40th Fluid Dynamics Conference and Exhibit
28 June - 1 July 2010, Chicago, Illinois

AIAA 2010-4991

Relaxation Techniques for High-Order Discretizations of Steady Compressible Inviscid Flows

Francesca Iacono* and Georg May†

Graduate school AICES, RWTH Aachen University, Schinkelstr. 2, 52062 Aachen, Germany

Z.J. Wang‡

Department of Aerospace Engineering, Iowa State University, 2271 Howe Hall, 50011 Ames, IA

There does not exist a definite best strategy to advance high-order discretizations of inviscid compressible flows to steady state. The need for a wide range of possible time-relaxation strategies is determined by different possible difficulties: stability issues, stiffness of the matrix, memory limitations, ill-conditioning, and slow convergence. We outline the main available methods and identify the kind of problems for which they are more suitable. We extensively describe the idea of the matrix-free Squared Preconditioning and apply it for the first time also to a Newton iteration nonlinearly preconditioned by means of the flow solver. We apply this approach to inviscid flows in order to study the tuning of its many parameters and to assess its potential.

I. Introduction

In the recent past, research effort has been focused on the development of numerical methods for conservation laws which are locally conservative, high-order accurate in smooth regions of the solution, geometrically flexible, computationally efficient, and simply formulated. Some desired features are still open to improvements, such as the ability to show sharp and monotone shock transitions and the comparability of computational times with low-order methods.

Our attention concentrates on high-order local discontinuous semidiscretization methods. The *Spectral Difference* (SD) method for conservation laws on unstructured grids utilizes the concept of discontinuous and high-order local representations to achieve conservation and high accuracy in a manner similar to the *Discontinuous Galerkin*[1–6] (DG) and *Spectral Volume*[7–10] (SV) methods, but is based on the finite difference formulation. Actually it can be seen as a quadrature-free DG method.[11] The SD method has been developed by Liu *et al.*[12,13] and Wang *et al.*[14] As a matter of fact, the DG, SV, and SD methods are similar in that they share the same solution space, i.e., the space of piecewise-continuous polynomials, and that Riemann solvers are used at the element interfaces to provide solution coupling between the discontinuous elements and appropriate numerical dissipation necessary to achieve stability. In addition, all of these approaches are locally conservative at the element level, making them suitable for problems with discontinuities. They differ on how the degrees of freedom are chosen, and how they are updated. The SD method has an important advantage over the DG and SV methods, namely that no integrals have to be evaluated to compute the residuals.

Our flow solver is based on a spatial discretization of SD type[15,16] on triangular grids. In 2008 Van den Abeele *et al.* demonstrated that the SD discretization can lead in some cases to a weak instability.[17] Hence, one of our current research interests is other (stable) types of discretization, such as the CPR (*Correction Procedure via Reconstruction*) formulation,[18] and a new formulation of SD using an approximation of the divergence operator on Raviart-Thomas elements.[19] While no instability arises in the solution of flows that we are considering even with the standard SD discretization, we decided to also consider the DG discretization.

*Ph.D. candidate, AIAA Student Member.

†Junior Professor in 'High-Order Methods for Simulation of Multiphase Flow', AIAA Member.

‡Professor of Aerospace Engineering, Associate Fellow of AIAA.

The first DG method was introduced in 1973 by Reed and Hill and a major development was carried out by Cockburn *et al.* in a series of papers,[1–5] in which they established a framework to easily solve nonlinear time-dependent hyperbolic conservation laws using a DG discretization in space with exact or approximate Riemann solvers at interface fluxes and TVB limiter to achieve nonoscillatory properties for strong shocks.

In this article though, we deal with the phase which follows the spatial discretization of the equations, i.e. the relaxation in pseudo-time performed in order to reach steady-state solutions. Starting from the initial application of time-explicit multistage schemes, we cover the main phases till the development of hybrid relaxation techniques. As a further step, we enclose the flow solver into a Nonlinear GMRES method (which we call '*Nonlinear Preconditioning approach*'). This is possible by seeing the flow solver as a fixed-point iteration and by then solving it using a Nonlinear GMRES algorithm.[20,21] For this approach a large number of parameters have to be chosen and tuned: we carry out parametric studies and consider scenarios of different kinds, such as matrix-explicit/matrix-free. Numerical simulations of inviscid flow along a smooth bump are performed: for a stretched mesh the Nonlinear Preconditioning approach performs better than standard flow solver without nonlinear preconditioning.

The remainder of this article is structured in the following way. In the next section, we outline the techniques that are commonly applied for time-integration in this context. Section III describes extensively the idea of matrix-free Squared Preconditioning, which was first presented in Ref. 22. Section IV describes our approach, i.e., nonlinear preconditioning for a matrix-free Newton-Krylov method. The results of numerical experiments are presented in Section V: effort was invested for validation of the new approach and better understanding of the trade-off of parameters.

II. Time-Explicit and Time-Implicit Flow Solvers

The Euler equations can be written as nonlinear conservation laws of the form:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u}$ is the vector of the conservative variables. The attention is here focused on steady-state solutions, obtained by advancing the equations in a so-called 'pseudo-time' variable t^*

$$\frac{\partial \mathbf{u}}{\partial t^*} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{0}. \tag{2}$$

The nonlinear equations which are taken into consideration derive from the spatial semidiscretization of Eq. (2) and look like this

$$\frac{d\mathbf{U}}{dt^*} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{3}$$

where $\mathbf{U}$ is the global vector of all the degrees of freedom, and $\mathbf{R}$ is the nonlinear residual vector of the spatial semidiscretization terms.

For this kind of equations there is not a definite solution strategy that shows to be the best in all situations. Many factors determine a certain solution method to be more suitable than the other. In the first place, time-explicit relaxation can be considered. *Runge-Kutta multistage schemes* have been widely used.[6,23] For steady-state problems, which receive our attention here, time accuracy during the relaxation process is of no importance, so that the chosen CFL number can be determined just by stability considerations. Moreover, *geometric multigrid* is well-known to be an efficient convergence accelerator in this context. Limitations are given, however, by stability limits: the linear stability limit for the combination of Runge-Kutta time stepping together with SD and DG discretization applied to the one-dimensional linear advection equation decays as m^{-2} for growing order of accuracy m. Moreover, multigrid has problems in some situations: when the problem is stiff (near stagnation points, at shocks, across the sonic line and when high-aspect-ratio cells are used inside the boundary layer) and when the flow aligns itself with the computational mesh (directional decoupling).

The previously mentioned problems suggested to take into consideration time-implicit relaxation too, which happens to outperform time-explicit integration for high orders of approximation.[24] As noted before, we are interested in steady-state solutions, i.e., in solutions to the system of nonlinear equations

$$\mathbf{R}(\mathbf{U}) = \mathbf{0}. \tag{4}$$

Therefore, one possibility for time-implicit integration is given by *Newton's method*:

$$\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n \left(\mathbf{U}^{n+1} - \mathbf{U}^n\right) = -\mathbf{R}(\mathbf{U}^n), \tag{5}$$

which can also be written as

$$\begin{aligned} \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n \Delta\mathbf{U}^n &= -\,\mathbf{R}(\mathbf{U}^n), \\ \mathbf{U}^{n+1} &= \mathbf{U}^n + \Delta\mathbf{U}^n. \end{aligned} \tag{6}$$

In Eqn. (6) one can clearly distinguish the two steps: computation of the Newton update $\Delta\mathbf{U}^n$ by solving a linear system and Newton update of the solution $\mathbf{U}^n$. When the linear system is solved by means of a Krylov subspace method, the method is called a *Newton-Krylov method*. If a GMRES method is used to solve the linear system, the name *Nonlinear GMRES method* has also been used.[21]

Newton's method only works if the initial value $\mathbf{U}^0$ is close enough to the true zero of the functions $\mathbf{R}(\mathbf{U})$. Numerical experiments show that the time necessary to reach the Newton's convergence bubble is often too large, so that other methods become of interest. The *backward Euler* (or *damped Newton*) method gives a better condition number to the linear system to be solved:

$$\begin{aligned} \left(\frac{1}{\Delta t^*}\mathbb{I} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n\right) \Delta\mathbf{U}^n &= -\mathbf{R}(\mathbf{U}^n), \\ \mathbf{U}^{n+1} &= \mathbf{U}^n + \Delta\mathbf{U}^n, \end{aligned} \tag{7}$$

where Δt^* is the timestep. The superscript $*$ denotes that we are not dealing with the real time, but pseudo-time. By letting the timestep Δt^* go to infinity, one recovers the Newton's method. We solve the linear system in Eqs. (7) by a GMRES algorithm.[25] Algorithm II.1 shows the nested structure of the iterations in these methods.

Algorithm II.1 *Newton-Krylov method*
Given $\boldsymbol{U}^0$*:*

1. ***Outer iteration.*** *For* $n = 0, 1, 2, \ldots$ *Do*
2. *Compute the right-hand side* $-\boldsymbol{R}(\boldsymbol{U}^n)$
3. *If (converged) Stop*
4. ***Inner iteration.*** *Solve the linear system* $\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right|_n \Delta\boldsymbol{U}^n = -\boldsymbol{R}(\boldsymbol{U}^n)$ *(Newton's method) or* $\left(\frac{1}{\Delta t^*}\mathbb{I} + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right|_n\right) \Delta\boldsymbol{U}^n = -\boldsymbol{R}(\boldsymbol{U}^n)$ *(damped Newton method) by GMRES method*
5. $\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta\boldsymbol{U}^n$
6. *EndDo*

Implicit time-integration usually requires the storage of the residual Jacobian matrix $\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n$. In particular, the matrix is often computed and 'frozen' for a certain number of iterations for computational efficiency. This matrix becomes very large when dealing with high-order discretization of 2D and 3D flows. To overcome the restrictions imposed by the necessity to store such large matrices, both *Jacobian-free* and *matrix-free* methods have been considered and developed.[26] Attention should be paid to the terminology: 'matrix-free' implies that no matrix of any type is formed or stored. However, usually, a matrix (or a set of matrices) is formed for preconditioning purposes. These matrices can be simpler than the true Jacobian of the problem, so the algorithm is then properly and in a more general meaning said to be 'Jacobian-free'.

III. A Matrix-Free Newton-Krylov Method: Squared Preconditioning

Let us present here the main steps which characterize the matrix-free time-implicit flow solver presented in Ref. 22. Let us focus on the solution of the linear system arising by the application of the (damped) Newton method (step 4 of Algorithm II.1). At each Newton iteration n let us define $A^{(n)} = \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n$ if we choose the Newton's method and $A^{(n)} = \frac{1}{\Delta t^*}\mathbb{I} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n$ if we take the damped Newton method. Let $\mathbf{b}^{(n)} = -\mathbf{R}(\mathbf{U}^n)$ and $\mathbf{x}^{(n)} = \Delta\mathbf{U}^n$. When not necessary we drop the superscript (n) for the sake of simplicity.

