Adaptive Higher-Order Finite Element Methods for Transient PDE Problems Based on Embedded Higher-Order Implicit Runge-Kutta Methods

Pavel Solin[a,b], Lukas Korous[c]

[a] *Department of Mathematics and Statistics, University of Nevada, Reno, USA*
[b] *Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Prague*
[c] *Charles University, Prague, Czech Republic*

Abstract

We present a new class of adaptivity algorithms for time-dependent partial differential equations (PDE) that combines adaptive higher-order finite elements (*hp*-FEM) in space with arbitrary (embedded, higher-order, implicit) Runge-Kutta methods in time. Weak formulation is only created for the stationary residual of the equation, and the Runge-Kutta method is supplied via its Butcher's table. Around 30 Butcher's tables for various Runge-Kutta methods with numerically verified orders of local and global truncation errors are provided. New time-dependent benchmark problem with known exact solution that contains a moving front of arbitrary steepness is introduced, and used to compare the quality of seven embedded implicit higher-order Runge-Kutta methods. Numerical experiments also include a comparison of adaptive low-order FEM and *hp*-FEM with dynamically changing meshes. All numerical results presented in this paper are easily reproducible in the FEMhub Online Lab (http://femhub.org) and using the Hermes open source library (http://hpfem.org/hermes).

Keywords: Runge-Kutta method, Butcher's table, Finite element method, Automatic adaptivity, Dynamically changing meshes, Reproducible research

Email addresses: solin@unr.edu (Pavel Solin), lukas.korous@gmail.com (Lukas Korous)

1. Introduction

In recent years, adaptive higher-order spatial discretizations including spectral elements (SEM), higher-order finite elements (*hp*-FEM), and higher-order Discontinuous Galerkin (*hp*-DG) methods have become very popular in computational engineering and science. See, e.g., [6, 10, 12, 14, 17, 22, 23] and the references therein (this list is largely incomplete).

The next task of great practical importance is to combine these techniques with adaptive implicit higher-order time integration methods to design robust higher-order methods for transient problems that are adaptive both in space and in time. To our best knowledge, this has not been done yet.

When spatially-adaptive algorithms are extended to transient problems, it is customary to speak about adaptivity with *dynamic* or *dynamically-changing meshes*. Mostly this is done with low-order methods both in space and in time [1, 2, 9, 15, 16, 18, 19, 24]. Sometimes these methods are wrongly referred to as "Space-Time Finite Element Discretizations", which creates an impression that a $d+1$ dimensional space-time continuum is meshed and discretized using $d+1$ dimensional finite elements.

Higher-order spatial discretizations with dynamically-changing meshes are far less frequent in the context of transient problems, and to our best knowledge they have been only combined with non-adaptive time discretizations or with adaptive lower-order implicit time stepping methods such as in [7, 21].

1.1. Need for Implicit Higher-Order Methods

Explicit time stepping is not practical in conjunction with spatial adaptivity. This can be seen on parabolic problems where for stability reasons the time step size Δt is limited by the square of the size of the smallest mesh element $O(\Delta h^2)$. This is a serious constraint even without spatial adaptivity. When Δh is further reduced by mesh refinements, the time step easily becomes prohibitively small. CFL-like conditions for hyperbolic problems usually limit the time step by $O(\Delta h)$ instead of $O(\Delta h^2)$ but the outcome is similar.

We are particularly interested in *embedded* implicit higher-order Runge-Kutta methods that provide at the end of each time step a pair of approximations with different orders of accuracy. The difference between these two approximations can be used as an a-posteriori error estimator. With such error estimator in hand, it is possible to design trivial algorithms for adaptive

time step control. These algorithms will not be discussed in this paper, but we will look at the quality of the error estimate provided by several embedded implicit higher-order Runge-Kutta methods.

1.2. Outline

The outline of the paper is as follows: Section 2 provides a brief review of Runge-Kutta methods and Butcher's tables for future reference. Section 3 explains how the spatial discretization of a transient problem can be combined with an arbitrary Runge-Kutta method. Section 4 presents a database of around 30 Butcher's tables with verified local and global truncation errors. Section 5 introduces a time-dependent benchmark problem with known exact solution that contains a moving front of arbitrary steepness. This benchmark is used to compare the quality of seven embedded implicit higher-order Runge-Kutta methods in Section 6. In Section 7 we describe an adaptive algorithm for time-dependent problems with dynamically-changing meshes that works with any Runge-Kutta method. The algorithm is illustrated numerically in the same section. Finally, conclusions and outlook are formulated in Section 8.

2. Brief Review of Runge-Kutta Methods

For an ordinary differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \tag{1}$$

an s-stage Runge-Kutta method has the form

$$y_{n+1} = y_n + \Delta t \sum_{j=1}^{s} b_j k_j \tag{2}$$

where

$$k_i = f\left(t_n + \Delta t c_i, y_n + \Delta t \sum_{j=1}^{s} a_{ij} k_j\right). \tag{3}$$

Here t_n is the last time level, Δt the time step, and a_{ij}, b_i and c_i are known constants. The unknowns k_i are called *stage derivatives* and their calculation is the most demanding part of the computation. With known stage derivatives, the new time level approximation y_{n+1} in (2) is evaluated easily.

2.1. Butcher's Tables

The constants a_{ij}, b_i and c_i in (2), (3) can be written in the form of a table [4],

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Table 1: Butcher's table of a general Runge-Kutta method.

The Runge-Kutta method is explicit if $a_{ij} = 0$ whenever $j \geq i$, diagonally implicit if $a_{ij} = 0$ whenever $j > i$, and fully implicit otherwise. For the moment we will not distinguish between these cases and assume an arbitrary $s \times s$ matrix A. However, the final computer code needs to distinguish between them for efficiency reasons.

2.2. Embedded Runge-Kutta Methods

A distinct place among Runge-Kutta methods belongs to *embedded methods*. These methods can be written using a Butcher's table that has two B-rows,

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s
	b_1^*	b_2^*	$\dots$	b_s^*

.

Table 2: Butcher's table of an embedded Runge-Kutta method.

The second B-row is used in conjunction with (2) to calculate an extra approximation

$$y_{n+1}^* = y_n + \Delta t \sum_{j=1}^{s} b_j^* k_j \tag{4}$$

whose order of accuracy is less than the order of accuracy of the original approximation y_{n+1}. Note that the extra approximation y^*_{n+1} is very cheap since the stages $k_1, k_2, \ldots, k_s$ calculated in (3) are reused without change.

The difference between these two approximations can be used as an a-posteriori error estimator,

$$e_{n+1} \approx y_{n+1} - y^*_{n+1}.$$

2.3. Butcher's Tables for Other Than Runge-Kutta Methods

Let us remark that the Butcher's syntax is not limited to traditional Runge-Kutta methods. For example, the widely used implicit Crank-Nicolson method has the Butcher's table

$$\begin{array}{c|cc} 1 & 1/2 & 1/2 \\ 0 & 0 & 0 \\ \hline & 1/2 & 1/2 \end{array}.$$

2.4. Resources for Runge-Kutta Methods

There is a vast amount of resources on Runge-Kutta methods and Butcher's tables. Many tables and references can be found in the original Butcher's book [4]. The Wikipedia page on Runge-Kutta methods [25] contains many useful Butcher's tables and references as well.

