A Two–dimensional Moving Finite Element Method with Local Refinement Based on A Posteriori Error Estimates

Jens Lang * Weiming Cao † Weizhang Huang ‡
Robert D. Russell §

November 14, 2002

Abstract

In this paper, we consider the numerical solution of time–dependent PDEs using a finite element method based upon rh–adaptivity. An adaptive horizontal method of lines strategy equipped with a posteriori error estimates to control the discretization through variable time steps and spatial grid adaptations is used. Our approach combines an r–refinement method based upon solving so–called moving mesh PDEs with h–refinement. Numerical results are presented to demonstrate the capabilities and benefits of combining mesh movement and local refinement.

1 Introduction

In the numerical simulation of multiscale dynamic processes an important aspect is to generate grids, or meshes, adapted to the solutions. Numerous examples demonstrate that adaptive mesh strategies can greatly reduce the errors and the computational effort for finite element methods (FEMs). This approach has been applied in a wide range of important physical and industrial contexts such as problems in fluid dynamics (e.g., reactive flow in a piston engine and flow around a pitching airfoil or moving bodies) and semiconductor device fabrication (e.g., modeling oxide flow, crystal growth, or phase change). Traditionally, the quite robust h–method is applied, where the mesh is locally refined or coarsened

*Department of Mathematics, Darmstadt University of Technology, Schlossgartenstr. 7, 64289 Darmstadt, Germany (lang@mathematik.tu-darmstadt.de).

†Department of Applied Mathematics, The University of Texas at San Antonio, San Antonio, Texas 78249, U.S.A. (wcao@math.utsa.edu).

‡Department of Mathematics, The University of Kansas, Lawrence, KS 66045, U.S.A. (huang@math.ukans.edu).

§Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (rdr@cs.sfu.ca).

by adding or deleting mesh points. This is often combined with the p–method, for which the polynomial degree in each element is selected in accordance with the smoothness properties of the solution. The hp–adaptive FEM has proven successful at accurately resolving and following important solution features for a wide variety of practical problems [5, 36].

Our goal here is to complement the h-adaptive strategy with an r–method, which dynamically redistributes the mesh points in time. As is well–known, moving mesh methods are superior at reducing dispersive errors in the vicinity of wave fronts while local refinement methods can, in principle, add enough degrees of freedom to resolve any fine scale structure. We expect that combining mesh movement with local refinement generally will not only make the global error control possible for the r–method, but also avoid the need for excessive local refinements (cf. Fig. 1 below), and produce grids that are better aligned with and closely follow the solution features.

Not surprisingly, rh–adaptive methods have been considered to some extent in previous studies. Adjerid and Flaherty [2] present a one–dimensional moving mesh FEM with local refinement for parabolic PDEs. Their approach is extended by Arney and Flaherty [1] to the two–dimensional case, where clusters of mesh points are built up and moved with an error–dependent speed. To prevent mesh tangling, Gropp [17] introduces a local uniform refinement strategy with moving grids. Several techniques for the creation and annihilation of moving nodes have been advocated by Kuprat [28]. Although no local refinement is utilized in the two– and three–dimensional mesh update strategies proposed by Johnson and Tezduyar [27] and Nkonga and Guillard [32], they are also of special interest since they show the usefulness of moving techniques for industrial applications. Finally, in recent work Habashi and coworkers [13, 20] have investigated the potential of a so-called mesh optimization methodology (MOM), which has both r– and h–refinement components, for higher dimensional CFD problems.

A general and robust rh–method requires a well–posed general procedure to determine the movement of the mesh points smoothly in time, especially in higher spatial dimensions. Miller, who first introduced a type of moving FEM [31], and his co–workers propose using the finite element residuals to steer the mesh movement [12]. Similar ideas are utilized by Baines [6]. We present in this paper an rh–method that interconnects the h–refinement with a general moving mesh method developed recently in [25, 26]. This r–method is based on the gradient flow equation of a functional which measures the approximation difficulty of the physical solution. It has been shown general and reliable for a variety of practical model problems [9, 11].

The basic idea of our rh-adaptive strategy is as follows: we first discretize both the physical and moving mesh PDEs in time by a Rosenbrock–Wanner–type scheme. Then a finite element approximation is applied in each time step. A hierarchical error estimator developed in [30] is used to construct the monitor function for the moving mesh PDE and to move the grid accordingly. After moving the mesh, we recalculate the physical solution and estimate the error. For elements with error estimates exceeding the tolerance, local refinements are

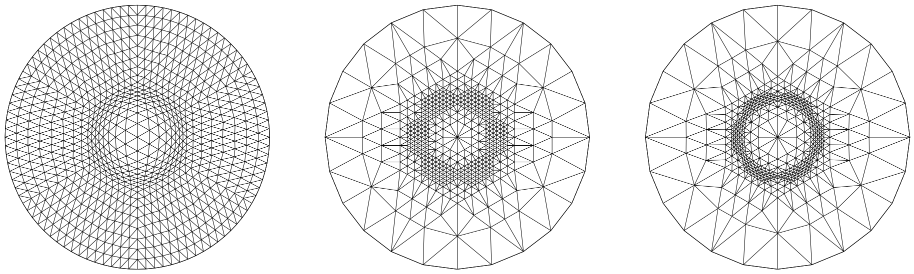

Figure 1: Approximation of the hyperbolic function $u = \mathrm{sech}\,(\,50\,(x^2+y^2-0.09))$ on $\Omega = \{(x,y) \in R^2 : x^2+y^2 \leq 1\}$: r–refinement with $N = 817$ yields an error $\|e\|_{L^2(\Omega)} = 3.55$E–2 (left), h–refinement with $N = 541$ yields $\|e\|_{L^2(\Omega)} = 1.91$E–2 (middle), and rh–refinement with $N = 541$ yields $\|e\|_{L^2(\Omega)} = 1.57\mathrm{E}-2$ (right). While the r–method wastes too many nodes, the combined rh–method results in a mesh well–fitted to the exponentional behaviour of the function and shows its potential compared to the pure h–method.

applied until the tolerance is completely satisfied. In regions with errors far less than the tolerance, grid points are deleted. We test such an adaptive strategy with nonlinear PDEs from fluid flow and combustion. Numerical results show that the combined rh–method reduces significantly the number of degree of freedoms to achieve a prescribed error tolerance.

The paper is organized as follows. In section 2 the basic physical PDEs and moving mesh PDEs are described. The temporal and spatial discretization schemes are introduced in section 3. After describing the error estimation technique which is in turn used to define the monitor function in section 4, the adaptive algorithm based upon the rh–method is described in section 5. We then give two numerical examples to demonstrate the performance of the algorithm, followed by some conclusions.

2 Model Problem and Moving Mesh Method

2.1 Model Problem

We shall consider a system of physical PDEs of the form

$$\left\{ \begin{array}{rcl} H(\boldsymbol{x},t,\boldsymbol{u},\nabla\boldsymbol{u})\,\partial_t\boldsymbol{u} & = & \nabla\cdot(D(\boldsymbol{x},t,\boldsymbol{u},\nabla\boldsymbol{u})\,\nabla\boldsymbol{u}) + F(\boldsymbol{x},t,\boldsymbol{u},\nabla\boldsymbol{u}) \\ & & \text{in } \Omega\times I, \\ B(\boldsymbol{x},t,\boldsymbol{u},\nabla\boldsymbol{u})\,\boldsymbol{u} & = & b(\boldsymbol{x},t,\boldsymbol{u}) \quad \text{on } \partial\Omega\times I, \\ \boldsymbol{u}(\boldsymbol{x},t_0) & = & \boldsymbol{u}_0(\boldsymbol{x}) \quad \text{in } \Omega, \end{array} \right. \tag{1}$$

where $I = (t_0, t_E]$ is a given time interval, $\Omega \subset \mathbb{R}^2$ is a bounded open domain with smooth boundary $\partial\Omega$, and $\nabla = (\partial_x, \partial_y)^T$ is the gradient operator. The boundary operator B defines an appropriate system of boundary conditions such that there exists a unique isolated vector–valued solution $\boldsymbol{u}(\boldsymbol{x}, t)$ for all time $t \in I$. Equation (1) is quite general and covers a wide variety of practically relevant problems such as reaction–diffusion and Navier–Stokes equations.

2.2 Moving Mesh PDEs

To discretize (1) with an adaptive finite element method, we need to construct a time dependent mesh, denoted as $\Omega_h(t)$, on the domain Ω. It can be generated for each time level t with any of a variety of mesh generators, such as a Delaunnay triangulation or front advancing method. An alternative approach is to start with an auxiliary domain Ω_c and a fixed mesh Ω_c^h on it, and then take $\Omega_h(t)$ as the image of Ω_c^h under a suitably defined mapping $\boldsymbol{x}(\boldsymbol{\xi}, t) : \Omega_c \to \Omega$ – see Fig. 2.