The considered problem is to solve a linear system $A\mathbf{x} = \mathbf{b}$. Preconditioning is needed. Here we imagine the preconditioner to be an operator, since we may not always be able to write the preconditioner as a matrix. The left-preconditioned GMRES algorithm is the GMRES algorithm applied to the system

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}, \tag{8}$$

where P is the preconditioner at the outer Newton iteration n. One usually assumes that the preconditioning matrix P is fixed, i.e., it does not change from one inner GMRES iteration to the following. However, in some cases, the matrix P is not available explicitly. Instead, the operation $P^{-1}\mathbf{q}$, where $\mathbf{q}$ is a vector, is the result of some unspecified computation, possibly another iterative process. In such cases, it may well happen that P is not a constant operator, i.e., P_j, where j is the index for the linear iteration.

In the literature different GMRES-like iterations schemes have been proposed, where it is allowed to take a different preconditioner in each linear step. One of them, proposed and analyzed by Van der Vorst,[27] is a recursive variant of the GMRES algorithm, called *Recursive GMRES* (GMRESR). We choose the *Flexible GMRES algorithm* (FGMRES), introduced in Ref. 28.

Since we are using the GMRES algorithm, only a matrix-vector product routine has to be provided and the matrix A does not need to be explicitly built. This is true for the preconditioner as well. Whenever in the GMRES algorithm a vector $\mathbf{q}$ needs to be preconditioned, we solve $P_j\mathbf{q}_{precond} = \mathbf{q}$, where $P_j := \mathcal{P}_j(A)$ is the polynomial that is implicitly constructed in a few iteration steps of GMRES. In this way, the preconditioner P_j is an approximation of the matrix A. The number of GMRES steps $n_{Krylov}(j)$ can be different at each iteration j and is low in comparison with the global number of Krylov vectors used to solve the linear system $A\mathbf{x} = \mathbf{b}$.

Let r be the restart parameter. Algorithm III.1 describes the inner iterations of step 4 in Algorithm II.1 for the outer Newton iteration n.

Algorithm III.1 *Flexible GMRES with Left Preconditioning*

1. $\boldsymbol{s}_0 = [\mathcal{P}_0(A)]^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0),\ \beta = \|\boldsymbol{s}_0\|_2,\ and\ \boldsymbol{v}_1 = \dfrac{\boldsymbol{s}_0}{\beta}$
2. *For* $j = 1, \ldots, r$, *Do*
3. $\quad \boldsymbol{z}_j = A\boldsymbol{v}_j\ and\ \boldsymbol{w} = [\mathcal{P}_j(A)]^{-1}\boldsymbol{z}_j$
4. $\quad$ *For* $i = 1, \ldots, j$, *Do*
5. $\quad\quad h_{i,j} = (\boldsymbol{w}, \boldsymbol{v}_i)$
6. $\quad\quad \boldsymbol{w} = \boldsymbol{w} - h_{i,j}\boldsymbol{v}_i$
7. $\quad$ *EndDo*
8. $\quad h_{j+1,j} = \|\boldsymbol{w}\|_2\ and\ \boldsymbol{v}_{j+1} = \dfrac{\boldsymbol{w}}{h_{j+1,j}}$
9. *EndDo*
10. $V_r = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r],\ \bar{H}_r = \{h_{i,j}\}_{1 \le i \le j+1,\, 1 \le j \le r}$
11. $\boldsymbol{y}_r = \arg\min_{\boldsymbol{y}} \|\beta\boldsymbol{e}_1 - \bar{H}_r\boldsymbol{y}\|_2\ and\ \boldsymbol{x} = \boldsymbol{x}_0 + V_r\boldsymbol{y}_r$

12. If satisfied Stop, else set $\boldsymbol{x}_0 := \boldsymbol{x}$ *and go to step 1*

May *et al.*[22] named this method *Squared Preconditioning*, because the preconditioning is realized by applying the GMRES method on two nested levels:

(i) $A\mathbf{x} = \mathbf{b}$ solved by GMRES, left-preconditioned by the preconditioning operator $\mathcal{P}_j(A)$ at inner GMRES iteration j;

(ii) $\mathbf{w} = [\mathcal{P}_j(A)]^{-1}\mathbf{z}_j \approx A^{-1}\mathbf{z}_j$ computed by unpreconditioned GMRES with a low number $n_{Krylov}(j)$ of Krylov vectors.

We said that in the GMRES context we do not need the explicit matrix A, but we did not mention yet which matrix-vector product routine we want to consider. Let us go back to our original system of nonlinear equations to be solved. Both for the Newton's method (6) and for the damped Newton method (7), the matrix A in the linear system contains the residual Jacobian $\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n$. A common approach substitutes the matrix-vector product by a first-order series approximation:

$$\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_n \mathbf{q} \approx \frac{\mathbf{R}(\mathbf{U}^n + \varepsilon\,\mathbf{q}) - \mathbf{R}(\mathbf{U}^n)}{\varepsilon}, \tag{9}$$

with ε small. This is what Chan and Jackson called *Directional Differencing Discrete Newton* (DDDN) method.[29] We choose in particular $\varepsilon := \sqrt{\|\mathbf{U}^n\| \cdot \varepsilon_{rel}}$, with ε_{rel} close to the machine round-off error. This eliminates the need to store the entire matrix, at the expense of one additional residual evaluation for each matrix-vector product performed. The Flexible GMRES[28] framework affords the opportunity to extend the matrix-free approach to the preconditioner as well. Also in this case one can recognize two nested levels (i) and (ii), where, however, on both levels the GMRES algorithm is performed **matrix-free**.

Memory efficiency is realised through this method, as the need to store matrices is eliminated at both the linear solver and preconditioner levels. The final matrix-free Newton-Krylov method is presented in Algorithm III.2 (where the superscript n has again been introduced).

Algorithm III.2 *Matrix-free Newton-Krylov method*
Given $\boldsymbol{U}^0$*:*

1. ***Outer iteration.*** *For* $n = 0, 1, 2, \ldots$ *Do*
2. *Compute the right-hand side* $\boldsymbol{b}^n = -\boldsymbol{R}(\boldsymbol{U}^n)$
3. *If (converged) Stop*
4. $\boldsymbol{x}_0^{(n)} = \boldsymbol{0}$
5. ***Inner iteration.***

(a) $\boldsymbol{s}_0 = \left[\mathcal{P}_0(A^{(n)})\right]^{-1}\left(\boldsymbol{b}^{(n)} - A^{(n)}\boldsymbol{x}_0^{(n)}\right)$

(b) $\beta = \|\boldsymbol{s}_0\|_2$, *and* $\boldsymbol{v}_1 = \dfrac{\boldsymbol{s}_0}{\beta}$

(c) For $j = 1, \ldots, r$, *Do*

(d) $\quad \boldsymbol{z}_j = A^{(n)}\boldsymbol{v}_j$

(e) $\quad \boldsymbol{w} = \left[\mathcal{P}_j(A^{(n)})\right]^{-1}\boldsymbol{z}_j.$

(f) $\quad$ *For* $i = 1, \ldots, j$, *Do*

(g) $\quad\quad h_{i,j} = (\boldsymbol{w}, \boldsymbol{v}_i)$

(h) $\quad\quad \boldsymbol{w} = \boldsymbol{w} - h_{i,j}\boldsymbol{v}_i$

(i) $\quad$ *EndDo*

(j) $\quad h_{j+1,j} = \|\boldsymbol{w}\|_2$ *and* $\boldsymbol{v}_{j+1} = \dfrac{\boldsymbol{w}}{h_{j+1,j}}$

(k) EndDo

(l) $V_r = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$, $\bar{H}_r = \{h_{i,j}\}_{1 \le i \le j+1,\, 1 \le j \le r}$

(m) $\boldsymbol{y}_r = \arg\min_{\boldsymbol{y}} \|\beta \boldsymbol{e}_1 - \bar{H}_r \boldsymbol{y}\|_2$ *and* $\boldsymbol{x}^{(n)} = \boldsymbol{x}_0 + V_r \boldsymbol{y}_r$

(n) If satisfied Stop, else set $\boldsymbol{x}_0^{(n)} = \boldsymbol{x}^{(n)}$ *and go to step 5a*

6. $\Delta \boldsymbol{U}^n = \boldsymbol{x}^{(n)}$

7. $\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta \boldsymbol{U}^n$

8. *EndDo*

At steps 5a and 5d of Algorithm III.2 the matrix-vector products $A^{(n)}\mathbf{x}_0^{(n)}$ and $A^{(n)}\mathbf{v}_j$ are performed by using Eq. (9). In the steps 5a and 5e of the algorithm, applying $\left[\mathcal{P}_j(A^{(n)})\right]^{-1}$ corresponds to solving the linear system $A^{(n)}\mathbf{s}_0 = \mathbf{b}^{(n)} - A^{(n)}\mathbf{x}_0^{(n)}$ (respectively, $A^{(n)}\mathbf{w} = \mathbf{z}_j$) by matrix-free unpreconditioned GMRES with a number $n_{Krylov}(j)$ of Krylov vectors.

IV. Nonlinear Preconditioning for the Newton-Krylov Method

As flows get more complicated, numerical methods to solve them become more expensive, both on a storage and on a computational point of view. We already spent some words on stability issues which limit the allowed timestep, stiffness of the matrix, memory limitations which do not allow to store the residual Jacobian matrix, ill-conditioning issues, slow convergence. The larger the number of 'difficulties' in the problem to be solved, the larger the number of choices to be done (which combination of methods leads to the solution in the cheapest way) and the number of parameters to be tuned.

Here we deal with an idea which works as a nonlinear preconditioner for a Newton-Krylov method. We present the idea in two different manners.

IV.A. Fixed-Point Iteration

In this case the method is seen as a fixed-point iteration for a function $\mathcal{F}(\mathbf{U})$, depending on $\mathbf{R}(\mathbf{U})$. In 1997 Wang[21] applied the idea of *Nonlinear GMRES* to a generic CFD code, following an idea given in Ref. 20. Any CFD code which makes use of time relaxation to reach steady state can be expressed as the action of an arbitrary operator $\mathcal{M}$:

$$\mathbf{U}^{k+1} = \mathcal{M}(\mathbf{U}^k). \tag{10}$$

Here $\mathcal{M}$ can be imagined to be any kind of iterative solver, such as time-explicit multigrid scheme or time-implicit Newton-Krylov method. The linear system in the Newton-Krylov method can be preconditioned, for example by means of an ILU(p) decomposition or by Squared Preconditioning, as described in Section III. Equation (10) can be seen as a fixed-point iteration corresponding to the Newton iteration (6) for $\mathbf{R}(\mathbf{U}) = \mathbf{0}$.