3. Application to PDE Problems

Let us consider a general (nonlinear, time-dependent) partial differential equation

$$\frac{\partial u}{\partial t} = f(x_1, \ldots, x_d, t, u, \nabla u)$$

that after the semi-discretization is space takes the form

$$M \frac{\mathrm{d}Y}{\mathrm{d}t} = F(t, Y). \tag{5}$$

Here M is the mass matrix, F a (nonlinear) vector-valued function with N components, and $Y = Y(t)$ an N-vector of time-dependent unknown solution coefficients. Without time-dependent equation data equation (5) is autonomous, $F = F(Y)$. Explicit dependence of F on t may come, for example, from time-dependent equation coefficients, time-dependent boundary

conditions, time-dependent forcing term (such as, e.g., heat sources or electric change density), or time-dependent geometry.

For the application of the Runge-Kutta method (2), (3) we write (5) formally as

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = M^{-1}F(t, Y), \tag{6}$$

although the mass matrix is never inverted in practise.

In the light of (6), equations (2) and (3) become

$$Y_{n+1} = Y_n + \Delta t \sum_{j=1}^{s} b_j K_j \tag{7}$$

and

$$MK_i = F\left(t_n + \Delta t c_i, Y_n + \Delta t \sum_{j=1}^{s} a_{ij} K_j\right). \tag{8}$$

3.1. Newton's Method

Any standard method can be used to solve the nonlinear algebraic system (8). We will use the Newton's method as the nonlinearity F usually is differentiable. Equation (8) is rewritten as

$$\begin{pmatrix} M & 0 & \dots & 0 \\ 0 & M & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & M \end{pmatrix} \begin{pmatrix} K_1 \\ K_2 \\ \vdots \\ K_s \end{pmatrix} - \begin{pmatrix} F\left(t_n + \Delta t c_1, Y_n + \Delta t \sum_{j=1}^{s} a_{1j} K_j\right) \\ F\left(t_n + \Delta t c_2, Y_n + \Delta t \sum_{j=1}^{s} a_{2j} K_j\right) \\ \vdots \\ F\left(t_n + \Delta t c_s, Y_n + \Delta t \sum_{j=1}^{s} a_{sj} K_j\right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{9}$$

The $Ns \times Ns$ Jacobian matrix of the left-hand side has the form

$$\boldsymbol{J} = \{J_{ij}\}_{i,j=1}^{s}$$

where each J_{ij} is an $N \times N$ matrix of the form

$$J_{ij} = \delta_{ij} M - a_{ij} h J(t_n + \Delta t c_i, Y_n + \Delta t \sum_{j=1}^{s} a_{ij} K_j).$$

Here J stands for the Jacobian matrix $\mathrm{D}F(t, Y)/\mathrm{D}Y$, and δ_{ij} is the Kronecker delta.

3.2. Efficiency

Clearly, in the finite element method all the nonzero matrices J_{ij} have the same sparsity structure that, moreover, is identical to the sparsity structure of M. Furthermore, for every i the matrix $J(t_n + \Delta t c_i, Y_n + \Delta t \sum_{j=1}^{s} a_{ij} K_j)$ is the same in all J_{ij}, $1 \leq j \leq s$. If equation (8) is autonomous, then J_{ij} is the same for all $1 \leq i, j \leq s$.

If the time-integration is explicit then the solution of system (8) is reduced to the solution of s linear algebraic systems with the mass matrix M on the left-hand side. If the time-integration method is diagonally-implicit (DIRK), then the solution of system (8) is reduced to the solution of s nonlinear systems with a $N \times N$ Jacobian matrix. The full $Ns \times Ns$ Jacobian matrix should be used only with fully implicit methods whose Butcher's table contains nonzero elements above the diagonal.

4. Database of Verified Butcher's Tables

After an extensive literature and Internet search we collected around 30 Butcher's tables. Since some tables were incomplete (pending various "simple calculations") and some contained typos, we implemented and tested all of them in the FEMhub Online Laboratory [8]. Moreover, for each table we verified numerically the order of local and global truncation errors. This was done for both the lower and higher-order variants in embedded methods. The tables and verification scripts in Python can be found in the published worksheet "Arbitrary RK Method" in the FEMhub Online Lab. To access it, click on the "Published Worksheets" button on the login screen.

For illustration let us show two graphs that are part of the published worksheet "Arbitrary RK Method" in the FEMhub Online Lab. They correspond to a 5-stage embedded SDIRK method by Cash [5] of orders 2 and 4, and show that indeed the orders of the global truncation error are 2 and 4, respectively. Note that in the log-log plot, the order of the method is revealed by the slope of the convergence curve.

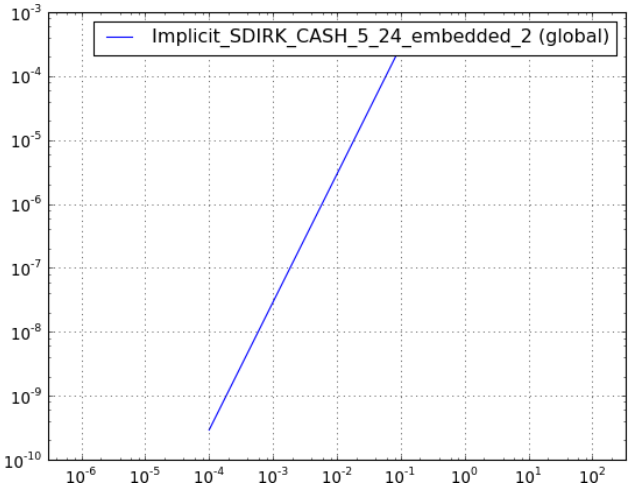

Figure 1: Log-log plot of the global truncation error for the second-order variant of the embedded implicit SDIRK method by Cash [5].

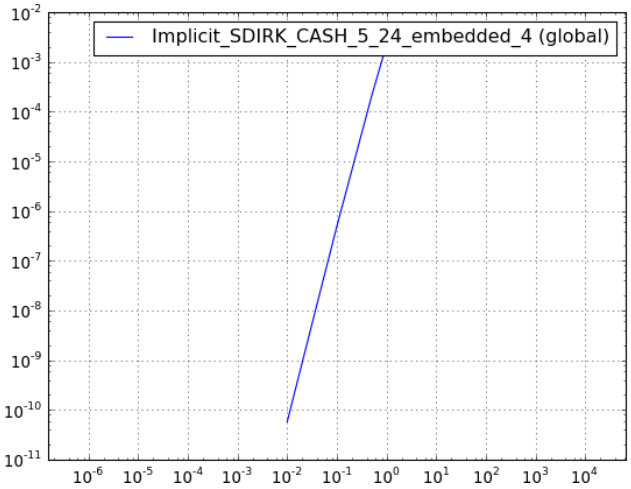

Figure 2: Log-log plot of the global truncation error for the fourth-order variant of the embedded implicit SDIRK method by Cash [5].