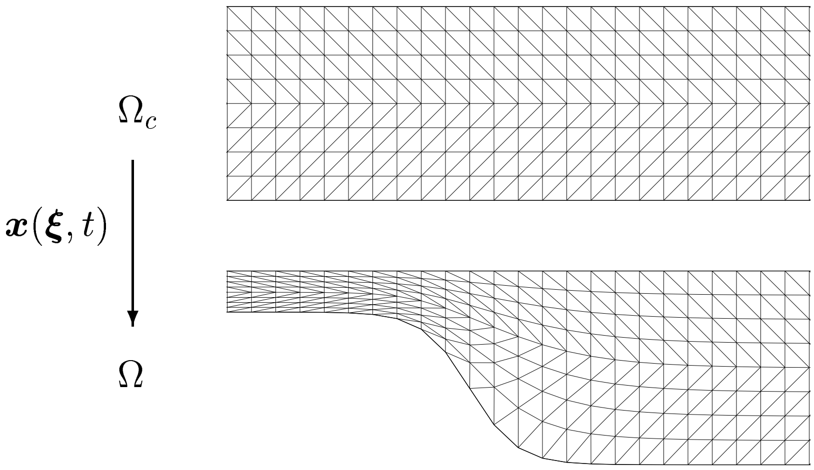

Figure 2: The moving mesh is the image of a reference mesh given on Ω_c (top) through a time–dependent coordinate transformation $\boldsymbol{x}(\boldsymbol{\xi}, t)$: $\Omega_c \to \Omega$.

To determine the time–dependent mapping $\boldsymbol{x}(\boldsymbol{\xi}, t)$, Huang and Russell [25, 26] propose using a time dependent PDE system, the so-called Moving Mesh PDEs (MMPDEs), which are based on the elliptic mesh generation methods.

Following [22], $\boldsymbol{x}(\boldsymbol{\xi},t)$ is determined by solving the MMPDEs

$$S(\boldsymbol{x},G)\,\partial_t\boldsymbol{x} = \sum_{i,j=1,2} D_{ij}(\boldsymbol{x},G)\,\frac{\partial^2\boldsymbol{x}}{\partial\xi_i\,\partial\xi_j} + \sum_{i=1,2} C_i(\boldsymbol{x},G)\,\frac{\partial\boldsymbol{x}}{\partial\xi_i} \quad \text{in } \Omega_c\times I\,, \tag{2}$$

where

$$D_{ij}(\boldsymbol{x},G) = \nabla\xi_i\cdot G^{-1}\nabla\xi_j, \qquad \text{and} \qquad C_i(\boldsymbol{x},G) = -\nabla\xi_i\cdot(\nabla\cdot G^{-1}),$$

and the so–called monitor function $G = G(\boldsymbol{u}(\boldsymbol{x},t))$ is a 2×2 symmetric positive definite matrix. The coefficient $S(\boldsymbol{x},G)$ is used to adjust the time scale of mesh movement. Following Huang [22], we choose

$$S(\boldsymbol{x},G) = \Theta\cdot\sqrt{(D_{11})^2+(D_{22})^2+(C_1)^2+(C_2)^2}, \tag{3}$$

where Θ is a user defined smoothing parameter. The smaller Θ is, the faster the adaptive mesh responds to changes of the monitor function, and the larger Θ is, the smoother the mesh movement.

It is clear from (2) that the monitor function is the link between the solution of the physical PDEs and the adaptive mesh defined by $\boldsymbol{x}(\boldsymbol{\xi},t)$. A proper definition of G is crucial for the construction of a mesh well–adapted to the solution. Typically, one expects to have higher mesh density in regions where the solution is steep or the error of the approximation is large. There are various forms of possible monitor functions which emphasize different aspects of mesh qualities, such as concentration, orthogonality, or alignment [9, 25]. Here we choose the monitor function G to depend upon an a-posteriori error estimate of the local spatial discretization. Its specific definition is given in section 4.2.

The MMPDEs (2) have to be supplemented with suitable boundary conditions. Dirichlet-type boundary conditions are a straightforward and robust choice. They are used to fix corner points and to incorporate temporal variations of the physical domain. We use Dirichlet boundary conditions for $\boldsymbol{x}(\boldsymbol{\xi},t)$ determined from the solution of a one dimensional MMPDE. More precisely, given a boundary segment Γ of $\partial\Omega$, let Γ_c be the corresponding boundary segment of $\partial\Omega_c$. Parameterize Γ and Γ_c by arclength coordinates ζ and s, respectively. Then the mapping $\boldsymbol{x}(\boldsymbol{\xi},t)$ on Γ is determined by the solution $\zeta(s,t)$ of the one dimensional MMPDE

$$\Theta\partial_t\zeta = [M\partial_s\zeta]^{-2}(M\partial_s^2\zeta + \partial_s M\,\partial_s\zeta)\,,$$

where M is the projection of the two dimensional monitor function G along the boundary, i.e., if $\vec{s}$ is the unit tangent vector along the boundary then $M(s,t) = \vec{s}^{\,T}G\vec{s}$.

2.3 Coupled Physical and Moving Mesh PDEs

Under the mapping $\boldsymbol{x}(\boldsymbol{\xi},t)$, we can transform the physical PDEs (1) into a system involving the computational coordinate $\boldsymbol{\xi}$, viz.,

$$\left\{\begin{array}{rcl} H(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})\,(\partial_t\hat{\boldsymbol{u}}-\partial_t\boldsymbol{x}\cdot J^{-T}\hat{\nabla}\hat{\boldsymbol{u}}) & = & \\ \multicolumn{3}{l}{|J|^{-1}\hat{\nabla}\cdot(|J|J^{-1}D(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})\,J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})+F(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})} \\ & & \text{in } \Omega_c\times I, \\ B(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})\,\hat{\boldsymbol{u}} & = & b(\boldsymbol{x},t,\hat{\boldsymbol{u}}) \quad \text{on } \partial\Omega_c\times I, \\ \hat{\boldsymbol{u}}(\boldsymbol{\xi},t_0) & = & \boldsymbol{u}_0(\boldsymbol{x}(\boldsymbol{\xi},t_0)) \quad \text{in } \Omega_c, \end{array}\right. \tag{4}$$

where $\hat{\boldsymbol{u}}=\hat{\boldsymbol{u}}(\boldsymbol{\xi},t)=\boldsymbol{u}(\boldsymbol{x}(\boldsymbol{\xi},t),t)$, $\hat{\nabla}$ is the gradient operator with respect to $\boldsymbol{\xi}=(\xi_1,\xi_2)^T$, and $J=\partial\boldsymbol{x}/\partial\boldsymbol{\xi}$ is the Jacobian of the mapping $\boldsymbol{x}(\boldsymbol{\xi},t)$. The additional term $\partial_t\boldsymbol{x}\cdot J^{-T}\hat{\nabla}\hat{\boldsymbol{u}}$ on the left–hand side can be viewed as a correction for the convective effects of the mesh motion.

Recall that $\boldsymbol{x}(\boldsymbol{\xi},t)$ and $\hat{\boldsymbol{u}}(\boldsymbol{x},t)$ are interconnected through the monitor function G in the MMPDEs (2). Both of them are time dependent unknown functions. Introducing a new solution vector $\boldsymbol{U}=(\hat{\boldsymbol{u}},\boldsymbol{x})^T$, we rewrite equations (2) and (4) as an expanded system

$$\left\{\begin{array}{rcl} P(t,\boldsymbol{U})\,\partial_t\boldsymbol{U} & = & f(t,\boldsymbol{U}), \quad t\in I, \\ \boldsymbol{U}(\boldsymbol{\xi},t_0) & = & \boldsymbol{U}_0(\boldsymbol{x}(\boldsymbol{\xi},t_0)), \end{array}\right. \tag{5}$$

where

$$P(t,\boldsymbol{U})=\begin{pmatrix} H(t,\boldsymbol{U}) & -H(t,\boldsymbol{U})J(\boldsymbol{x})^{-T}\hat{\nabla}\hat{\boldsymbol{u}} \\ 0 & S(\boldsymbol{x},G) \end{pmatrix}$$

and

$$f(t,\boldsymbol{U})=\begin{pmatrix} |J|^{-1}\hat{\nabla}\cdot(|J|J^{-1}D(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})\,J^{-T}\hat{\nabla}\hat{\boldsymbol{u}})+F(\boldsymbol{x},t,\hat{\boldsymbol{u}},J^{-T}\hat{\nabla}\hat{\boldsymbol{u}}) \\ \displaystyle\sum_{i,j=1,2} D_{ij}(\boldsymbol{x},G)\,\frac{\partial^2\boldsymbol{x}}{\partial\xi_i\,\partial\xi_j}+\sum_{i=1,2} C_i(\boldsymbol{x},G)\,\frac{\partial\boldsymbol{x}}{\partial\xi_i} \end{pmatrix}.$$

The initial and boundary conditions for $\boldsymbol{U}$ are taken from those in (1) and (2), respectively. This system is highly nonlinear and stiff in general.

3 Time–Space Discretization

Equation (5) is discretized first in time and then in space, an approach which is known as Rothe's method or the horizontal method of lines. After discretizing in time we end up with a system of spatial problems which are solved by the finite element method. The spatial discretization error can be assessed by standard error estimators for stationary problems [7, 30]. We note that the adaptive Rothe's method differs from the commonly used adaptive method of lines (MOL) approach (for a comparison see [14]).

