The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

JACOBIAN-FREE MULTIGRID PRECONDITIONER FOR DG-SEM FOR ATMOSPHERIC FLOW

Philipp Birken[1], Andreas Dedner[2], Johannes Kasimir[1], Robert Klöfkorn[1]

[1] Center for Mathematical Sciences, Lund University, Box 117, Lund, 22100, Sweden
[2]Mathematics Institute, University of Warwick, Coventry, CV4 7AL, UK

Key words: DGSEM, FV, implicit, multigrid, preconditioner, matrix-free, atmospheric, DUNE, DUNE-FEM

Summary. High fidelity fluid simulations have important applications in science and engineering, examples include numerical weather prediction and simulation aided design. Discontinuous Galerkin (DG) methods are promising high order discretizations for simulating unsteady compressible fluid flow in three dimensions. Systems arising from such discretizations are often stiff and require implicit time integration. This motivates the study of fast, parallel, low-memory solvers for the resulting algebraic equation systems.

For (low order) finite volume (FV) discretizations, multigrid (MG) methods have been successfully applied to steady and unsteady fluid flows. But for high order DG methods applied to flow problems, such solvers are currently lacking.

The lack of efficient solvers suitable for contemporary computer architectures inhibits wider adoption of DG methods. This motivates our research to construct a Jacobian-free preconditioner for high order DG discretizations. The preconditioner is based on a multigrid method constructed for a low order finite volume discretization defined on a subgrid of the DG mesh. Numerical experiments on atmospheric flow problems show the benefit of this approach.

1 INTRODUCTION

The Discontinuous Galerkin Spectral Element method has emerged as one of the most promising discretizations to replace second order finite volume methods in industrial simulations of turbulent, time dependent, fluid flow.

- It is of high order, a property considered to be of critical importance for efficient Large-Eddy-Simulation (LES) of turbulent and wall bounded flow.
- It can handle the complex geometries necessary in real world applications.
- It has computational advantages on modern computer architectures, and lends itself well to parallelism based on domain decomposition.

However, many applications also require implicit time stepping to avoid excessive time step restrictions. Examples include situations when the mesh size varies considerably within the domain, as is common for wall bounded flows, as well as low Mach flows, typical for atmospheric problems. The type of problems considered here involve phenomena on a wide range of time scales. Particularly, we assume the phenomena of interest to occur over relatively long time scales

compared to the fastest dynamics. This is typical for applications such as weather forecasts where simulations stretch over days or weeks, while the Euler equations they are based on also describe much faster processes e.g. sound waves. Explicit time stepping methods are unsuitable for such problems because of the stability condition they impose on the time steps. Instead it is suitable to use implicit time stepping methods. This allows decoupling the resolution of the spatial discretization from that of the temporal. Implicit discretizations require efficient solvers to yield improvement over explicit time stepping methods.

Implicit time stepping requires solving large sparse linear systems of equations, which in turn requires good preconditioners to be competitive computationally. As pointed out in [2], to achieve the peak floating point performance of modern processors, it is necessary to reuse data loaded into the CPU, i.e. the arithmetic intensity of the algorithms must be sufficiently high. Algorithms based on frequent matrix-vector products with assembled matrices are unlikely to achieve sufficient arithmetic intensity, and it is therefore necessary to consider matrix-free or Jacobian-free algorithms [18, 23] and preconditioners suitable for such solvers. The lack of efficient and grid independent solvers for DG discretizations of compressible time dependent flow problems limits the applicability of high order discretizations in many practical applications. Previous attempts are [1, 6, 10, 15, 26]. Robust and efficient preconditioning techniques for Jacobian-free Newton-Krylov (JFNK) solvers are therefore an active area of research.

Multigrid methods have been shown to scale optimally in many settings, in particular for elliptic problems, but also for finite volume discretizations of compressible flows [11, 5, 4, 7]. For the multigrid method to be Jacobian-free, the smoother must be Jacobian-free, restricting the possible choices. Recently, a new Jacobian-free multigrid preconditioner for use in JFNK methods was suggested in [7, 17]. The approach uses a geometric multigrid method defined on a sub cell finite volume discretization (also known as low-order-refined (LOR) preconditioning). It is therefore similar to p-multigrid, but allows to make use of existing smoothers for FV methods, such as [3, 6].

This motivates us to build on that foundation. We follow the framework from [8], which makes use of high order time integration, and a JFNK solver with a smart choice of tolerances. We extend the preconditioner from [7, 17] to the discretized 2D Euler equations with a source term for gravity, as is common for atmospheric flows.

The paper is organized as follows. In Sections 2.1 and 2.2 we briefly recall the main building blocks of the DG discretization and implicit time stepping, focusing on first order problems. In Sections 2.3 and 2.3.3 we introduce the proposed solver and preconditioner for DG discretizations. In Section 3 we investigate the effectiveness of this approach for the rising bubble test case, and conclude with discussing the overall results in Section 4.

