European Congress on Computational Methods
in Applied Sciences and Engineering (ECCOMAS 2012)
J. Eberhardsteiner et.al. (eds.)
Vienna, Austria, September 10-14, 2012

EFFICIENT TIME INTEGRATION FOR DISCONTINUOUS GALERKIN METHOD FOR THE UNSTEADY 3D NAVIER-STOKES EQUATIONS

Philipp Birken[1], Gregor Gassner[2], Mark Haas[3] and Claus-Dieter Munz[2]

[1]University of Kassel
Institute of Mathematics, Heinrich-Plett-Str. 40, 34132 Kassel, Germany
e-mail: birken@mathematik.uni-kassel.de
supported by the DFG in the context of the SFB/TR TRR 30

[2] University of Stuttgart
Institute of Aerodynamics and Gas Dynamics, Pfaffenwaldring 21, 70569 Stuttgart, Germany
e-mail: {gassner,munz}@iag.uni-stuttgart.de
supported by the DFG in the context of the *Schwerpunktprogramm MetStroem* and the cluster of excellence *Simulation Technology (SimTech)*

[3] Bosch GmbH
70442 Stuttgart, Germany

Keywords: Discontinuous Galerkin, Unsteady flows, Navier-Stokes, Three dimensional problems, Time integration, Tolerance Scaling

Abstract. *We look at adaptive time integration in the context of discontinuous Galerkin methods for the three dimensional unsteady compressible Navier-Stokes equations. Several explicit and implicit schemes will be compared.*

1 INTRODUCTION

In this paper we consider the aspect of time integration in the context of discontinuous Galerkin methods (DG) for the three dimensional unsteady compressible Navier-Stokes equations. The method of lines approach leads in this context typically to very stiff problems, where timesteps in explicit methods are driven by stability alone and in implicit methods by accuracy alone. Thus, if the resulting large linear or nonlinear equation systems can be solved efficiently, implicit methods are preferrable. In contrast to finite volume methods, where implicit methods are standard [19], the lack of these efficient solvers in the DG context has limited the use of implicit schemes.

So far, a number of people have looked at DG methods for unsteady flows. Wang and Mavriplis looked at different time integration schemes for unsteady Euler equations in two dimensions [37]. They found higher order DIRK methods to be superior to both BDF methods and the implicit midpoint rule. As a solver, a p-multigrid method was employed. Kanevsky et al. [22] considered both Euler and Navier-Stokes equations in two dimensions and suggested the use of implicit-explicit (IMEX) schemes. A JFNK method was used to solve the nonlinear systems. Persson and Perraire looked at the two dimensional unsteady Navier-Stokes equations with BDF time integration in conjunction with an ILU preconditioned Newton-Krylov solver [29]. The use of Rosenbrock-W methods for the Euler equations in meteorology was considered by St-Cyr et al. [35]. Another class of method is the so-called Space-Time-DG scheme developed by van der Vegt and van der Veen [24] where time is simply treated as an additional dimension in the DG discretization.

These articles leave a number of questions unanswered. First of all, only two dimensional problems are considered. Second, with the exception of the study by Wang and Mavriplis, the articles provide proofs of concept and only little comparison. Furthermore, the question of the best time integration method depends both on the specific application and on the method used to solve the resulting nonlinear equation systems. This is in particular shown by the studies [21] for finite element discretizations of the unsteady incompressible Navier-Stokes equations and [4] for finite volume discretizations of the unsteady laminar Navier-Stokes equations. Finally, a reasonable assessment of time integration schemes needs to take into account time adaptive calculations for realistic test cases.

To tackle these questions, we use a fixed solver for the appearing nonlinear systems, namely a preconditioned Jacobian-Free Newton-Krylov (JFNK) solver. As a preconditioner, we employ the newly suggested ROBO-SGS (for Reduced Offdiagonal Block-Order) that was shown to be efficient and parallely scalable [5]. This preconditioner uses very little memory, which is of crucial importance to tackle 3D problems using a DG method. In particular, ROBO-SGS exploits the hierarchic properties of a modal basis, which is why we use the modal DG scheme with nodal integration of Gassner et al. [16].

Then, we consider different explicit and implicit time integration schemes and compare them regarding their errors, convergence orders and computational efficiency for several test problems. In particular, we will employ time adaptive schemes. Tolerance calibration and scaling [33] is used to connect the tolerance, which is based on basic error estimates to the actual error. In the end, this will result in methods where the user choses a tolerance for the time integration error, which then automatically defines the tolerances for the Newton solver and the GMRES method. Furthermore, it gives a tool to compare the error of different time integration schemes.

For explicit time integration methods, we will consider explicit Runge-Kutta methods and the Space-Time-Expansion DG of Gassner et al. [25]. As implicit methods, we will consider

Explicit step Singly Diagonally Implicit Runge-Kutta (ESDIRK) schemes, Singly Diagonally Implicit Runge-Kutta (SDIRK) schemes and Rosenbrock methods. BDF methods are not considered for several reasons. First of all, all studies so far showed that for time adaptive flow computations, other methods are more efficient. Furthermore, these methods are not A-stable, but only A(α)-stable for orders greater than two. However, as we will demonstrate, some of the eigenvalues of the DG discretized problems are typically very close to the imaginary axis, rendering these methods unsuitable.

The outline of the paper is as follows: First we will describe the governing equations and the DG methodology. Then we will describe the different time integration schemes, after which we will briefly describe the JFNK method employed. Finally, numerical results are presented.

2 GOVERNING EQUATIONS

The Navier-Stokes equations are a second order system of conservation laws (mass, momentum, energy) modeling viscous compressible flow. Written in conservative variables density ρ, momentum $\mathbf{m}$ and energy per unit volume ρE:

$$\begin{aligned}
\partial_t \rho + \nabla \cdot \mathbf{m} &= 0, \\
\partial_t m_i + \sum_{j=1}^{d} \partial_{x_j}(m_i v_j + p\delta_{ij}) &= \frac{1}{Re}\sum_{j=1}^{d} \partial_{x_j} S_{ij} + q_i, \qquad i = 1 \dots d \\
\partial_t(\rho E) + \nabla \cdot (H\mathbf{m}) &= \frac{1}{Re}\sum_{j=1}^{d} \partial_{x_j}\left(\sum_{i=1}^{d} S_{ij} v_i - \frac{1}{Pr} W_j\right) + q_e.
\end{aligned}$$

Here, d stands for the number of dimensions, H for the enthalpy per unit mass, **S** represents the viscous shear stress tensor and W the heat flux. As the equation are dimensionless, the Reynolds number Re and the Prandtl number Pr appear. The equations are closed by the equation of state for the pressure $p = (\gamma - 1)\rho e$, where we assume an ideal gas. Finally, q_e denotes a possible source term in the energy equation, whereas $\mathbf{q} = (q_1, ..., q_d)^T$ is a source term in the momentum equation, for example due to external forces.

3 SPATIAL DISCRETIZATION

We employ the mixed modal-nodal Discontinuous Galerkin scheme which has been suggested by Gassner et al. [16]. One of the main advantages of this method is that it allows the use of elements of arbitrary shape (i.e. tetrahedrons, prisms, pyramids, hexahedrons, ...) with high order of accuracy.

3.1 The Discontinuous Galerkin Method

We consider conservation laws of the form

$$\mathbf{u}_t + \nabla \cdot \vec{f}(\mathbf{u}) = \mathbf{q}(t, \mathbf{u}), \tag{1}$$

with suitable initial and boundary conditions in a domain $\Omega \times [0,T] \subset R^d \times R_0^+$. Here, $\mathbf{u} = \mathbf{u}(\vec{x}, t) \in R^{d+2}$ is the state vector, $\vec{f}(\mathbf{u}) = \vec{f}^C(\mathbf{u}) - \vec{f}^D(\mathbf{u}, \nabla\mathbf{u})$ is the physical flux, where $\vec{f}^C(\mathbf{u})$ is the convective (i.e. hyperbolic) and $\vec{f}^D(\mathbf{u}, \nabla\mathbf{u})$ the diffusive (i.e. parabolic) flux component. The possibly time and space dependent source term is given by $\mathbf{q}(t, \vec{x}, \mathbf{u})$.

