniform grids, however, the local truncation error is formally only linear in the discretization stepsizes Δx_i. In order to overcome this formal difficulty and to derive an analogue of the basic stepsize ΔT of the time discretization, one may introduce (see e.g. [17]) a sufficiently differentiable virtual grid function $\xi(r)$ on $[0,1]$ which is implicitly defined by

$$\begin{aligned} x_i &= \xi(r_i)\ ,\ \ i = 1, ..., n_x \\ r_i &= (i-1)\Delta r\ ,\ \Delta r = 1/(n_x - 1)\ ,\ \ i = 1, \ldots, n_x\ . \end{aligned}$$

The following expansion for the local trunction error of the FD-approximation of (1) at each interior node x_i can be derived:

$$\tau^{\Delta r}(x_i, t) = \sum_{k=1}^{K/2} \Delta r^{2k} \Xi_k(x_i, t)\ +\ O(\Delta r^{K+1})\ , i = 2, ..., n_x - 1\ .$$

The error order at the boundary nodes may be perturbed. However, provided that the problem is of parabolic nature, these local perturbations of the truncation error are damped out within each time step and do not appear in the approximation error of the step.

3 Extrapolation in Space and Time

3.1 Error Estimates

First, we assume that a grid function $\xi(r)$ is given explicitly. Basic stepsizes ΔT and $\Delta R = 1/(n_x - 1)$ as well as prescribed numbers of subdivisions, say $\bar{j}$ and $\bar{\mu}$, may be given. Furthermore, a space step sequence $\{m_\mu\}$ is required, e.g. $\{1, 2, 4, \ldots\}$. Then, similar to the time stepsize subdivision, a sequence of computational grids

$$\mathcal{G}^\mu = \{x_i^\mu | x_i^\mu = \xi(r_i), r_i = (i-1)\Delta r_\mu, i = 1, \ldots, m_\mu(n_x - 1) + 1\}$$

can be constructed. Now, integrating from t_{L} to t_{R} with method (5) for a sequence of time stepsizes $\Delta t_j = \Delta T/n_j,\ j = 1, \ldots, \bar{j}$ on all grids $\mathcal{G}^\mu,\ \mu = 1, \ldots, \bar{\mu}$ yields approximations

$$\{U(x_i^\mu, t_{\mathrm{R}}; \Delta r_\mu, \Delta t_j)\}_{\mu=1,\ldots,\bar{\mu}}^{j=1,\ldots,\bar{j}}\ .$$

For the approximation errors, at best a perturbed expansion can be derived. Furthermore, even for quite simple problems, severe order restrictions exist [13]. Here, we assume that the following expansion describes the essential

error behavior of one integration step:

$$U(x_i^\mu, t_\mathrm{R}; \Delta r_\mu, \Delta t_j) - u(x_i^\mu, t_\mathrm{R}) \;=\; \sum_{\rho=0,l=0;(\rho,l)\neq(0,0)}^{\rho=M,l=N} e_{\rho,l}(x_i^\mu, t_\mathrm{R}) \Delta r_\mu^{2\rho} \Delta t_j^l + \; O(\Delta r_\mu^{2M+1}) + O(\Delta t_j^{N+1}) \; . \quad (10)$$

Numerical experiments show that the dominant error terms behave according to (10); see [15].

As an extension of the notation for one dimensional extrapolation, we identify $T_{\mu,1,j,1}$ with $U(\bar{x}_i, t_\mathrm{R}; \Delta r_\mu, \Delta t_j)$. As r–extrapolation is possible only at common nodes, say $\{\bar{x}_i\}$, this identification may include an appropriate restriction operation. For a fixed index pair $(\mu, 1)$, the t–extrapolation process is applied to construct approximations $T_{\mu,1,j,k}$. For a fixed pair (j,k), r–extrapolation uses the scheme

$$T_{\mu,\nu,j,k} = T_{\mu,\nu-1,j,k} + \frac{T_{\mu,\nu-1,j,k} - T_{\mu-1,\nu-1,j,k}}{(m_j/m_{\mu-\nu+1})^2 - 1} \; , \; \mu = 2, \ldots, \bar{\mu}; \; \nu = 2, \ldots, \mu. \quad (11)$$