2 GOVERNING EQUATIONS, DISCRETIZATION and SOLVERS

We consider a general class of time dependent nonlinear advection-reaction problems for a vector valued function $\mathbf{U}\colon (0,T)\times\Omega\to\mathbb{R}^r$ with $r\in\mathbb{N}^+$ components of the form

$$\partial_t \mathbf{U} = \mathcal{L}(\mathbf{U}) \quad := \quad -\nabla\cdot F_c(\mathbf{U}) + S(\mathbf{U}) \quad \text{in } (0,T]\times\Omega \tag{1}$$

in $\Omega\subset\mathbb{R}^d$, $d=1,2,3$. Suitable initial and boundary conditions have to be added. F_c describes the convective flux and S a source term. Note that all the coefficients in the partial differential equation are allowed to depend explicitly on the spatial variable x and on time t but to simplify the presentation we suppress this dependency in our notation.

For the discretization, we use a method of lines approach based on first discretizing the differential operator in space using a DG approximation, and then solving the resulting system of ODEs using a time stepping scheme.

2.1 Spatial Discretization

Given a tessellation $\mathcal{T}_h$ of the computational domain Ω with $\cup_{E\in\mathcal{T}_h} E = \Omega$ we denote with Γ_i the set of all intersections between two elements of the grid $\mathcal{T}_h$, and the set of all intersections, also with the boundary of the domain Ω, is denoted by Γ.

We consider a discrete space V_h^k spanned by Lagrange type basis functions $\boldsymbol{\psi}_i(x)$ based on tensor product Gauss-Legendre (GL) quadrature nodes. This yields diagonal mass matrices as discussed in detail in [12, 21, 22].

After fixing the grid and the discrete space, we seek

$$\mathbf{U}_h(t,x) = \sum_i \mathbf{U}_i(t)\boldsymbol{\psi}_i(x) \in V_h^k$$

by discretizing the spatial operator $\mathcal{L}(\mathbf{U})$ in (1) with boundary conditions by defining for all test functions $\boldsymbol{\psi} \in V_h^k$,

$$\langle \boldsymbol{\psi}, \mathcal{L}_h(\mathbf{U}_h)\rangle := \langle \boldsymbol{\psi}, K_h(\mathbf{U}_h)\rangle + \langle \boldsymbol{\psi}, I_h(\mathbf{U}_h)\rangle. \tag{2}$$

Hereby, we have the element integrals

$$\langle \boldsymbol{\psi}, K_h(\mathbf{U}_h)\rangle \quad := \quad \sum_{E\in\mathcal{T}_h} \int_E \big(F_c(\mathbf{U}_h) : \nabla\boldsymbol{\psi} + S(\mathbf{U}_h)\cdot\boldsymbol{\psi}\big), \tag{3}$$

and the surface integrals (by introducing an appropriate numerical flux H_c for the convection term)

$$\langle \boldsymbol{\psi}, I_h(\mathbf{U}_h)\rangle \quad := \quad -\sum_{e\in\Gamma} \int_e H_c(\mathbf{U}_h) : [\![\boldsymbol{\psi}]\!]_e, \tag{4}$$

with $[\![\mathbf{U}]\!]_e$ denoting the classic jump of $\mathbf{U}$ over $e \in \Gamma$. The convective numerical flux H_c is chosen to be the HLLC flux in the numerical experiments shown here, (see [25] for details).

2.2 Temporal discretization

After spatial discretization, we get a system of ODEs for the coefficients of $\mathbf{U}(t)$ which reads

$$\mathbf{U}'(t) \quad = \quad f(\mathbf{U}(t)) \text{ in } (0,T] \tag{5}$$

with $f(\mathbf{U}(t)) = M^{-1}\mathcal{L}_h(\mathbf{U}_h(t))$. $\mathbf{U}(0)$ is given by the projection of $\mathbf{U}_0$ onto V_h^k.

We use Singly-Diagonally Implicit Runge Kutta (SDIRK) methods because of their simplicity and relatively low cost compared to fully implicit schemes. A SDIRK method is defined by a lower triangular Butcher tableau with constant nonzero diagonal elements. In every stage, an equation system of the form

$$\mathbf{G}(\mathbf{U}_s) := \mathbf{U}_s - \alpha\Delta t f(\mathbf{U}_s) - \bar{\mathbf{U}}_s = \mathbf{0} \tag{6}$$

is to be solved, where U_s is the stage value at stage s. $\bar{U}_s$ and α are the previous stage value and the diagonal entry in the Butcher tableau. The systems are solved using a Jacobian-free Newton-Krylov method (see [20]). The preconditioning strategy applied in the Krylov solver is the main topic of this article.

2.3 Solver

The implicit time stepper requires solving one nonlinear algebraic equation system for every stage in the SDIRK method. To solve the systems we use a Jacobian-free Newton-Krylov method with a multigrid preconditioner based on a low order discretization of the problem.