We derive the DG method by first subdividing the domain Ω into non-overlapping grid cells Q. In each grid cell we approximate the state vector using a local polynomial approximation of the form

$$\mathbf{u}\left(\vec{x},t\right)\big|_{Q} \approx \mathbf{u}^{Q}\left(\vec{x},t\right) = \sum_{j=1}^{N} \hat{\mathbf{u}}_j\left(t\right)\varphi_j^{Q}\left(\vec{x}\right), \tag{2}$$

where in our case, $\{\varphi_j^{Q}\left(\vec{x}\right)\}_{j=1,\ldots,N}$ are modal hierarchical orthonormal basis functions and $\hat{\mathbf{u}}$ are the corresponding coefficients in that cell. The basis functions are constructed from a monomial basis with a simple Gram-Schmidt orthogonalization algorithm for arbitrary (reference) grid cell types. The dimension of the local approximation space depends on the spatial dimension d and the polynomial degree p

$$N = N(p,d) = \frac{(p+d)!}{p!d!}. \tag{3}$$

The next step of our approximation is to define how the unknown degrees of freedom $\hat{\mathbf{u}}_j\left(t\right)$ are determined. Neglecting the source term for now, we insert the approximate solution (2) into the conservation law (1), multiply with a smooth test function $\phi = \phi\left(\vec{x}\right)$ and integrate over Q to obtain

$$\langle \mathbf{u}_t^{Q} + \nabla\cdot\vec{f}\left(\mathbf{u}^{Q}\right), \phi\rangle_Q = 0, \tag{4}$$

where $\langle .,.\rangle_Q$ denotes the $L_2(Q)$ scalar product over Q. We proceed with an integration by parts to obtain

$$\langle \mathbf{u}_t^{Q}, \phi\rangle_Q + \left(\vec{f}\left(\mathbf{u}\right)\cdot\vec{n}, \phi\right)_{\partial Q} - \langle \vec{f}\left(\mathbf{u}^{Q}\right), \nabla\phi\rangle_Q = 0, \tag{5}$$

where $(.,.)_{\partial Q}$ denotes the surface integral over the boundary of the element Q. As the approximate solution is in general discontinuous across grid cell interfaces, the trace of the flux normal component $\vec{f}\left(\mathbf{u}\right)\cdot\vec{n}$ is not uniquely defined. To get a stable and accurate discretization, several choices for the numerical approximation are known. Here, we use the HLLC flux [36]. For a purely convective problem inserting the trace approximation $\vec{f}\left(\mathbf{u}^{Q}\right)\cdot\vec{n} \approx g^{C}\left(\mathbf{u}^{-},\mathbf{u}^{+},\vec{n}\right)$ into equation (5) would yield

$$\langle \mathbf{u}_t^{Q}, \phi\rangle_Q + \left(g^{C}\left(\mathbf{u}^{-},\mathbf{u}^{+},\vec{n}\right), \phi\right)_{\partial Q} - \langle \vec{f}^{C}\left(\mathbf{u}^{Q}\right), \nabla\phi\rangle_Q = 0. \tag{6}$$

We denote by $(.)^{-}$ values at the inner side of a cell interface, i.e. values that depend on $\mathbf{u}^{Q}$ and by $(.)^{+}$ values that depend on the neighbor cells sharing the interface with the cell Q.

The handling of the diffusive part of the flux is a little more delicate for DG methods, because the jump in the gradients needs special handling. Several authors have suggested solutions for this problem [28, 8, 1, 2, 3] and all of these have been used in conjunction with implicit temporal discretizations. In this work we apply the so-called dGRP flux by Gassner, Lörcher and Munz [13, 15, 14, 26]. This is a symmetric interior penalty based method that has been derived in a way that optimizes stability, i. e. minimizes the eigenvalues of the DG operator and yields optimal order of convergence for mixed hyperbolic/parabolic PDEs such as the compressible Navier-Stokes equations [13, 15]. From a technical point of view this flux introduces an approximation of the trace of the flux normal component $\vec{f}^{D}\left(\mathbf{u},\nabla\mathbf{u}\right)\cdot\vec{n} \approx g^{\text{dGRP}}\left(\mathbf{u}^{-},\nabla\mathbf{u}^{-},\mathbf{u}^{+},\nabla\mathbf{u}^{+},\eta,\vec{n}\right)$, where η is a parameter that depends on the geometry of the cell Q and its neighbor and the local order of the polynomial approximation (2). To ensure adjoint consistency an additional surface flux term $\mathbf{h}\left(\mathbf{u}^{-},\mathbf{u}^{+},\vec{n}\right)$ is introduced via two integrations by parts [15] yielding the final DG formulation

$$\langle \mathbf{u}_t^{Q}, \phi\rangle_Q + \left(\mathbf{g}^{\mathbf{C}} - g^{\text{dGRP}}, \phi\right)_{\partial Q} - \left(h, \nabla\phi\right)_{\partial Q} - \langle \vec{f}^{C} - \vec{f}^{D}, \nabla\phi\rangle_Q = 0. \tag{7}$$

3.2 Nodal Integration

The computation of the volume and surface quadrature operators can be a very expensive task if standard methods such as Gaussian quadrature are used, which is caused by the high number of polynomial evaluations required for computing the fluxes. Based on the nodal DG scheme developed by Hesthaven and Warburton [18], Gassner et al. developed a way of constructing efficient quadrature operators that work on arbitrarily shaped elements, see [16] for further details. This has the advantage that the number of degrees of freedom does not depend on the element shape as it would be for a purely nodal scheme when using elements other than tetrahedra. As points to define the nodal basis, Legendre-Gauss-Lobatto (LGL) points are used on edges and then a method called LGL-type nesting is used to determine the interior points, which leads to a small Lebesgue constant.

The coexistence of modal and nodal elements is quite natural for a DG scheme since the transformation from modal ($\hat{\mathbf{u}}$) to nodal ($\tilde{\mathbf{u}}$) degrees of freedom is nothing else but a polynomial evaluation of the modal polynomials at the nodal interpolation points which can be expressed in the form of a matrix-vector-multiplication:

$$\tilde{\mathbf{u}} = \mathbf{V}\hat{\mathbf{u}}. \tag{8}$$

Here $\mathbf{V}$ is a Vandermonde matrix containing the evaluations of the modal polynomials at the interpolation points. The back transformation can be implemented using the inverse of the Vandermonde matrix:

$$\hat{\mathbf{u}} = \mathbf{V}^{-1}\tilde{\mathbf{u}}. \tag{9}$$

Should the number of nodal interpolation points not be equal to the number of modal degrees of freedom, as it is the case for elements other than tetrahedra, the inverse $\mathbf{V}^{-1}$ is defined using a least squares procedure based on singular value decomposition [16].

The nodal DG method can be conveniently formulated in terms of matrices representing the discrete integrals in (7):

$$\mathbf{M}\tilde{\mathbf{u}}_t + \sum_{i=1}^{\text{nFaces}} \left(\underbrace{\mathbf{M}_i^S \tilde{\mathbf{g}}_i - \mathbf{N}_i \tilde{\mathbf{h}}_i}_{\text{surface integral}} \right) - \sum_{k=1}^{d} \underbrace{\mathbf{S}_k \tilde{\mathbf{f}}_k}_{\text{volume integral}} = 0. \tag{10}$$

Should the equations be nonlinear such as e.g. the compressible Navier-Stokes equations the nonlinearity is present in the evaluation of the fluxes. In Eq. (10), $\tilde{\mathbf{f}}_k,\ k = 1, ..., d$ are the vectors of flux evaluations at all nodal points, while $\tilde{\mathbf{g}}_i$ and $\tilde{\mathbf{h}}_i$ stand for the evaluations of the surface flux approximations at the nodal points of the element face i. The operators in Eq. (10) are designed to act on nodal input vectors and to produce a nodal output. Using Eq. (9) all the operators in Eq. (10) can be modified in order to produce a modal output yielding the mixed modal-nodal DG method:

$$\hat{\mathbf{u}}_t = -\mathbf{V}^{-1}\mathbf{M}^{-1} \left(\sum_{i=1}^{\text{nFaces}} \left(\mathbf{M}_i^S \tilde{\mathbf{g}}_i - \mathbf{N}_i \tilde{\mathbf{h}}_i \right) - \sum_{k=1}^{d} \mathbf{S}_k \tilde{\mathbf{f}}_k \right). \tag{11}$$

4 TIME INTEGRATION METHODS

Equation (11) represents an ordinary differential equation in the cell Q. If we combine the modal coefficient vectors in one vector $\underline{\mathbf{u}} \in \mathbb{R}^m$, we obtain a large system of ODEs

$$\underline{\mathbf{u}}(t) = \underline{\mathbf{f}}(\underline{\mathbf{u}}(t)), \tag{12}$$

where $\underline{\mathbf{f}}$ is a vector valued function corresponding to the right hand side (11) for the whole grid. Generally, a vector with an underbar will denote a vector from $\mathbb{R}^m$. We will now consider a number of classes of time integration schemes for the solution of (12), in particular SDIRK, ESDIRK, Rosenbrock, Space Time Expansion with local time-stepping and explicit RK. From each of these, we will consider a few specific methods. Generally, we will call the time step size Δt and $\underline{\mathbf{u}}^n$ is the numerical approximation to $\underline{\mathbf{u}}(t_n)$.

