Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **10**, No. 2, pp. 261-274

DOI: 10.4208/aamm.OA-2017-0004
April 2018

A Study of Preconditioned Jacobian-Free Newton-Krylov Discontinuous Galerkin Method for Compressible Flows on 3D Hexahedral Grids

Wanglong Qin[1,2], Junwei Cai[1], Hongqiang Lu[2,*], Peter K. Jimack[3] and M. A. Walkley[3]

[1] *Department of Air Force, The 28th Resarch Institute of China Electronics Technology Group Corporation, Nanjing 210007, China*
[2] *Department of Aerodynamics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*
[3] *School of Computing, University of Leeds, Leeds LS2 9JT, UK*

Received 7 January 2017, Accepted (in revised version) 7 August 2017

Abstract. Storage requirement and computational efficiency have always been challenges for the efficient implementation of discontinuous Galerkin (DG) methods for real life applications. In this paper, a fully implicit Jacobian-Free Newton-Krylov (JFNK) method is developed in the context of DG discretizations for the three-dimensional compressible Euler and Navier-Stokes equations. Compared with the Jacobian-based methods, the Jacobian-Free approach saves the storage for the Jacobian matrix which can be of great importance for DG methods. Three types of preconditioners are investigated in which the block diagonal preconditioner requires the least storage, while the block LU-SGS and ILU0 preconditioners require more storage but are more computationally efficient. An implicit time-stepping strategy is adopted for the stability of the current solver, which is based upon a hexahedral spatial mesh and the nonlinear solver package Kinsol is used to improve the computational efficiency and robustness. Numerical results demonstrate that the preconditioned JFNK-DG solver can substantially reduce the storage requirement compared with the Jacobian based method without significantly compromising accuracy or efficiency. Furthermore, as a good compromise between efficiency and storage requirement, the ILU0 preconditioner shows the best choice of the preconditioners presented.

AMS subject classifications: 65M10, 78A48

Key words: Discontinuous Galerkin, Jacobian-free, implicit time-stepping, preconditioner.

*Corresponding author.
Emails: qinwanglong@126.com (W. L. Qin), caijunwei@126.com (J. W. Cai), hongqiang.lu@nuaa.edu.cn (H. Q. Lu), p.k.jimack@leeds.ac.uk (P. K. Jimack), m.a.walkley@leeds.ac.uk (M. A. Walkley)

 ©2018 Global Science Press

1 Introduction

Discontinuous Galerkin (DG) methods using high order approximations have become an attractive alternative for the solutions of systems of conservative laws [1–4]. The attractive features, such as high-order accuracy, great geometry flexibility, straightforward implementation of h/p adaptation and parallel computing make it suitable for aerodynamic applications [5–9]. The same as the classical finite element methods, DG methods can achieve high-order accuracy on grids by means of high-order polynomial approximation within elements while the physics of wave propagation is accounted for by means of solving Riemann problems at element interfaces, as in upwind finite volume methods. Despite these advantages, computational efficiency has always been a challenge in DG methods for their use in real life applications. It is noted in [15–18] that the explicit Runge-Kutta methods are not good choices for DG schemes in steady-state simulations due to their severe stability limitations. Thus in this case the use of an implicit time integration is almost mandatory. The Newton-Krylov method [10, 34] is considered as a robust and efficient approach for solving the nonlinear algebraic equations that arise from a DG discretization and has been used in [11–13]. However in order to speed up convergence, this method often combines with preconditioners, such as block diagonal (BD) [14], block Gauss Siedel (BGS) [15], Lower-Upper Symmetry Gauss Siedel (LU-SGS) factorization [16] and Incomplete Lower-Upper factorization with zero fill-in (ILU0) [17]. Nevertheless there is a very large storage requirement for the sparse Jacobian matrix and the preconditioners, especially for three-dimensional simulations, which provides a limitation on DG schemes when the grid density and/or the order of polynomial approximation increases [18]. Since only the product of the Jacobian matrix and a vector is required in the Krylov subspace methods, a difference quotient of the nonlinear function can be used as an approximation, which circumvents the construction and storage of the Jacobian matrix [19, 20]. A good preconditioning is still required in order to obtain satisfactory performance, making the schemes not completely matrix-free. Nevertheless, this can be formed based upon an approximation of the true Jacobian matrix which is easy to implement [21–23]. Regarding the specific Krylov subspace method, in this work we consider only GMRES since it is appropriate for non-symmetric and indefinite linear systems.