Newton's method is applied to solve (6) with $\mathbf{U}_s^0 = \bar{\mathbf{U}}_s$:

$$\mathbf{G}'(\mathbf{U}_s^k)\delta\mathbf{U}_s = -\mathbf{G}(\mathbf{U}_s^k), \tag{7}$$

$$\mathbf{U}_s^{k+1} = \mathbf{U}_s^k + \delta\mathbf{U}_s. \tag{8}$$

The iteration is terminated when

$$|\mathbf{G}(\mathbf{U}_s^k)| < \text{TOL } |\mathbf{G}(\mathbf{U}_s^0)| \tag{9}$$

where TOL is a specified tolerance. The linear system (7) is solved using GMRES with a preconditioner based on a multigrid method, which will be explained in the following subsections. The tolerance for the termination of the Krylov method is selected adaptively using the second Eisenstat-Walker criterion [13] (setting the parameters $\gamma = 0.1$ and $\alpha = 1$).

GMRES is used in a Jacobian-free manner, i.e. using a finite difference approximation of the Jacobian-Vector product, to avoid the storage of large Jacobian matrices, which for higher order DG methods are prohibitively large:

$$\mathbf{G}'(\mathbf{U})\mathbf{y} \approx \frac{1}{\epsilon}\Big[\mathbf{G}(\mathbf{U} + \epsilon\mathbf{y}) - \mathbf{G}(\mathbf{U})\Big], \tag{10}$$

where $\epsilon = \sqrt{\epsilon_{mach}}/|\mathbf{y}|$ and ϵ_{mach} is the machine precision.

The main difficulty here now lies in finding an effective Jacobian-free preconditioner. The setup excludes some common options such as Gauss-Seidel or incomplete LU preconditioners. Instead we choose a geometric multigrid preconditioner based on an auxiliary low order discretization of the problem.

2.3.1 Low order approximation preconditioner

We make use of a finite volume discretization, defined by the description in Section 2.1 with polynomial degree $k = 0$. It is defined on a subgrid of the DG grid in such a way that the two discretizations have the same number of degrees of freedom. The low order spatial discretization on the refined grid defines a function space V_h^0 consisting of piece-wise constant functions and an operator f_{low}, such that the low order spatial discretization of the problem reads

$$\boldsymbol{u}'(t) = f_{low}(\boldsymbol{u}(t)), \tag{11}$$

where $\boldsymbol{u}$ is the corresponding vector of the degrees of freedom. An example of a subgrid using an equidistant subdivision is shown in Figure 1 for a DG space of order three.

Using the same temporal discretization as for the high order problem, the spatially low order discretization gives rise to a nonlinear system corresponding to eqn. (6)

$$\mathbf{g}(\boldsymbol{u}_s) := \boldsymbol{u}_s - \alpha\Delta t f_{low}(\boldsymbol{u}_s) - \bar{\boldsymbol{u}}_s = \mathbf{0}, \tag{12}$$

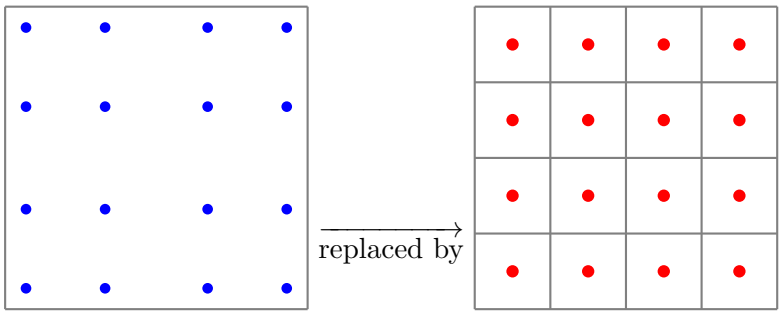


Figure 1: (Left) Gauß-Legendre (GL) nodes for $k = 3$ in one grid cell and (right) FV cell averages in subgrid.

and the linear system

$$\mathbf{g}'(\boldsymbol{u}_s^k)\delta\boldsymbol{u}_s = -\mathbf{g}(\boldsymbol{u}_s^k) \tag{13}$$

corresponding to eqn. (7).

Assuming we have a preconditioner $\mathbf{q}^{-1}(\boldsymbol{u}_s^k) \approx \mathbf{g}'(\boldsymbol{u}_s^k)^{-1}$ for (13), we define a preconditioner for eqn. (7) to be

$$\mathbf{Q}^{-1}\mathbf{U}_s^k := \mathbf{T}^{-1}\mathbf{q}^{-1}\mathbf{T}\mathbf{U}_s^k, \tag{14}$$

where $\mathbf{T}$ is a *transfer function* between the two discretizations (see Section 2.3.2). Note that the transfer function can also be combined with a pre or post smoothing step, which is discussed in Section 2.3.3.

2.3.2 Transfer functions

We now define mappings between the degrees of freedom of the high and the low order discretization. The vectors $\mathbf{U}_s, \mathbf{U}_s^k, \bar{\mathbf{U}}_s$ in the original discretization represent functions in V_h^k. Correspondingly, $\boldsymbol{u}_s, \boldsymbol{u}_s^k, \bar{\boldsymbol{u}}_s$ represent solutions of the low order discretization, i.e. functions in V_h^0. The transfer functions between V_h^k and V_h^0 are defined such that the functions are close *in some sense*.