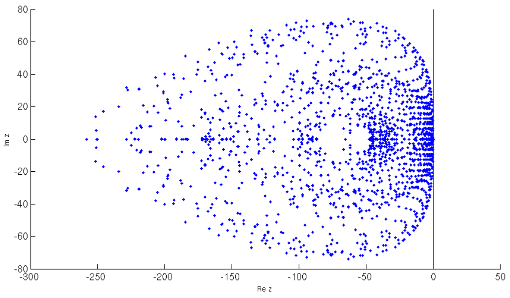

Figure 1: Eigenvalues for a 2D problem with $Re = 100$, 6×6 mesh, 4th order in space

In figure (4), a plot of the eigenvalues of (12) at t_0 for the described DG discretization of a 2D problem is shown. As can be seen, there are eigenvalues directly on the imaginary axis. For this reason we won't consider BDF methods, since these are only $A(\alpha)$-stable for orders greater than two. Furthermore, it shows that the problem is stiff, since the largest real part is -250.

4.1 Explicit Methods

Explicit time integration methods are very popular in the DG community. While the implementation is easy compared to implicit methods, they have the drawback of a bounded stability region. For the purely hyperbolic case, the timestep is limited by the CFL condition

$$\Delta t_Q \leq \sigma^C \frac{1}{2N+1} \frac{\Delta x}{\max\left(\lambda_Q^C\right)}, \tag{13}$$

where σ^C is a user-defined parameter and $\max\left(\lambda_Q^C\right)$ stands for the maximum of the hyperbolic eigenvalues in a cell Q. Different from high order finite volume methods, the degree of the polynomial approximation influences the timestep significantly as the constraint scales with $2N+1$. In the paracolic case, the stability condition turns out to be even more restrictive:

$$\Delta t_Q \leq \sigma^D \frac{1}{N^2} \frac{\Delta x^2}{\max\left(\lambda_Q^D\right)}, \tag{14}$$

where again σ^D is a user-defined parameter and $\max\left(\lambda_Q^D\right)$ stands for the maximum parabolic eigenvalue in the cell Q. In addition to the fact that this stability condition scales with Δx^2, which is a problem for all kinds of spatial discretizations, the scaling imposed by the DG polynomial on the maximum eigenvalue becomes quadratic. These conditions haven to be fulfilled in every cell in order to obtain a stable scheme.

4.1.1 Explicit Runge-Kutta Methods

Explicit Runge-Kutta (ERK) methods are more or less the standard for time integration in conjunction with DG spatial discretizations. They are straightforward to implement and offer good performance in serial and parallel computing environments. In this work we focus on an efficient low storage ERK (LSERK) methods of fourth order with five stages ($s = 5$) of Carpenter and Kennedy [**?**]. The scheme can be implemented efficiently in the following form:

$$\begin{aligned} &\underline{\mathbf{u}} = \underline{\mathbf{u}}^n \\ &\underline{\mathbf{k}} = \underline{\mathbf{0}} \\ &i \in [1, ..., s] : \begin{cases} \underline{\mathbf{k}} = a_i \, \underline{\mathbf{k}} + \Delta t \mathbf{f} \left(\underline{\mathbf{u}} \right) \\ \underline{\mathbf{u}} = \underline{\mathbf{u}} + b_i \, \underline{\mathbf{k}} \end{cases} \\ &\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{u}} \end{aligned} \tag{15}$$

4.1.2 Space Time Expansion with Time Consistent Local Time-Stepping

While all the other time integration schemes considered here are actually ODE solvers and hence, are based on the method of lines, we now consider a different approach. Lörcher et al. developed the so-called Space Time Expansion (STE) method for DG discretizations [25, 12] which is very similar to Harten's variant but has the feature of time consistent local time-stepping. There it was also shown that the approach is in general a predictor-corrector method. Since the method is not well known we will describe it in greater detail here.

The basic idea is to advance a PDE in time by integrating it from t^n to t^{n+1}:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \int_{t^n}^{t^{n+1}} \nabla \cdot \vec{f}(\mathbf{u}) dt. \tag{16}$$

Since the temporal evolution of $\mathbf{u}\left(\vec{x}, t\right)$ is unknown, there is no obvious solution for the integral in Eq. (16). The idea of the STE scheme is to construct a high order predictor $\mathbf{p}\left(\vec{x}, t\right)$ approximating $\mathbf{u}\left(\vec{x}, t\right)$ in the time interval $[t^n, t^{n+1}]$ in each cell and insert it into eq. (16). This way one obtains a corrector equation that yields the actual time update:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \int_{t^n}^{t^{n+1}} \nabla \cdot \mathbf{f}\left(\mathbf{p}\left(\vec{x}, t\right)\right). \tag{17}$$

The integral can be solved numerically using e. g. Gaussian quadrature:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \frac{\Delta t}{2} \sum_{j=1}^{nGP} \omega_j \nabla \cdot \mathbf{f}\left(\mathbf{p}\left(\vec{x}, t_j\right)\right). \tag{18}$$

The construction of the predictor used to be a delicate task which has earned the method a bad reputation and rendered it unattractive to prospective users. In recent years a number of efficient and easy to implement methods have been devised. Here, we use continuous explicit Runge-Kutta methods, as suggested in [12].

While an STE timestep is less expensive than one of a classical ERK method, the stability of the STE method is somewhat limited resulting in approximately the same computational effort for both schemes. The scheme becomes interesting once one considers the main drawback of all explicit methods, namely that the stability conditions (13) and (14) must not be violated in any cell. Since classical methods such as the ones in section (4.1.1) require all cells to be advanced using a uniform timestep, such schemes can become very inefficient, since it is not uncommon that the minimum timestep in the computational domain is several orders of magnitude smaller than the maximum one. Good examples that cause this behavior would be *hp* refinement for the resolution of boundary layers or shocks. When considering this, the idea of advancing each cell individually only abiding to its own stability constraint becomes very attractive. Lörcher et al. have constructed such a temporal method for DG spatial discretizations in [25]. The main idea is to consider the evolution equations (17) for the degrees of freedom of each individual cell separately, instead of all of the degrees of freedom in the computational domain Ω at once. To summarize this method, we first combine Eq. (17) with the DG operator in Eq. (11) to obtain an evolution equation for the degrees of freedom in an individual cell Q

$$\hat{\mathbf{u}}_Q^{n+1} = \hat{\mathbf{u}}_Q^{n} - \int_{t^n}^{t^{n+1}} \mathbf{V}^{-1} \left(\sum_{i=1}^{\text{nFaces}} \left(\mathbf{M}_i^S \tilde{\mathbf{g}}_i - \mathbf{N}_i \tilde{\mathbf{h}}_i \right) - \sum_{dim=1}^{d} \mathbf{S}_{dim} \tilde{\mathbf{f}}_{dim} \right) dt. \tag{19}$$

A straightforward analysis of Eq. (19) turns out that it can be split into an operator for the the volume terms

$$\mathbf{R}_V \left(\tilde{\mathbf{p}}_Q \right) = V^{-1} \sum_{dim=1}^{d} \mathbf{S}_{dim} \tilde{\mathbf{f}}_{dim}, \tag{20}$$

and an operator for the surface terms

$$\mathbf{R}_S \left(\tilde{\mathbf{p}}_Q, \tilde{\mathbf{p}}_Q^{+} \right) = V^{-1} \left(\sum_{i=1}^{\text{nFaces}} \left(\mathbf{M}_i^S \tilde{\mathbf{g}}_i - \mathbf{N}_i \tilde{\mathbf{h}}_i \right) \right), \tag{21}$$

allowing us to rewrite Eq. (19) into

$$\hat{\mathbf{u}}_Q^{n+1} = \hat{\mathbf{u}}_Q^{n} + \int_{t^n}^{t^{n+1}} \mathbf{R}_V \left(\tilde{\mathbf{p}}_Q \right) dt - \int_{t^n}^{t^{n+1}} \mathbf{R}_S \left(\tilde{\mathbf{p}}_Q, \tilde{\mathbf{p}}_Q^{+} \right) dt. \tag{22}$$

Assuming that there is a predictor $\tilde{\mathbf{p}}_Q(t) \approx \tilde{\mathbf{u}}_Q(t)$ available in each cell, the time integral of the volume operator can be solved without further considerations since it only depends on $\tilde{\mathbf{p}}_Q(t)$, provided that $\Delta t_Q = t_Q^{n+1} - t_Q^n$ does not violate the stability conditions (13) and (14). The surface part is the difficult one, since it involves not only the local predictor $\tilde{\mathbf{p}}_Q(t)$ but also predictor values from neighboring cells $\tilde{\mathbf{p}}_Q^{+}(t)$ in order to compute the time integral of the surface terms.