The sequences of t– and r–extrapolations are interchangeable, that is, t–extrapolation may be applied to any fixed (μ, ν) pair. Using (10), the leading error terms of the approximations $T_{\mu,\nu,j,k}$ can be derived:

$$T_{\mu,\nu,j,k} - u(\bar{x}_i, t_\mathrm{R}) \;\doteq\; (-1)^{\nu+1} \beta_{\mu,\nu} e_{\nu,0}(\bar{x}_i, t_\mathrm{R}) \Delta R^{2\nu} + (-1)^{k+1} \gamma_{j,k} e_{0,k}(\bar{x}_i, t_\mathrm{R}) \Delta T^k, \quad (12)$$

with $\beta_{\mu,\nu} = (m_{\mu-\nu+1} \cdot \ldots \cdot m_\mu)^{-2}$ and $\gamma_{j,k} = (n_{j-k+1} \cdot \ldots \cdot n_j)^{-1}$. The leading mixed error term ($\sim \Delta R^{2\nu} \Delta T^k$) is neglected as it is usually small compared to the other terms. In the stepsize control algorithm, this term is taken into account, but for ease of presentation we suppress this rather technical extension of the algorithm described below.

The general r–extrapolation scheme has two drawbacks. First, as mentioned earlier, the highest r–extrapolated approximation is available only on the coarsest grid. So, to continue the integration, we must obtain sufficiently accurate initial values on all finer grids. Second, an explicit grid function is needed. As a consequence, only one space extrapolation is applied and the extrapolated approximations are used for the error check only. The unextrapolated solution on the fine grid is used to continue the integration (on both the fine and the coarse grids). The fine grid construction uses a uniform subdivision of each coarse grid interval. Thus, the associated virtual grid function is not the same as the one defined implicitly by the coarse grid. This difference can be interpreted as a small perturbation in the expansion (10) and can be neglected.

The algorithmic realization of our two grid extrapolation scheme is as follows. First, a full time integration step with t–extrapolation and convergence monitor is taken on the coarse grid. Using the error estimates $\hat{\epsilon}^t_{1,k-1} = ||T_{1,1,k,k-1} - T_{1,1,k,k}||$, the convergence monitor fixes κ, the number of subdivisions of ΔT to get t–convergence, i.e. $\hat{\epsilon}^t_{1,\kappa} \leq \mathrm{tol}_t$ holds. Then, the whole time integration process is repeated on the fine grid providing the approximations $T_{2,1,j,k}, j = 1, \ldots, \kappa; k = 1, \ldots, j$. This time, the convergence monitor is switched off. Using r–extrapolation, the approximations $T_{2,2,j,k}$ are calculated. The dominant space discretization error and a refined time discretization error are estimated by

$$a)\ \hat{\epsilon}^x = ||T_{2,1,\kappa,\kappa} - T_{2,2,\kappa,\kappa}||\ ,\ \ b)\ \hat{\epsilon}^t = ||T_{2,2,\kappa,\kappa-1} - T_{2,2,\kappa,\kappa}||, \tag{13}$$

respectively. Using (10), the leading error terms turn out to be

$$a)\ \epsilon^x \doteq 1/4||e_{1,0}(\bar{x}_i, t_{\mathrm{R}})||\Delta R^2\ ,\ \ b)\ \epsilon^t \doteq \gamma_{\kappa,\kappa-1}||e_{0,\kappa-1}(\bar{x}_i, t_{\mathrm{R}})||\Delta T^{\kappa-1}\ . \tag{14}$$

Integration is continued if both convergence checks

$$a)\ \hat{\epsilon}^x \leq \mathrm{tol}_x\ \text{ and }\ b)\ \hat{\epsilon}^t \leq \mathrm{tol}_t \tag{15}$$

are satisfied. Otherwise, the basic time stepsize is reduced.
All estimates are calculated using the weighted root–mean–square norm

$$||\varepsilon|| = \sqrt{\frac{1}{n_x}\sum_{i=1}^{n_x}\frac{1}{n_{\mathrm{pde}}}\sum_{j=1}^{n_{\mathrm{pde}}}\left(\frac{\varepsilon_{j,i}}{w_{j,i}}\right)^2}\ , \tag{16}$$

where $\varepsilon_{j,i}$ denotes the absolute error estimate of component j of U at the i–node. The weights $w_{j,i}$ are computed internally from

$$w_{j,i} = \max\{|u_{j,i}|, s^j_{\mathrm{abs}}\},$$

where s^j_{abs} are user prescribed thresholds. Special extensions of this simple weighting are available.