An obvious choice for $\mathbf{T}$ would be the L^2 projection. However, in our experience, the simpler mapping of interpolating the polynomials in V_h^k in the cell centers of the cells defining V_h^0 gives the same preconditioner performance, while being much easier to implement on unstructured grids and requiring less computational cost. A detailed comparison of different approaches shows the interpolation based transfer functions offer the best performance in terms of preconditioner iterations and computing time [16, 17] and was therefore chosen for the experiments in this paper. Further discussions on this topic can be found in [27].

2.3.3 Multigrid method for finite volume approximations

The multigrid method for the finite volume problem is an agglomeration multigrid method, described in [7, 17]. The method is defined by choosing restriction and prolongation operators, $\mathbf{R}$, $\mathbf{P}$, and a *smoother*, $\mathbf{S}(\cdot,\cdot)$.

We start with a sequence of increasingly refined quad meshes $\mathcal{T}_l$ for $l = 0 \dots L$, where $\mathcal{T}_l$ is generated by splitting each quad $E \in \mathcal{T}_{l-1}$ into 2^d new quads denoted by $C(E)$. The approximation on level l is represented by a vector $\mathbf{U}_l$.

For restriction we use the agglomeration operator $\mathbf{R}$ mapping a vector containing the cell averages on the tessellation $\mathcal{T}_l$ to a vector containing the implied cell averages on the coarser tessellation $\mathcal{T}_{l-1}$

$$\left(\mathbf{R}\mathbf{U}_l\right)_E = \frac{1}{|C(E)|} \sum_{\mathbf{k} \in C(E)} U_{l,\mathbf{k}}, \quad E \in \mathcal{T}_{l-1}. \tag{15}$$

For prolongation we use the injection operator $\mathbf{P}$. It assigns cell averages in child cells to the value in their parent cell. It can be expressed as

$$\left(\mathbf{P}\mathbf{U}_l\right)_{\mathbf{k}} = U_{l,E}, \quad \forall \mathbf{k} \in C(E), \quad E \in \mathcal{T}_l. \tag{16}$$

The smoother $\mathbf{S}(\cdot,\cdot)$ is a linear iterative method that converges to the solution of (13). It is designed to quickly reduce high frequency components of the error. We use a smoother based on pseudo time iteration. It is described in detail in section 2.3.4.

The multigrid algorithm consists of a V- or W-cycle with pre- and post-smoothing. On the coarsest grid level we apply the smoother twice. The whole multigrid method for solving equation (13) can be written as

1. Compute $\mathbf{MG}_l(\mathbf{x}, \mathbf{b};\ \boldsymbol{u}_s^k)$:
2. $\mathbf{x} := \mathbf{S}(\mathbf{x}, \mathbf{b})$
3. If $l > 0$
   - $\mathbf{r} := \mathbf{R}(\mathbf{g}'(\boldsymbol{u}_s^k)\mathbf{x} - \mathbf{b})$
   - $\mathbf{v} := \mathbf{0}$
   - Repeat once for V-cycle and twice for W-cycle
     $\mathbf{v} := \mathbf{MG}_{l-1}(\mathbf{v}, \mathbf{r};\ \mathbf{R}\boldsymbol{u}_s^k)$
   - $\mathbf{x} := \mathbf{x} - \mathbf{P}\mathbf{v}$
4. $\mathbf{x} := \mathbf{S}(\mathbf{x}, \mathbf{b})$
5. Return $\mathbf{x}$

If the smoother is effective, the iteration is mesh independent and converges to the solution of (13) in a few iterations.

2.3.4 Smoother / Pseudo time iteration

A common approach to solve linear systems such as (13) arising from computational fluid dynamics models is so called pseudo-time stepping. Smoothing can be both done during the multigrid cycle, i.e., on the finite-volume data before or after prolongation/restriction. Smoothing is also possible on the DG data before the transfer to the finite-volume grid or after reconstruction of the DG data. In both cases we use a similar pseudo-time stepping approach which we describe here only for the finite-volume data.

Instead of solving (13) directly, a pseudo time variable is introduced

$$\frac{\delta \mathbf{w}}{\delta \tau} = -\mathbf{g}(\boldsymbol{u}_s^k) - \mathbf{g}'(\boldsymbol{u}_s^k)\mathbf{w} \tag{17}$$

to construct an ODE with a steady state at the solution of the linear system. The ODE is stable if the original spatial discretization of the system is stable. In the next step, an explicit Runge-Kutta (RK) time integration scheme is used to integrate in pseudo time from some initial guess. To get an idea of how the optimization of the Runge-Kutta method is done, see [3]. Note, that $\mathbf{g}'$ is approximated in the same way as described before for the DG setup.