By defining $\mathcal{F}(\mathbf{U}) := \mathbf{U} - \mathcal{M}(\mathbf{U})$, solving Eq. (10) is equivalent to finding the roots of $\mathcal{F}$:

$$\mathcal{F}(\mathbf{U}) = \mathbf{0}. \tag{11}$$

(Convergence is obtained when $\mathbf{U}^{k+1} = \mathbf{U}^k$.) System (11) can then be solved by Newton's method:

$$\begin{aligned} \left.\frac{\partial \mathcal{F}}{\partial \mathbf{U}}\right|_k \Delta \mathbf{U}^k &= -\mathcal{F}(\mathbf{U}^k), \\ \mathbf{U}^{k+1} &= \mathbf{U}^k + \Delta \mathbf{U}^k, \end{aligned} \tag{12}$$

or by damped Newton method.

One could point out that this approach does not produce any benefit, because any method used to solve Eqn. (11) approximately can be applied to the original nonlinear system in Eq. (4) as well. However, there is a difference in the convergence behavior. Regarding Eqn. (11), regardless of the method used to solve it, the outer method, the driver, is still Newton's method, and thus quadratic convergence is achievable. Whereas, if the same method is applied to Eqn. (4) directly, that method determines the convergence of the outer loop and the resulting method may converge less rapidly and, in particular, quadratic convergence may be lost.

IV.B. Nonlinear Left-Preconditioning Operator

Here we apply a nonlinear operator to the residual: instead of solving $\mathbf{R}(\mathbf{U}) = \mathbf{0}$ we solve

$$\mathcal{P}^{-1}\left(\mathbf{R}(\mathbf{U})\right) = \mathbf{0}. \tag{13}$$

Let us see the residual $\mathbf{R}$ as an operator:

$$\begin{array}{rlll} \mathbf{R}: & \mathcal{D} \subseteq \mathbb{R}^{N_{DOF}} & \to \mathbb{R}^{N_{DOF}} \\ & \mathbf{U} & \mapsto \mathbf{R}(\mathbf{U}), \end{array} \tag{14}$$

where N_{DOF} is the number of global degrees of freedom. We then introduce a definition for the operator $\mathcal{P}^{-1} : \mathbb{R}^{N_{DOF}} \to \mathbb{R}^{N_{DOF}}$ as follows:

$$\begin{array}{rlll} \mathcal{P}^{-1} \circ \mathbf{R}: & \mathcal{D} \subseteq \mathbb{R}^{N_{DOF}} & \to \mathbb{R}^{N_{DOF}} \\ & \mathbf{U} & \mapsto (\mathcal{P}^{-1} \circ \mathbf{R})(\mathbf{U}) = \mathcal{P}^{-1}\left(\mathbf{R}(\mathbf{U})\right) := \mathbf{U} - \mathcal{M}(\mathbf{U}). \end{array} \tag{15}$$

From the definition follows

$$\frac{\partial\left(\mathcal{P}^{-1}(\mathbf{R})\right)}{\partial \mathbf{U}} = I - \frac{\partial \mathcal{M}}{\partial \mathbf{U}}, \tag{16}$$

Hence Newton's method applied to Eq. (13) gives

$$\begin{aligned} \left(I - \left.\frac{\partial \mathcal{M}}{\partial \mathbf{U}}\right|_k\right) \Delta \mathbf{U}^k &= \mathcal{M}(\mathbf{U}^k) - \mathbf{U}^k, \\ \mathbf{U}^{k+1} &= \mathbf{U}^k + \Delta \mathbf{U}^k, \end{aligned} \tag{17}$$

which corresponds exactly to Eq. (12).

Let us stress here that in this paper we use only matrix-free GMRES to solve (17), because we do not have direct ways to compute the Jacobian matrix $\left.\frac{\partial \mathcal{M}}{\partial \mathbf{U}}\right|_k$. The algorithm is analogous to Algorithm III.2, with $A^{(k)} = I - \left.\frac{\partial \mathcal{M}}{\partial \mathbf{U}}\right|_k$, $\mathbf{b}^{(k)} = \mathcal{M}(\mathbf{U}^k) - \mathbf{U}^k$, and $\mathcal{P}_j^{(k)}$ being given by the identity operator because Squared Preconditioning is not applied here. As regards the choice of the operator $\mathcal{M}$ we have different options. In the past Chan and Jackson[29] chose the SSOR method, while Bijl and Carpenter used the nonlinear multigrid method.[30] We have in our code the option of choosing both matrix-explicit and matrix-free GMRES solvers, as also time-explicit relaxation techniques.

V. Implementation and Numerical Results

We implemented the Nonlinear Preconditioning framework in our code. Comparison of performances is now possible: in fact, the solution possibilities at our disposal are now several and vary on different levels within the solver $\mathcal{M}$ itself. We introduce here the solution methods and the corresponding notations that we will be using.

TE (time-explicit) 3-stage Runge-Kutta (RK3) method is used. Multigrid can be incorporated as convergence accelerator thanks to the Full Approximation Storage algorithm.[31] Parameters to be tuned are:

- CFL number on the fine grid;
- if multigrid is used:
  - number of grids,
  - type of multigrid cycle (saw-tooth or W-cycle),
  - CFL number on the coarsest grid.

TIME (time-implicit matrix-explicit) Matrix-explicit Newton-Krylov method is used. In particular, GMRES method is used to solve the linear system. Newton or damped Newton's method is used: in the latter case one can tune the way the CFL number is increased. Preconditioning for the linear system is performed by ILU decomposition. Parameters to be tuned are:

- if damped Newton's method is used, CFL number;
- relative tolerance and maximum number of iterations as stopping criteria for the GMRES algorithm;
- restart parameter for the GMRES algorithm;
- fill-in level p in the preconditioner ILU(p);

TIMF (time-implicit matrix-free) Matrix-free Newton-Krylov method is used. In particular, GMRES method is used to solve the linear system. Newton or damped Newton's method is used: in the latter case one can tune the way the CFL number is increased. Preconditioning for the linear system is performed by Squared Preconditioning. Parameters to be tuned are:

- if damped Newton's method is used, CFL number;
- relative tolerance and maximum number of iterations as stopping criteria for the GMRES algorithm;
- restart parameter for the GMRES algorithm;
- number of Krylov methods in the squared preconditioner;

TENP (TE as nonlinear preconditioner) Additional parameters to be tuned are:

- relative tolerance and maximum number of iterations as stopping criteria for the GMRES algorithm;
- restart parameter for the GMRES algorithm;
- number of calls to the flow solver to which $\mathcal{M}$ corresponds.

TIMENP (TIME as nonlinear preconditioner) See TENP. ILU(p) preconditioning for the linear systems arising in the matrix-explicit Newton-Krylov method to solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$ can be applied ('w/ PC') or not ('no PC').

TIMFNP (TIMF as nonlinear preconditioner) See TENP. Squared Preconditioning for the linear systems arising in the matrix-free Newton-Krylov method to solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$ can be applied ('w/ SP') or not ('no PC').

This large flexibility in the combination of the elements of the solver is essential to develop best-practice strategies for different cases.

In the implementation of the time-implicit relaxation we let the library PETSc[32] drive the GMRES algorithm, to solve both Eq. (4) (when solved time-implicitly) and Eq. (11). The default convergence test in PETSc is based on the l_2-norm of the residual. If we refer to the solution of a linear system $A\mathbf{x} = \mathbf{b}$, we have convergence at n-th iteration[a] if

$$\frac{\|\mathbf{b} - A\mathbf{x}_n\|_2}{\|\mathbf{b}\|_2} \leq rtol. \tag{18}$$

So one of the parameters to be fixed is the relative tolerance $rtol$. We also have to fix M, the maximum number of iterations (i.e., computed Krylov vectors) within the GMRES algorithm, and the number of iterations after which the GMRES algorithm is restarted (i.e., number of stored Krylov vectors). When solving Eq. (11), we also have to choose N, the number of calls to the flow solver to which the operator $\mathcal{M}$ corresponds. Typically $N = 1$, because of computational costs. Let us recall that the fixed-point iteration is always performed as a matrix-free **unpreconditioned** GMRES.

V.A. Specral Difference Discretization for Flow over a Smooth Bump

We first consider an SD discretization. In this section we focus on the inviscid flow over a smooth bump (see Fig. 1) at free stream Mach number $M_\infty = 0.3$. We consider here the flow over a bump of very slight slope. The reason why we consider such a low bump is that we want also to perform computations on a stretched grid. Since our code does not allow for curved elements, taking a 'stretched' bump is the only way to also obtain a stretched grid valid for computations. These are performed using the SD scheme on a regular triangular mesh consisting of 1482 elements. In our simulations we choose order of accuracy $m = 3$.

[a]Here a simplified version of the convergence test has been given. See Ref. 32 for the complete test.

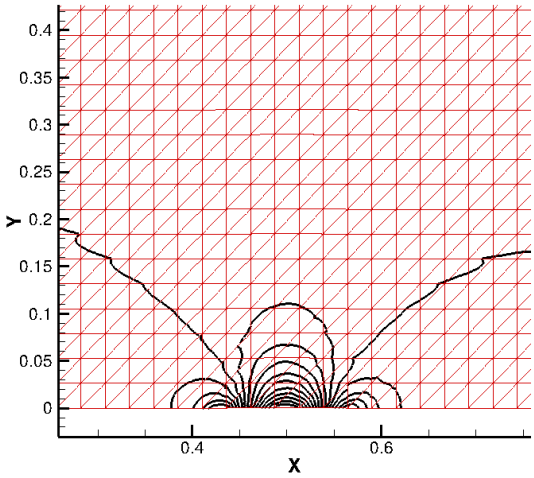

Figure 1. Smooth bump: Detail of the computational mesh (1482 elements) and Mach contour lines. The height of the bump is so slight (the crest is about $y = 0.00069$ at $x = 0.5$) that it can hardly be seen in the picture.

V.A.1. Effect of Nonlinear Preconditioning on GMRES

First of all we consider the effect of the nonlinear preconditioner on the matrix-free Newton-Krylov method. In Fig. 2 the convergence history of different solvers for the first linear system to be solved is shown. We iterate until the norm of the residual has dropped by 3 orders of magnitude. The linear system solved is $\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_0 \Delta\mathbf{U}^0 = -\mathbf{R}(\mathbf{U}^0)$ in the TIMF case, and $\left.\frac{\partial \mathcal{F}}{\partial \mathbf{U}}\right|_0 \Delta\mathbf{U}^0 = -\mathcal{F}(\mathbf{U}^0)$ for the other test cases. The black solid line corresponds to matrix-free GMRES without preconditioning; in fact, it shows the slowest convergence. The other two solid lines correspond to matrix-free GMRES with Squared Preconditioning, while the dotted lines represent test cases where the approach presented in Section IV is applied. It can be observed that, analogously to Squared Preconditioning, this approach accelerates the convergence rate.