3.2 Stepsize Selection

As usual, we derive formulas for stepsizes ΔR, ΔT which would have been optimal in the current step, i.e. $\epsilon^x = \mathrm{tol}_x$, $\epsilon^t = \mathrm{tol}_t$, and use them for the next step. Upon expanding the leading error coefficients $e_{.,.}$ in (14) according to

$$e_{.,.}(\bar{x}_i, t_{\mathrm{R}}) = 0 + \Delta T\frac{\partial}{\partial t}e_{.,.}(\bar{x}_i, t_{\mathrm{L}}) + \ldots,$$

we derive proposed new values

$$a)\ \Delta R^{\text{new}} = \sqrt[2]{\frac{\text{tol}_x}{\hat{\epsilon}^x} \cdot \frac{\Delta T}{\Delta T^{\text{new}}}}\,\Delta R\ ,\quad b)\ \Delta T^{\text{new}} = \sqrt[\kappa]{\frac{\text{tol}_t}{\hat{\epsilon}^t}}\,\Delta T\ . \tag{17}$$

Here, (17b) is the usual formula for time step adaptation; cf. (8). Formula (17a), however, is not suited for a direct stepsize adaptation. We allow only a global coarsening ($\Delta R^{\text{new}} = 2\Delta R$), refinement ($\Delta R^{\text{new}} = \Delta R/2$) or reuse ($\Delta R^{\text{new}} = \Delta R$) of ΔR. Inserting these possible choices into (17a), three time stepsize restrictions for ΔT^{new} are derived. The optimal triple $(\Delta T^{\text{new}}_{s^{\text{new}}}, s^{\text{new}}, \Delta R^{\text{new}})$ is then determined by an extension of the principle of minimizing the work per unit step mentioned earlier. In order to adapt the spatial mesh locally, we also use the following equidistribution procedure. Local error estimates $\hat{\epsilon}^x_i$ of the form (13a) are calculated for each node of the coarse grid by an obvious nodal application of the weighted root–mean–square norm (16). Threshold values are defined by

$$\begin{aligned}
\sigma^+ &= \hat{\epsilon}^x_{\max}\ ,\ \text{if}\ \hat{\varepsilon}^x_{\max} \le \text{tol}_x,\\
\sigma^+ &= \sqrt{\hat{\epsilon}^x_{\max} \cdot \text{tol}_x}\ ,\ \ \text{if}\ \hat{\epsilon}^x_{\max} > \text{tol}_x,\\
\sigma^- &= 1/6\,\text{tol}_x\ ,
\end{aligned}$$

with $\hat{\epsilon}^x_{\max} = \max\{\hat{\epsilon}^x_i\}$. The general grid adaptation rules are

- if $\hat{\epsilon}^x_i < \sigma^-$: node x_i may be eliminated,
- if $\hat{\epsilon}^x_i \ge \sigma^-$: node x_i cannot be eliminated,
- if $\hat{\epsilon}^x_i > \sigma^+$: subdivide Δx_{i-1} and Δx_i.

Some heuristics are imposed to regularize the adaptation and the procedure is completed by a final smoothing process to obtain a quasi–uniform grid satisfying $\Delta x_i/\Delta x_{i+1} \in [1/q, q]$ with $q = 2.5$.

Solution approximations at inserted nodes are computed by means of local monotone piecewise cubic Hermite interpolation of [8].

In accordance with the well known methods for stiff integration (e.g. the codes LSODE, SDIRK4, EULSIM) we do not made any attempt to estimate or to control the global error propagation over the whole integration interval.

4 Special Features

4.1 Moving Grids

A quite popular and widely used approach to increase robustness and efficiency of an MOL treatment of PDE problems is the use of moving grids. One may distinguish two different categories of moving grid techniques:

- the nodes of the grid are moved such that the values of a certain monitor function are equidistributed in space; see, e.g., [1, 5].
- the nodes are moved to reduce the dynamics of the solution on the new coordinates; see, for example, [10, 14, 16].