5. Time-Dependent Benchmark Problem With Known Exact Solution That Contains a Moving Front

To enable comparison of adaptive time-integration methods, we propose a simple model problem

$$\frac{\partial u}{\partial t} - \Delta u = f \tag{10}$$

in a square domain $\Omega = (x_0, x_1) \times (y_0, y_1)$ and time interval $(0, t_1)$. Equation (10) is equipped with homogeneous Dirichlet boundary conditions, zero initial condition, and the right-hand side f corresponds to an exact solution

$$u(x, y, t) = B(x, y)\arctan(t)\left(\frac{\pi}{2} - \arctan(S(R(x, y) - t))\right)$$

where $R(x, y) = \sqrt{x^2 + y^2}$ is the radius and

$$B(x, y) = \frac{(x - x_0)(x - x_1)(y - y_0)(y - y_1)}{C}$$

is a bubble function that enforces the Dirichlet boundary conditions.

The solution contains a moving front whose steepness can be influenced via the parameter S. The constant C serves for scaling purposes only. Fig. 3 shows the solution u for the values $x_0 = 0$, $x_1 = 10$, $y_0 = -5$, $y_1 = 5$, $C = 1000$ and $t = 5$. The steepness of the moving front is $S = 2$ (left) and $S = 20$ (right). As time approaches t_1, the adaptivity algoritm should coarsen the mesh so that no trace of a moving front is left.

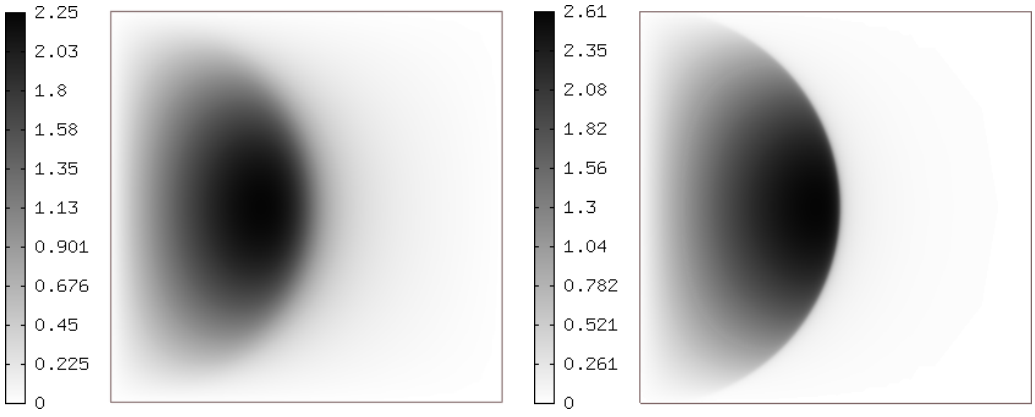

Figure 3: Exact solution u for $S = 2$ (left) and $S = 20$ (right) at $t = 5$.

6. Comparison of Selected Embedded Implicit Higher-Order Runge-Kutta Methods

Since we are going to compare temporal errors, we choose a small value of steepness $S = 2$. The mesh contains 16×16 fourth-order elements and 3969 degrees of freedom (DOF), and it virtually eliminates the spatial error. The values of the remaining parameters are left as before, and time step is chosen to be $\Delta t = 0.1$. We begin with the embedded method by Cash [5] of orders 2 and 3. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 4. Magnitudes of temporal error estimate and exact error in H^1-norm are shown in Fig. 5.

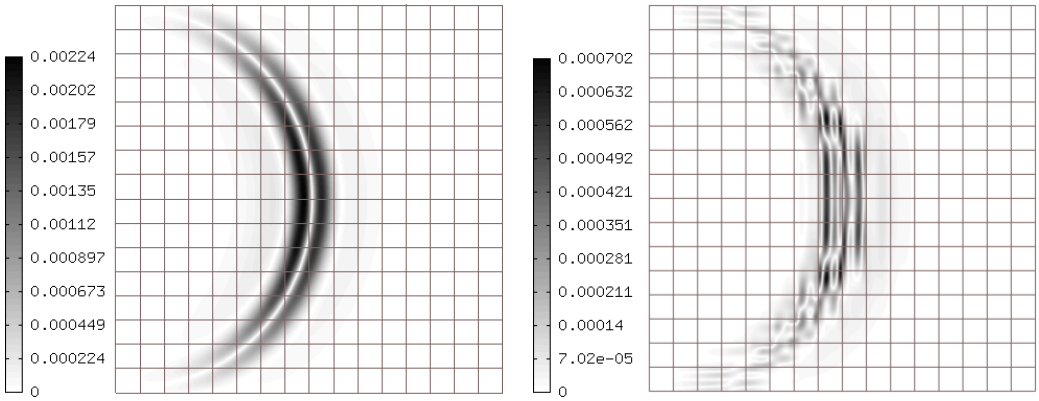

Figure 4: Error estimate (left) and exact error (right) for the embedded method by Cash [5] (orders 2 and 3) at $t = 5$.

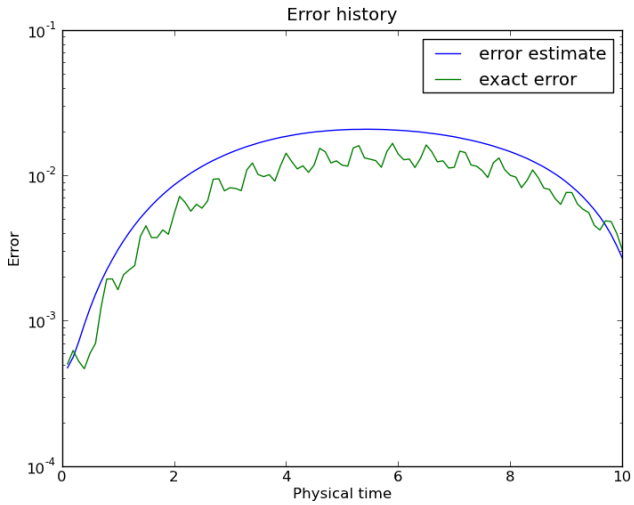

Figure 5: Magnitudes of temporal error estimate and exact error as functions of time.

Next we consider the embedded method by Billington [3] of orders 2 and 3. This method fails. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 6. Magnitudes of temporal error estimate and exact error are shown in Fig. 7.