3.1 Time Discretization

Since we will incorporate the h–refinement into our spatial discretization, a suitable time integration method for (5) should be able to easily and accurately handle the addition and deletion of unknowns associated with the h–refinement. In this regard, one–step time integrators are preferred over multi–step methods such as backward differentiation formulas (BDFs). When mesh points are added or deleted, a multi–step method must usually be restarted at lower order, whereas a one–step method can continue at higher order. Among the class of one–step methods, linearly implicit Rosenbrock–Wanner–type schemes (ROW) are attractive since they completely avoid the solution of nonlinear problems, so no Newton–like iteration need to be controlled. Working the Jacobian or an approximation of it directly into the integration formula, ROW–methods possess optimal linear stability properties for stiff equations [33, 21]. We apply the ROW–methods given in Lang [30], which are suitable for an error–controlled solution of parabolic PDEs. Applied to the initial–value problem (5) with step size $\tau_n > 0$ an s–stage ROW–method has the recursive form

$$\left\{ \begin{array}{rcl} \left(\dfrac{1}{\tau_n\gamma}P(t_n,\boldsymbol{U}_n) - A_n\right)\boldsymbol{U}'_{ni} & = & f(t_{ni},\boldsymbol{U}_{ni}) - P(t_n,\boldsymbol{U}_n)\displaystyle\sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_n}\boldsymbol{U}'_{nj} + \\ & & + \left(P(t_n,\boldsymbol{U}_n) - P(t_{ni},\boldsymbol{U}_{ni})\right)\boldsymbol{Z}_{ni} + \tau_n\gamma_i\, C_n, \quad i = 1,\ldots,s, \end{array} \right. \tag{6}$$

where the internal values are given by

$$t_{ni} = t_n + \alpha_i\tau_n, \quad \boldsymbol{U}_{ni} = \boldsymbol{U}_n + \sum_{j=1}^{i-1} a_{ij}\boldsymbol{U}'_{nj}, \quad \boldsymbol{Z}_{ni} = (1-\sigma_i)\boldsymbol{Z}_n + \sum_{j=1}^{i-1}\frac{s_{ij}}{\tau_n}\boldsymbol{U}'_{nj},$$

and

$$\begin{array}{rcl} A_n & \approx & \partial_u\left(f(t,\boldsymbol{U}) - P(t,\boldsymbol{U})\boldsymbol{Z}\right)_{|t=t_n,\boldsymbol{U}=\boldsymbol{U}_n,\boldsymbol{Z}=\boldsymbol{Z}_n}, \\ C_n & \approx & \partial_t\left(f(t,\boldsymbol{U}) - P(t,\boldsymbol{U})\boldsymbol{Z}\right)_{|t=t_n,\boldsymbol{U}=\boldsymbol{U}_n,\boldsymbol{Z}=\boldsymbol{Z}_n}. \end{array}$$

The approximation $\boldsymbol{U}_n$ to the solution $\boldsymbol{U}(t_n)$ is used to compute a new approximate solution at time level $t_{n+1} := t_n + \tau_n$ using a linear combination of $\boldsymbol{U}_n$ and the solutions of (6), viz.,

$$\boldsymbol{U}_{n+1} = \boldsymbol{U}_n + \sum_{i=1}^{s} m_i\boldsymbol{U}'_{ni}. \tag{7}$$

An approximation to the temporal derivation $\partial_t \boldsymbol{U}$ for the next time step is constructed by computing

$$\boldsymbol{Z}_{n+1} = \boldsymbol{Z}_n + \sum_{i=1}^{s} m_i \left(\sum_{j=1}^{i} \frac{c_{ij} - s_{ij}}{\tau_n} \boldsymbol{U}'_{ni} + (\sigma_i - 1)\boldsymbol{Z}_n \right). \tag{8}$$

To start the above ROW–method, an approximation $\boldsymbol{Z}_0$ of $\partial_t \boldsymbol{U}$ at t_0 is required. The stage number s and the formula coefficients are chosen to obtain a desired order of consistency and good stability properties for stiff equations. A nice feature of linearly implicit ROW–methods is that all linear systems for the intermediate values $\boldsymbol{U}'_{ni}, i = 1, \ldots, s$, involve the same operator. The boundary conditions for $\boldsymbol{U}'_{ni}$ are readily obtained applying the ROW–method (6) to the boundary conditions for $\boldsymbol{U}$, which are understood as algebraic equations of the form (5) with $P(t, \boldsymbol{U}) \equiv 0$ [30].

Various ROW–methods have been constructed in [30] to integrate systems of the form (5). In our computations, we use the solver Ros2 [34] (see Tab. 1 for the defining formula coefficients), which has second–order accuracy for arbitrary A_n and C_n. This property allows us to freeze the coefficients in the MMPDEs (2) during one time step without order reduction. Specifically, for all $t \in (t_n, t_{n+1})$ we use $D_{ij}(\boldsymbol{x}_n, G(\boldsymbol{u}_n))$ instead of $D_{ij}(\boldsymbol{x}(t), G(\boldsymbol{u}(\boldsymbol{x}(t), t)))$ and $C_i(\boldsymbol{x}_n, G(\boldsymbol{u}_n))$ instead of $C_i(\boldsymbol{x}(t), G(\boldsymbol{u}(\boldsymbol{x}(t), t)))$ for $i, j = 1, 2$. Here, $\boldsymbol{x}_n$ and $\boldsymbol{u}_n$ denote approximations of the mapping $\boldsymbol{x}$ and the solution $\boldsymbol{u}$ at t_n. Since the position of mesh points need not to be determined as precisely as the solution of the physical PDE, it is generally unnecessary to solve the MMPDEs to very high accuracy. We have found the approach of freezing the coefficients to be quite efficient and robust with respect to the choice of the constant parameter Θ in (3).

The time step size is also adapted in order to control the temporal error. For ROW–methods, a second solution of lower order, say $\hat{p}$, can be computed by an embedded formula

$$\begin{aligned} \hat{\boldsymbol{U}}_{n+1} &= \boldsymbol{U}_n + \sum_{i=1}^{s} \hat{m}_i \boldsymbol{U}'_{ni}\,, \\ \hat{\boldsymbol{Z}}_{n+1} &= \boldsymbol{Z}_n + \sum_{i=1}^{s} \hat{m}_i \left(\sum_{j=1}^{i} \frac{c_{ij} - s_{ij}}{\tau_n} \boldsymbol{U}_{ni} + (\sigma_i - 1)\boldsymbol{Z}_n \right), \end{aligned}$$

where the original weights m_i in (7) and (8) are simply replaced by $\hat{m}_i$. If p is the order of $\boldsymbol{U}_{n+1}$, we call such a pair of formulas of order $p(\hat{p})$. The difference between these solutions is used to compute the local error estimator

$$r_{n+1} = \|\boldsymbol{U}_{n+1} - \hat{\boldsymbol{U}}_{n+1}\|\,, \tag{9}$$

where $\|\cdot\|$ is a weighted norm defined for vector–valued functions $\boldsymbol{v} = (v_1, \ldots, v_q)^T$

as

$$\|\boldsymbol{v}\| = \left(\frac{1}{q} \sum_{i=1}^{q} \left(\frac{\|v_i\|_{L_2(\Omega_c)}}{\mathrm{ATOL}_i + \|U_{n+1,i}\|_{L_2(\Omega_c)} \cdot \mathrm{RTOL}_i} \right)^2 \right)^{1/2} . \qquad (10)$$

The tolerances ATOL_i and RTOL_i are selected to accurately reflect the scale of the problem. The predicted new time step is

$$\tau_{n+1} = \min\left(10\,,\, \max\left(0.1\,,\, \frac{\tau_n}{\tau_{n-1}} \left(\frac{\mathrm{TOLT} \cdot r_n}{r_{n+1} \cdot r_{n+1}} \right)^{1/(\hat{p}+1)} \right) \right) \tau_n\,, \qquad (11)$$

where TOLT is the prescribed error tolerance. This formula is related to a discrete PI–controller first established in the pioneering work of GUSTAFFSON, LUNDH, and SÖDERLIND [18, 19].

$\gamma = 1.707106781186547e+00$	
$a_{11} = 0.000000000000000e+00$ $a_{21} = 5.857864376269050e-01$ $a_{22} = 0.000000000000000e+00$	$\alpha_1 = 0.000000000000000e+00$ $\alpha_2 = 1.000000000000000e+00$
$c_{11} = 5.857864376269050e-01$ $c_{21} = 1.171572875253810e+00$ $c_{22} = 5.857864376269050e-01$	$s_{11} = 0.000000000000000e+00$ $s_{21} = 3.431457505076198e-01$ $s_{22} = 0.000000000000000e+00$
$\gamma_1 = 1.707106781186547e+00$ $\gamma_2 = -1.707106781186547e+00$	$\sigma_1 = 0.000000000000000e+00$ $\sigma_2 = 5.857864376269050e-01$
$m_1 = 8.786796564403575e-01$ $m_2 = 2.928932188134525e-01$	$\hat{m}_1 = 5.857864376269050e-01$ $\hat{m}_2 = 0.000000000000000e+00$

Table 1: Set of coefficients for ROS2, which is of order 2(1).

3.2 Space Discretization

Having discretized in time, we use the finite element method to solve equation (6) supplemented with the discretized boundary conditions. Let $\mathcal{T}_h$ be an admissible finite element mesh on Ω_c at $t = t_n$, and $S_h^q \subset H^1(\Omega_c)$ be the associated finite dimensional space consisting of all continuous functions that are polynomials of order q on each finite element $T \in \mathcal{T}_h$ and that vanish on boundaries where Dirichlet–type conditions are given. Taking the $L_2(\Omega_c)$–inner product of (6) with test functions $\boldsymbol{\phi} \in S_h^q$, the standard Galerkin finite element approximation $\boldsymbol{U}'_{h,ni} \in S_h^q$ for the intermediate values $\boldsymbol{U}'_{ni}$ is required to satisfy

$$(L_n \boldsymbol{U}'_{h,ni}, \boldsymbol{\phi}) = (r_{ni}, \boldsymbol{\phi}) \quad \text{for all } \boldsymbol{\phi} \in S_h^q\,. \qquad (12)$$

Here, L_n is the weak representation of the differential operator on the left–hand side in (6) and $r_{ni} = r_{ni}(\boldsymbol{U}'_{h,n1}, \ldots, \boldsymbol{U}'_{h,ni-1})$ stands for the entire right–hand side in (6). Since L_n is independent of the index i, its calculation is required only once each time step. The finite element solution at the time level t_{n+1} is computed as

$$\boldsymbol{U}_{h,n+1} = \boldsymbol{U}_{h,n} + \sum_{i=1}^{s} m_i \, \boldsymbol{U}'_{h,ni} \,, \tag{13}$$

with $\boldsymbol{U}_{h,n}$ being an approximation to $\boldsymbol{U}(t_n)$. The linear systems (12) are solved by a preconditioned Krylov subspace method, viz., an efficient combination of BICGSTAB [35] and an incomplete LU–factorization.