The sequence in Fig. (2) illustrates a typical situation where local time-stepping is applied. The three cells Q_1, Q_2 and Q_3 all have different maximum stable timesteps and thus can reach different maximum target time levels t_i^{n+1} as depicted by the dashed bars in the first figure. As already stated, to update the degrees of freedom of a cell Q_i using Eq. (22) we need the predictor values in the cell itself as well as the ones of all neighbor cells up to the target time level t_k^{n+1}. It is clear that in this example, the condition is only fulfilled for cell Q_2. From this observation we can derive a general condition that has to be fulfilled by a cell in order to be advanced in time, the so-called "evolve condition"

$$t_i^{n+1} \leq \min \left\{ t_j^{n+1} \right\} \forall j : Q_j \cap Q_i \neq 0. \tag{23}$$

In order to make the method time consistent and fully conservative, the time integral of the surface operator has to be applied to both cells sharing a common side as depicted in the second figure of the sequence. After a full update of cell Q_2, all data is available for computing the predictor $\tilde{\mathbf{p}}_{Q_2}(t)$ and the next maximum stable timestep at time level t_2^{n+1}. The next cell that fulfills the evolve condition in our example would be Q_1. The interesting part in this update

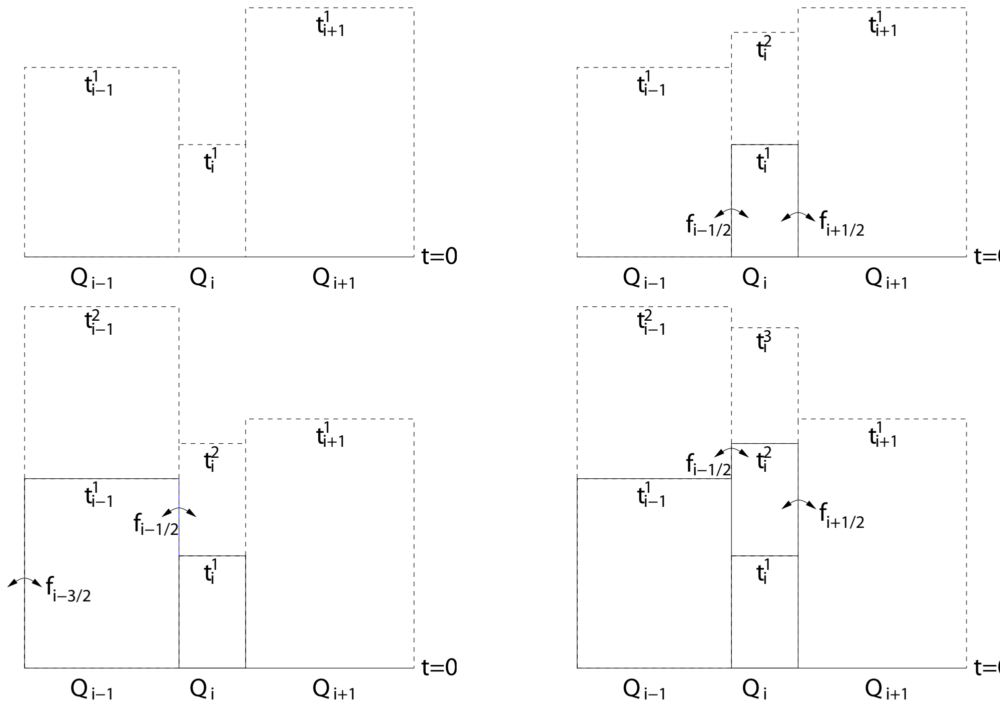

Figure 2: Sequence of steps 1-4 of a computation with 3 different elements and local time-stepping

process is the time integral at the interface with cell Q_2 since a part of this integral has already been computed during the update process of Q_2. Since it is mathematically equivalent to split the integral of an interval into a sum of an arbitrary number K of subintervals

$$\int_{t^n}^{t^{n+1}} ...dt = \int_{t^n}^{t^1} ...dt + \int_{t^1}^{t^2} ...dt + ... + \int_{t^{K-2}}^{t^{K-2}} ...dt + \int_{t^{K-1}}^{t^{n+1}} ...dt,$$

we split the interval $\left[t_1^n, t_1^{n+1}\right]$ into the intervals $\left[t_1^n, t_2^{n+1}\right]$ and $\left[t_2^{n+1}, t_1^{n+1}\right]$ which yields

$$\int_{t_1^n}^{t_1^{n+1}} \mathbf{R}_S\left(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2\right) dt = \int_{t_1^n}^{t_2^{n+1}} \mathbf{R}_S\left(\tilde{\mathbf{p}}_1^n, \tilde{\mathbf{p}}_2^n\right) dt + \int_{t_2^{n+1}}^{t_1^{n+1}} \mathbf{R}_S\left(\tilde{\mathbf{p}}_1^n, \tilde{\mathbf{p}}_2^{n+1}\right) dt.$$

This way we obtain a consistent local time stepping method which is not possible using e. g. ERK methods. This method can be implemented with very little additional logic once the STE method is available. One only has to make sure that the evolve condition (Eq. (23)) is not violated if a cell is to be advanced in time and that the time integrals over the surface operators are split in a consistent manner. It is interesting to note that this process only works due to the fact that the STE scheme naturally allows the imposition of time dependent boundary conditions without loss of accuracy, which is a problem for Runge Kutta methods, see Carpenter et al. [6].

4.2 Implicit Methods

Implicit methods can be constructed to have unbounded stability regions, but require the solution of linear or nonlinear equation systems. This makes them more expensive per step and more difficult to implement. Nonetheless the severe timestep limitation of explicit schemes makes implicit timestepping schemes attractive.

4.2.1 DIRK methods

A general implicit Runge-Kutta methods requires solving a nonlinear equation system with $s \cdot m$ unknowns. We will therefore restrict ourselves to so called diagonally implicit Runge-Kutta methods or short DIRK methods. Given coefficients a_{ij} and b_i, such a method with s stages can be written as

$$\underline{\mathbf{U}}_i = \underline{\mathbf{u}}^n + \Delta t \sum_{j=1}^{i} a_{ij} \underline{\mathbf{f}}(\underline{\mathbf{U}}_j), \quad i = 1, ..., s \tag{24}$$

$$\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{u}}^n + \Delta t \sum_{i=1}^{s} b_i \underline{\mathbf{f}}(\underline{\mathbf{U}}_i). \tag{25}$$

Thus, all entries of the Butcher array in the strictly upper triangular part are zero. If additionally all values on the diagonal of $\mathbf{A}$ are identical, the scheme is called singly diagonally implicit, short SDIRK. Furthermore, we require, that the coefficients $\mathbf{b}$ and the last line of the matrix $\mathbf{A}$ coincide. Thus, the resulting scheme is, if A-stable, also L-stable. This class of schemes is called stiffly accurate and is popular for the solution of differential algebraic equations (DAEs). Note that the application of (25) is therefore not necessary, since then $\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{U}}_s$.