In this paper, a Jacobian-Free Newton-Krylov approach for the discontinuous Galerkin method is developed on hexahedral grids with a specific focus upon significantly reducing the storage requirement against the original Jacobian based solver. A novel feature of our work is that we conduct a comparative study of several preconditioners in order to investigate their computational efficiency in the framework of the JFNK-DG solver. To maximize the robustness of the three-dimensional solver, the Kinsol package [24–26] is used combining with an implicit time-stepping strategy. The developed preconditioned JFNK-DG method is used to compute a variety of flow problems on hexahedral grids to demonstrate its accuracy, efficiency and robustness. Numerical results demonstrate that the preconditioned Jacobian-Free Newton-Krylov approach works well with DG method

in solving the three-dimensional Euler and Navier-Stokes equations.

2 Governing equations

The Navier-Stokes equations governing unsteady compressible viscous flow can be expressed as

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t}+\frac{\partial \mathbf{F}_k(\mathbf{U}(\mathbf{x},t))}{\partial x_k}=\frac{\partial \mathbf{G}_k(\mathbf{U}(\mathbf{x},t))}{\partial x_k}, \tag{2.1}$$

where the summation convention has been used. The conservative variable vector $\mathbf{U}$, inviscid flux vector $\mathbf{F}$ and viscous flux vector $\mathbf{G}$ are defined by

$$\mathbf{U}=\begin{Bmatrix}\rho\\ \rho u_i\\ \rho e\end{Bmatrix},\quad \mathbf{F}_j=\begin{Bmatrix}\rho u_j\\ \rho u_i u_j+p\delta_{ij}\\ \rho h u_j\end{Bmatrix}\quad\text{and}\quad \mathbf{G}_j=\begin{Bmatrix}0\\ \sigma_{ij}\\ u_l\sigma_{lj}+q_j\end{Bmatrix}, \tag{2.2}$$

where ρ,p and e denote the density, static pressure and total energy per unit mass, respectively. Furthermore, u_i is the velocity vector of the flow in the coordinate direction x_i. The components of the viscous stress tensor σ_{ij} and the heat flux vector q_j are given by

$$\sigma_{ij}=\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij},\quad q_j=\frac{1}{\gamma-1}\frac{\mu}{Pr}\frac{\partial T}{\partial x_j}, \tag{2.3}$$

where T is the temperature of the fluid. With the state equation for perfect gas, the system of equations can be completed:

$$p=(\gamma-1)\rho\left(e-0.5u_j u_j\right). \tag{2.4}$$

In the above equations, μ denotes the molecular viscosity, γ the ratio of the specific heats and Pr is the dimensionless Prandtl number, which is taken 0.72 for air. Neglecting viscous effects, the left-hand-side of Eq. (2.1) represents the Euler equations governing unsteady compressible inviscid flows.

3 JFNK-DG method

3.1 DG discretization

Using a mixed formulation [12], Eq. (2.1) can be reformulated as

$$\mathbf{s}=\nabla\mathbf{U}, \tag{3.1a}$$

$$\frac{\partial \mathbf{U}}{\partial t}+\nabla\cdot\mathbf{f_c}(\mathbf{U})-\nabla\cdot\mathbf{f_v}(\mathbf{U},\mathbf{s})=0, \tag{3.1b}$$

where $\mathbf{s}$ is an introduced auxiliary variable, while $\mathbf{f_c}(\mathbf{U})$ and $\mathbf{f_v}(\mathbf{U},\mathbf{s})$ are the inviscid and viscous flux tensors. By introducing suitable piecewise polynomial test functions ϕ_h and approximate solutions $\mathbf{U}_h$ at each cell K, we obtain the weak formulations:

$$\int_K \phi_h \mathbf{s}_h d\Omega = \int_K \phi_h \nabla \mathbf{U}_h d\Omega + \int_{\partial K} \phi_h^- \left(\hat{\mathbf{U}}_h - \mathbf{U}_h^-\right)\cdot \mathbf{n} d\sigma, \tag{3.2a}$$

$$\int_K \phi_h \frac{\partial \mathbf{U}_h}{\partial t} d\Omega - \int_K \nabla \phi_h \cdot (\mathbf{f_c}(\mathbf{U}_h) - \mathbf{f_v}(\mathbf{U}_h,\mathbf{s}_h)) d\Omega + \int_{\partial K} \phi_h (\mathbf{f_c}(\mathbf{U}_h) - \mathbf{f_v}(\mathbf{U}_h,\mathbf{s}_h))\cdot \mathbf{n} d\sigma = 0, \tag{3.2b}$$

where Ω is the domain, σ the boundary of Ω. The variables with superscript "$-$" denote the values inside the cell K and $\hat{\mathbf{U}}_h \cdot \mathbf{n}$ denotes the numerical flux function. $\mathbf{f_c}(\mathbf{U}_h)\cdot\mathbf{n}$ and $\mathbf{f_v}(\mathbf{U}_h,\mathbf{s}_h)\cdot\mathbf{n}$ are inviscid and viscous numerical fluxes, respectively. The inviscid numerical flux $\mathbf{f_c}(\mathbf{U}_h)\cdot\mathbf{n}$ can be handled as in the Finite Volume (FV) method. Here we use the well-known Local Lax-Friedrichs (LLF) [27] or Roe flux [28] for the approximate solution of the Riemann problem. As for the viscous numerical flux, [2, 29, 30] have proposed feasible approaches. However the well-known BR2 scheme [17] is used in the current solver.

3.2 Jacobian-Free Newton-Krylov approach

Using auxiliary variables in Eq. (3.2a) to substitute them in Eq. (3.2b), the discrete form can be reformulated as a nonlinear ordinary differential equation system of the form

$$\mathbf{M}\frac{d\mathbf{u}}{dt} + \mathbf{R}(\mathbf{u}) = \mathbf{0}, \tag{3.3}$$

where $\mathbf{u}$ is the global vector of unknown degrees of freedoms (DOFs) and $\mathbf{M}$ represents the global block diagonal mass matrix. At this point a temporal discretization is required. When we are interested in steady-state solutions, it is possible to set $\frac{du}{dt}$ to be zero. However it is preferable to use an implicit time-stepping scheme with excellent stability properties. This allows large time steps to be selected (and often allows steady-state to be reached more efficiently when a good "initial guess" is not available) and results in a nonlinear algebraic system $\mathbf{G}(\mathbf{u}_{n+1}) = 0$ in time step t^{n+1}. Using the implicit backward Euler method, $\mathbf{G}(\mathbf{u}_{n+1})$ can be expressed as

$$\mathbf{G}(\mathbf{u}_{n+1}) = \mathbf{M}\frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{\Delta t} + \mathbf{R}(\mathbf{u}_{n+1}). \tag{3.4}$$

In this expression, $\mathbf{u}_n$ and $\mathbf{u}_{n+1}$ are the solution vectors at the times t_n and t_{n+1}, respectively.