4 Error Estimates and Monitor Functions

4.1 Error Estimates

Once all $\boldsymbol{U}'_{h,ni} \in S_h^q$ have been computed, an a-posteriori error estimate can be employed to assess the spatial error distribution. We adopt here a technique known as hierarchical error estimation — see, e.g., BORNEMANN, ERDMANN, and KORNHUBER [8], DEUFLHARD, LEINEN and YSERENTANT [15], BANK and SMITH [4]. More precisely, let the approximation subspace S_h^{q+1} admit a decomposition

$$S_h^{q+1} = S_h^q \oplus Z_h^{q+1} \,, \tag{14}$$

where Z_h^{q+1} is the subspace spanned by all additional basis functions that are required to extend the space S_h^q to the higher order space S_h^{q+1}. Hierarchical error estimates are used to calculate the bound on the spatial error by evaluating components in the space Z_h^{q+1} only. In LANG [30], this technique has been carried over to time–dependent nonlinear problems. Following the approach developed there, an a posteriori error estimator $\boldsymbol{E}_{n+1}(\boldsymbol{\xi}) \in Z_h^{q+1}$ for the finite element solution $\boldsymbol{U}_{h,n+1}$ is defined as a linear combination of terms of the form

$$\boldsymbol{E}_{n+1}(\boldsymbol{\xi}) = \boldsymbol{E}_{n0}(\boldsymbol{\xi}) + \sum_{i=1}^{s} m_i \, \boldsymbol{E}_{ni}(\boldsymbol{\xi}) \,, \tag{15}$$

where $\boldsymbol{E}_{n0} \in Z_h^{q+1}$ measures the projection error of the initial value $\boldsymbol{U}_{h,n}$ and $\boldsymbol{E}_{ni}$ estimates the spatial error of the intermediate value $\boldsymbol{U}'_{h,ni}$. More precisely, we compute $\boldsymbol{E}_{n0}$ from the equation

$$(L_n \boldsymbol{E}_{n0}, \boldsymbol{\phi}) = (L_n \, (\bar{\boldsymbol{U}}_{h,n} - \boldsymbol{U}_{h,n}), \boldsymbol{\phi}) \quad \text{for all } \boldsymbol{\phi} \in Z_h^{q+1} \,, \tag{16}$$

with $\bar{\boldsymbol{U}}_{h,n}$ representing the initial solution computed on a well–fitted mesh at time t_n, and $\boldsymbol{U}_{h,n}$ being its projection onto S_h^q. (Were the computational mesh Ω_c^h to be coarsened, $\boldsymbol{E}_{n0}$ estimates the resulting loss of resolution for the previous finite element solution $\bar{\boldsymbol{U}}_{h,n}$.) The stage error estimator $\boldsymbol{E}_{ni} \in Z_h^{q+1}$ satisfies

$$(L_n \boldsymbol{E}_{ni}, \boldsymbol{\phi}) = (r_{ni}(\boldsymbol{U}'_{h,n1} + \boldsymbol{E}_{n1}, \ldots, \boldsymbol{U}'_{h,ni-1} + \boldsymbol{E}_{ni-1}), \boldsymbol{\phi}) - (L_n \boldsymbol{U}'_{h,ni}, \boldsymbol{\phi}) \tag{17}$$

for all $\boldsymbol{\phi} \in Z_h^{q+1}$. The computation of the error estimator $\boldsymbol{E}_{n+1}$ only requires the solution of linear systems. The expense of the error estimation can be further reduced by replacing system (17) with a block diagonal approximation in a standard way [15, 30]. The stage error estimators $\boldsymbol{E}_{ni}$ are used successively to improve the approximation of the nonlinear term r_{ni}. Here, we apply linear finite elements on triangular meshes and measure the spatial errors in the space of quadratic functions.

In our context of rh–refinement, the error estimates $\boldsymbol{E}_n(\boldsymbol{\xi})$ are in fact scaled such that we basically only use spatial error estimates for the $\hat{\boldsymbol{u}}$–component of solution $\boldsymbol{U} = (\hat{\boldsymbol{u}}, \boldsymbol{x})^T$ and not the nodes $\boldsymbol{x}$. That is, we set $\mathrm{RTOL}_i = \infty$ for all $\boldsymbol{x}$–components in (10). As discussed previously, this is because our experience has shown that control of spatial errors for $\hat{\boldsymbol{u}}$ is generally sufficient to maintain adequate precision for grid placement as well.

4.2 Monitor Function

To construct the monitor function G, we follow [11] and first define an error function $\mathcal{E}_{n+1}$ which describes the estimated error per unit area at each node of the physical domain. Specifically, letting $\boldsymbol{x}_p$ be a mesh point in $\Omega_h(t_{n+1})$ and $\boldsymbol{\xi}_p = \boldsymbol{\xi}(\boldsymbol{x}_p, t_{n+1})$ the corresponding mesh point in Ω_c, we define

$$\mathcal{E}_{n+1}(\boldsymbol{x}_p) = \frac{\|\boldsymbol{E}_{n+1}\|_{C(\boldsymbol{\xi}_p)}}{\int_{C(\boldsymbol{\xi}_p)} d\boldsymbol{\xi}}, \tag{18}$$

where $C(\boldsymbol{\xi}_p) \subset \Omega_c$ is the union of neighbouring grid cells having $\boldsymbol{\xi}_p$ as one of their vertices. Clearly, regions with larger $\mathcal{E}_{n+1}$ need higher mesh concentration. To avoid overcrowding the mesh points in regions of maximum errors, we introduce a cut–off function of $\mathcal{E}_{n+1}$ as follows

$$\bar{\mathcal{E}}_{n+1}(\boldsymbol{x}_p) = \begin{cases} 0.8 \cdot \max_{\boldsymbol{x}_p} \mathcal{E}_{n+1}(\boldsymbol{x}_p) & \text{if } \mathcal{E}_{n+1}(\boldsymbol{x}_p) > 0.8 \cdot \max_{\boldsymbol{x}_p} \mathcal{E}_{n+1}(\boldsymbol{x}_p) \\ \mathcal{E}_{n+1}(\boldsymbol{x}_p) & \text{otherwise}. \end{cases}$$

The monitor function is then defined as

$$G(\boldsymbol{x}_p, t_{n+1}) = \sqrt{1 + \alpha \left(\frac{\bar{\mathcal{E}}_{n+1}(\boldsymbol{x}_p)}{\max_{\boldsymbol{x}_p} \bar{\mathcal{E}}_{n+1}(\boldsymbol{x}_p)} \right)^2} \cdot I \tag{19}$$

for all mesh points $\boldsymbol{x}_p$ in $\Omega_h(t_{n+1})$, where α is an intensity parameter used to control the influence of the error function $\bar{\mathcal{E}}_{n+1}$ on the mesh concentration. The monitor function defined pointwise is extended to a function $G(\boldsymbol{x}, t_{n+1})$ for all $\boldsymbol{x} \in \Omega_h(t_{n+1})$ by linear interpolation.

To increase the smoothness of the mesh distribution and also to reduce the stiffness of the MMPDEs (2), it is common practice to smooth the monitor function by a local averaging. Given a non–negative integer M, we use the monitor function $G^{(M)}(\boldsymbol{x}, t_{n+1})$ defined by the iterative process

$$\left\{\begin{array}{rcl} G^{(0)}(\boldsymbol{x}_p, t_{n+1}) & := & G(\boldsymbol{x}_p, t_{n+1}) \quad \text{for all } \boldsymbol{x}_p\,, \\[2ex] m = 0, 1, \ldots, M-1: & & \\[2ex] G^{(m+1)}(\boldsymbol{x}_p, t_{n+1}) & := & \dfrac{\int_{C(\boldsymbol{\xi}_p)} G^{(m)}(\boldsymbol{x}(\boldsymbol{\xi}, t_{n+1}), t_{n+1})\, d\boldsymbol{\xi}}{\int_{C(\boldsymbol{\xi}_p)} d\boldsymbol{\xi}} \quad \text{for all } \boldsymbol{x}_p\,. \end{array}\right. \tag{20}$$

This smoothing algorithm has proven to work quite satisfactorily in practice. In our computations, we take $M = 6$.

5 rh–Adaptive Algorithm

In this section, we describe the coupling of the r–method which involves solving system (5) with the monitor function defined in (20) and the h–method which is based upon using a posteriori error estimates introduced in (15).