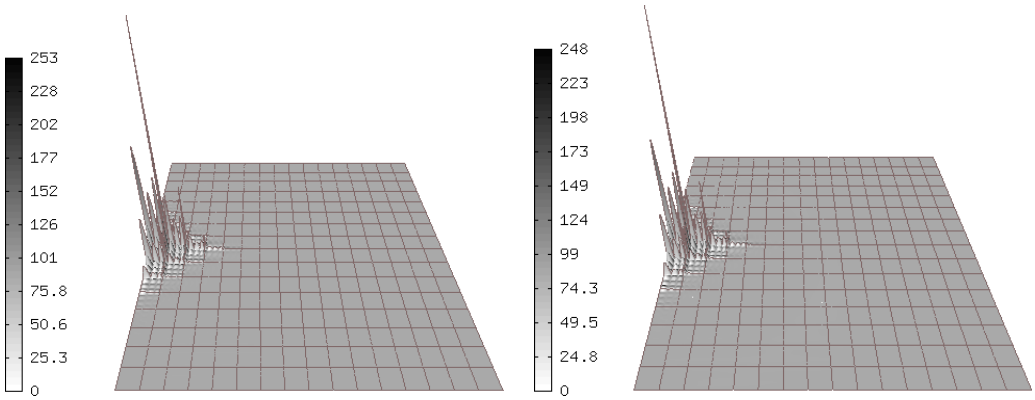

Figure 6: Error estimate (left) and exact error (right) for the embedded method by Billington [3] (orders 2 and 3) at $t = 5$.

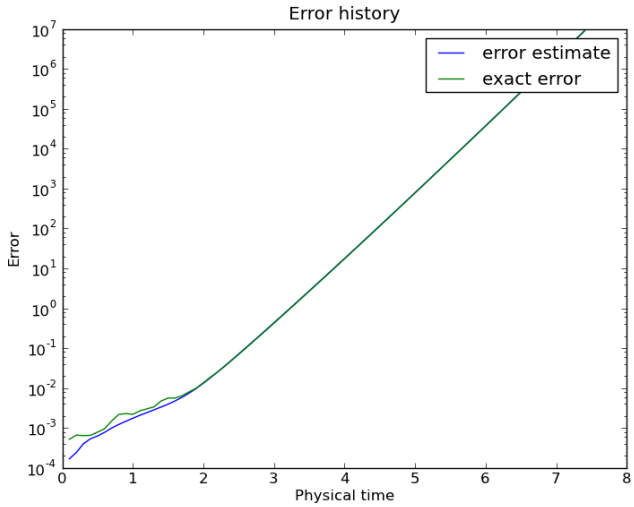

Figure 7: Magnitudes of temporal error estimate and exact error as functions of time.

Next let us look at the TR-BDF2 method [11] of orders 2 and 3. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 8. Magnitudes of temporal error estimate and exact error are shown in Fig. 9.

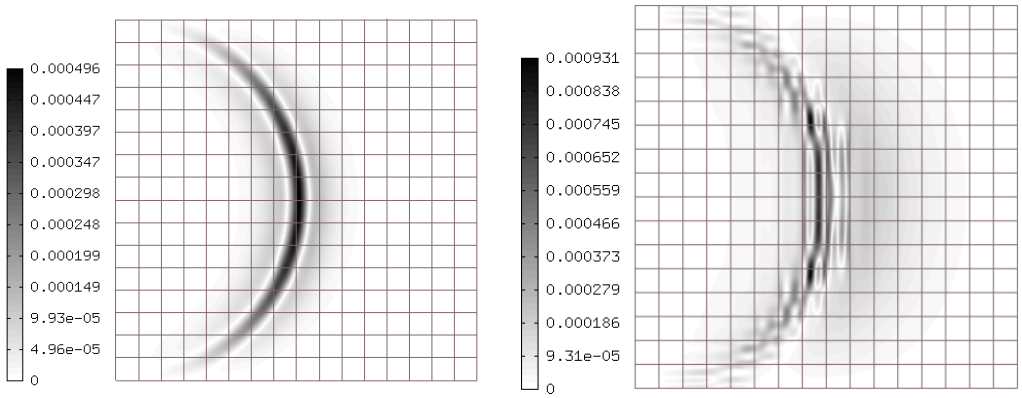

Figure 8: Error estimate (left) and exact error (right) for the embedded method TR-BDF2 [11] (orders 2 and 3) at $t = 5$.

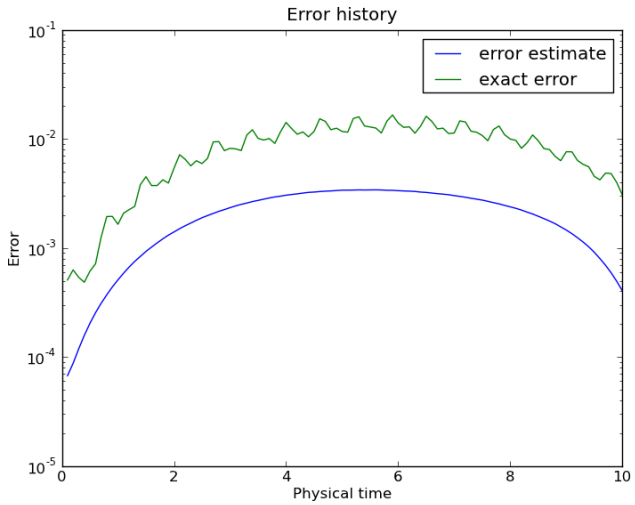

Figure 9: Magnitudes of temporal error estimate and exact error as functions of time.

The next method that we test is TR-X2 [11]. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 10. Magnitudes of temporal error estimate and exact error are shown in Fig. 11.

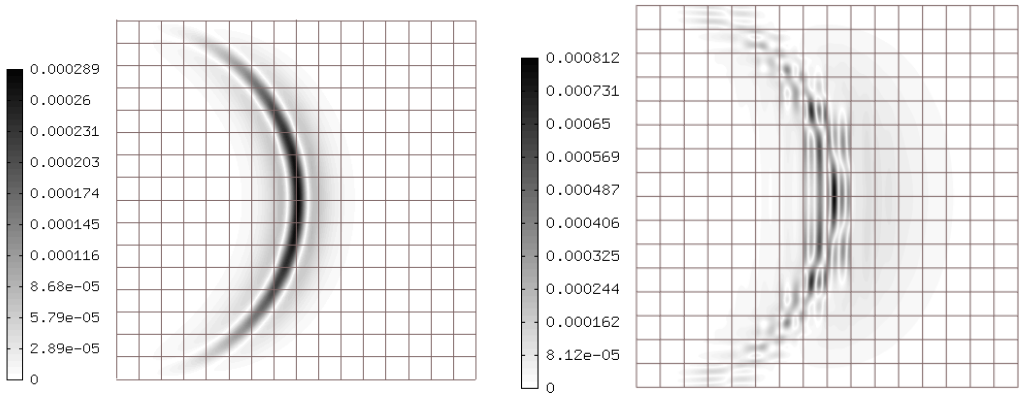

Figure 10: Error estimate (left) and exact error (right) for the embedded method TR-X2 [11] (orders 2 and 3) at $t = 5$.