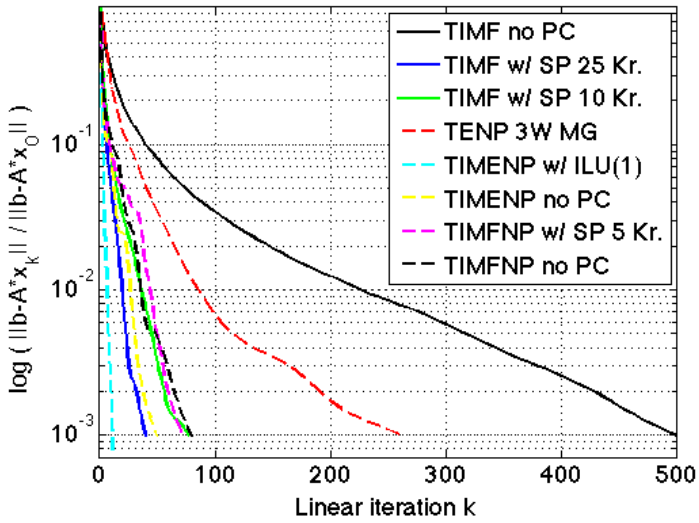

Figure 2. Linear residual vs. Linear iterations for the first linear system to be solved. Residuals are, independently for each test case, normalized with respect to the first residual, in order to highlight the convergence behavior. PC stands for preconditioning, SP for Squared Preconditioning and Kr. for the number of Krylov vectors used in the Squared Preconditioning, 3W MG stands for W-cycle multigrid on 3 grids.

V.A.2. Parameter Selection for Solution Schemes

As we observed at the beginning of this section, the different solution schemes studied in this work contain a number of parameters which must be chosen optimally in order to fully exploit the potentiality of each scheme. Multiple levels of iteration nesting are involved, hence the selection of these parameters requires extreme care. Let us consider the test case TIMENP w/ PC. Different parameters have to be tuned for the Newton strategy and the GMRES solver on the two nested levels. We restrict here to 3 of these parameters, namely the number of Krylov vectors in the two GMRES strategies (we do not use restart and we use the maximum number of iterations as stopping criterion) and the level of fill-in in the ILU(p) preconditioning. We first fix the maximum number of Krylov vectors to be 5 in the GMRES method solving Eq. (12) and to be 10 in the GMRES in Eq. (7). Fig. 3 shows the behavior for different values of fill-in in the ILU preconditioning. As expected, the best preconditioning effect is given by high levels of fill-in. If we take

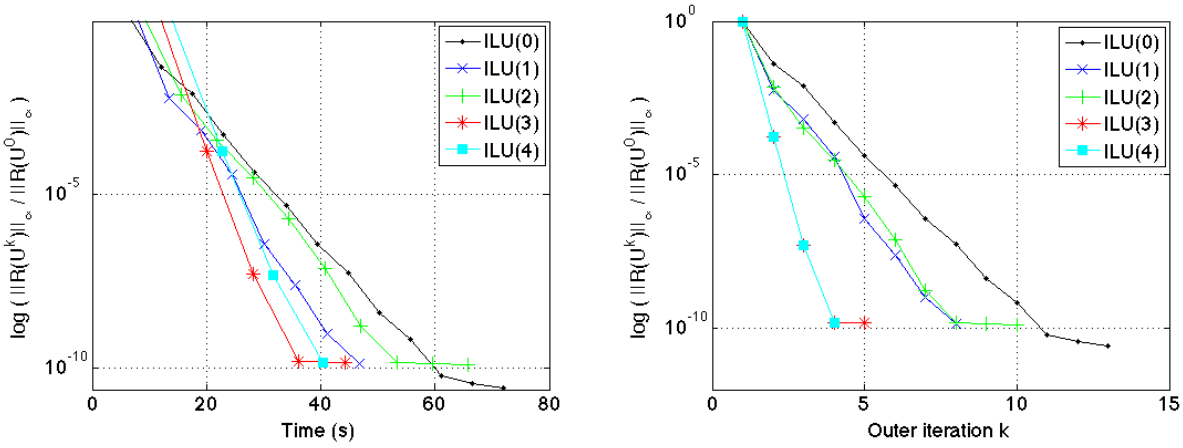

Figure 3. Effect of changing the level of fill-in in the test case TIMENP. Left: reduction of the residual against runtime. Right: reduction of the residual against the number of Newton iterations.

into account runtime as well, ILU(3) is the computationally optimal preconditioner for the problem under consideration.

Having fixed the level of fill-in to be 3, we now want to determine the best number of search directions in the GMRES method for the Newton-Krylov method in Eq. (12). Fig. 4 shows the behavior for different numbers of search directions. On the one side, increasing the number of search directions provides a more

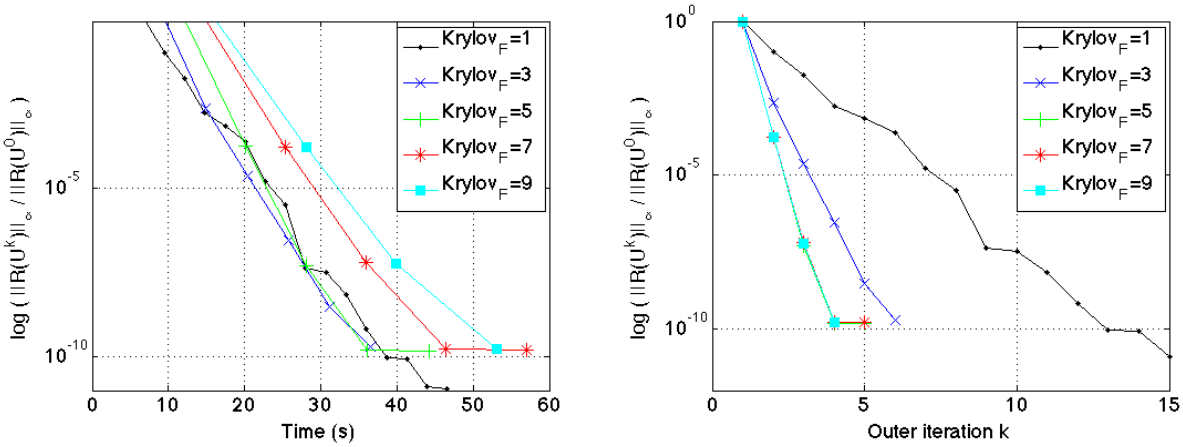

Figure 4. Effect of changing the number of search directions in the GMRES method to solve $\mathcal{F}(\mathbf{U}) = \mathbf{0}$ in the test case TIMENP. Left: reduction of the residual against runtime. Right: reduction of the residual against the number of Newton iterations.

accurate solution to the linear system, which arises at every step of the Newton's method. Therefore, the Newton's method approaches quadratic convergence as the number of search directions, i.e., Krylov vectors, is increased. On the other side, a larger number of search directions means computational overhead. As a

compromise we fix the number of search directions in the Newton-Krylov method of Eq. (12) to be 5.

Finally, we study the number of search directions in the GMRES method in Eq. (7). Fig. 5 shows the behavior for different numbers of search directions. The best choice seems to be 10 Krylov vectors. In

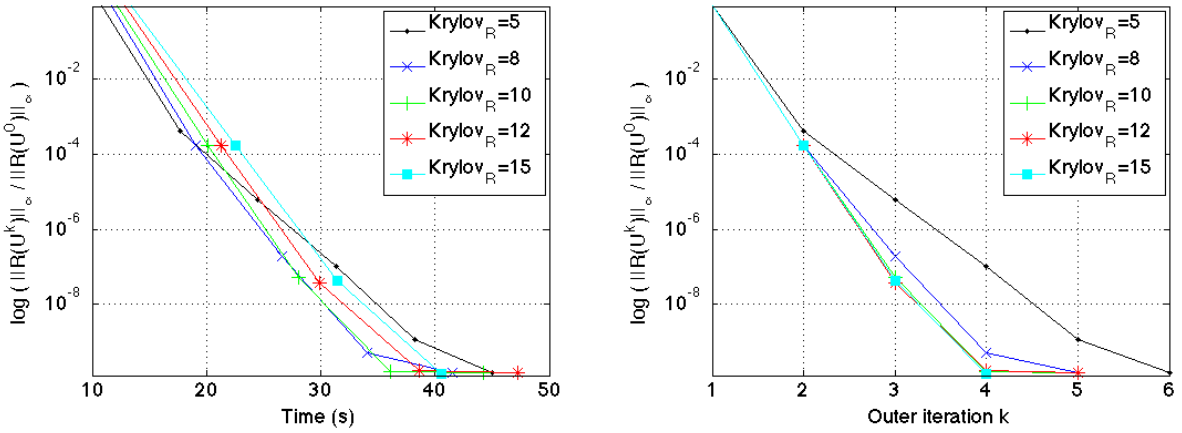

Figure 5. Effect of changing the number of search directions in the GMRES method to solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$ in the test case TIMENP. Left: reduction of the residual against runtime. Right: reduction of the residual against the number of Newton iterations.

fact, in order to reach the steady-state solution, solving accurately the linear systems in the Newton-Krylov method is not the best option. Such linear systems should be solved exactly in the context of an exact Newton's method in order to reach the quadratic convergence, but here we are using a damped Newton method with finite CFL numbers. We have observed that in the time needed by Newton's method to reach the ball of quadratic convergence, the damped-Newton strategy has usually already converged to levels of accuracy commonly required by engineering applications. This is the reason why low values for the relative tolerance and for the maximum number of search directions give the most competitive computational times.

This parametric study has been introduced as an example of the complex kind of tuning that is required for these methods. Here the goal was solely to minimize the runtime required to converge the system of equations $\mathcal{F}(\mathbf{U}) = \mathbf{0}$ to a given tolerance level. However, one could conduct similar studies with other goal in minds, such as the convergence of integral quantities, and obtaining a different outcome. Material constraints such as memory can play a role too. For instance, we started from these results to tune the parameters for the simulation TIMENP w/ PC (see Table 3). There, however, we tuned the whole set of available parameters (including the relative tolerance, for example) and we aimed at the best possible compromise between fast convergence and memory storage. This explains why there we take $p = 1$ level of fill-in instead of $p = 3$.