Our refinement strategy consists of first calculating a preliminary finite element solution $\boldsymbol{U}_{h,n+1}$ and its approximate error $\boldsymbol{E}_{n+1}$ on a given mesh $\mathcal{T}_h^{(0)}$ for a time step τ_n. If $\|\boldsymbol{E}_{n+1}\| >$ TOLX, the local quantities $\eta_T := \|\boldsymbol{E}_{n+1}\|_T, T \in \mathcal{T}_h^{(0)}$, are used to locate regions where greater resolution is needed. To this end, we define a local error barrier $\eta_{bar} := \gamma \cdot \max_T \eta_T$, where $0 < \gamma < 1$ is a parameter. All elements $T \in \mathcal{T}_h^{(0)}$ with η_T larger than the barrier η_{bar} are selected for refinement. To ensure that at least a certain percentage of elements is refined, we iteratively reduce η_{bar} by the factor γ. In our computations, we set $\gamma = 0.8$ and repeat the selection process until at least 10% of all elements are marked for refinement. Then a finer grid $\mathcal{T}_h^{(1)}$ is created by locally refining each of the marked elements into four congruent triangles, and applying bisection afterwards to avoid slave nodes. This is the standard red–green refinement commonly used in two–dimensional adaptive codes [3, 16]. The solution and error estimator are computed anew on $\mathcal{T}_h^{(1)}$. This recursive process leads naturally to a sequence of improved spatial meshes

$$\mathcal{T}_h^{(0)} \subset \mathcal{T}_h^{(1)} \subset \ldots \subset \mathcal{T}_h^{(d)}\,.$$

It is stopped when $\|\boldsymbol{E}_{n+1}\| <$ TOLX on a certain $\mathcal{T}_h^{(d)}$. Clearly, a goal is that the automatic mesh moving technique should avoid excessive refinement. Moreover, if h–refinement is necessary, the depth d should at least be small.

If the time step cannot be accepted due to insufficient temporal accuracy, i.e., $r_{n+1} >$ TOLT in (9), the time step is rejected, all refinements done in this step are deleted, and the computation is repeated with a reduced value of τ_n given by formula (11). Otherwise, we proceed in time, continuing until the final time t_E is reached.

If $\|\boldsymbol{E}_{n+1}\| < \frac{1}{3}$TOLX or h–refinement took place during a time step, mesh coarsening is performed after the time step has been accepted. This process helps to reduce degrees of freedom in regions where they are no longer needed. More precisely, an element $T \in \mathcal{T}_h^{(d)}$ is removed only if η_T is below $\frac{1}{10}\eta_{bar}$ for the triangle and the three corresponding triangular elements created by a local refinement step (see [29] for more details). To control the global error, mesh movement, local refinement and unrefinement are repeated until the prescribed spatial tolerance TOLX is completely satisfied. This algorithm is summarized in a flow chart in Fig. 3.

We allow also for a complete remeshing whenever the mesh is severely distorted at a specific time. The creation of a new base mesh is similar to the generation of a suitable initial mesh. We first create a quasi–uniform mesh and solve system (5) with a small scaling parameter $\Theta = \Theta_{meshing}$, which causes a sequence of small time steps due to a fast mesh movement. Then the error estimates are computed, and local mesh refinement is performed as described above. In contrast to the original rh–adaptive algorithm, we coarsen the mesh after each time step in order to approximately equidistribute the error estimators. If the integration results in excessive refinement, the computation is redone with a finer base mesh. The parameter $\Theta_{meshing}$ for a remeshing at t_n has to be chosen in such a way that the auxiliary integration is finished before the time $t_{n+1} = t_n + \tau_n$ determined by (11). We have found that $\Theta_{meshing} = 0.01 \cdot \Theta$ is generally quite sufficient.

Once a new base mesh has been constructed, the solution and the error estimator on it are determined using linear interpolation, and the time integration proceeds. In order to provide data for the computation of the projection error $\boldsymbol{E}_{n0}$ in (16), we save the solution at the previous grid until it is no longer needed because advancement to a subsequent time level has been successful.

6 Numerical Examples

6.1 Burgers' Equation

Our first test is for the well-known Burgers' equation

$$\partial_t u = \nu \nabla^2 u - u\partial_x u - u\partial_y u, \qquad \text{in } \Omega \times (0.25, 1.5]$$

where Ω is the unit square. The initial and Dirichlet boundary conditions are chosen such that the exact solution is

$$u(x,y,t) = 1/[1 + e^{(x+y-t)/(2\nu)}].$$

We consider the case of a moderately small diffusion coefficient $\nu = 0.005$.

With this example we shall mainly demonstrate the benefits of the combined rh–method described in section 5 over the pure h– and r–method by comparing the number of degree of freedoms required by the three methods to attain a similar solution accuracy. Our results for the r–method are from [9], where a linear finite element approximation on a 2048–triangular mesh and a constant time step size τ=1E-3 are employed. For such an r–adaptive approach, the L^1–norm of the solution error varies between 1E-4 and 1E-3 over the entire time

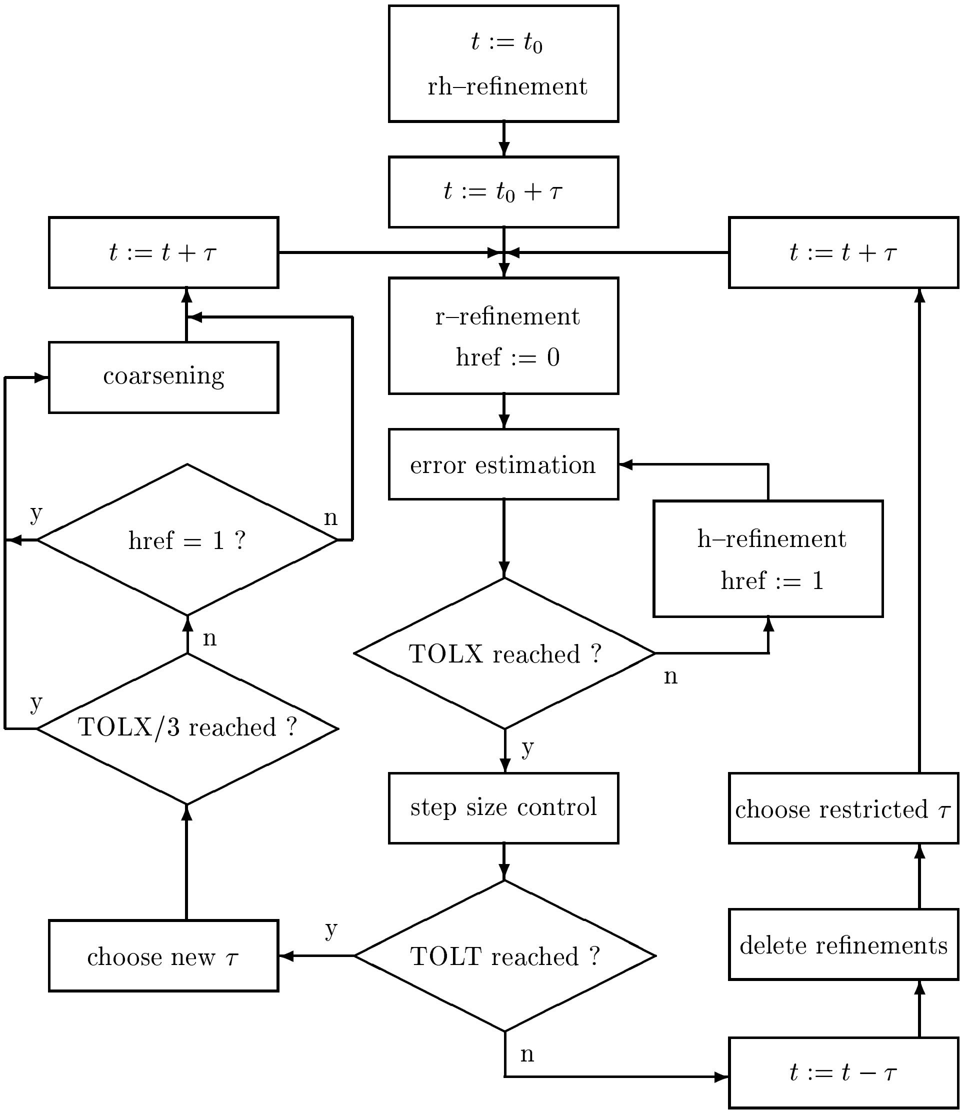

Figure 3: Flow chart of the entire rh–adaptive algorithm.

interval. To reach a comparable solution accuracy, we choose the tolerances TOLT=TOLX=5E-4 for the h– and r–method (see Fig. 4). In addition, we set ATOL_i=1E-6 and RTOL_i=1.0 in (10). Recall that x and y are not utilized for spatial error control as described in section 4. The mesh movement is controlled by $\Theta = 10.0$ in (3) and $\alpha = 50.0$ in (19).

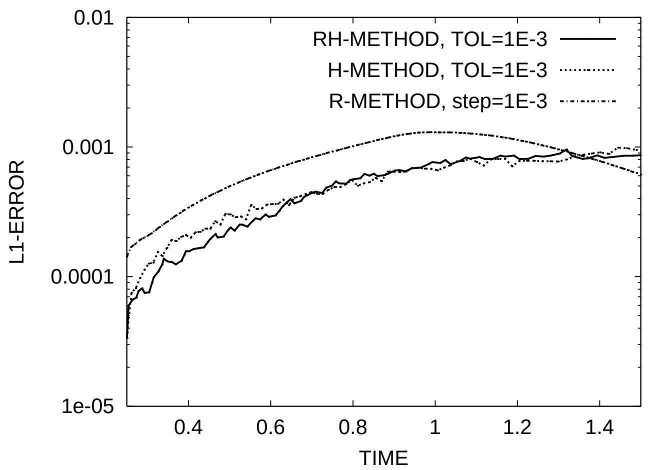

Figure 4: Burgers' equation: Temporal evolution of local L^1–errors for linear finite elements. All computations give comparable local error. The r–method in [9] is applied with a constant time step 0.001.