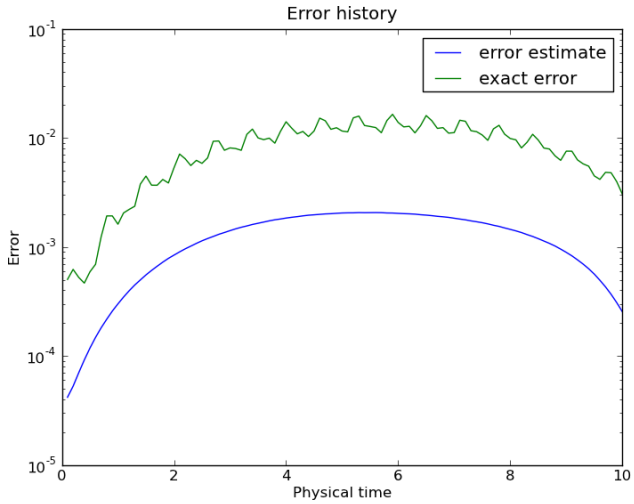

Figure 11: Magnitudes of temporal error estimate and exact error as functions of time.

Next on our list is the embedded method by Cash [5] of orders 2 and 4. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 12. Magnitudes of temporal error estimate and exact error are shown in Fig. 13.

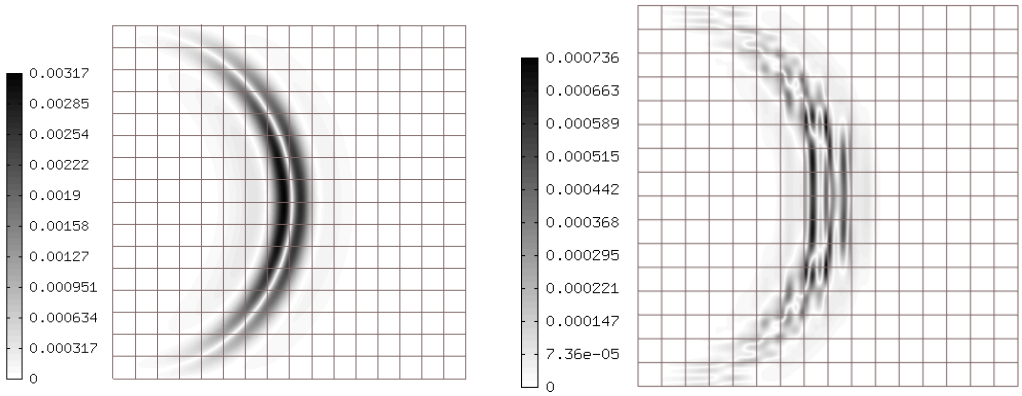

Figure 12: Error estimate (left) and exact error (right) for the embedded method by Cash [5] (orders 2 and 4) at $t = 5$.

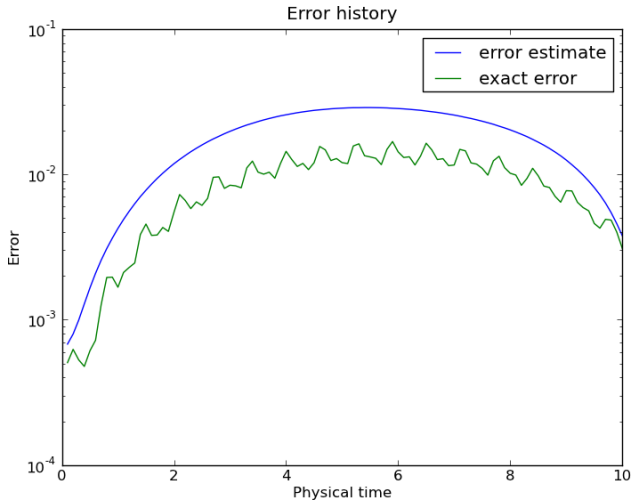

Figure 13: Magnitudes of temporal error estimate and exact error as functions of time.

Next we look at the embedded method by Cash [5] of orders 3 and 4. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 14. Magnitudes of temporal error estimate and exact error are shown in Fig. 15.

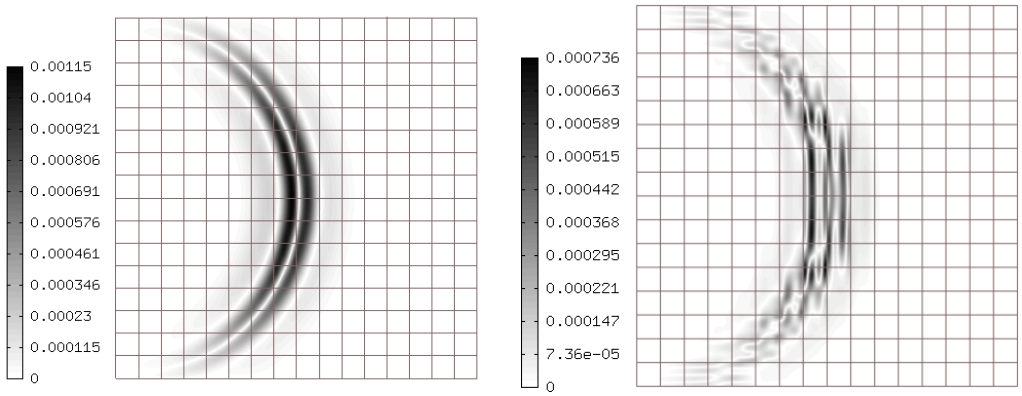

Figure 14: Error estimate (left) and exact error (right) for the embedded method by Cash [5] (orders 3 and 4) at $t = 5$.

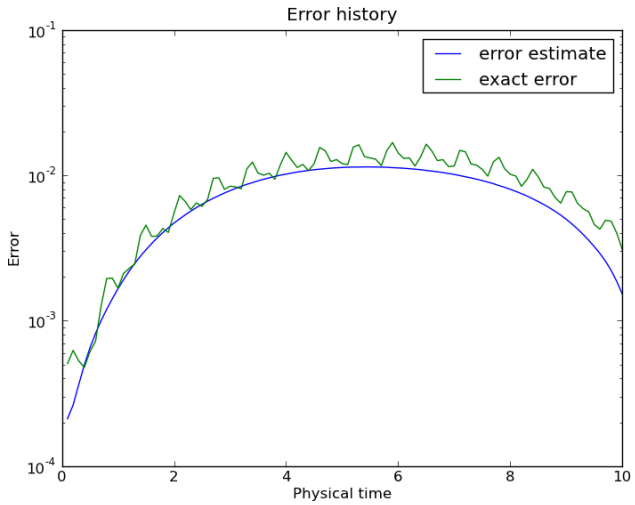

Figure 15: Magnitudes of temporal error estimate and exact error as functions of time.

The last method on our list is the embedded method by Ismail [13] of orders 4 and 5. The corresponding error estimate and exact error for $t = 5$ are shown in the left and right parts of Fig. 16. Magnitudes of temporal error estimate and exact error are shown in Fig. 17.