As our method uses a static regridding device to control and equilibrate the spatial errors, introducing a moving grid with the second objective is quite natural. Introducing moving nodes, i.e. $x_i \to x_i(t)$, transforms the original system (1) into an augmented system of the form

$$\begin{aligned} B(x,t,u,u_x)(\dot{u} - u_x\dot{x}) &= f(x,t,u,u_x,(D(x,t,u)u_x)_x), && (18)\\ h(x,t,u,u_x,\ldots) &= 0\ . && (19) \end{aligned}$$

Several choices for the grid determining equation (19) have been proposed. A widely used approach is to minimize the (weighted) time derivative of the augmented system

$$\dot{u}^T\dot{u} + \alpha\dot{x}^2 = \min\ .$$

Solving this minimization problem and equipping the resulting equation with an additional regularization term (to avoid node crossing) yields the general grid determination equation for (19)

$$\dot{u}^T u_x + \alpha\dot{x} - \lambda x_{xx} = 0\ .$$

This type of equation has been used successfully in several methods. Unfortunately, often a problem depending tuning of the parameters α and λ is required. To have as few tuning parameters as possible in our moving grid technique, the α–term is omitted. With this *a priori* decision and λ in the range $[0.01, 0.25]$, quite good results for a set of typical test problem can be obtained. Even better results are obtained using special moving grid equations, e.g. fixing one node to a special solution value combined with special grid equations for all other nodes. Note that λ must be viewed as an internally scaled parameter. Changing units of the solution and/or the coordinates does not change the algorithmic performance.

4.2 Global Solution Representation

At least the graphical solution display requires a solution representation at points other than on the computational grid. Our global solution representation and evaluation scheme uses two quite different Hermite interpolation variants. We observe that doing interpolation first in time then in space gives better results than vice versa.

Interpolation in Time. The basic idea is to compute accurate approximations of solution derivatives at the endpoints of each integration interval by

extrapolation of divided differences of appropriate solution approximations. Properties and a general implementation of this approach have been studied in [9]. For an application in the adaptive MOL context, the basic algorithm must be modified as the space grid may change from step to step. As the problem is stiff, derivative information from only the right endpoint t_{R} is used. As, in general, a matrix B may appear in the left hand side of the problem formulation (1) the first time derivative can not be calculated directly via a relationship of the form $u' = f(u)$. Assume that κ lines of the time extrapolation tableau have been computed in the step, i.e. the error of the highest extrapolated solution approximation at t_R is proportional to $\Delta T^{\kappa+1}$. If we require an interpolation error of at least the same order, an interpolation polynomial with minimal degree κ is neccessary. Consequently, $(\kappa - 1)$ sufficiently accurate derivatives at t_R are required, that is, the error of the approximation to $u^{(k)}$ should be at least of order $\Delta T^{\kappa+1-k}$. This is achieved by forming the divided backward differences

$$r_j^{(k)} = \frac{1}{\Delta t_j^k} \nabla^k u_{n_j} \quad k = 1, \ldots, \rho \leq \kappa, \quad j = k, \ldots, \kappa. \tag{20}$$

For $r^{(k)}$, at best $\kappa + 1 - k$ different approximations are available. This allows extrapolating $(\kappa - k)$–times, yielding an approximation error proportional to $\Delta T^{\kappa+1-k}$, the minimal required order and the best attainable order under the given restrictions. So, the only choice for ρ in (20) is $\kappa - 1$. Using Newton's interpolation formula, the interpolating polynomial is constructed and by means of a Horner–like scheme evaluated efficiently.

Interpolation in Space. The piecewise cubic monotone Hermite interpolation, [8], is also used to interpolate for output purposes.

Output Generation. To free the user from writing code for output generation, some useful output options can be selected. For both time and space interpolation, four modes are available:

- interpolation at points uniformly distributed between t_0 and t_{end} (x_{L} and x_{R});
- interpolation at points uniformly distributed within each integration interval $[t_{\mathrm{L}}, t_{\mathrm{R}}]$ (grid interval Δx_i);
- interpolation such that the maximum interval between two successive interpolation points is less than or equal to a value $\Delta t_{\max}^{\mathrm{IP}}$ ($\Delta x_{\max}^{\mathrm{IP}}$);
- interpolation at user prescribed output points t_i^{IP} (x_i^{IP}).

Usually, the additional amount of work for constructing and evaluating the interpolants is small compared to the work for solving the PDE.