In Fig. 5, we plot the evolution of the number of grid points needed to reach the required accuracy. Not surprisingly, the h–method needs significantly more nodes than the other methods. The rh–method does a better job than the r–method, especially at the beginning and the end of the computation. There the length of the moving solution front is shorter than in the middle of the time interval, as shown in Fig. 7. The rh–method is able to adapt to increasing and decreasing nonuniformities through moving the mesh towards an error distribution rather than devoting excessive effort to adding too many points. A closer examination of the results for the time interval $[0.9, 1.1]$ in Fig. 6 shows the main advantage of the rh–method: First the r–method moves the nodes into regions of insufficient accuracy to ensure the required tolerance is satisfied. Then, when this is no longer possible, the h–method helps by refining (or coarsening) afterwards. In contrast, the pure h–method constantly refines and coarsens the mesh.

6.2 Flame problem

Our second example is a more practically relevant combustion problem modeling the propagation of a laminar flame through a heat absorbing obstacle

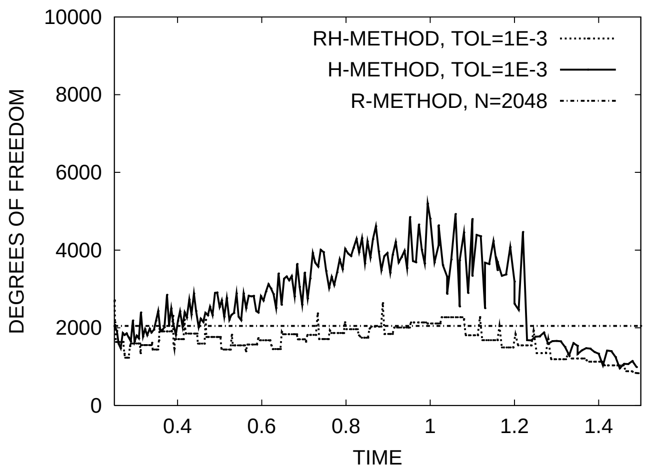

Figure 5: Burgers' equation: Number of grid points (for linear finite elements) needed to reach the accuracies shown in Fig. 4.

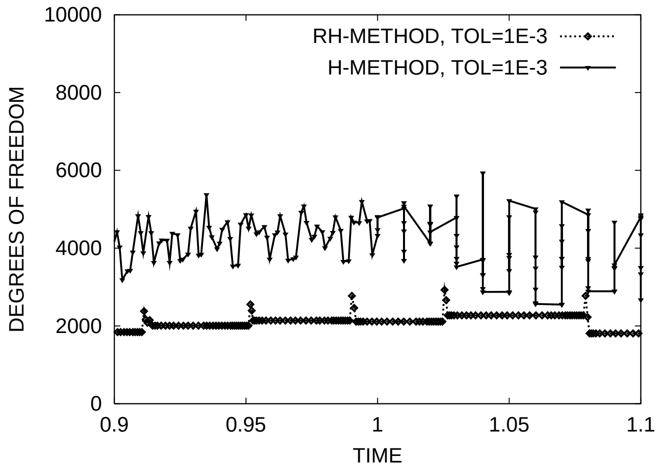

Figure 6: Burgers' equation: Temporal evolution of the number of grid points for linear finite elements in the time interval [0.9,1.1]. Whereas the rh–method keeps the number of grid points constant over a longer period, the h–method has to refine and coarse constantly due to the moving solution.

(see [30]). The equations for the dimensionless temperature T and the species

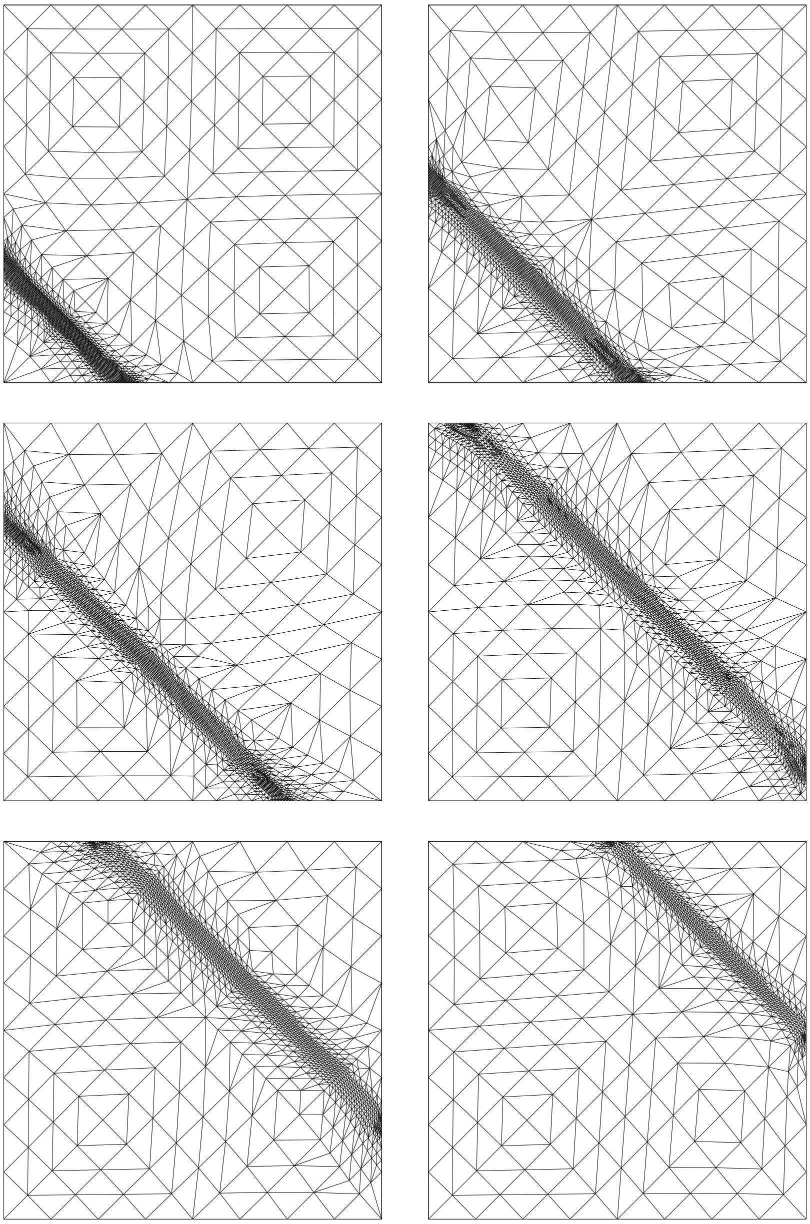

Figure 7: Burgers' equation: Selection of moving grids at different time points.

concentration C read

$$
\begin{aligned}
\partial_t T - \nabla^2 T \quad &= \omega, \\
\partial_t C - \tfrac{1}{Le}\nabla^2 C \quad &= -\omega,
\end{aligned}
$$

where ω is determined by an Arrhenius law

$$
\omega \; = \; \frac{\beta^2}{2Le}\, C\, e^{\frac{\beta(T-1)}{1+\alpha(T-1)}} \, .
$$

We set $Le = 1$, $\beta = 10$, and $\alpha = 0.8$. The physical domain $\Omega = [0,60] \times [0,16]$ is covered by two parallel cooled rods with rectangular cross section of length $L = 15$ and width $H = 4$ (see also Fig. 8). The absorption of heat is modeled by the boundary condition $\partial_n T = -\kappa T$, where the heat loss parameter κ is set to 0.1. On the left boundary Dirichlet conditions corresponding to the burnt state $T = 1$ and $C = 0$ are prescribed. The remaining boundary conditions are of homogeneous Neumann type. The initial solution is a right–travelling flame located left of the obstacle:

$$
T(x,y,0) \quad = \quad \begin{cases} 1 & \text{for} \quad x \le 9\,, \\ e^{9-x} & \text{for} \quad x > 9\,, \end{cases}
$$

$$
C(x,y,0) \quad = \quad \begin{cases} 0 & \text{for} \quad x \le 9\,, \\ 1 - e^{Le\,(9-x)} & \text{for} \quad x > 9\,. \end{cases}
$$

For the given κ, the flame speed slows down in the interior of the channel. The flame becomes curved, but manages to pass through.

We choose the tolerances for the h– and rh–method as TOLX=TOLT=5E-4 and set ATOL$_i$=1E-6 and RTOL$_i$=1.0 for all components in (10). Since the time scale of the underlying combustion process demands fast mesh adaptation, we use $\Theta = 0.1$ in (3) and $\alpha = 50.0$ in (19). In both cases, linear finite elements are used.

In Fig. 8 and Fig. 9, the moving meshes and the corresponding temperature level lines are depicted at various times. The moving grids follow the dynamics of the problem. Grid points lying at the front as well as at the back of the flame move towards the main combustion region. As before, the rh–method needs fewer points than the h–method to ensure comparable resolution. A reduction in the number of mesh points by up to a factor four can be observed from Fig. 10. The number of time steps chosen by the integrator ROS2 are 429 and 499 for the h– and rh–method, respectively. Closer examination reveals that in numerous instances the moving technique uses small time integration steps due to a sudden change in the local grid dynamics from coarsening — see Fig. 11 for cases where the time integrator is forced to reduce the time step.