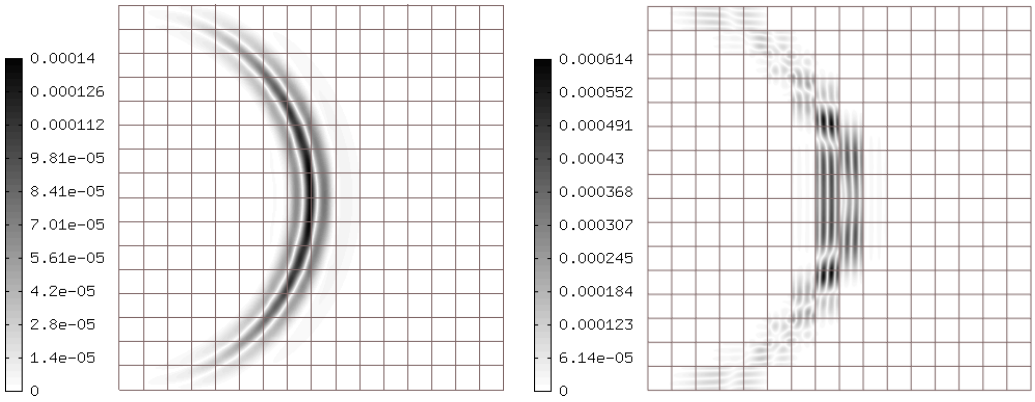

Figure 16: Error estimate (left) and exact error (right) for the embedded method by Ismail [13] (orders 4 and 5) at $t = 5$.

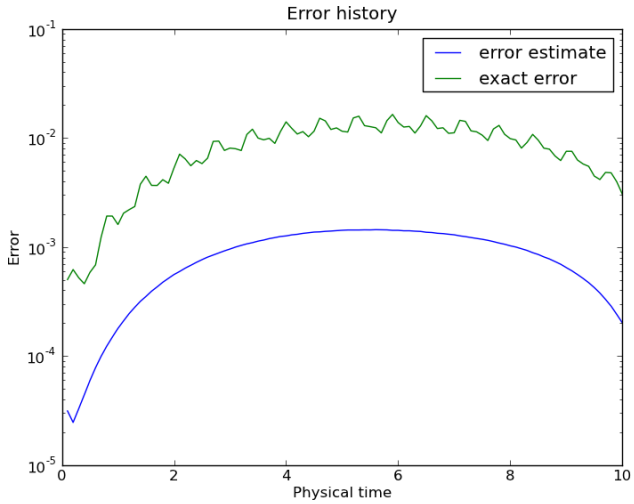

Figure 17: Magnitudes of temporal error estimate and exact error as functions of time.

6.1. Discussion of Results

The reader can see that (with the exception of the Billington's method) all methods were capable of identifying correctly the spatial distribution of the temporal approximation error. The magnitude of the temporal error was best estimated by the Cash 23, Cash 24 and Cash 34 methods. Remarkably good results were obtained by the Cash 34 method. The magnitude of the temporal error was significantly underestimated by the TR-BDF2, TR-X2 and Ismail methods. In particular the last is a disappointment since it uses seven stages, and thus it takes lots of CPU time compared to all other methods.

7. Spatial Adaptivity with Dynamically-Changing Meshes and Arbitrary Runge-Kutta Methods in Time

Given the limited length of the paper, we have to refer to [7, 20, 21] for basic ideas of multimesh assembling. In particular [7, 21] describe how the multimesh assembling is used to construct adaptive *hp*-FEM with dynamically changing meshes for transient PDE problems. However, papers [7, 21] are based on the Rothe's method where the temporal discretization is incorporated into the weak formulation. Thus the implementation of each new time stepping method means a new weak formulation and new computer code. In order to circumvent this problem, the present study employs the Method of lines which is more flexible in working with time stepping methods.

Let us assume that we just finished nth time step and have a locally refined mesh τ^n that was obtained using automatic *hp*-adaptivity, and an approximation u^n on it. The adaptivity algorithm for the next time step is as follows:

Step 1 (Global mesh derefinement):
Each time step with the exception of the first one begins with a global mesh derefinement. This step is implementation-dependent, and thus let us present three options that are available in the Hermes library:

- Reset the mesh τ^n to a very coarse base mesh τ_{base}. Reset the polynomial degrees of all elements to the initial polynomial degree p_{init}.
- Remove m refinement layers uniformly from all elements in the mesh τ^n. Here m is usually one, two or at most three. Reset the polynomial degrees of all elements to the initial polynomial degree p_{init}.
- Remove just one refinement layer uniformly from all elements in the mesh τ^n. Decrease the polynomial degrees of all mesh elements by one.

The first option is mathematically cleanest, meaning that the mesh obtained on the new time level is completely independent from the mesh that was generated during the last time step. The last option is fastest, but the sequence of dynamical meshes generated in this way may have on average more DOF than needed. The second option is a compromise. The coarsened mesh is denoted by τ_r^{n+1} where $r = 0$ is the counter of adaptivity steps. This is the initial mesh for the adaptive procedure that will lead to the next time level

approximation u^{n+1}.

Step 2 (Construction of a solution pair):
The mesh τ_r^{n+1} is refined globally in both h and p [22] and the resulting mesh is denoted by $\tau_{r,\ ref}^{n+1}$. Problem (8) is solved on the mesh $\tau_{r,\ ref}^{n+1}$ for the stage vectors $K_{r,\ 1}^{n+1}, K_{r,\ 2}^{n+1}, \ldots, K_{r,\ s}^{n+1}$. The stage vectors are combined linearly with the basis functions on the mesh $\tau_{r,\ ref}^{n+1}$ to obtain their equivalents in the finite element space on the mesh $\tau_{r,\ ref}^{n+1}$. These continuous, piecewise polynomial functions are denoted by $u_{r,\ K_1}^{n+1}, u_{r,\ K_2}^{n+1}, \ldots, u_{r,\ K_s}^{n+1}$. We add up these functions, weighted with the coefficients $b_1, b_2, \ldots, b_s$ and the time step, to obtain

$$u_{r,\ K}^{n+1} = \Delta t \sum_{j=1}^{s} b_j u_{r,\ K_j}^{n+1}.$$

Next, the function

$$u_{r,\ ref}^{n+1} = u^n + u_{r,\ K}^{n+1}$$

is projected to the mesh τ_0^{n+1} to extract its lower-order part. Since the approximations u^n and $u_{r,\ K}^{n+1}$ are defined on different meshes which, however, have a common coarse master mesh, the multimesh assembling technique [21] is employed. This is essential, since the multimesh technique ensures no loss of information during the projection. The result of the projection is denoted by u_r^{n+1}. Hence we have two approximations with different orders of accuracy, u_r^{n+1} and $u_{r,\ ref}^{n+1}$.