4.3 Root Finding

In principle, PDEX1M offers an option for locating zeros of switching functions

$$\phi_l(x,t,u(x,t)) = 0,\ l = 1,\ldots,n_{\text{swif}}\ . \tag{21}$$

In general, the roots of each switching function form a complicated curve in the $(x,t)-$plane and it is a challenging problem to approximate this curve accurately and efficiently. Instead of attacking this general problem, our root finder tries to determine all roots of the switching functions

$$\phi_l(\bar{x},t,u(\bar{x},t)) = 0,\ l = 1,\ldots,n_{\text{swif}},\ \bar{x} \in \mathcal{G}^s\ . \tag{22}$$

Here, $\mathcal{G}^s$ denotes a grid (of size n_x^s, say) usually the computational (fine) grid. Switching from (21) to (22) is nothing other than "discretizing" (21) over the grid $\mathcal{G}^s$. So the roots of $n_s = n_{\text{swif}} \cdot n_x^s$ switching functions of the form

$$\phi_l^i = \phi_l(t,u(\bar{x}_i,t)) = 0 \tag{23}$$

must be found. The root finding algorithm uses the continuous solution representation described above. First, it checks if there was a sign change in one of the switching functions (23). Then, for all index pairs (l,i) where the sign of ϕ_l^i has changed, a scalar Newton method is invoked to find the associated zero $t_{\text{zero}}^{l,i}$. Finally, the left–most zero $t_l = \min\{t_{\text{zero}}^{l,i}\}$ is determined. Depending on the user's choice, the integration is stopped at t_l or continued. Special care has been taken to realize this basic algorithm in a robust and efficient fashion but details are beyond the scope of our paper.

5 Numerical Examples

The numerical experiments have been carried out using the current version of the software package PDEX1M. For all results presented below, no specific adaptation of internal parameters was done. However, the package allows an easy modification of most of the internal parameters, for example λ, cf. Section 4.1. As well, the additional features just mentioned can be selected easily by setting appropriate option flags. A graphical user interface is currently under construction.

Example 1: Automobile Catalytic Converter

The model of an automobile catalytic converter [6] leads to a system of 11 PDEs of the general form (1)–(4). The 11 state variables are 3 temperatures (gas, converter surface and converter hull) and 2×4 species concentrations (gas, converter surface).

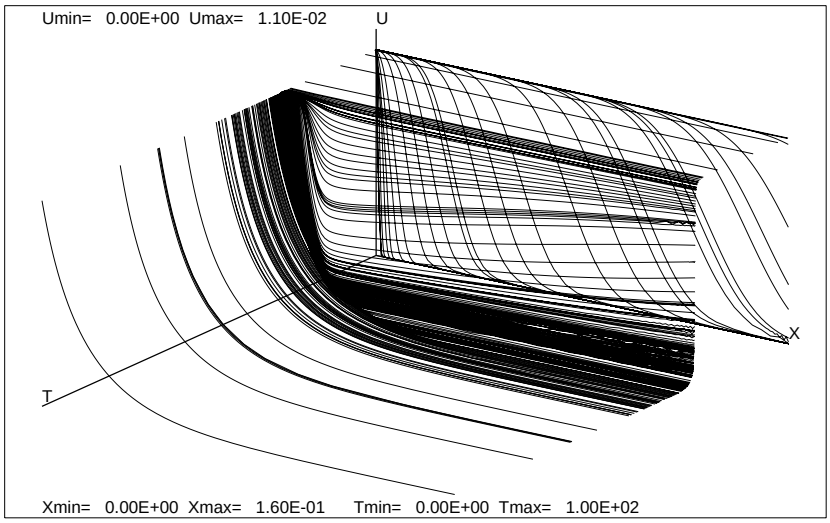


Figure 1: Solution component C_3H_6 (example 1)

To reduce air pollution, the study of the startup phase is of great importance. The simulation presented in this paper assumes that the polluted air enters with a linearly increasing temperature (and with constant velocity) at the left boundary. In Fig. 1, the concentration of the pollutant C_3H_6 is shown in the (x,t)–plane. The numerical solution on the computational grid is plotted at all internally selected integration points. The displayed time interval is $[0,100]$ (whereas the actual integration interval was $[0,1000]$).

Initially, the converter is rapidly filled with the inflowing gas. This appears in Fig. 1 as a front moving from left to right. To resolve this fast process ($t \in [0,0.2]$), small time steps and a grid with local refinements within the front position arise automatically. After the initial filling process is completed, the temperature is not yet high enough to start the reaction so that larger time steps and a coarse grid appear to be appropriate. At $t = 25$, the reaction starts and reduces the concentration of C_3H_6. Steep gradients at the left boundary require both small stepsizes in time and a very dense grid at the left boundary. At $t = 50$, the dynamics of the problem dies out and accordingly time steps increase. Finally, the process becomes nearly stationary allowing very large steps. The simulation from $t = 100$ to $t = 1000$ requires only 5 more steps.