3 NUMERICAL EXPERIMENT

This test case considers the convection of a warm air bubble. The stiffness of this problem follows from the difference in speed between the rising bubble and the fast sound waves also present in the solution. Our implementation follows the description of the test case in [24] for the evolution of a warm bubble in a neutrally stratified atmosphere. The bubble is positioned at a certain height, it will start to rise and develop vortex structures in interaction with the cooler surrounding air. The governing equations are the compressible Euler equations in potential temperature formulation with gravitational force and using a BG-fix approach for well balancing as described in [9]. We choose basis functions of the DG space of degree $k = 3$ throughout. The domain is $[0,\ 1000] \times [0,\ 2000]\mathrm{m}^2$ and the environmental atmosphere is determined by the formulas

$$\tilde{\theta} = 303.15K, \quad \tilde{T} = T_0 - zgc_p^{-1}, \quad \tilde{p} = p_0(\tilde{T}T_0^{-1})^{c_p/R_d}. \tag{18}$$

In the test case, $c_p = 1,005\mathrm{J/(kg\ K)}$, $c_v = 717.95\mathrm{J/(kg\ K)}$ and $g = 9.80665\mathrm{m/s}^2$. The initial atmosphere is the sum of this neutrally stratified environmental atmosphere and a perturbation induced by a deviation of the potential temperature with the shape of a "flattened" Gaussian pulse, given by

$$\theta'(x,z) = A_0 \begin{cases} 1.0, & r := ||(x,z) - (x_0,z_0)|| < a, \\ exp(-(r-a)^2/s^2), & 0 \le r - a \le 3s, \\ 0, & \text{else}, \end{cases} \tag{19}$$

where $A_0 = 0.5\mathrm{K}$, $x_0 = 500\mathrm{m}$, $z_0 = 520\mathrm{m}$, $a = 50\mathrm{m}$ and $s = 100\mathrm{m}$. The introduction of the perturbation θ' into the initial state follows the same way as described in [9].

On all sides of the domain we impose slip boundary conditions. The system is integrated until 1200s of model time.

To test the effectiveness of the multigrid preconditioner we run the test case with an explicit 4-stage 3rd order SSP method described in [19] and implemented in DUNE-FEM-DG [12]. The stable time step size used at a vertical resolution of $\Delta x = 12.5m$ was $\Delta t = 0.002558s$.

For the implicit time stepping we use an SDIRK method of order 2 with 2 stages known as Ellsiepen's method (20) with $\alpha = 1 - \frac{\sqrt{2}}{2}$, see [14]. We use a fixed time step of $\Delta t = 5s$ and $\Delta t = 10s$. Other time step sizes have been tested but did yielded inferior results. For these two time step sizes different configurations of the solver will be tested.

$$\begin{array}{c|cc} \alpha & \alpha & \\ 1 & 1-\alpha & \alpha \\ \hline & 1-\alpha & \alpha \\ & 1-\beta & \beta \end{array} \tag{20}$$

The internal Newton solve are run with a relative coarse tolerance of $\mathrm{TOL} = 10^{-3}$.

For the preconditioner there are a variety of configuration parameters which are encrypted in the following key:

$$\mathbf{mg}\, a\, b\, c\, d\, e\, f\, G \tag{21}$$

The **mg** is an abbreviation for multigrid and the other letters have the following meaning:

- **a,b:** Number of **pre** and **post** smoothing steps on the DG solution
- **c,d:** Number of **pre** and **post** smoothing steps on the finest FV level
- **e,f:** Number of **pre** and **post** smoothing steps on the intermediate FV levels
- **G:** Either **V** or **W**, denoting the cycle of the multigrid method

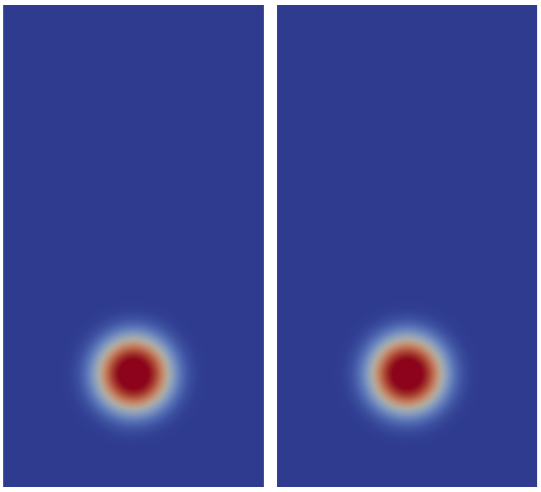
Since the test cases use polynomial degree $k = 3$ the multigrid preconditioner has at least $l + 2$ levels (going from DG to FV fine as shown in Figure 1), where l is the number of levels existing in the mesh prior to the FV refinement. For the runs at $\Delta x = 25m$ we have $l = 2$ and for $\Delta x = 12.5m$ we have $l = 3$, which means 4 or 5 multigrid levels, respectively. For example, the configuration $\mathbf{mg}111111V$ means multigrid in V-cycle mode with one pre and post smoothing step on the DG solution as well as on all FV levels.