The point about DIRK schemes is, that the computation of the stage vectors is decoupled and instead of solving one nonlinear system with sm unknowns, the s nonlinear systems (24) with m unknowns have to be solved. This corresponds to the sequential application of several implicit Euler steps in two possible ways. With the starting vectors

$$\underline{\mathbf{s}}_i = \underline{\mathbf{u}}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \underline{\mathbf{f}}(\underline{\mathbf{U}}_j), \tag{26}$$

we can solve for the stage values

$$\underline{\mathbf{U}}_i = \underline{\mathbf{s}}_i + \Delta t a_{ii} \underline{\mathbf{f}}(\underline{\mathbf{U}}_i). \tag{27}$$

The equation (27) corresponds to a step of the implicit Euler method with starting vector $\underline{\mathbf{s}}_i$ and time step $a_{ii}\Delta t$. Note that a_{ii} is typically smaller than one and thus the equation systems are easier to solve than a system arising from an implicit Euler discretization with the same Δt. Second, if a time adaptive strategy is used, the higher order leads to larger time steps compared to the implicit Euler method and thus for a certain tolerance, we have less than s times the number of nonlinear systems to solve.

Additionally, it is usually possible to give a second set of coefficients $\hat{b}$, which define for the otherwise identical Butcher tableau a method of lower order. This can be used for the estimation of the time integration error, as will be explained later.

A method of third order (SDIRK3) with an embedding of second order was developed by Cash [7]. This method is A- and L-stable.

Furthermore, there is the class of ESDIRK schemes, where the first step of the Runge-Kutta schemes is explicit. This allows to have a stage order of two, but also means that the methods cannot be algebraically stable. The use of these schemes in the context of compressible Navier-Stokes equations was analyzed by Bijl et al. in [4] where they were demonstrated to be more efficient than BDF methods for engineering accuracies. They suggested the six stage method ESDIRK4 of fourth order with an embedded method of third order and the four stage method

ESDIRK3 of third order with an embedded method of second order, both designed in [23]. Again, these schemes are A- and L-stable.

The use of a first explicit stage involving $\underline{\mathbf{f}}(\underline{\mathbf{u}}^n)$ only, allows to reuse the last stage derivative from the last time step in the first stage, since $\underline{\mathbf{f}}(\underline{\mathbf{u}}^{n+1})$ from the last time step is just the same thing.

4.2.2 Rosenbrock methods

To circumvent the solution of nonlinear equation systems, Rosenbrock methods can be used [17]. The idea is to linearize a DIRK scheme, thus sacrificing some stability properties, as well as accuracy, but reducing the computational effort per time step. Therefore, this class of schemes is also referred to as linearly implicit or semi-implicit. More precise, an s-stage Rosenbrock method with coefficients $\tilde{a}_{ij}$, γ_{ij}, b_i, $\gamma_i = \sum_{j=1}^{i} \gamma_{ij}$ and $a_i = \sum_{j=1}^{i-1} \tilde{a}_{ij}$ is given by

$$\begin{aligned}
(\mathbf{I} - \gamma_{ii}\Delta t\mathbf{W})\underline{\mathbf{k}}_i &= \underline{\mathbf{f}}(t_n + a_i\Delta t, \tilde{\underline{\mathbf{U}}}_i) + \Delta t\mathbf{W}\sum_{j=1}^{i-1}\gamma_{ij}\underline{\mathbf{k}}_j + \Delta t\gamma_{ii}\partial_t\underline{\mathbf{f}}(t_n, \underline{\mathbf{u}}_n), \quad i = 1, ..., s \\
\tilde{\underline{\mathbf{U}}}_i &= \underline{\mathbf{u}}^n + \Delta t\sum_{j=1}^{i-1}\tilde{a}_{ij}\underline{\mathbf{k}}_j, \quad i = 1, ..., s \qquad (28)\\
\underline{\mathbf{u}}^{n+1} &= \underline{\mathbf{u}}^n + \Delta t\sum_{i}^{s} b_i\underline{\mathbf{k}}_i.
\end{aligned}$$

Note that in the autonomous case, the last term on the right hand side is zero.

Thus, per time step, s linear equation systems with the same system matrix and different right hand sides have to be solved. If, instead of the exact Jacobian $\frac{\partial \underline{\mathbf{f}}}{\partial \underline{\mathbf{u}}}(t_n)$, an approximation W is used, we obtain so called Rosenbrock-Wanner methods or short ROW methods. If the linear system is solved using a Krylov subspace method, the scheme is called a Krylov-ROW method.

The decreased accuracy of Rosenbrock methods compared to SDIRK methods, will later result in the time step selector chosing smaller time steps for Rosenbrock methods. Regarding stability, it is possible to design A-stable and even L-stable schemes. In [21], Rosenbrock methods are compared to SDIRK methods in the context of the incompressible Navier-Stokes equations and found to be competitive, if not superior.

The efficient implementation of Rosenbrock methods is done using a set of auxiliary variables

$$\underline{\mathbf{U}}_i = \Delta t\sum_{j=1}^{i}\gamma_{ij}\underline{\mathbf{k}}_j$$

and circumvents the matrix-vector multiplication in the previous formulation. Using the identity

$$\underline{\mathbf{k}}_i = \frac{1}{\Delta t}\left(\frac{1}{\gamma_{ii}}\underline{\mathbf{U}}_i - \sum_{j=1}^{i-1} c_{ij}\underline{\mathbf{U}}_j\right)$$

with coefficients c_{ij} explained below, we obtain

$$\begin{aligned}(\mathbf{I}-\gamma_{ii}\Delta t\mathbf{W})\underline{\mathbf{U}}_i &= \Delta t\gamma_{ii}\underline{\mathbf{f}}(t_n+a_i\Delta t,\hat{\underline{\mathbf{U}}}_i)+\gamma_{ii}\sum_{j=1}^{i-1}c_{ij}\underline{\mathbf{U}}_j+\Delta t^2\gamma_{ii}\gamma_i\partial_t\underline{\mathbf{f}}(t_n,\underline{\mathbf{u}}_n),\\ \hat{\underline{\mathbf{U}}}_i &= \underline{\mathbf{u}}^n+\sum_{j=1}^{i-1}a_{ij}\underline{\mathbf{U}}_j,\\ \underline{\mathbf{u}}^{n+1} &= \underline{\mathbf{u}}^n+\sum_{i}^{s}m_i\underline{\mathbf{U}}_i,\end{aligned}\tag{29}$$

where again in the autonomous case, the last term on the right hand side is zero. The relation between the two sets of coefficients is the following:

$$\mathbf{C}=\text{diag}(\gamma_{11}^{-1},...,\gamma_{ss}^{-1})-\mathbf{\Gamma}^{-1},\qquad \mathbf{A}=\tilde{\mathbf{A}}\mathbf{\Gamma}^{-1},\qquad \mathbf{m}^T=\mathbf{b}^T\mathbf{\Gamma}^{-1}.$$

Here, we consider the method ROS34PW2 from [31].

4.3 Adaptive time step size selection

For unsteady flows, we need to make sure that the time integration error can be controlled. To do this, we will estimate the time integration error and select the time step size accordingly. For DIRK and Rosenbrock methods, this is done using the embedded schemes of a lower order $\hat{p}$. Comparing the local truncation error of both schemes, we obtain the following estimate for the local error of the lower order scheme:

$$\mathbf{l}\approx\Delta t_n\sum_{j=1}^{s}(b_i-\hat{b}_i)\underline{\mathbf{k}}_i,\tag{30}$$

respectively, for the efficient formulation (29) of the Rosenbrock method

$$\mathbf{l}\approx\sum_{j=1}^{s}(m_i-\hat{m}_i)\underline{\mathbf{U}}_i.$$

The error estimate is then used to determine the new step size. To do this, we decide beforehand on a target error tolerance, which we implement using a common fixed resolution test [34]. This means that we define the error tolerance per component via

$$d_i=RTOL|u_i^n|+ATOL.\tag{31}$$

Then we compare this to the local error estimate via requiring

$$\|\mathbf{l}./\mathbf{d}\|\leq 1,$$

where . denotes a pointwise division operator. For the norm, we are going to use the 2-norm, since GMRES is based on that. A more substantial and less heuristic choice of norm would be desirable for the future.