Table 5 gives a comparison of the amount of work for solving this example with different prescribed tolerances $\mathrm{tol}_x = \mathrm{tol}_t = \mathrm{tol}$. In counting the number of function evaluations (for numerical Jacobian approximation (nfcnj) and discretization (nfcn)), LU–decompositions (ndec) and solves (nsol), we accumulate both the fine grid and the coarse grid calls. CPU–time (CPU) is in seconds (for SUN SPARC 10). Note that the average number of nodes on

the fine grid, $\bar{n}_x^F$, is proportional to $1/\sqrt{\text{tol}_x}$.

tol	CPU	nstep	nfcnj	nfcn	ndec	nsol	$\bar{n}_x^F$
$2.5 \cdot 10^{-3}$	167	150	7268	1275	1001	3086	83
$1.0 \cdot 10^{-3}$	295	176	8878	1804	1316	4511	115
$2.5 \cdot 10^{-4}$	720	208	10350	2760	1686	7122	205

Table 1: Amount of work for different tolerances (example 1)

Example 2: Shear Band Formation

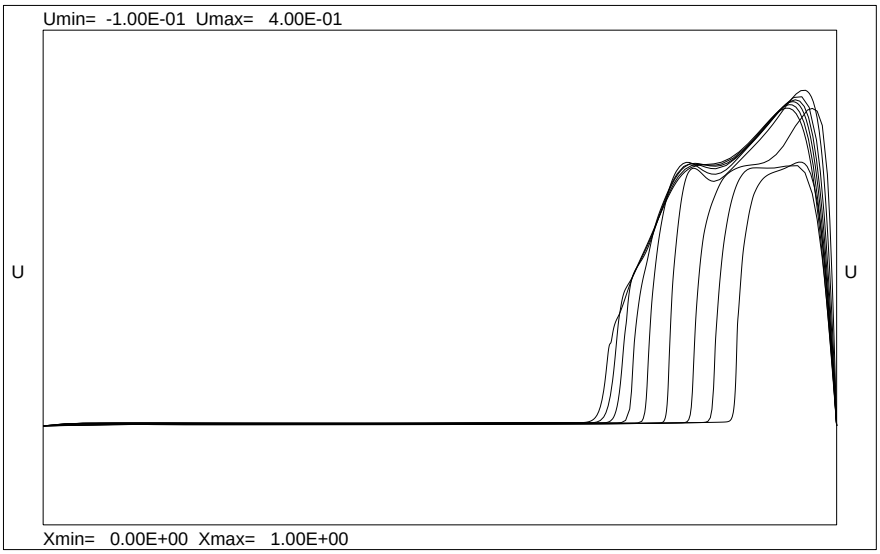


Figure 2: Temperature T (example 2)

This rather challenging test problem, [7], is used to illustrate the special features mentioned in Section 4. The simulation problem is as follows:

$$
\begin{aligned}
& x \in [0,1]\ , \quad t \in [0, 3.3], \\
& u_t = v, \\
& v_t = (G(T)u_x)_x + v_{xx}/\text{Re}, \\
& T_t = T_{xx}/(\text{PrRe}) + (v_x)^2/\text{Re}, \\
& G(T) = \frac{1}{2}((1+G_\infty) - (1-G_\infty)\tanh((T-T_m)/\Delta T)), \\
& u(x,0) = v(x,0) = T(x,0) = 0, \\
& T(0,t) = 0\ , \quad T(1,t) = 0, \\
& v(0,t) = 0\ , \ v(1,t) = V(t),
\end{aligned}
$$