Parametric studies analogous to the previous one deepened our knowledge of the approach. As concerns multigrid techniques, when the TENP case is chosen, multigrid accelerates the convergence with respect to $\mathcal{M}$ being simple multistage Runge-Kutta without multigrid. However, in our experience, TENP showed to be the slowest converging technique in most of the cases and using multigrid did not change this fact. In order to get some benefit from multigrid, we tried choosing $\mathcal{M}$ to be a few iterations (2 to 10) of multigrid Runge-Kutta relaxation. Even though this reduced the number of nonlinear iterations necessary to converge, the computational time increased till it often became larger than the computational time without multigrid. In our experience, the best strategy for using multigrid is to choose $\mathcal{M}$ to be 2 multistage Runge-Kutta iterations with multigrid. Section V.B.1 focuses on multigrid. As regards TIMENP and TIMFNP, we tried to apply the *Hybrid Multilevel method*,[33] but that did not accelerate the convergence.

V.A.3. Effect of Nonlinear Preconditioning on Nonlinear Iterations

Let us now give a global overview to the most meaningful test cases run for the bump of Fig. 1. The flow here considered is quite simple, hence we expect the classical solution strategies, built without Nonlinear Preconditioning, to achieve better performance. In Fig. 6 and Fig. 7 the most interesting test cases are plotted. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning. We look at the results from 3 different perspectives. Fig. 6 (left) shows time-to-residual-reduction perspective. In Fig. 7 (left) we plotted the time-to-C_L-convergence aspect.

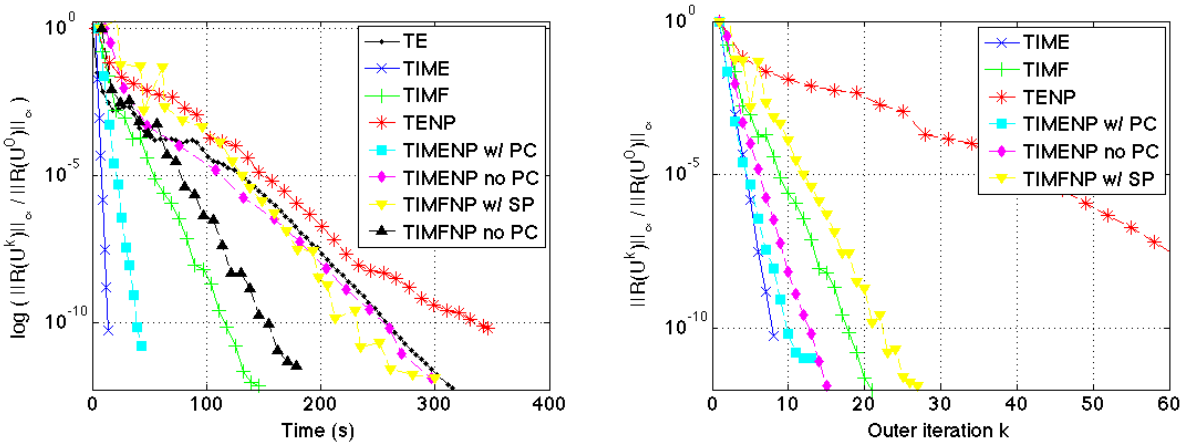

Figure 6. Smooth bump with SD discretization: residual reduction. $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ **vs. time. Right:** $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ **vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.**

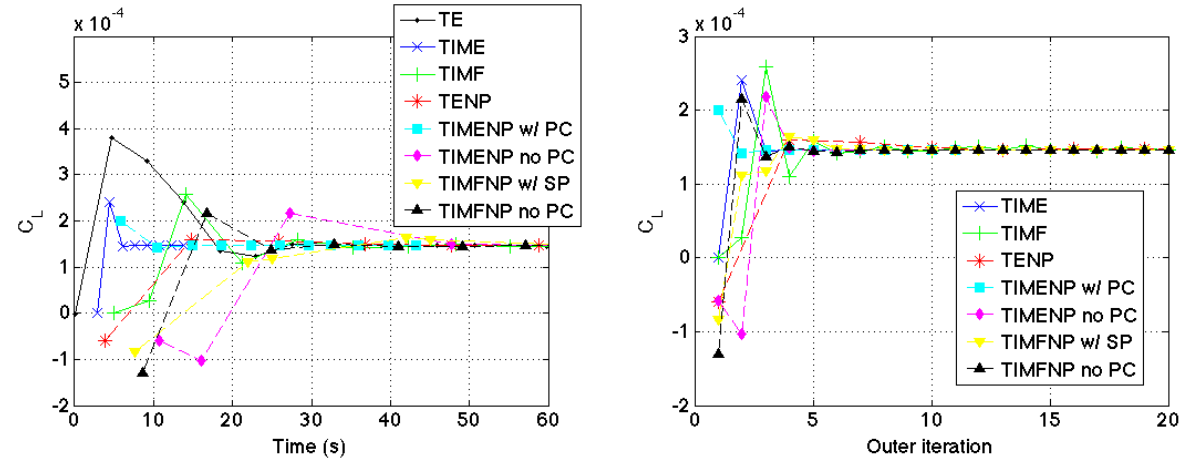

Figure 7. Smooth bump with SD discretization: C_L **convergence. Left: Lift coefficient** C_L **vs. time. Right: Lift coefficient** C_L **vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.**

Memory requirement offers a third point of view (see Table 1). For all the computations we gave an

Test case	Memory (MB)
TE	1.0
TIME	100.2
TIMF	16.9
TENP	13.2
TIMENP w/ PC	95.6
TIMENP no PC	59.9
TIMFNP w/ SP	16.9
TIMFNP no PC	15.3

Table 1. Estimates of memory requirement for flow over a smooth bump with SD discretization.

estimate of the memory needed to store the vectors and (in matrix-explicit computations) the matrices having dimension proportional to

$$N_{DOF} = 4 \times N_m \times N_e, \tag{19}$$

where N_m is the number of solution nodes in each element for order of accuracy m and N_e is the number of elements in the triangulation. Table 2 contains the estimates that we used. We indicate by $\#Restart_{\mathbf{R}}$ and $\#Restart_{\mathcal{F}}$ the restart parameter for the GMRES algorithm in the Newton-Krylov method in Eq. (12) and in Eq. (7), repectively. $\#Krylov_{SP}$ is the number of Krylov vectors used for the Squared Preconditioning. The storage needed for the system matrix and the ILU preconditioner is deduced thanks to the PETSc $-log\_summary$ profiling option.[32] All the computations have been executed in double precision, i.e., 8 bytes

Type of method	Entries to be stored
RK3 w/o MG	$3 \times N_{DOF}$
RK3 w/ MG on L grids	$3 \times \sum_{\ell=0}^{L} N_{DOF}^{(\ell)}$
ME GMRES	PETSc matrices storage$+\#Restart_{\mathbf{R}} \times N_{DOF}$
MF GMRES	$(\#Restart_{\mathbf{R}} + \#Krylov_{SP}) \times N_{DOF}$
if w/ Nonlinear Preconditioning	add $\#Restart_{\mathcal{F}} \times N_{DOF}$

Table 2. Estimates of the number of entries to be stored for vectors and matrices whose dimension is proportional to N_{DOF}.

of memory have been counted for each entry to be stored.

Memory limitations can be an important factor in the choice of the solution method. Here the fastest converging method, i.e., TIME, is also the one which needs more memory. Table 3 lists at great length the parameters for the simulations that we ran. For each single computation the parameters have been tuned in order to obtain a good compromise between fast convergence and storage requirement.

As suggested, the flow considered here is very simple, and therefore Nonlinear Preconditioning brings no advantage if compared to 'simple' flow solvers (TE, TIME and TIMF). In particular, TIME is the best-practice strategy if one is interested both in residual reduction and in the convergence of the integral quantities. However, if memory limitations are present TIMF or TE would be the solution methods of choice.

V.B. Discontinuous Galerkin Discretization for Flow over a Smooth Bump

As we already mentioned, the SD discretization has proved to be weakly unstable in some cases. Hence, we implemented a DG discretization as well. We considere here the same test case as in Section V.C. However, we now use a DG discretization. We consider here the flow of Section V.A. In particular, we use a DG discretization for the inviscid flow over the bump represented in Fig. 1.

V.B.1. Effect of Multigrid on the Nonlinear Preconditioning Approach

We focus here on the test case TENP with different settings. We assigned different values to N, i.e., the number of 3-stage Runge-Kutta steps in the solver $\mathcal{M}$. We ran simulations with multigrid on 3 grids

Test	solve $\mathcal{F}(\mathbf{U}) = \mathbf{0}$				solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$							
	$rtol_{\mathcal{F}}$	$M_{\mathcal{F}}$	$Rest_{\mathcal{F}}$	N	Relax-ation	Sto-rage	$rtol_{\mathbf{R}}$	$M_{\mathbf{R}}$	$Rest_{\mathbf{R}}$	PC	MG	CFL
TE	-				Shu-RK3	-	-	-	-	-	3 gr. s.t.	2.7
TIME	-				GMRES	ME	1e-2	30	30	ILU(1)	-	1e5
TIMF	-				GMRES	MF	1e-2	40	40	SP w/ 15 Kr.	-	1e5
TENP	1e-2	40	40	2	Shu-RK3	-	-	-	-	-	3 gr. s.t.	2.7
TIMENP w/ PC	1e-1	5	5	1	GMRES	ME	1e-1	10	10	ILU(1)	-	1000
TIMENP no PC	1e-1	30	30	1	GMRES	ME	1e-1	90	30	-	-	1000
TIMFNP w/ SP	1e-1	30	30	1	GMRES	MF	1e-1	20	20	SP w/ 5 Kr.	-	1000
TIMFNP no PC	1e-2	20	20	1	GMRES	MF	1e-2	30	30	-	-	1000

Table 3. Simulations of the flow over a smooth bump, SD discretization. In a GMRES algorithm, $rtol$ is the relative tolerance used as stopping criterion, M the maximum number of iterations, and $Rest$ is the number of iterations after which the GMRES algorithm is restarted. N is the number of calls to the flow solver to which the operator $\mathcal{M}$ corresponds. As regards the storage of the matrix of the system, ME stands for matrix-explicit and MF for matrix-free. ILU(p) indicates the Incomplete LU factorization with p-level of fill-in. SP indicates Squared Preconditioning with the corresponding number of search directions Kr. Shu-RK3 indicates Shu's 3-stage Runge-Kutta scheme.[6] In the multigrid procedure s.t. stands for saw-tooth cycle.

(respectively, 1482, 342, and 72 elements on each grid) and we tried saw-tooth cycle and W-cycle. Fig. 8 and Fig. 9 report the results.