7 Conclusion

We have presented a finite element method based on a combined rh–mesh refinement strategy. Major purposes are (i) to incorporate an r–refinement strategy

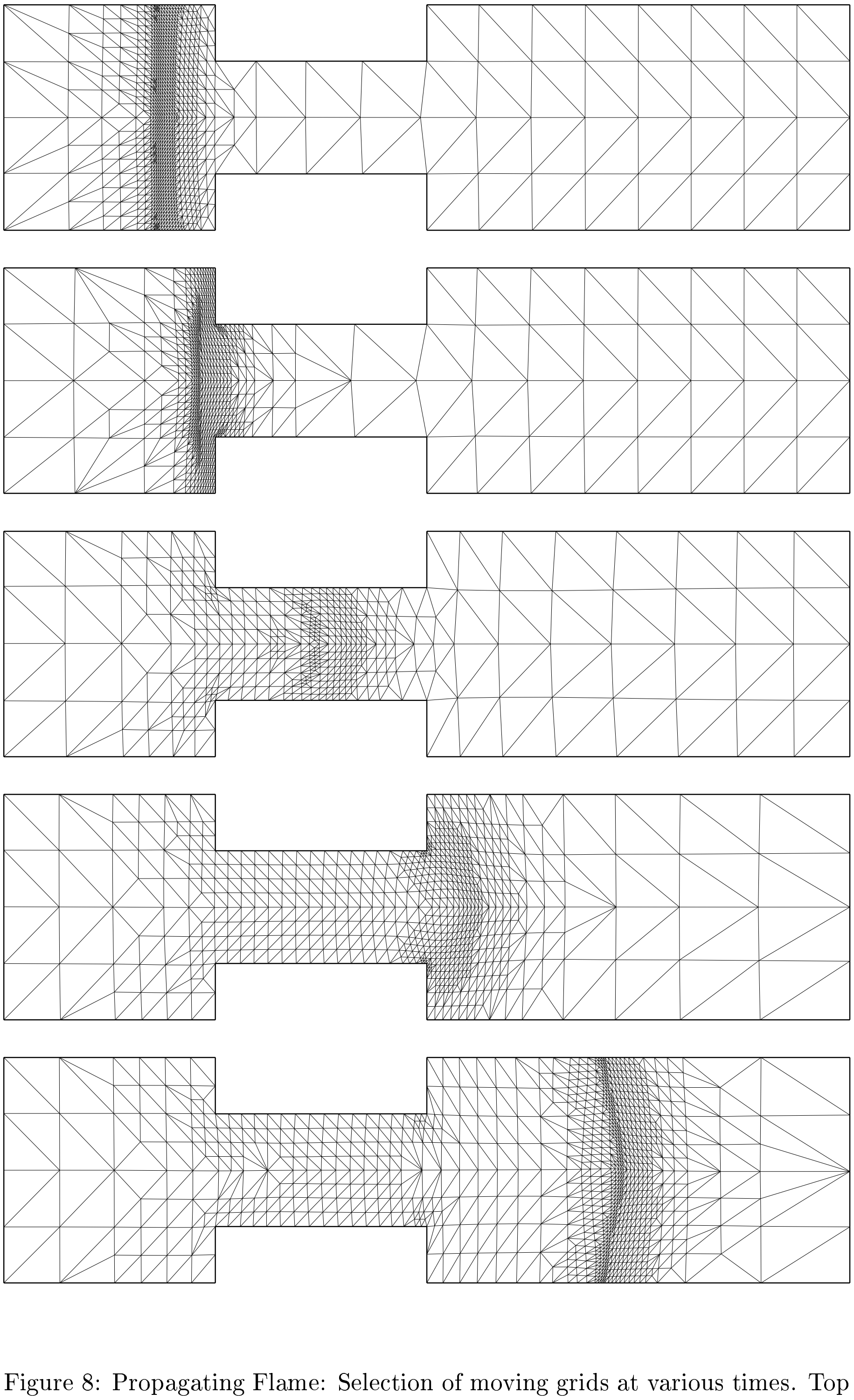

Figure 8: Propagating Flame: Selection of moving grids at various times. Top to bottom: $t = 2.27, 5.43, 19.1, 35.4, 50.0$.

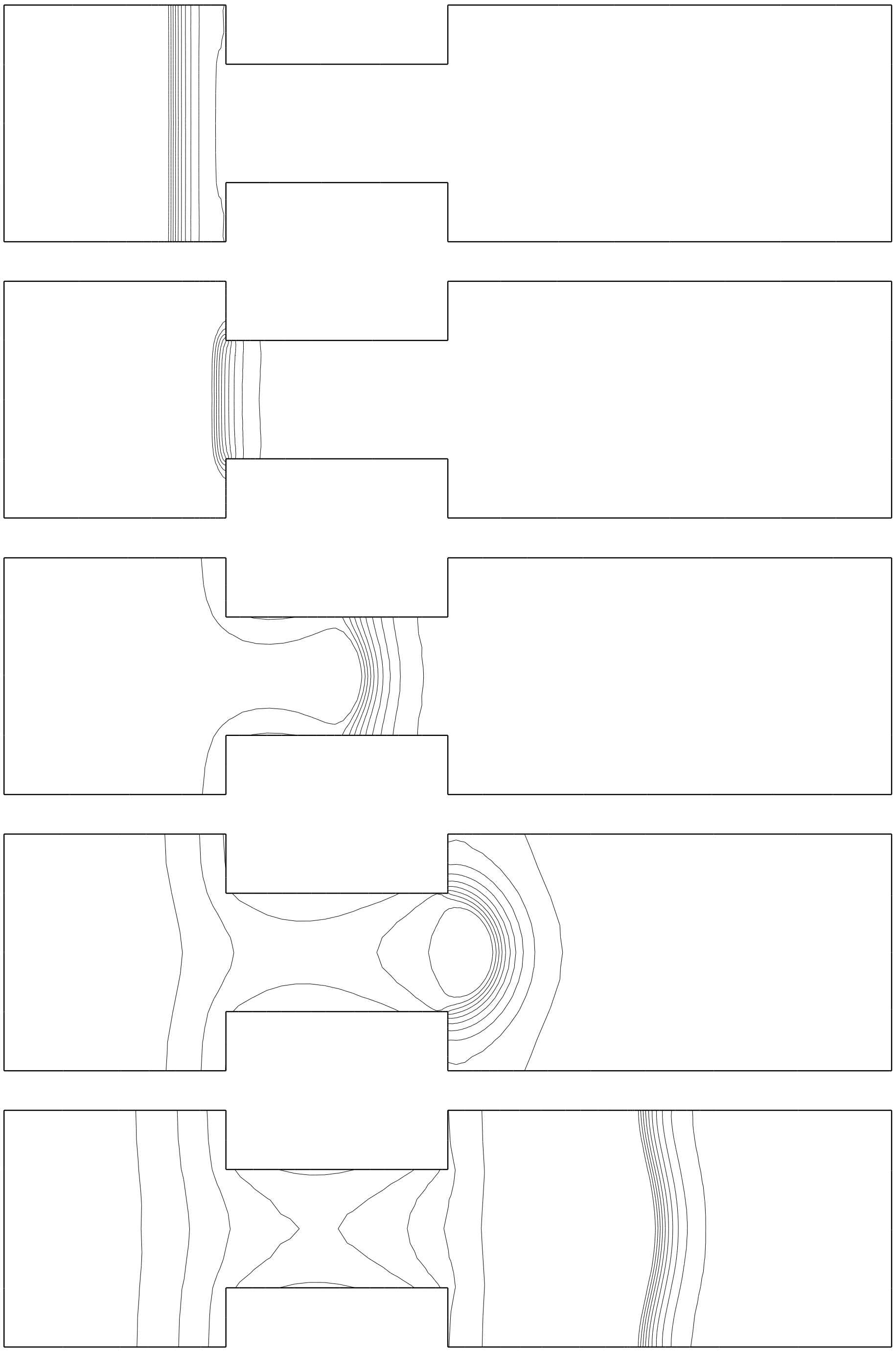

Figure 9: Propagating Flame: Selection of temperature level lines at various times. Top to bottom: $t = 2.27, 5.43, 19.1, 35.4, 50.0$.

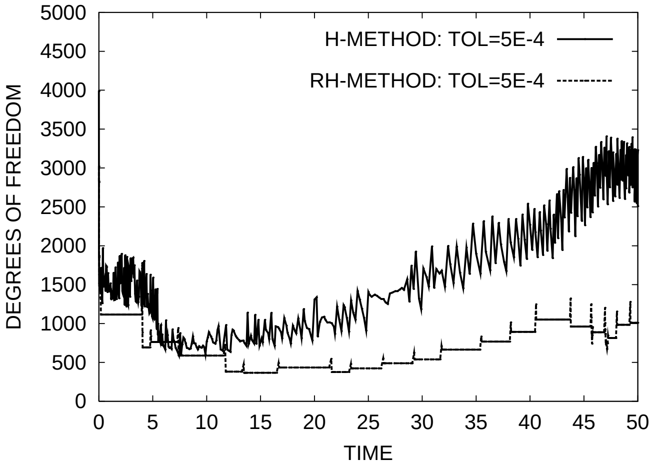

Figure 10: Propagating Flame: Number of grid points chosen for the h– and rh–method.