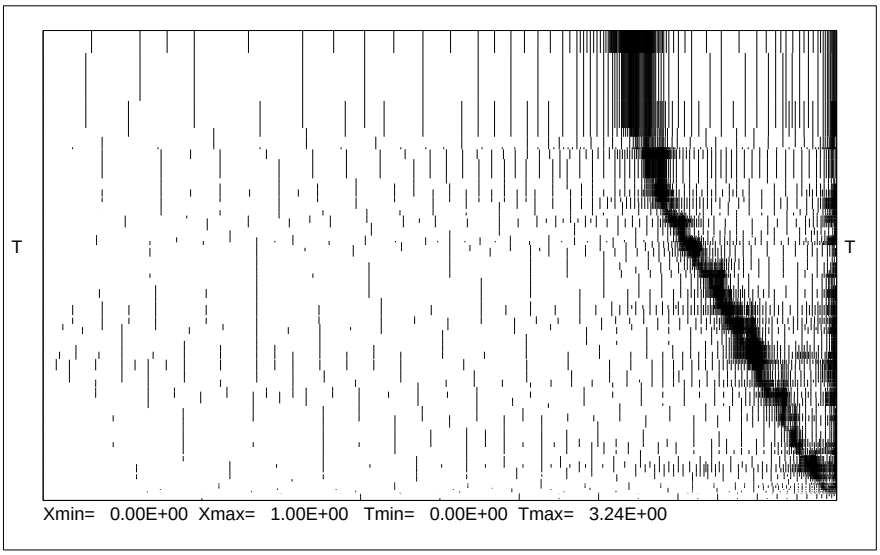


Figure 3: Time–space adaptive grid (example 2)

with

$$V(t) = \begin{cases} V_0(t/r), & 0 < t < r, \\ V_0, & r < t < d - r, \\ V_0(d-t)/r, & d - r < t < d, \\ 0, & d < t. \end{cases}$$

The parameters are Re = 100, Pr = 50, $G_\infty = 0.05$, $T_m = 0.03$, $\Delta T = 0.01$, $V_0 = 0.5$, $d = 1.5$, and $r = 0.05$. The numerical solution for the temperature T is displayed in Fig. 2 at 10 output points t_i uniformly distributed in the integration interval. A steep front develops near the right boundary and stays there during the whole integration. Another front travels slowly to the left. As the resolution of these fronts is critical for the numerical solution, dense grids are used in these regions. The computational grid, in the (x, t)–plane, is plotted in Fig. 3 The behavior of component v is even more complicated. A front ($v \approx 0.1$) travels to the left then back to the right and finally stops. A second front ($v \approx 0.3$) stays for a while at the right boundary. Finally, this peak is damped and moves to the left.

In Table 2 an integration with the default method is compared to an integration with the moving grid option switched on ($\lambda = 0.1$, prescribed tolerance: $\mathrm{tol}_x = \mathrm{tol}_t = 10^{-3}$, thresholds: $s^j_{\mathrm{abs}} = 1.$). Although this problem is not well suited for the use of a moving grid, the number of steps is nearly halved. As the number of PDEs is increased, a loss in the overall efficiency can be observed. In the last row of Table 2 are the results of a standard run with the root finding option switched on.

variant	CPU	nstep	nfcnj	nfcn	ndec	nsol	$\bar{n}_x^F$
standard	82	124	3190	2403	994	2113	152
moving grid	87	73	2550	1259	742	2067	141
root finding	95	124	3190	2403	994	2113	152

Table 2: Amount of work for different variants (example 2)

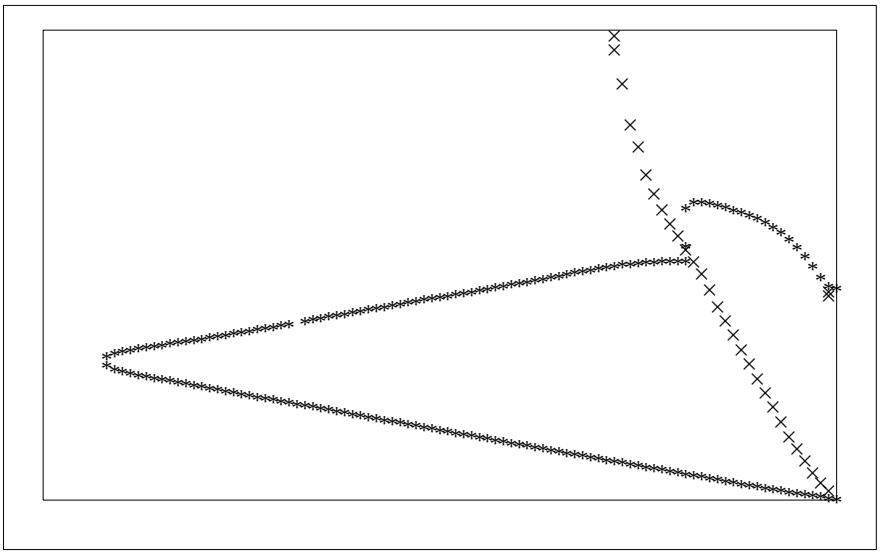


Figure 4: Roots of switching functions (example 2)