The next question is, how the time step has to be chosen, such that the error can be controlled. The classical method is the following, also called EPS (error per step) control [17]:

$$\Delta t_{new}=\Delta t_n\cdot\|\mathbf{l}./\underline{\mathbf{d}}\|^{1/(\hat{p}+1)}.\tag{32}$$

This is combined with two safety factors to avoid volatile increases or decreases in time step size:

$$\begin{aligned} \text{if } \|\mathbf{l}./\mathbf{d}\| \geq 1, \qquad & \Delta t_{n+1} = \Delta t_n \max(f_{min}, f_{safety}\|\mathbf{l}./\mathbf{d}\|^{1/\hat{p}+1}) \\ \text{else} \qquad & \Delta t_{n+1} = \Delta t_n \min(f_{max}, f_{safety}\|\mathbf{l}./\mathbf{d}\|^{1/\hat{p}+1}). \end{aligned}$$

Here, we choose $f_{min} = 0.3$, $f_{max} = 2.0$ and $f_{safety} = 0.9$. Finally, we use tolerance scaling and calibration [33]. The first is useful, since, although for the above controller, convergence of the method for $TOL \to 0$ can be proven, the relation between global error and tolerance is typically of the form

$$\|e\| = \tau \cdot TOL^{\alpha}, \tag{33}$$

where α is smaller than one, but does not depend strongly on the problem solved. Therefore, an internal rescaling $TOL' = \mathcal{O}(TOL^{1/\alpha})$ is done, so that the user obtains a behavior where a decrease in the rescaled tolerance TOL' leads to a corresponding decrease in error. Furthermore, it is useful to calibrate the solver, such that $TOL = TOL' = TOL_0$ for a specific tolerance TOL_0:

$$TOL' = TOL_0^{(\alpha-1/\alpha)} TOL^{1/\alpha}. \tag{34}$$

5 SOLVING EQUATION SYSTEMS

5.1 Inexact Newton method

To solve the appearing nonlinear systems, we use an inexact Newton's method. As a termination criterion we use a residual based one with a relative tolerance, resulting in

$$\|\underline{\mathbf{F}}(\underline{\mathbf{u}}_k)\| \leq \epsilon = \epsilon_r \|\underline{\mathbf{F}}(\underline{\mathbf{u}}_0)\|.$$

If the iteration does not converge after a maximal number of iterations, the time step will be repeated with half the time step size. In particular, we will use the inexact Newton's method from [10], where the linear system in the k-th Newton step is solved only up to a relative tolerance, given by a forcing term η_k. This can be written as:

$$\begin{aligned} \left\| \left.\frac{\partial \underline{\mathbf{F}}(\underline{\mathbf{u}})}{\partial \underline{\mathbf{u}}}\right|_{\underline{\mathbf{u}}^{(k)}} \Delta \underline{\mathbf{u}} + \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)}) \right\| &\leq \eta_k \|\underline{\mathbf{F}}(\underline{\mathbf{u}}_k)\| \\ \underline{\mathbf{u}}^{(k+1)} &= \underline{\mathbf{u}}^{(k)} + \Delta u, \quad k = 0, 1, ... \end{aligned} \tag{35}$$

In [11], the choice of the sequence of forcing terms is discussed and it is proved that for a properly chosen sequence of forcing terms, the convergence can be quadratic. They also construct a sequence of this type, which is employed here.

5.2 Jacobian-free GMRES

The linear systems are solved using a Jacobian-free Krylov subspace method, in particular the GMRES method of Saad and Schultz [32], which was explained by McHugh and Knoll [27] to perform better than others in the Jacobian-free context. In their basic version, Krylov subspace schemes need the Jacobian and in addition a preconditioning matrix to improve convergence speed, which is storage wise a huge problem. Furthermore, the use of approximate Jacobians to save storage and CPU time leads to a decrease of the convergence speed of the

Newton method. To get rid of the Jacobian, the idea is that in Krylov subspace methods, the Jacobian appears only in the form of matrix vector products $\underline{\mathbf{Av}}_i$ which can be approximated by a difference quotient

$$\underline{\mathbf{Av}}_i \approx \frac{\underline{\mathbf{F}}\left(\bar{\underline{\mathbf{u}}} + \epsilon\underline{\mathbf{v}}_i\right) - \underline{\mathbf{F}}(\bar{\underline{\mathbf{u}}})}{\epsilon} = \underline{\mathbf{v}}_i - a_{\nu+1}\Delta t\frac{\underline{\mathbf{f}}\left(\bar{\underline{\mathbf{u}}} + \epsilon\underline{\mathbf{v}}_i\right) - \underline{\mathbf{f}}(\bar{\underline{\mathbf{u}}})}{\epsilon}. \tag{36}$$

The parameter ϵ is a scalar, where smaller values lead to a better approximation but may lead to truncation errors. A simple choice for the parameter, that avoids cancellation but still is moderately small is given by Quin, Ludlow and Shaw [30] as

$$\epsilon = \frac{\sqrt{eps}}{\|\Delta\underline{\mathbf{u}}\|_2},$$

where eps is the machine accuracy. Second order convergence is obtained up to ϵ-accuracy if proper forcing terms are employed, since it is possible to view the errors coming from the finite difference approximation as arising from inexact solves.

5.3 Preconditioner

A much more sophisticated method that is very good preconditioner for compressible flow problems is the symmetric block Gauss-Seidel-method (SGS), which corresponds to solving the equation system

$$(\mathbf{D} + \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{x}^P. \tag{37}$$

This needs a huge amount of storage, therefore, we employ ROBO-SGS, which makes use of the fact that the modal DG scheme has a hierarchical basis which also gives the element Jacobians a hierarchical structure. This gives us the opportunity to reduce the inter-element

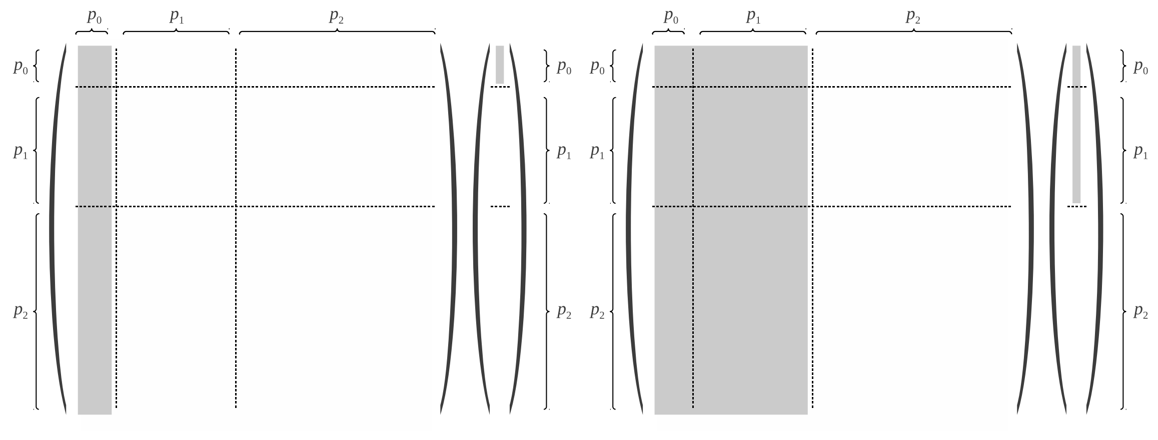

Figure 3: Reduced versions of the off-diagonal blocks of the Jacobian, $k = 0$ and $k = 1$ variants

coupling by neglecting all derivatives of higher-order degrees of freedom of the neighboring cells in the Jacobian. For example, in the case $k = 0$ we take only the DOFs of the neighbors into account, which correspond to the integral mean values of the conserved quantities, see Fig. (3). However, we keep not only the derivatives of the remaining degrees of freedom with respect to themselves, but to all degrees of freedom, resulting in a rectangular structure in the off diagonal blocks of the preconditioner. For $k = p$, we keep everything, thus recovering the original block SGS preconditioner. If we formally set $k = -1$, we neglect all off diagonal block entries, thus recovering block Jacobi. Here, we employ ROBO-SGS-1, which gives a very good compromise between efficiency and storage.