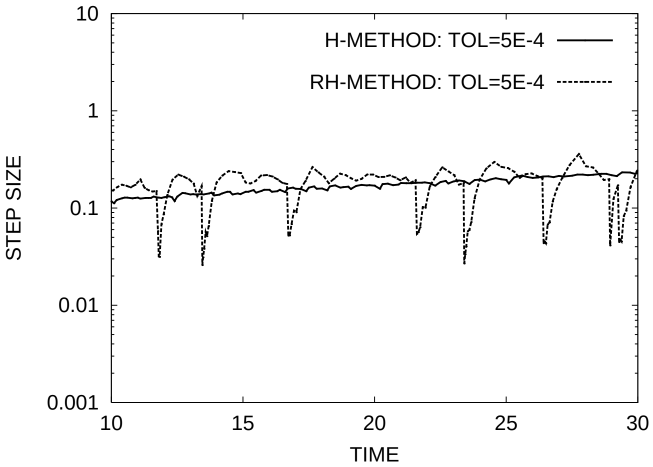

Figure 11: Propagating Flame: History of time steps chosen by $ROS2$ for the h– and rh–method.

into an h–refinement finite element code [30] to provide more efficiency by having better mesh alignment and (ii) to enhance an effective r–method (as used, e.g., in [11]) with global error control using h–refinement. The finite element method is based upon the horizontal method of lines. For it, the physical PDEs are integrated in time with a Rosenbrock–Wanner–type method. Hierarchical error estimates are used to guide both the mesh movement and local refinement. The general r–refinement method, originally developed in [25, 26], is based on solving a set of moving mesh PDEs.

The implementation of r–refinement here is fairly straightforward and has not been extensively tested to see that parameters are optimized for this well-tested h–refinement code. In addition, there has not as yet been a rigorous theoretical analysis of the method. Nevertheless, the overall feasibility of the general rh–refinement approach in this context is apparent. The numerical results are quite promising, demonstrating that a combined mesh refinement method can significantly reduce the number of degree of freedoms needed to reach a prescribed error tolerance. We anticipate that a considerably more efficient implementation of this method can be developed which will be ideal for solving a large class of time dependent problems with multiple-scales. The task of finding the most efficient rh–refinement method for time-dependent PDEs can be daunting given the number interconnected parameters and possible strategies for computing the solution and grid as the solution evolves. For example, r–movement can be done for only a relatively coarse mesh. Another approach worth investigating is to modify the form of the MMPDE as recently introduced in [23, 24]. Appropriately choosing the monitor function, the error during the r–refinement steps can be better coordinated with the error form for the finite element method with h–refinement. For steady state solutions, this could provide a mesh optimization analogous in principle to that in [13, 20]. Finally, it is natural and straightforward to apply such an rh–method to problems with moving boundaries. These are all issues which will be investigated in the future.

Acknowledgement. The first author is grateful to Peter Deuflhard from the Konrad–Zuse–Institut in Berlin for his support and to Robert D. Russell for his support from NSERC (Canada) grant OGP-0008781 during a visit to Simon Fraser University. The second and third author were supported in part by NSF grants DMS–0209313 and DMS–007424, respectively.

References

[1] D.C. Arney and J.E. Flaherty, An adaptive mesh–moving and local refinement method for time–dependent partial differential equations. *ACM Trans. Math. Software*, 16, 48–71 (1990).

[2] S. Adjerid, and J.E. Flaherty, A moving–mesh finite element method with local refinement for parabolic partial differential equations. *Comp. Meth. Appl. Mech. Engrg.*, 55, 3–26 (1986).

[3] R.E. Bank, PLTMG: A software package for solving elliptic partial differential equations - user's guide 8.0. SIAM, 1998.

[4] R.E. Bank and R.K. Smith, A posteriori error estimates based on hierarchical bases. *SIAM J. Numer. Anal.*, 30, 921–935, (1993).

[5] I. Babuška and M. Suri, The p and h-p version of the finite element method, an overview. *Comput. Meth. Appl. Mech. Engrg.*, 80, 5–26 (1990).

[6] M.J. Baines, Moving Finite Elements. Oxford University Press, Oxford, 1994.

[7] F.A. Bornemann, An adaptive multilevel approach to parabolic equations. III. 2D error estimation and multilevel preconditioning. *IMPACT of Comput. in Sci. and Engrg.*, 4, 1–45 (1992).

[8] F.A. Bornemann, B. Erdmann, and R. Kornhuber, A posteriori error estimates for elliptic problems in two and three space dimensions. *SIAM J. Numer. Anal.*, 33, 1188–1204, (1996).

[9] W. Cao, W. Huang and R.D. Russell, An r-adaptive finite element method based upon moving mesh PDEs. *J. Comput. Phys.*, 149, 221–244 (1999).

[10] W. Cao, W. Huang and R.D. Russell, A study of monitor functions for two–dimensional adaptive mesh generation. *SIAM J. Sci. Comput.*, 20 (1999), 1978–1994.

[11] W. Cao, W. Huang and R.D. Russell, An error indicator monitor function for an r-adaptive finite element method, *J. Comp. Phys.* 170, 871–892 (2001).

[12] N. Carlson and K. Miller, Design and application of a gradient–weighted moving finite element method. *SIAM J. Sci. Comput.*, 19, 798 (1998).

[13] J. Dompierre, M. Vallet, Y. Bourgault, M. Fortin, and W. Habashi, Anisotropic mesh adaption: towards user-independent, mesh-independent and solver-independent CFD. Part III: Unstructured grids.. *Internat. J. Num. Methods in Fluids*, 39, 675–702 (2002).

[14] P. Deuflhard, J. Lang, and U. Nowak, Adaptive algorithms in dynamical process simulation. in: H. Neunzert (ed.), Progress in Industrial Mathematics at ECMI'94, 122–137, Wiley–Teubner, 1996.

[15] P. Deuflhard, P. Leinen, and H. Yserentant, Concepts of an adaptive hierarchical finite element code. *IMPACT of Comput. in Sci. and Engrg.*, 1, 3–35 (1989).

[16] B. Erdmann, J. Lang, and R. Roitzsch, KASKADE manual, version 2.0. TR93–5, Konrad–Zuse–Zentrum für Informationstechnik Berlin, 1993.

[17] W.D. Gropp, Local uniform mesh refinement with moving grids. *SIAM J. Sci. Stat. Comput.*, 8, 292–304 (1987).

[18] K. Gustafsson, Control–theoretic techniques for stepsize selection in implicit Runge-Kutta methods. *ACM Trans. Math. Software*, 20, 496–517 (1994).

[19] K. Gustafsson, M. Lundh, and G. Söderlind, A PI stepsize control for the numerical solution of ordinary differential equations. *BIT*, 28, 270–287 (1988).

[20] W. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin and M. Vallet, Anisotropic mesh adaption: towards user-independent, mesh-independent and solver-independent CFD. Part I: General principles. *Internat. J. Num. Methods in Fluids*, 32, 725–744 (2000).

[21] E. Hairer and G. Wanner, Solving ordinary differential equations II, stiff and differential-algebraic problems. Second revised edition, Springer–Verlag, Berlin, Heidelberg, New York, 1996.

[22] W. Huang, Practical aspects of formulation and solution of moving mesh partial differential equations. *J. Comput. Phys*, 171, 753–775 (2001).

[23] W. Huang, Variational mesh adaptation: isotropy and equidistribution. *J. Comput. Phys.*, 174, 903–924 (2001).

[24] W. Huang and with W. Sun, Variational mesh adaptation II: Error estimates and monitor functions, *J. Comput. Phys.* (To appear).

[25] W. Huang and R.D. Russell, A high dimensional moving mesh strategy. *Appl. Numer. Math.*, 26, 63–76 (1997).

[26] W. Huang and R.D. Russell, Moving mesh strategy based upon a gradient flow equation for two dimensional problems. *SIAM J. Sci. Comput.*, 20, 998–1015 (1999).

[27] A.A. Johnson and T.E. Tezduyar, Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Comp. Meth. Appl. Mech. Engrg.*, 119, 73–94 (1994).

[28] A.P. Kuprat, Creation and annihilation of nodes for the moving finite element method. Ph.D., University of California, Berkeley, 1992.

[29] J. Lang, Adaptive FEM for reaction–diffusion equations. *Appl. Numer. Math.*, 26, 105–116 (1998).

[30] J. Lang, Adaptive multilevel solution of nonlinear parabolic PDE systems. Theory, algorithm, and applications. Lecture Notes in Computational Science and Engineering 16, Springer–Verlag, Berlin, Heidelberg, New York, 2000.

[31] K. Miller and R.N. Miller, Moving finite elements I. *SIAM J. Numer. Anal.*, 18, 1019–1032 (1981).

[32] B. Nkonga and H. Guillard, Godunov type method on non–structured meshes for three–dimensional moving boundary problems. *INRIA Report*, 1883 (1993).

[33] H.H. Rosenbrock, Some general implicit processes for the numerical solution of differential equations. *Computer J.*, 5, 329–331 (1963).

[34] A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, and F.A. Potra, Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers. *Atmos. Environ.*, 31, 3459–3472 (1997).

[35] H.A. van der Vorst, BI–CGSTAB: A fast and smoothly converging variant of BI–CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat.*, 13, 631–644 (1992).

[36] O.C. Zienkiewicz, J.G. Zhu and N.G. Gong, Effective and practical h-p adaptive analysis strategies for the finite element method. *Int. J. Numer. Meth. Engrg.*, 28, 879–891 (1989).