As a simple switching function, we prescribe $0 = v - 0.1$ and $0 = T - 0.1$, and require the zeros on a fixed equidistant grid with 101 nodes. The detected zeros are displayed in the (x, t)–plane in Fig. 4. The roots of the first function $(*)$ show the front movement (indicated by $v = 0.1$), first to the left, then back to the right and finally stationary. The moving front of the T–component is displayed by the roots of the second switching function $(\times)$. Although quite a large number of roots have to be determined, the additional amount of work, including the continuous solution representation in space and time, is less than 20%.

References

[1] S. Adjerid, J.E. Flaherty: *A Moving Finite Element Method with Error Estimation and Refinement for One–Dimensional Time Dependent Partial Differential Equations.* SIAM J. Numer. Anal., **23** 778–796(1986)

[2] P. Deuflhard: *Order and Stepsize Control in Extrapolation Methods.* Numer. Math. **41** 399–422 (1983)

[3] P. Deuflhard: *Recent Progress in Extrapolation Methods for ODEs.* SIAM Review **27** 505–535 (1985)

[4] P. Deuflhard, U. Nowak: *Extrapolation Integrators for Quasilinear Implicit ODEs.* In: P. Deuflhard, B. Enquist, eds., *Large Scale Scientific Computing. Progress in Scientific Computing* **7** 37–50 (1987)

[5] E.A. Dorfi, L.O'C. Drury: *Simple Adaptive Grids for 1–D Initial Value Problems.* J. Comput. Phys. **69** 175–195 (1987)

[6] G. Eigenberger, J. Frauhammer, T. Kirchner: private communication (1993)

[7] J.E. Flaherty, P.K. Moore: *Integrated space–time adaptive hp–refinement methods for parabolic systems.* Appl. Numer. Math. **16** 317–341 (1995)

[8] F.N. Fritsch, J. Butland: *Piecewise Cubic Hermite Intepolation Package.* Preprint UCRL–87285, Lawrence Livermore Nat. Lab. (1985)

[9] E. Hairer, A. Ostermann: *Dense Output for Extrapolation Methods.* Numer. Math. **58** 419–439 (1990)

[10] J.M. Hyman: *Moving Mesh Methods for Partial Differential Equations.* In: L. Goldstein, S. Rosencrans, G. Sod, eds., *Mathematics Applied to Science* 129–153 (1988)

[11] J. Lawson, M. Berzins: *Towards an Automatic Algorithm for the Numerical Solution of Parabolic Equations using the Method of Lines.* In: J.R. Cash, I. Gladwell, eds., *Proc. of the 1989 ODE Conference* IMA Conference Series (1992)

[12] Ch. Lubich: *Linearly Implicit Extrapolation Methods for Differential–Algebraic Systems.* Numer. Math. **55** 129–145 (1989)

[13] Ch. Lubich, A. Ostermann: *Runge–Kutta Methods for Parabolic Equations and Convolution Quadrature.* ETH Zürich, Seminar für Angewandte Mathematik, Research Report No. 91–06 (1991)

[14] K. Miller, R.N. Miller: *Moving Finite Elements I.* SIAM J. Numer. Anal. **18** 1019–1032 (1981)

[15] U. Nowak: *Adaptive Linienmethoden für nichtlineare parabolische Systeme in einer Raumdimension.* Technical Report TR 93–14, Konrad–Zuse–Zentrum Berlin (1993)

[16] L.R. Petzold: *An adaptive Moving Grid Method for One–Dimensional Systems of Partial Differential Equations and its Numerical Solution.* Proc. Workshop on Adaptive Methods for Partial Differential Equations, Renselaer Polytechnic Institute (1988)

[17] R.K. de Rivas: *On the Use of Nonuniform Grids in Finite Difference Equations.* J. Comput. Phys. **10** 202–210 (1972)

[18] R.F. Sincovec, N.K. Madsen: *Software for Nonlinear Partial Differential Equations.* ACM Trans. Math. Software **1** 232–260 (1975)