Step 3 (Mesh adaptation):
With the solution pair u_r^{n+1} and $u_{r,\ ref}^{n+1}$ in hand, we perform one step of *hp*-adaptivity on the mesh τ_r^{n+1} [22]. The result of the mesh adaptation step is denoted by τ_{r+1}^{n+1}. With this new mesh in hand, we can increase the counter of adaptivity steps $r := r+1$ and return to Step 1. The adaptivity algorithm is stopped when the norm of the relative error estimate

$$err_r = \frac{\|u_{r,ref}^{n+1} - u_r^{n+1}\|}{\|u_{r,ref}^{n+1}\|}$$

in some adaptivity step r is below a given relative error tolerance. Then we define $u^{n+1} := u_r^{n+1}$, $\tau^{n+1} := \tau_r^{n+1}$, and proceed to the next time step.

7.1. Comparison of low-order h-FEM and hp-FEM with dynamical meshes

The benchmark problem from Section 5 can be used to compare the performance of adaptive low-order h-FEM and adaptive hp-FEM. We keep the same parameter values as in Section 5, including the time step $\Delta t = 0.1$ and front steepness $S = 20$. In all cases, a relative error tolerance for spatial adaptivity of 1% is used as a stopping criterion. For this computation we use the fully implicit fifth-order three-stage Radau IIA method that virtually eliminates the temporal error.

Figs. 18 – 20 show approximate solutions and meshes for $t = 2.5$, $t = 5.0$ and $t = 7.5$ obtained with adaptive hp-FEM. In the mesh images, the numbers stand for polynomial degrees. All elements are quadrilaterals. Two colors within one element mean different directional polynomial degrees.

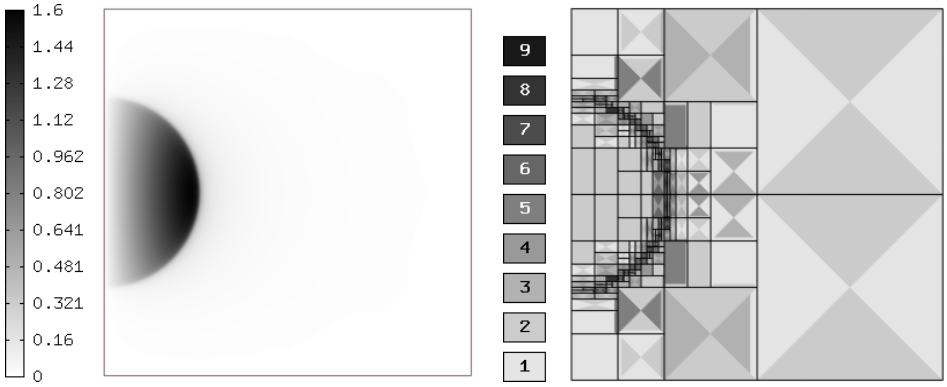

Figure 18: Approximate solution and mesh, time $t = 2.5$.

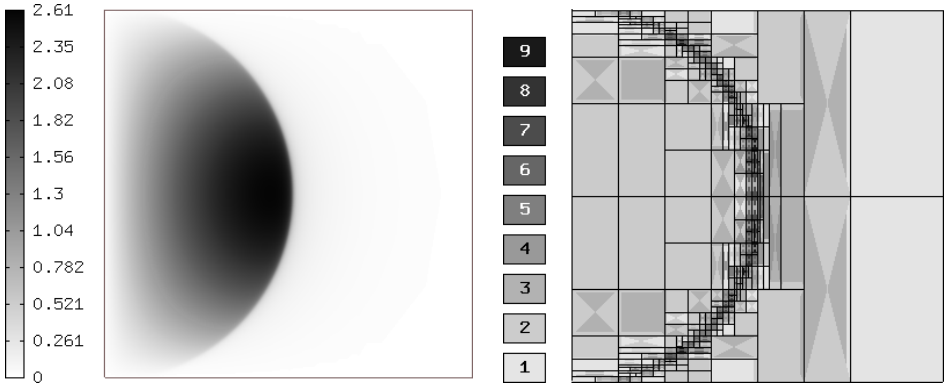

Figure 19: Approximate solution and mesh, time $t = 5.0$.

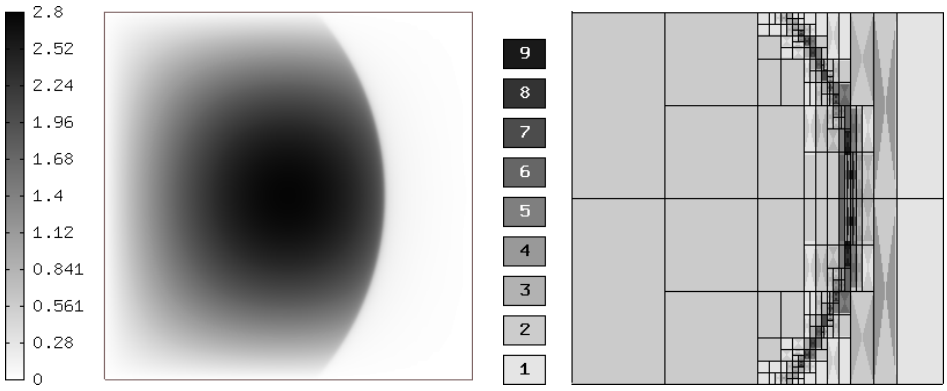

Figure 20: Approximate solution and mesh, time $t = 7.5$.

In Fig. 21 we present an analogous result to the *hp*-FEM mesh shown in the right part of Fig. 18, but now for adaptive *h*-FEM with $p = 2$. While for the *hp*-FEM the number of DOF at $t = 2.5$ was 2637 and for *h*-FEM with $p = 2$ it was 6707. For *h*-FEM with $p = 1$ the problem size grew rapidly and the computation was stopped after it exceeded 60000 DOF.

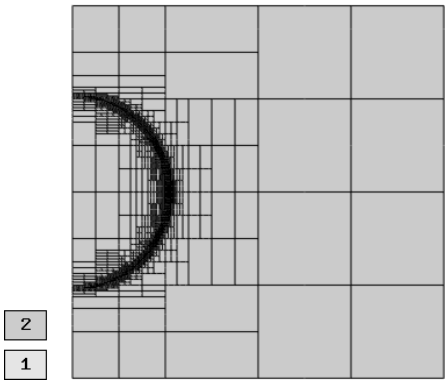

Figure 21: Mesh for adaptive *h*-FEM with $p = 2$ at $t = 2.5$.

A graphical overview of how the number of DOF evolves with physical time for these three computations is shown in Fig. 22. The fact that the number of DOF is not growing or descending monotonically is due to the stopping criterion. In each time step the adaptivity algorithm stops immediately after the error estimate is less than 1%, but the final value can differ from one time step to another. If the value of the error estimate in step $n+1$ is greater than in step n, then the number of DOF in step $n + 1$ can be less compared to step n. Whether this happens or not is random.