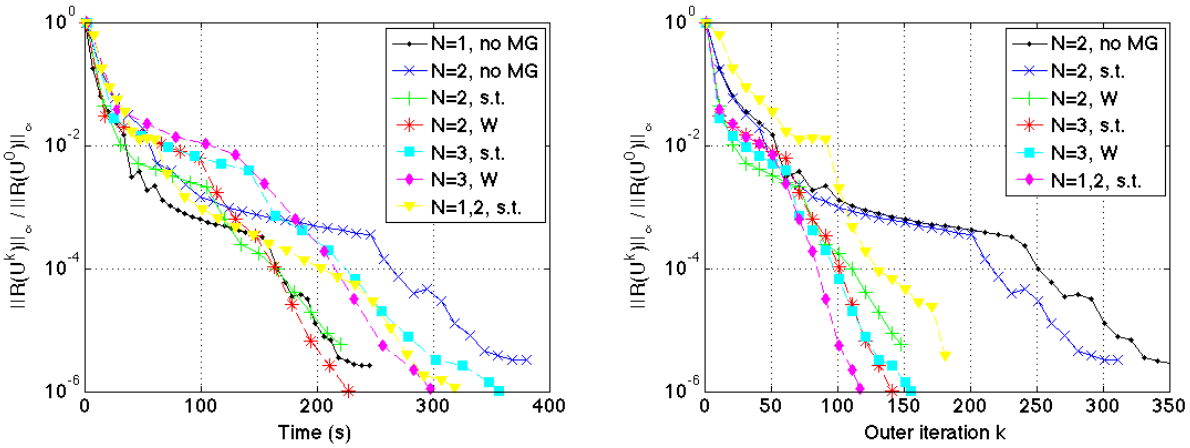

Figure 8. Smooth bump with DG discretization: residual reduction. Left: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. time. Right: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. nonlinear iterations in the Newton-Krylov method. N is the number of smoothing steps in the solver $\mathcal{M}$. MG stands for multigrid, s.t. denotes the saw-tooth cycle and W the W-cycle.

The residual decreases in a similar way for the simulation with $\mathcal{M}$ being $N = 1$ step of 3-stage RK method without multigrid and for the simulation with $\mathcal{M}$ being $N = 2$ steps of 3-stage RK method with multigrid. This is the reason why we also tried a simulation where we started running the first kind of simulation and we then switched to the second kind of simulation when the residual had dropped of 2 orders of magnitude (see 'N=1,2, s.t.' in the figures). Larger values of N penalize time performances, because one single Newton iteration becomes more expensive. At the same time, large values of N accelerate the convergence of the lift coefficient.

V.C. Spectral Difference Discretization for Flow over a Smooth Bump with Stretched Mesh

We want now to test the new approach on a more difficult test case. Here we adopt again the SD discretization. One known difficulty is represented by stiff problems, arising for example when highly-stretched cells

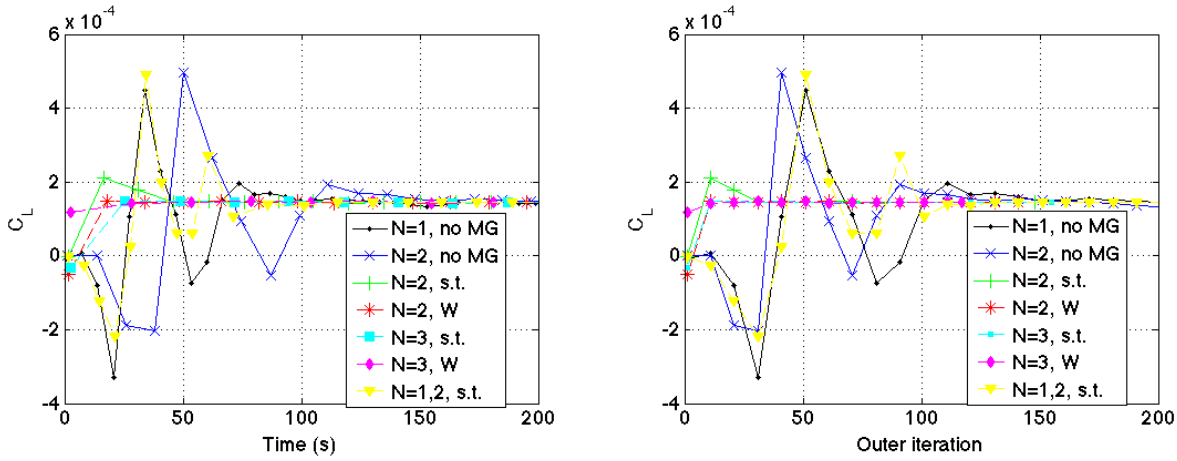

Figure 9. Smooth bump with DG discretization: C_L convergence. Left: Lift coefficient C_L vs. time. Right: Lift coefficient C_L vs. nonlinear iterations in the Newton-Krylov method. N is the number of smoothing steps in the solver $\mathcal{M}$. MG stands for multigrid, s.t. denotes the saw-tooth cycle and W the W-cycle.

are present in the grid. For this reason we consider a highly-stretched mesh (see Fig. 10). The inviscid flow

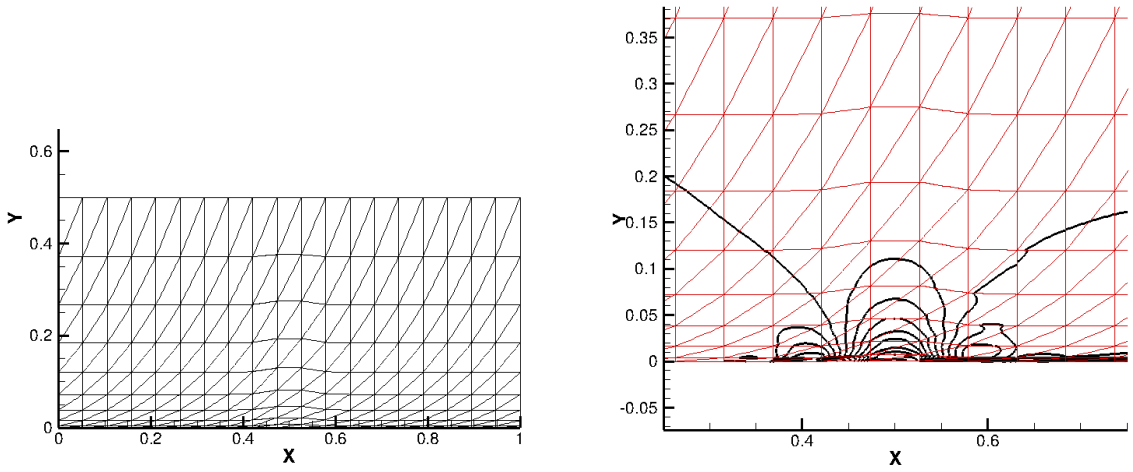

Figure 10. Flow over a smooth bump: stretched computational mesh (342 elements) on the left; detail of the bump and Mach contour lines on the right ($m = 3$). The height of the bump is so slight (the crest is at about $y = 0.00069$) that it can hardly be seen in the picture.

has free-stream Mach number $M_\infty = 0.3$. In this case as well we plot the behavior of the most relevant test cases in Fig. 11 and Fig. 12. Simple time-explicit test cases have here not been reported because too slow to converge. Table 4 reports storage requirement. Details on the choice of the parameters are reported in Table 5.

Here we can basically distinguish between matrix-explicit methods (TIME and TIMENP w/ PC) and matrix-free methods (TIMF, TENP and TIMFNP w/ SP). In both cases the Nonlinear Preconditioning framework performs better. In fact, when considering the reduction of the residual norm, TIMENP w/ PC is the fastest among the matrix-explicit methods and TENP is the fastest among the matrix-free ones.

V.C.1. Higher Order of Accuracy

Given the previous results, we choose then to perform the same computations for order of accuracy $m = 4$ as well. The results, shown in Fig. 13, confirm the previous outcome.

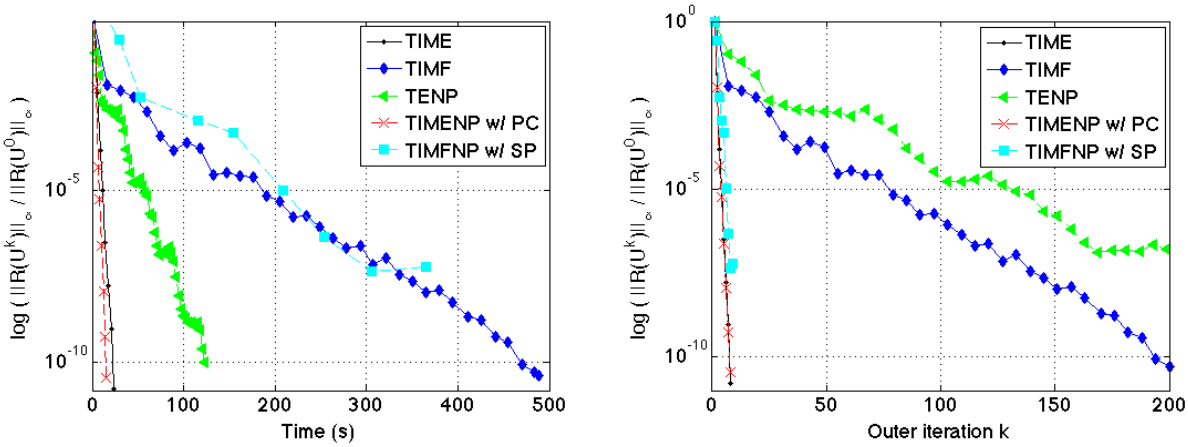

Figure 11. Smooth bump with stretched mesh and SD discretiazione: residual reduction. Left: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. time. Right: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.

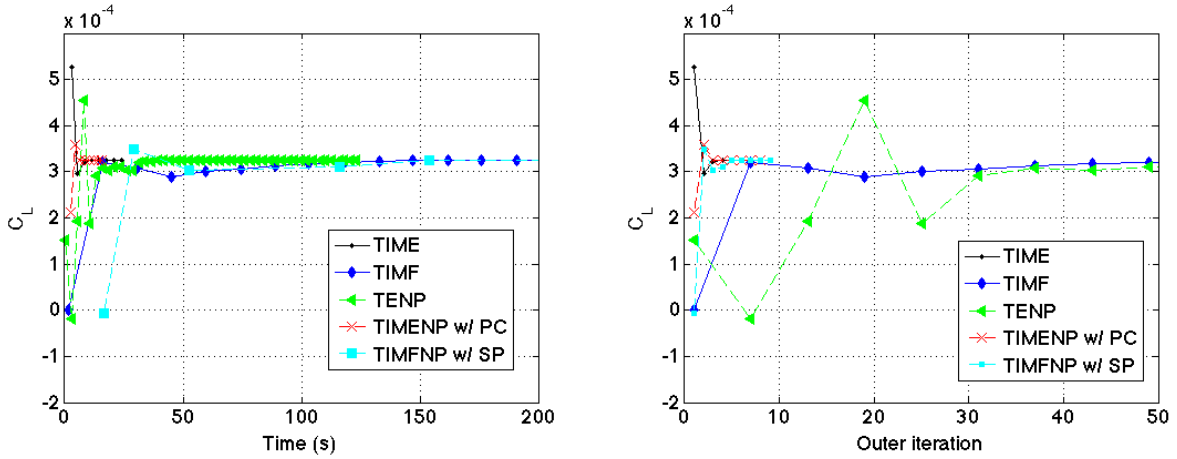

Figure 12. Smooth bump with stretched mesh and SD discretization: C_L convergence. Left: Lift coefficient C_L vs. time. Right: Lift coefficient C_L vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.