The solution of the test case at various points in time for the explicit and one of the considered implicit configurations is displayed in Figure 2.

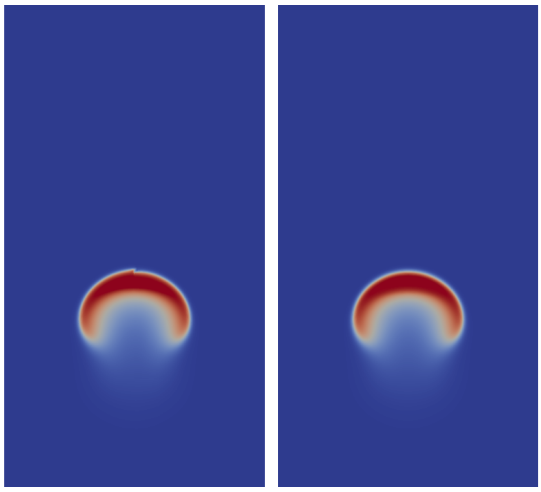
Figure 3 contains a selection of different configurations of the multigrid method for two time step sizes. In Figure 3a we compare the non-preconditioned implicit method with the preconditioned method The different configurations include either no pre and post smoothing step on the DG solution and a mix of both, one or two smoothing steps on the FV levels as well as using a V-cycle and a W-cycle. Clearly, using a W-cycle does not significantly reduce the number of iterations but since this involves much more operator evaluations it is not competitive in terms of CPU time with the V-cycle version. The use of W-cycle has therefore been abandoned. Figure 3a also includes a comparison with p-multigrid which is not competitive compared to the FV based multigrid version. The two best configurations in terms of minimal linear iterations and minimal CPU time are $\mathbf{mg}001111V$ and $\mathbf{mg}111111V$, where the additional pre and post smoothing on the DG solution reduces the number of linear iterations at the cost of more operator evaluations. Some of the configurations, e.g. $\mathbf{mg}112222W$, yield a competitive number of linear iterations but the overall application of the multigrid method is much more expensive leading to a termination of the simulation due to exceeding the reserved wall time window on the super computer. In any case these combinations can be ruled out since other configurations are much faster.

In Figure 3b we compare the number of operator calls (DG and FV combined) for the different combinations from Figure 3a. It is observed that the number of operator calls corresponds to the number of linear iterations. Therefore, the number of operator calls is not further considered in this investigation.

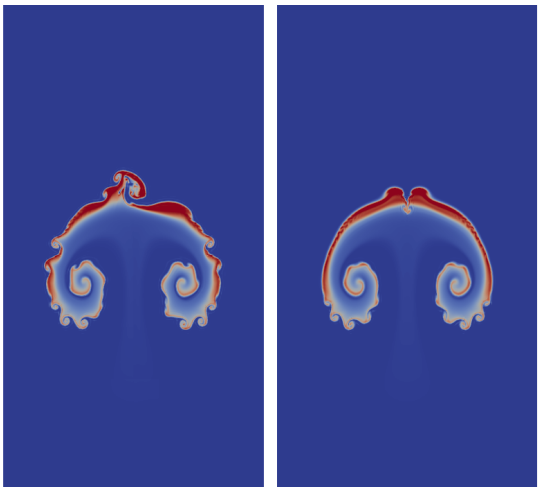
Figure 3c compares the two best configurations from Figure 3a and 3b for different grid width $\Delta x = 25m$ and $\Delta x = 12.5m$. The number of linear iterations is not grid independent but grows with decreasing grid width. However, in this particular case the growth less than a factor of 2 which is still acceptable since the time step size was kept constant at $\Delta t = 5s$.