6 NUMERICAL RESULTS

6.1 Test cases

The first test case is about the convection of a 2D vortex in a periodic domain, which is designed to be isentropic for inviscid flows [9]. However, we will consider this in the setting of laminar flows. The initial conditions are taken as freestream values $(\rho_\infty, v_{1_\infty}, v_{2_\infty}, T_\infty) = (1, v_{1_\infty}, 0, 1)$, but are perturbed at t_0 by a vortex $(\delta v_1, \delta v_2, \delta T)$ centered at $(\tilde{x}_1, \tilde{x}_2)$, given by the formulas

$$\delta v_1 = -\frac{\alpha}{2\pi}(x_2 - \tilde{x}_2)e^{\phi(1-r^2)},$$
$$\delta v_2 = \frac{\alpha}{2\pi}(x_1 - \tilde{x}_1)e^{\phi(1-r^2)},$$
$$\delta T = -\frac{\alpha^2(\gamma-1)}{16\phi\gamma\pi^2}(x_1 - \tilde{x}_1)e^{2\phi(1-r^2)},$$

where ϕ and α are parameters and r is the euclidian distance of a point (x_1, x_2) to the vortex center, which is set at $(\tilde{x}_1, \tilde{x}_2) = (0, 0)$ in the domain $[-7, 7] \times [-3.5, 3.5]$. The parameter α can be used to tweak the speed of the flow in the vortex, whereas ϕ determines the size and is chosen as $\phi = 1$. Here, we will employ $\alpha = 4.0$, $v_{1_\infty} = 0.5$ and $t_{end} = 4.0$. The Reynolds number is set to $Re = 100$ and the grid is chosen cartesian with 100×50 cells. For the reference solution, the LSERK4 method is used with $\Delta t = 0.0001$. The spatial order is four.

As a second test case, we consider a density wave on the 2D square $[0, 2] \times [0, 2]$ for $t \in [0, 1]$, governed by the Euler equations. Here, the exact solution is given by

$$\rho(\mathbf{x}, t) = 2(1 + 0.5\sin(\pi|\mathbf{x} - \mathbf{v}t|)),$$

which is used as a reference and defines the initial conditions at $t = 0$. Otherwise, all the primitive variables are equal to two. The grid is chosen cartesian with 10×20 cells and the spatial order is five.

The third test case is vortex shedding behind a 3D sphere at Mach 0.3 and Reynolds number of 1,000. The unstructured grid consists of 21,128 hexahedral cells. To obtain initial conditions and start the initial vortex shedding, we begin with free stream data and linear polynomials. Furthermore, to cause an initial disturbance, the boundary of the sphere is chosen to be noncurved at first. In this way, we compute 120 s of time using an explicit time integration scheme. Then we switch to polynomials of degree three and compute another 10 s of time using the explicit method. From this, we start the actual computations with a polynomial degree of four and a curved representation of the boundary. In total, we have 739,480 unknowns for this test case.

For the first time step, we chose $\Delta t = 0.0065$, from then on the steps are determined by the time adaptive algorithm. This time step size was chosen such that the resulting embedded error estimate is between 0.9 and 1, leading to an accepted time step. Then, we perform the computations on a time interval of 30 seconds. The initial solution and the result at the end when using ESDIRK4 time integration with a tolerance of 10^{-3} are shown in figure 4, where we see isosurfaces of λ_2=-10^{-4}, a common vortex identifier [20]. Note that there is no visual difference between the results for ESDIRK4 and LSERK4.

6.2 Order of convergence

First of all, we'll look at the order obtained by the different methods by comparing the results obtained for fixed time step calculations of $\Delta t = 1.0, 0.5, ..., 0.03125$ with those of a reference

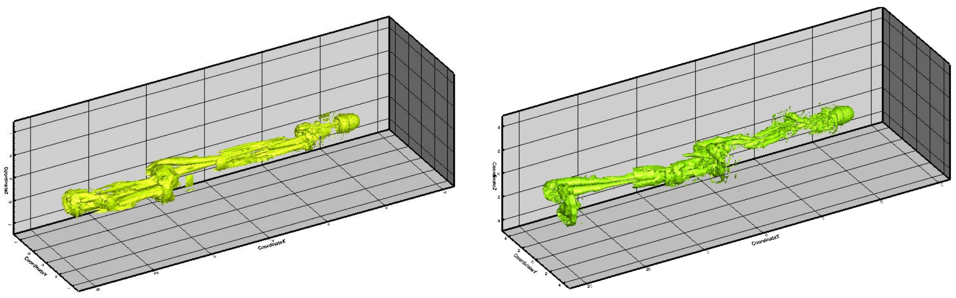

Figure 4: Isosurfaces of λ_2=-10^{-4} for initial (left) and final solution of sphere problem (right).

solution for the first variant of the vortex convection problem and the density wave problem. The nonlinear systems are solved up to a relative tolerance of $1E - 7$. As can be seen in figure 6.2, the theoretical order is obtained by all methods considered. ESDIRK4 reaches the error of the reference solution at some point. It is striking that all third order methods produce almost the same error. Since the Rosenbrock method is obtained from a linearization of SDIRK 3, this means that these problems are only weakly nonlinear.

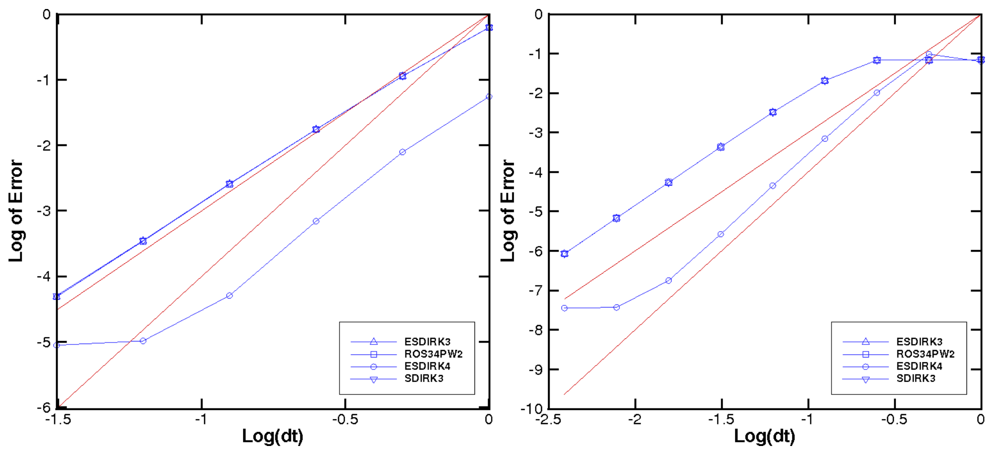

Figure 5: Order of the methods; Vortex convection problem (left), Density wave problem (right)

6.3 Tolerance scaling

To determine the correct tolerance scaling, we consider the vortex convection problem and the density wave problem. These problems are then solved by the different implicit time adaptive schemes for different tolerances. Comparing with the reference solutions, the obtained relation between TOL and the error in the discrete L_2 norm for different methods can be seen in table 6.3 and 6.3. Note that this is not the norm that the error estimator works in, since that is the 2-norm. As can be seen, ESDIRK 3, ESDIRK 4 and ROS34PW2 produce similar errors, whereas SDIRK 3 produces errors that are ten times smaller for the same tolerance. However, SDIRK 3 has a better relation between TOL and error for the vortex convection problem, whereas the other method have a better relation for the density wave problem. From these values we deduce the values for α in (33) to be those in (6.3).

Finally, the new tolerance TOL' is calibrated via (34) at $TOL = 1.E - 4$, leading to a factor of 10 for ROS34PW2 and rougly 2.78 for the other methods.

TOL	SDIRK 3	ESDIRK 3	ESDIRK 4	ROS34PW2
10^{-2}	3.989257e-2	4.854288e-1	2.964846e-1	1.618823e-1
10^{-3}	5.093144e-3	6.277344e-2	4.103350e-2	2.847235e-2
10^{-4}	6.070536e-4	7.383178e-3	8.313921e-3	5.922446e-3
10^{-5}	7.023503e-5	8.109867e-4	1.155345e-3	1.180773e-3
10^{-6}	1.327953e-5	8.366576e-5	1.235267e-4	2.191002e-4

Table 1: Relation between TOL and error in discrete L_2 norm for different methods, vortex convection problem

TOL	SDIRK 3	ESDIRK 3	ESDIRK 4	ROS34PW2
10^{-2}	1.411442e-3	1.662924e-2	1.315162e-2	1.485843e-2
10^{-3}	1.801086e-4	1.704417e-3	1.160576e-3	3.064769e-3
10^{-4}	1.878412e-5	1.774249e-4	1.272843e-4	5.401451e-4
10^{-5}	1.301709e-6	1.762944e-5	1.439278e-5	9.672057e-5
10^{-6}	1.427034e-6	1.758030e-6	1.459053e-6	1.735513e-5

Table 2: Relation between TOL and error in discrete L_2 norm for different methods, density wave problem

6.4 Time adaptive calculations

Now, we will consider time adaptive calculations and compare different time stepping schemes for the sphere problem, where we used a tolerance of 10^{-3}. The results are shown in table 4. As can be seen, the explicit scheme LSERK4 is by far the slowest with about a factor of five between ESDIRK4 and LSERK4. Thus, even when considering that a fourth order explicit scheme is probably overkill for this problem, the message is that explicit Runge-Kutta methods are significantly slower than implicit methods for this test case. Furthermore, the local-time stepping RKCK scheme is about 20% slower than ESDIRK4. Note that for this test case, the time step of the explicit scheme is constrained by the CFL condition and not the DFL condition. Therefore, it can be expected that for higher Reynolds number and finer discretizations at the boundary, the difference in efficiency is even more significant.