Test case	Memory (MB)
TIME	22.9
TIMF	4.6
TENP	3.3
TIMENP w/ PC	21.8
TIMFNP w/ SP	6.5

Table 4. Estimates of memory requirement for flow over a smooth bump with stretched mesh and SD discretization.

Test	solve $\mathcal{F}(\mathbf{U}) = \mathbf{0}$				solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$							
	$rtol_{\mathcal{F}}$	$M_{\mathcal{F}}$	$Rest_{\mathcal{F}}$	N	Relax-ation	Sto-rage	$rtol_{\mathbf{R}}$	$M_{\mathbf{R}}$	$Rest_{\mathbf{R}}$	PC	MG	CFL
TIME	-				GMRES ($fr = 2$)	ME	1e-4	150	30	ILU(1)	-	1e4
TIMF	-				GMRES	MF	1e-2	50	50	SP w/ 10 Kr.	-	550
TENP	1e-2	40	40	1	Shu-RK3	-	-	-	-	-	-	2.5
TIMENP w/ PC	1e-2	5	5	1	GMRES	ME	1e-2	10	10	ILU(1)	-	1000
TIMFNP w/ SP	1e-2	5	5	1	GMRES	MF	1e-2	150	50	SP w/ 30 Kr.	-	1000

Table 5. Simulations of the flow over a smooth bump, with stretched mesh and SD discretization. In a GMRES algorithm, $rtol$ is the relative tolerance used as stopping criterion, M the maximum number of iterations, and $ResT$ is the number of iterations after which the GMRES algorithm is restarted. N is the number of calls to the flow solver to which the operator $\mathcal{M}$ corresponds. As regards the storage of the matrix of the system, ME stands for matrix-explicit and MF for matrix-free. fr indicates for how many linear iterations the stored matrix is frozen. ILU(p) indicates the Incomplete LU factorization with p-level of fill-in. SP indicates Squared Preconditioning with the corresponding number of search directions Kr. Shu-RK3 indicates Shu's 3-stage Runge-Kutta scheme.[6] In the multigrid procedure s.t. stands for saw-tooth cycle.

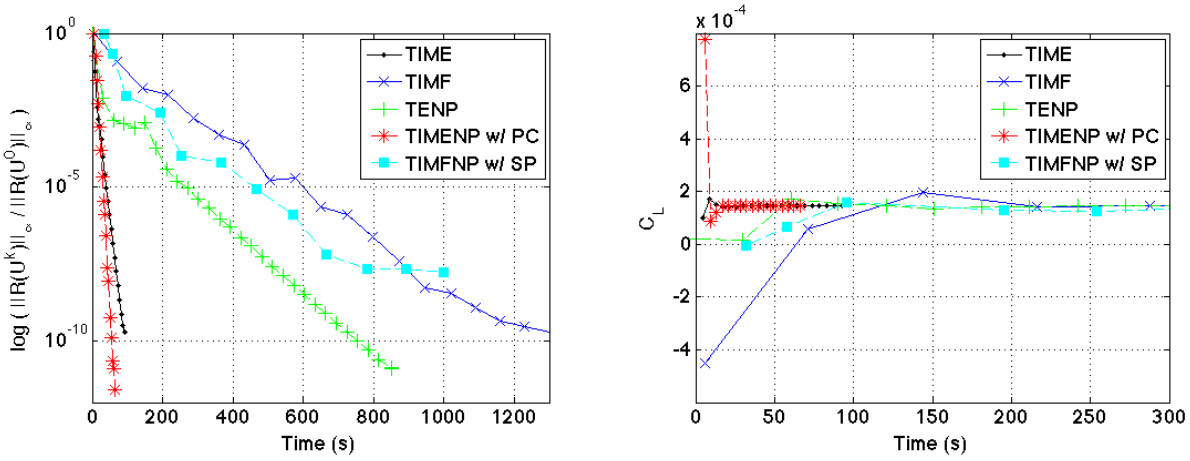

Figure 13. Smooth bump with stretched mesh and Sd discretization, order of accuracy $m = 4$. Left: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. time. Right: Lift coefficient C_L vs. time. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.

V.D. Discontinuous Galerkin Discretization for Flow over a Smooth Bump with Stretched Mesh

We considere here the same test case as in Section V.C. However, we now use the DG discretization. The algorithm implemented for the DG discretization differs from the SD algorithm in two details of the Newton-Krylov algorithm: the CFL number and the relative tolerance can now increase adaptively with increasing nonlinear iterations. Results are shown in Fig. 14 and in Fig. 15. Table 6 summarizes the parameters

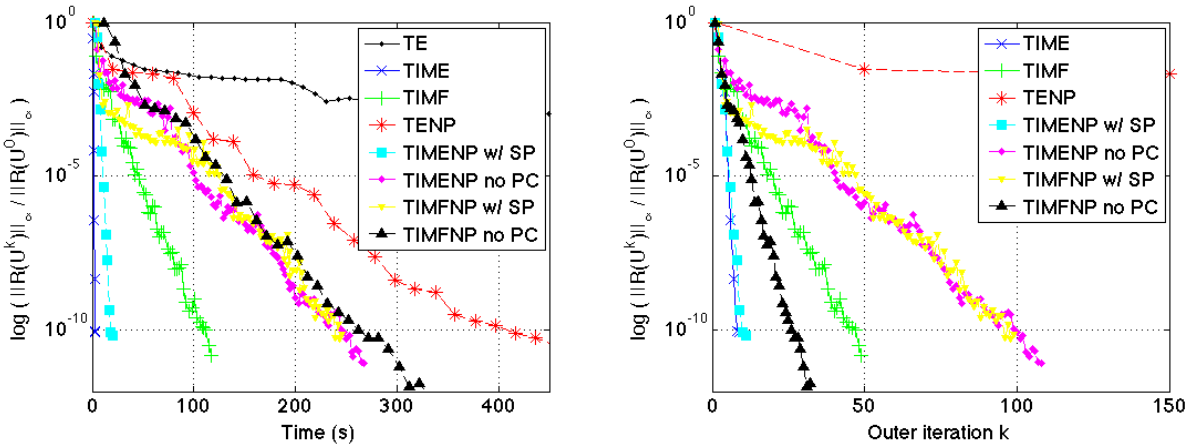

Figure 14. Smooth bump with stretched mesh and DG discretization: residual reduction. Left: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. time. Right: $\frac{\|\mathbf{R}(\mathbf{U}^n)\|_\infty}{\|\mathbf{R}(\mathbf{U}^0)\|_\infty}$ vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.

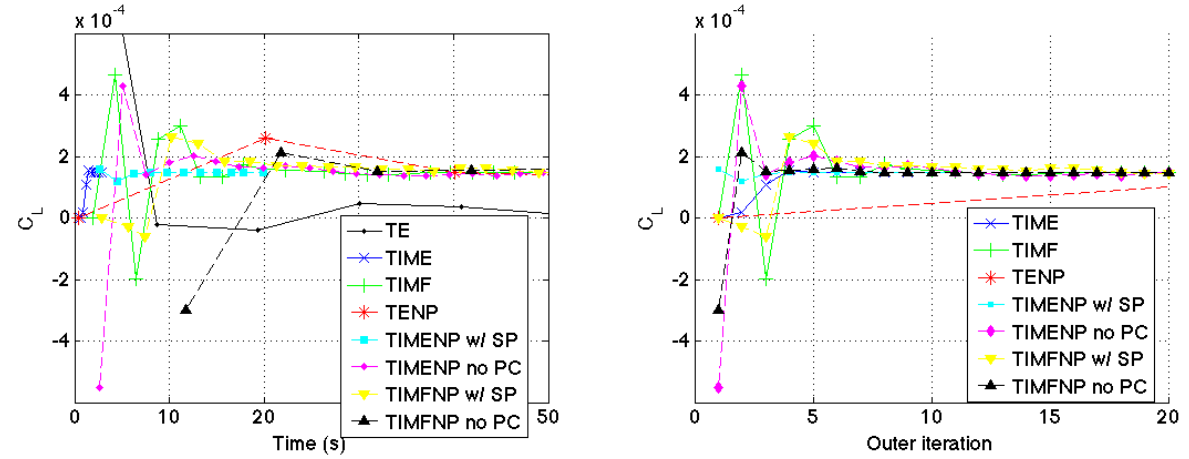

Figure 15. Smooth bump with stretched mesh and DG discretization: C_L convergence. Left: Lift coefficient C_L vs. time. Right: Lift coefficient C_L vs. nonlinear iterations in the Newton-Krylov method. PC stands for preconditioning, SP stands for Squared Preconditioning. Solid (respectively, dotted) lines denote test cases without (respectively, with) Nonlinear Preconditioning.

relative to the test cases that we ran.

Here the approaches without Nonlinear Preconditioning win. We observed the same for higher orders of accuracy ($m = 4$ and $m = 5$). In particular, focusing on the matrix-explicit approaches, we observed that the modified algorithm implemented for the DG discretization accelerates the convergence of the residual in the TIME test case of a factor of about 8 (respectively, 5) for $m = 3$ (respectively, $m = 4$) with respect to the algorithm for the SD discretization. At the same time the test case TIMENP w/ PC is accelerated only of a factor 1 or 2, so that TIME overperforms TIMENP w/ PC. Analogous considerations can be made in the matrix-free context.