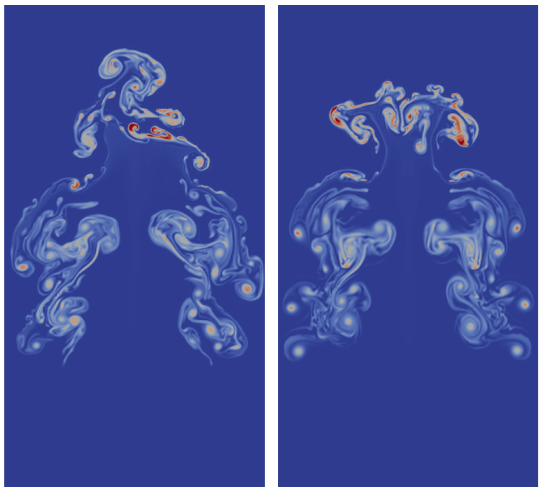
(a) $t = 0$



(b) $t = 360$



(c) $t = 720$



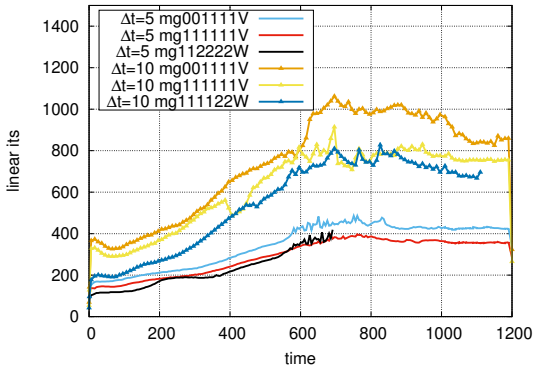
(d) $t = 1200$

Figure 2: Potential temperature for various times. Left, the solution computed at $\Delta x = 12.5m$ using the explicit time stepping with $\Delta t = 0.002558s$, and right the solution at $\Delta x = 12.5m$ using implicit time stepping with multigrid preconditioning and time step $\Delta t = 5s$. Note that the implicit time stepping allows to use a time step that is ≈ 2000 times larger than the explicit time step. Both schemes produce an acceptable solution.

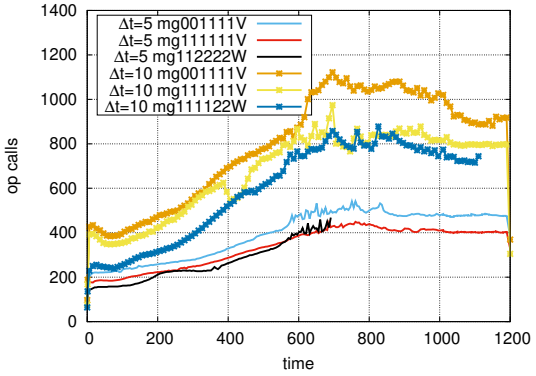
Finally, Figure 3d shows the linear iterations needed when using the non-preconditioned implicit version. Again the simulation terminated due to exceeding the allocated time window on the super computer.

In summary, at grid width $\Delta x = 12.5m$ the fastest run was **mg**111111V with $7915s$ to finish followed by **mg**0011111V with $8166s$, both using $\Delta t = 5s$. Compared to the explicit SSP3(4) time stepping, which needed $14000s$, this is roughly twice as fast. All other multigrid configurations were slower, including those using a larger time step size. At $\Delta x = 25m$, using a

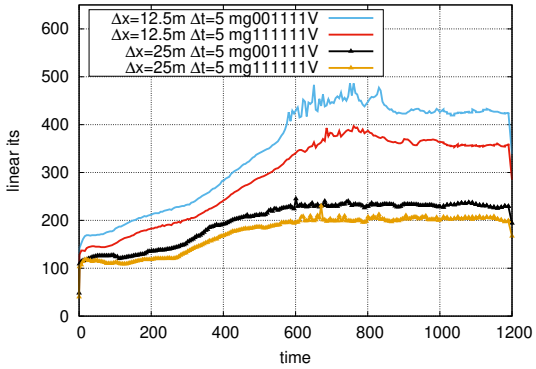
time step size of $\Delta t = 10s$ the configuration **mg**111111V needed 1181s followed by **mg**0011111V which 1196s. The explicit SSP3(4) time stepping took 1763s.



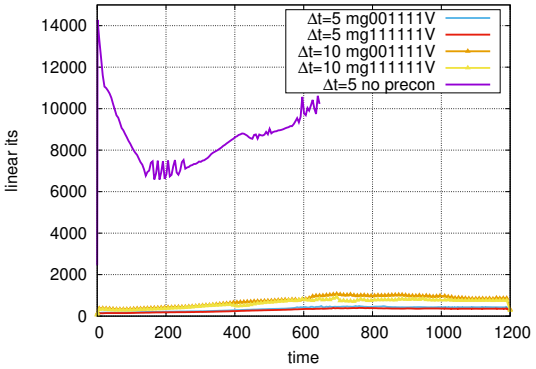
(a) Comparison of linear iteration counts for $\Delta t = 5s$ and $\Delta t = 10s$



(b) Comparison of operator calls for $\Delta t = 5s$ and $\Delta t = 10s$



(c) Comparison of linear iteration counts for $\Delta t = 5s$ for two different grid width $\Delta x = 12.5m$ and $\Delta x = 12.5m$.



(d) Comparison including non-preconditioned run.

Figure 3: Linear iterations for various configurations of the multigrid solver.

4 CONCLUSIONS

In this paper we presented a low-order-refined (LOR) preconditioning approach for high order discretizations of compressible fluid flows. The preconditioner is Jacobian free, uses little extra memory, and achieves close to optimal arithmetic intensity for large problems on parallel machines. The numerical experiments demonstrate competitiveness compared to explicit time stepping methods and Jacobian-based preconditioners such as ILU.

Future work will consider smoother methods tailored to improve the convergence rate for low Mach problems and for anisotropic grids.

REFERENCES

[1] F. Bassi, A. Ghidoni, S. Rebay, and P. Tesini. High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations. *Int. J. Num. Meth. Fluids*, 60:847–865, 2009.

[2] P. Bastian, E. Müller, S. Muething, and M. Piatkowski. Matrix-free multigrid block-preconditioners for higher order Discontinuous Galerkin discretisations. *J. Comput. Phys.*, 394:417–439, 2019.