When comparing the different implicit schemes, we can see that SDIRK3 is significantly slower than the ESDIRK methods, wheras ESDIRK3 is competitive with ESDIRK4. The worst scheme is the Rosenbrock method, which chooses time steps that are significantly smaller than those of the DIRK methods, which is in line with experiences from the finite volume case.

7 CONCLUSIONS

We considered different time integration schemes in the context of a modal DG scheme and employed tolerance scaling to make a more valid comparison. Without much additional memory, all implicit schemes outperform a standard explicit scheme for a wallbounded flow test case. However, only ESDIRK3 and ESDIRK4 manage to beat a local time stepping explicit scheme.

REFERENCES

[1] D. Arnold. *An Interior Penalty Finite Element Method with Discontinuous Elements*. PhD thesis, The University of Chicago, 1979.

	SDIRK 3	ESDIRK 3	ESDIRK 4	ROS34PW2
α	0.9	0.9	0.9	0.8

Table 3: Values for α chosen for different time integration methods

Scheme	Iter.	CPU in s	Memory in GB
LSERK4	-	346,745	16.3
RKCK	-	80,889	22.9
SDIRK3	22,223	110,844	19.0
ESDIRK3	14,724	73,798	19.0
ESDIRK4	14,639	66,051	19.0
ROS34PW2	60,449	239,869	19.0

Table 4: Efficiency of time integration schemes for vortex shedding behind a sphere.

[2] F. Bassi and S. Rebay. A high-order discontinuous Galerkin finite element method solution of the 2D Euler equations. *J. Comput. Phys.*, 138:251–285, 1997.

[3] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. A high-order accurate discontinuous finite element method fir inviscid an viscous turbomachinery flows. In R. Decuypere and G. Dibelius, editors, *Proceedings of 2nd European Conference on Turbomachinery, Fluid and Thermodynamics*, pages 99–108, Technologisch Instituut, Antwerpen, Belgium, 1997.

[4] H. Bijl, M. H. Carpenter, V. N. Vatsa, and Kennedy. C. A. Implicit time integration schemes for the unsteady compressible navier-stokes equations: Laminar flow. *J. Comp. Phys.*, 179:313–329, 2002.

[5] P. Birken, G. Gassner, M. Haas, and C.-D. Munz. A new class of preconditioners for discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations: ROBO-SGS. *J. Comp. Phys.*, submitted.

[6] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W.-S. Don. The Theoretical Accuracy of Runge-Kutta Time Discretizations for the Initial Boundary Value Problem: A Study of the Boundary Error . *SIAM J. Sci. Comp.*, 16(6):1241–1252, 1995.

[7] J. R. Cash. Diagonally implicit Runge-Kutta formulae with error estimates. *J. of the Inst. Math. and its Appl.*, 24:293–301, 1979.

[8] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection diffusion systems. *SIAM J. Numer. Anal.*, 35:2440–2463, 1998.

[9] F. Davoudzadeh, H. Mcdonald, and B. E. Thompson. Accuracy evaluation of unsteady CFD numerical schemes by vortex preservation. *Computers & Fluids*, 24:883–895, 1995.

[10] R. Dembo, R. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[11] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact newton method. *SIAM J. Sci. Comp.*, 17:16–32, 1996.

[12] G. Gassner, M. Dumbser, F. Hindenlang, and C.-D. Munz. Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors. *J. Comp. Phys.*, 230(11):4232–4247, May 2011.

[13] G. Gassner, F. Lörcher, and C.-D. Munz. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *J. Comput. Phys.*, 224:1049–1063, 2007.

[14] G. Gassner, F. Lörcher, and C.-D. Munz. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *J. Comput. Phys.*, 224(2):1049–1063, 2007.

[15] G. Gassner, F. Lörcher, and C.-D. Munz. A Discontinuous Galerkin Scheme based on a Space-Time Expansion II. Viscous Flow Equations in Multi Dimensions. *J. Sci. Comput.*, 34:260–286, 2008.

[16] G. J. Gassner, F. Lörcher, C.-D. Munz, and J. S. Hesthaven. Polymorphic nodal elements and their application in discontinuous galerkin methods. *Journal of Computational Physics*, 228(5):1573 – 1590, 2009.

[17] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer, Berlin, Series in Computational Mathematics 14, 3. edition, 2004.

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

[19] A. Jameson. Aerodynamics. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, volume 3: Fluids, chapter 11, pages 325–406. John Wiley & Sons, 2004.

[20] J. Jeong and F. Hussain. On the identification of a vortex. *J. Fluid Mech.*, 285:69–94, 1995.

[21] V. John and Rang. J. Adaptive time step control for the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engrg.*, 199:514–524, 2010.

[22] A. Kanevsky, M. H. Carpenter, D. Gottlieb, and J. S. Hesthaven. Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. *J. Comp. Phys.*, 225:1753–1781, 2007.

[23] C. A. Kennedy and M. H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl. Num. Math.*, 44:139–181, 2003.

[24] C. M. Klaij, J. J. W. van der Vegt, and H. van der Ven. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations. *J. Comp. Phys.*, 217(2):589–611, 2006.

[25] F. Lörcher, G. Gassner, and C.-D. Munz. A Discontinuous Galerkin Scheme Based on a Space-Time Expansion. I. Inviscid Compressible Flow in One Space Dimension. *Journal of Scientific Computing*, 32:175–199, 2007. 10.1007/s10915-007-9128-x.

[26] F. Lörcher, G. Gassner, and C.-D. Munz. An explicit discontinuous Galerkin scheme with local time-stepping for general unsteady diffusion equations. *J. Comput. Phys.*, 227(11):5649–5670, 2008.

[27] P. R. McHugh and D. A. Knoll. Comparison of standard and matrix-free implementations of several Newton-Krylov solvers. *AIAA J.*, 32(12):2394–2400, 1994.

[28] J. Peraire and P.-O. Persson. The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems. *SIAM J. Sci. Comput.*, 30(4):1806–1824, 2008.

[29] P.-O. Persson and J. Peraire. Newton-GMRES Preconditioning for Discontinuous Galerkin discretizations of the Navier-Stokes equations. *SIAM J. Sci. Comp.*, 30:2709–2733, 2008.

[30] N. Qin, D. K. Ludlow, and S. T. Shaw. A matrix–free preconditioned Newton/GMRES method for unsteady Navier-Stokes solutions. *Int. J. Num. Meth. Fluids*, 33:223–248, 2000.

[31] J. Rang and L. Angermann. New Rosenbrock W-Methods of Order 3 for Partial Differential Algebraic Equations of Index 1. *BIT*, 45:761–787, 2005.

[32] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[33] G. Söderlind and Wang. L. Adaptive time-stepping and computational stability. *J. Comp. Appl. Math.*, 185:225–243, 2006.

[34] G. Söderlind and Wang. L. Evaluating numerical ODE/DAE methods, algorithms and software. *J. Comp. Appl. Math.*, 185:244–260, 2006.

[35] A. St-Cyr and D. Neckels. A Fully Implicit Jacobian-Free High-Order Discontinuous Galerkin Mesoscale Flow Solver. In *Proceedings of the 9th International Conference on Computational Science*, ICCS 2009, pages 243–252, Berlin, Heidelberg, 2009. Springer-Verlag.

[36] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 1999.

[37] L. Wang and D. J. Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. *J. Comp. Phys.*, 225:1994–2015, 2007.