Test	solve $\mathcal{F}(\mathbf{U}) = \mathbf{0}$				solve $\mathbf{R}(\mathbf{U}) = \mathbf{0}$							
	$rtol_{\mathcal{F}}$	$M_{\mathcal{F}}$	$Rest_{\mathcal{F}}$	N	Relaxation	Storage	$rtol_{\mathbf{R}}$	$M_{\mathbf{R}}$	$Rest_{\mathbf{R}}$	PC	MG	CFL
TE	-				Shu-RK3	-	-	-	-	-	-	2.2
TIME	-				GMRES	ME	1e-2	20	20	ILU(2)	-	adap
TIMF	-				GMRES ($\varepsilon_{rel} = 10^{-3}$)	MF	1e-2	40	40	SP w/ 20 Kr.	-	adap
TENP	1e-2	40	40	1	Shu-RK3	-	-	-	-	-	-	1.8
TIMENP w/ PC	1e-2	40	40	1	GMRES	ME	1e-2	20	20	ILU(2)	-	1000
TIMENP no PC	1e-2	40	40	1	GMRES	ME	1e-2	20	20	-	-	1000
TIMFNP w/ SP	1e-1	30	30	1	GMRES	MF	1e-1	30	30	SP w/ 20 Kr.	-	1000
TIMFNP no PC	1e-2 ($\varepsilon_{rel} = 10^{-3}$)	40	40	1	GMRES ($\varepsilon_{rel} = 10^{-3}$)	MF	1e-2	20	20	-	-	1000

Table 6. Simulations of the flow over a smooth bump with stretched mesh, DG discretization. In a GMRES algorithm, $rtol$ is the relative tolerance used as stopping criterion, M the maximum number of iterations, and $Rest$ is the number of iterations after which the GMRES algorithm is restarted. N is the number of calls to the flow solver to which the operator $\mathcal{M}$ corresponds. As regards the storage of the matrix of the system, ME stands for matrix-explicit and MF for matrix-free. ε_{rel} is the parameter in the matrix-free approximation of Eq. (9). ILU(p) indicates the Incomplete LU factorization with p-level of fill-in. SP indicates Squared Preconditioning with the corresponding number of search directions Kr. Shu-RK3 indicates Shu's 3-stage Runge-Kutta scheme.[6] In the multigrid procedure s.t. stands for saw-tooth cycle. $adap$ means that we used an adaptive CFL number, increasing for decreasing residual norm.

VI. Conclusion

In this paper we merged two ideas: the matrix-free Squared Preconditioning introduced by May *et al.* in Ref. 22 and the Nonlinear GMRES algorithm implemented as an 'envelope' to a flow solver, proposed by Wigton *et al.* in Ref. 20 and resumed by Wang in Ref. 21. As a result, we implemented a matrix-free Newton-Krylov method, in which our flow solver is nested. In this way, the flow solver works as a 'preconditioner' for the Newton-Krylov driver and introduces more freedom in the process of tuning the best-practice strategy for solving a flow.

Numerical tests explored different aspects, such as parametric studies and the role played by multigrid in this approach. We solved inviscid flows along a smooth bump and we also considered a stretched grid as example of a stiff problem. We considered both SD and DG discretizations. For the stretched grid on SD discretization the new approach outperformed the other methods. We however believe that the test cases here considered are yet not representative of what we think could be the full potential of the presented approach, which is supposed to emerge in the solution of more complex flows.

Other aspects could be explored. The parameter ε in the matrix-free approximation of Eq. (9) seems to play a non-negligible effect on the convergence of the method. Different choices have been suggested in the literature and could be applied on the matrix-free approximation of the Jacobian matrix both of $\mathbf{R}$ and $\mathcal{F}$. Moreover, here we only considered left nonlinear preconditioning and Birken *et al.* suggested right nonlinear preconditioning to be potentially more favorable.[34] Its implementation in our framework would offer the chance to verify this conjecture.

Acknowledgments

Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111, and by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA8655-08-1-3060, is gratefully acknowledged. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

F. I. thanks professor Wang's group at Iowa State University, in particular T. Haga, and M. Bachmayr at RWTH Aachen University for the interesting exchange of ideas, which contributed to the development of

this work.

References

[1]Cockburn, B. and Shu, C.-W., "The Runge-Kutta Local Projection P^1-Discontinuous-Galerkin Finite Element Method for Scalar Conservation Laws," *IMA Preprint Series*, Vol. 388, January 1988.

[2]Cockburn, B. and Shu, C.-W., "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework," *Math. Comput.*, Vol. 52, No. 186, 1989, pp. 411–435.

[3]Cockburn, B., Lin, S.-Y., and Shu, C.-W., "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws III: One-Dimensional Systems," *J. Comput. Phys.*, Vol. 84, 1989, pp. 90–113.

[4]Cockburn, B., Hou, S., and Shu, C.-W., "The Runge-Kutta Discontinuous Galerkin Finite Element Method for Conservation Laws IV: the Multidimensional Case," *Math. Comput.*, Vol. 54, No. 190, 1990, pp. 545–581.

[5]Cockburn, B. and Shu, C.-W., "The Runge-Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems," *J. Comput. Phys.*, Vol. 141, 1998, pp. 199–244.

[6]Shu, C.-W. and Osher, S., "Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes," *J. Comput. Phys.*, Vol. 77, 1988, pp. 439–471.

[7]Wang, Z. J., "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids I: Basic Formulation," *J. Comput. Phys.*, Vol. 178, 2002, pp. 210–251.

[8]Wang, Z. J. and Liu, Y., "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids II: Extension to Two-Dimensional Scalar Equation," *J. Comput. Phys.*, Vol. 179, 2002, pp. 665–697.

[9]Wang, Z. J., Zhang, L., and Liu, Y., "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids IV: Extension to Two-Dimensional Systems," *J. Comput. Phys.*, Vol. 194, 2004, pp. 716–741.

[10]Liu, Y., Vinokur, M., and Wang, Z. J., "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids V: Extension to Three-Dimensional Systems," *J. Comput. Phys.*, Vol. 212, 2006, pp. 454–472.

[11]May, G., "On the Connection Between the Spectral Difference Method and the Discontinuous Galerkin Method," *Comm. Comp. Phys. (accepted for publication)*, Tech. Rep. AICES-2008-11, http://www.aices.rwth-aachen.de/research/preprints.

[12]Liu, Y., Vinokur, M., and Wang, Z. J., "Discontinuous Spectral Difference Method for Conservation Laws on Unstructured Grids," *Computational Fluid Dynamics 2004*, Springer Berlin Heidelberg, 2004, pp. 449–454.

[13]Liu, Y., Vinokur, M., and Wang, Z. J., "Spectral Difference Method for Unstructured Grids I: Basic Formulation," *J. Comput. Phys.*, Vol. 216, No. 2, 2006, pp. 780–801.

[14]Wang, Z. J., Liu, Y., May, G., and Jameson, A., "Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations," *J. Sci. Comput.*, Vol. 32, No. 1, July 2007, pp. 45–71.

[15]May, G. and Jameson, A., "A Spectral Difference Method for the Euler and Navier-Stokes Equations on Unstructured Meshes," *44th AIAA Aerospace Sciences Meeting and Exhibit*, Vol. 304, AIAA, January 2006.

[16]May, G., *A Kinetic Scheme for the Navier-Stokes Equations and High-Order Methods for Hyperbolic Conservation Laws*, Ph.D. thesis, Department of aeronautics and astronautics, Stanford University, September 2006.

[17]van den Abeele, K., Lacor, C., and Wang, Z. J., "On the Stability and Accuracy of the Spectral Difference Method," *J. Sci. Comput.*, Vol. 37, No. 2, 2008, pp. 162–188.

[18]Wang, Z. J. and Gao, H., "A Unifying Lifting Collocation Penalty Formulation for the Euler Equations on Mixed grids," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, AIAA, Orlando, Florida, January 2009.

[19]May, G. and Schöberl, J., "Analysis of a Spectral Difference Scheme with Flux Interpolation on Raviart-Thomas Elements," Tech. Rep. AICES-2010/04-8, AICES, http://www.aices.rwth-aachen.de/research/preprints, April 2010.

[20]Wigton, L. B., Uy, N. J., and young, D. P., ""GMRES Acceleration of Computational Fluid Dynamics Codes"," *AIAA-85-1494*, 1985.

[21]Wang, Z. J., "A Global GMRES/Multi-Grid Scheme for an Adaptive Cartesian/Quad Grid Flow Solver on Distributed Memory Machines," *AIAA 13th Computational Fluid Dynamics Conference, Snowmass Village, CO, June-July 1997*, No. 2048, AIAA, 1997.

[22]May, G., Iacono, F., and Jameson, A., "A Hybrid Multilevel Method for High-Order Discretization of the Euler Equations on Unstructured Meshes," *J. Comput. Phys.*, Vol. 229, 2010, pp. 3938–3956.

[23]Jameson, A., "Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method," *Appl. Math. Comput.*, Vol. 13, 1983, pp. 327–356.

[24]Iacono, F. and May, G., "Convergence Acceleration for Simulation of Steady-State Compressible Flows Using High-Order Schemes," *19th AIAA Computational Fluid Dynamics Conference, San Antonio, TX*, AIAA, http://www.aices.rwth-aachen.de:8080/aices/preprint/documents/AICES-2009-14.pdf, 2009.

[25]Saad, Y. and Schultz, M. H., "GMRES: a Generalized Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, No. 3, 1986, pp. 856–869.

[26]Knoll, D. A. and Keyes, D. E., "Jacobian-Free Newton-Krylov Methods: a Survey of Approaches and Applications," *J. Comput. Phys.*, Vol. 193, 2004, pp. 357–397.

[27]der Vorst, H. A. V., "GMRESR: a Family of Nested GMRES methods," *Num. Lin. Alg. Appl.*, Vol. I, No. 4, 1994, pp. 369–386.

[28]Saad, Y., "A Flexible Inner-Outer Preconditioned GMRES Algorithm," Tech. Rep. umsi-91-279, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, Minnesota, http://www-users.cs.umn.edu/ saad/PDF/umsi-91-279.pdf, 1991.

[29]Chan, T. F. and Jackson, K. R., "Nonlinearly Preconditioned Krylov Subspace Methods for Discrete Newton Algorithms," *SIAM J. Sci. Stat. Comput.*, Vol. 5, No. 3, 1984, pp. 533–542.

[30]Bijl, H. and Carpenter, M. H., "Iterative Solution Techniques for Unsteady Flow Computations Using Higher Order Time Integration Schemes," *Int. J. Numer. Meth. Fluids*, Vol. 47, No. 8-9, 2005, pp. 857–862.

[31]Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Math. Comput.*, Vol. 31, No. 138, 1977, pp. 333–390.

[32]Balay, S., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L. C., Smith, B., and Zhang, H., *PETSc Users Manual*, Argonne National Laboratory, http://www.mcs.anl.gov/petsc/petsc-as/snapshots/petsc-current/docs/manual.pdf, December 2008.

[33]May, G., Iacono, F., and Jameson, A., "Efficient Algorithms for High-Order Discretizations of the Euler and Navier-Stokes Equations," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, No. AIAA Paper 2009-182, January 2009.

[34]Birken, P. and Jameson, A., "On Nonlinear Preconditioners in Newton-Krylov Methods for Unsteady Flows," *Int. J. Numer. Meth. Fluids*, Vol. 62, No. 5, 2010, pp. 565–573.