[3] P. Birken. Optimizing Runge-Kutta smoothers for unsteady flow problems. *ETNA*, 39:298–312, 2012.

[4] P. Birken. *Numerical Methods for Unsteady Compressible Flow Problems*. CRC Press, 2021.

[5] P. Birken, J. Bull, and A. Jameson. Preconditioned Smoothers for the Full Approximation Scheme for the RANS Equations. *J. Sci. Comput.*, 78(2):995–1022, 2019.

[6] P. Birken, G. Gassner, M. Haas, and C. D. Munz. Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations. *J. Comput. Phys.*, 240:20–35, 2013.

[7] P. Birken, G. J. Gassner, and L. M. Versbach. Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods. *Int. J. Comput. Fluid Dyn.*, 33(9):353–361, 2019.

[8] D. S. Blom, P. Birken, H. Bijl, F. Kessels, A. Meister, and A. H. van Zuijlen. A comparison of rosenbrock and esdirk methods combined with iterative solvers for unsteady compressible flows. *Adv. Comp. Math.*, 42:1401–1426, 2016.

[9] S. Brdar, M. Baldauf, A. Dedner, and R. Klöfkorn. Comparison of dynamical cores for NWP models: comparison of COSMO and DUNE. *Theor. Comput. Fluid Dyn.*, 27(3-4):453–472, 2013.

[10] L. E. Carr, C. F. Borges, and F. X. Giraldo. Matrix-Free Polynomial-Based Nonlinear Least Squares Optimized Preconditioning and its Application to Continuous and Discontinuous Element-Based Discretizations of the Euler Equations. *J. Sci. Comput.*, 66:917–940, 2016.

[11] D. A. Caughey and A. Jameson. How Many Steps are Required to Solve the Euler Equations of Steady Compressible Flow: In Search of a Fast Solution Algorithm. *AIAA Paper 2001-2673*, 2001.

[12] A. Dedner and R. Klöfkorn. Extendible and Efficient Python Framework for Solving Evolution Equations with Stabilized Discontinuous Galerkin Method. *Commun. Appl. Math. Comput.*, 4:657–696, 2022.

[13] S.C. Eisenstat and H.F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM J. Sci. Comput.*, 17(1):16–32, 1996.

[14] P. Ellsiepen. *Zeits- und ortsadaptive Verfahren angewandt auf Mehrphasenprobleme poröser Medien*. PhD thesis, University of Stuttgart, 1999.

[15] M. Franco, P. O. Persson, and W. Pazner. Iterative subregion correction preconditioners with adaptive tolerance for problems with geometrically localized stiffness. *Commun. Appl. Math. Comput.*, 6:811–836, 2023.

[16] Kasimir, J. *Subgrid finite volume preconditioner for Discontinuous Galerkin implemented in the DUNE framework*. Master thesis, Lund University, 2021.

[17] Kasimir, J. and Versbach, L. M. and Birken, P. and Gassner, G. J. and Klöfkorn, R. An finite volume based multigrid preconditioner for dg-sem for convection-diffusion. In *Fluid Dynamics and Transport Phenomena*, volume 600 of *World Congress in Computational Mechanics and ECCOMAS Congress*, pages 1–12, 2021.

[18] D. Kempf, R. Heß, S. Müthing, and P. Bastian. Automatic Code Generation for High-Performance Discontinuous Galerkin Methods on Modern Architectures. *ACM Trans. Math. Softw.*, 47(1), 2020.

[19] David I. Ketcheson. Highly Efficient Strong Stability-Preserving Runge-Kutta Methods with Low-Storage Implementations. *SIAM J. Sci. Comput.*, 30(4):2113–2136, 2008.

[20] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193(2):357–397, 2004.

[21] D. A. Kopriva and G. Gassner. On the Quadrature and Weak Form Choices in Collocation Type Discontinuous Galerkin Spectral Element Methods. *J. Sci. Comput.*, 44:136–155, 2010.

[22] David A. Kopriva, Stephen L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *Int. J. Numer. Methods Eng.*, 53(1):105–122, 2002.

[23] M. Kronbichler and K. Kormann. Fast Matrix-Free Evaluation of Discontinuous Galerkin Finite Element Operators. *ACM Trans. Math. Softw.*, 45(3), 2019.

[24] A. Robert. Bubble Convection Experiments with a Semi-Implicit Formulation of the Euler Equations. *J. Atmos. Sci.*, 50, 1993.

[25] E.F. Toro. *Riemann solvers and numerical methods for fluid dynamics. A practical introduction. 2nd ed.* Berlin, Springer, 1999.

[26] L. Wang, W. Trojak, F. Witherden, and A. Jameson. Nonlinear p-multigrid preconditioner for implicit time integration of compressible navier–stokes equations with p-adaptive flux reconstruction. *J. Sci. Comput.*, 93, 2022.

[27] F. D. Witherden and P. E. Vincent. On Nodal Point Sets for Flux Reconstruction. *J. Comput. Appl. Math.*, 381, 2021.