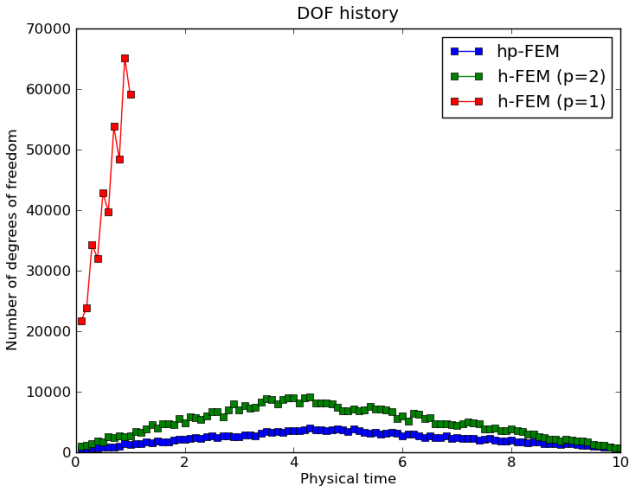

Figure 22: Problem size as a function of physical time.

8. Conclusion and Outlook

We presented a novel adaptive algorithm for time-dependent PDE problems that makes it possible to combine easily adaptive *hp*-FEM discretization in space with arbitrary Runge-Kutta methods in time. The Runge-Kutta method is selected at runtime by supplying its Butcher's table. We believe that this technique can be employed with minor changes for low-order finite element methods, finite volume methods, and Discontinuous Galerkin methods.

From the point of view of time integration, of particular interest are embedded implicit higher-order methods that can serve as a basis of very simple and efficient adaptive time-stepping schemes. We have numerically verified around 30 Butcher's tables and made them available in such a way that everyone can repeat the calculation in his/her own web browser via a few mouse clicks.

We introduced a new time-dependent benchmark problem with an exact solution that contains a moving front of arbitrary steepness. This benchmark was used to compare the quality of seven embedded implicit higher-order Runge-Kutta methods. The new benchmark problem was also used to compare the performance of adaptive low-order *h*-FEM with adaptive *hp*-FEM in space.

Our next steps will lead to the generalization of this method to multiphysics coupled problems, as well as to PDE systems where the time-

derivative is not present in all equations (such as in incompressible Navier-Stokes equations).

Acknowledgment

The work of the first author was supported by Subcontract No. 00089911 of Battelle Energy Alliance (U.S. Department of Energy intermediary) and by the Grant No. IAA100760702 of the Grant Agency of the Academy of Sciences of the Czech Republic.

References

[1] E. Bänsch: An Adaptive Finite-Element Strategy for the Three-Dimensional Time-Dependent Navier-Stokes Equations, J. Comput. Appl. Math. 36 (1991) 328.

[2] R. Bermejo, J. Carpio: A Space-Time Adaptive Finite Element Algorithm Based on Dual Weighted Residual Methodology for Parabolic Equations. SIAM J. Scientific Computing (2009), 3324 3355.

[3] S.R. Billington, Type-Insensitive Codes for the Solution of Stiff and Nonstiff Systems of Ordinary Differential Equations. In: Master Thesis, University of Manchester, United Kingdom (1983).

[4] J.C. Butcher: *Numerical Methods for Ordinary Differential Equations*, J. Wiley & Sons, 2003.

[5] J.R. Cash: Diagonally Implicit Runge-Kutta Formulae with Error Estimates, J. Inst. Maths Applies (1979) 24, 293-301.

[6] L. Demkowicz: *Computing with hp-Adaptive Finite Elements*, Taylor & Francis, 2007.

[7] L. Dubcova, P. Solin, J. Cerveny, P. Kus: Space and Time Adaptive Two-Mesh hp-FEM for Transient Microwave Heating Problems, Electromagnetics, Vol. 30, Issue 1, pp. 23 - 40, 2010.

[8] FEMhub Online Laboratory: http://femhub.org, retrieved June 25, 2011.

[9] Y.T. Feng, D. Peric: A Time-Adaptive Space-Time Finite Element Method for Incompressible Lagrangian Flows with Free Surfaces: Computational Issues, Comput. Methods Appl. Mech. Engrg. 190 (2000) 499-518.

[10] J. Hesthaven, T. Warburton: *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, 2008.

[11] M.E. Hosea, L.F. Shampine: Analysis and Implementation of TR-BDF2, Applied Numerical Mathematics 20, (1996) 21-37.

[12] P. Houston, C. Schwab, E. Süli: Discontinuous *hp*-Finite Element Methods for Advection-Diffusion-Reaction Problems, SIAM J. Numer. Anal. (2002) 2133-2163.

[13] F. Ismail et al: Embedded Pair of Diagonally Implicit Runge-Kutta Method for Solving Ordinary Differential Equations, Sains Malaysiana 39 (2010), 10491054.

[14] H.C. Krokowski et al: *hp-Finite Element Methods for Singular Perturbations*, Springer Berlin Heidelberg, 2008.

[15] J. Lang, D. Teleaga: Towards a Fully Space-Time Adaptive FEM for Magnetoquasistatics, IEEE Transactions on Magnetics (2008), 1238 - 1241.

[16] D. Meidner: Adaptive Space-Time Finite Element Methods for Optimization Problems Governed by Nonlinear Parabolic Systems, Ph.D. thesis, Universitt Heidelberg, Heidelberg, 2007.

[17] W. Mitchell: A Survey of hp-Adaptive Strategies for Elliptic Partial Differential Equations, Recent Advances in Computational and Applied Mathematics (T. E. Simos, ed.), Springer, 2011, 227-258.

[18] M. Moeller: Adaptive high-resolution finite element schemes, Ph.D. dissertation, Technical University of Dortmund, 2008.

[19] M. Schmich, B. Vexler: Adaptivity with Dynamic Meshes for Space-Time Finite Element Discretizations of Parabolic Equations. SIAM J. Scientific Computing (2008), 369 393.

[20] P. Solin, J. Cerveny, L. Dubcova, D. Andrs: Monolithic Discretization of Linear Thermoelasticity Problems via Adaptive Multimesh hp-FEM, J. Comput. Appl. Math 234 (2010) 2350 - 2357.

[21] P. Solin, L. Dubcova, J. Kruis: Adaptive hp-FEM with Dynamical Meshes for Transient Heat and Moisture Transfer Problems, J. Comput. Appl. Math. 233 (2010) 3103-3112.

[22] P. Solin, K. Segeth, I. Dolezel: *Higher-Order Finite Element Methods*, CRC Press, 2003.

[23] B. Stamm, T. P. Wihler: *hp*-Optimal Discontinuous Galerkin Methods for Linear Elliptic Problems, EPFL-IACS report 07.2007.

[24] L.L. Thompson, D. He: Adaptive Space-Time Finite Element Methods for the Wave Equation on Unbounded Domains, Comput. Methods in Appl. Mech. Engrg. (2005) 1947-2000.

[25] Wikipedia page on Runge-Kutta Methods: http://en.wikipedia.org/wiki/Runge-Kutta_methods. Retrieved June 25, 2011.