Newton's method is used for solving the nonlinear system which results in a linear system

$$\mathbf{J}(\mathbf{u}^{(k)})\delta^{(k+1)} = -\mathbf{G}(\mathbf{u}^{(k)}) \tag{3.5}$$

at the k^{th} Newton step (k=0,1,2,$\cdots$). In this system, $\mathbf{u}^{(k)}$ represents the k^{th} approximation to $\mathbf{u}_{n+1}$, typically commencing with the initial guess $\mathbf{u}^{(0)}=\mathbf{u}_n$. When the solution to Eq. (3.5) has been found, the approximation to $\mathbf{u}_{n+1}$ is updated as

$$\mathbf{u}^{(k+1)}=\mathbf{u}^{(k)}+\boldsymbol{\delta}^{(k+1)}.$$

The Jacobian matrix can be evaluated as

$$\mathbf{J}(\mathbf{u}^{(k)})=\frac{\partial\mathbf{G}\left(\mathbf{u}^{(k)}\right)}{\partial\mathbf{u}},$$

which is an $N\times N$ block sparse matrix, where N is the number of elements in the computational domain. The rank of each block is $M\times N_{dof}$, M being the number of variables of the Navier-Stokes equations and N_{dof} the numbers of DOFs for each variable. When Krylov subspace methods [10] are used for solving this linear system, a matrix-vector product is required at each GMRES iteration:

$$\boldsymbol{\omega}=\mathbf{J}(\mathbf{u}^{(k)})\mathbf{p}. \tag{3.6}$$

It can be observed that

$$\mathbf{J}(\mathbf{u}^{(k)})\mathbf{p}\approx\frac{\mathbf{G}(\mathbf{u}^{(k)}+\varepsilon\mathbf{p})-\mathbf{G}(\mathbf{u}^{(k)})}{\varepsilon}, \tag{3.7}$$

where ε is a small scalar. Hence the product can be estimated directly without knowing the matrix $\mathbf{J}(\mathbf{u}^{(k)})$. Although an additional cost of evaluation of function $\mathbf{G}$ is required, the more costly evaluation of the matrix $\mathbf{J}(\mathbf{u}^{(k)})$ is not needed at the start of the k^{th} Newton step and the large memory required for storing the matrix is saved. Since the storage of the Jacobian matrix grows quickly when the order increases for the DG method, the Jacobian-Free approach seems especially attractive.

3.3 Preconditioning

The Jacobian matrix arising from a DG discretization is generally ill-conditioned and the conditioning deteriorates as mesh size h is reduced or order p is increased, which makes the standard GMRES iteration converge very slowly. In order to speed up, preconditioning plays a significant role. In this work right preconditioning is used, which reformulates Eq. (3.5) as

$$(\mathbf{J}(\mathbf{u}^{(k)})\mathbf{P}^{-\mathbf{1}})(\mathbf{P}\boldsymbol{\delta}^{(k+1)})=-\mathbf{G}(\mathbf{u}^{(k)}), \tag{3.8}$$

where $\mathbf{P}$ is the preconditioning matrix. To achieve a fast convergence, $\mathbf{P}$ should be an approximation of the matrix $\mathbf{J}$, but much easier to invert and save.

The Jacobian matrix $\mathbf{J}$ can be decompose into a lower diagonal part $\mathbf{L}$, a block diagonal part $\mathbf{D}$ and an upper diagonal part $\mathbf{U}$ as

$$\mathbf{J}=\mathbf{L}+\mathbf{D}+\mathbf{U}. \tag{3.9}$$

When neglecting the off diagonal blocks **L** and **U**, the simplest block diagonal/Jacobi preconditioner can be obtained

$$\mathbf{P}=\mathbf{D}. \tag{3.10}$$

Block LU-SGS preconditioner is considered as a more sophisticated preconditioner for compressible flow simulations and has been widely used in FV and DG methods

$$\mathbf{P}=(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U}). \tag{3.11}$$

Another often-used preconditioner in DG schemes is the Incomplete LU factorization, in which an approximation of the Jacobian $\widetilde{\mathbf{J}}$ can be obtained

$$\mathbf{P}=\widetilde{\mathbf{J}}=\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}. \tag{3.12}$$

In these preconditioners, the block diagonal method is often used due to its easy implementation and small storage requirement. However this preconditioner fails to capture much information from the original Jacobian matrix. On the contrary, the block LU-SGS preconditioner and ILU0 preconditioner keep much of the Jacobian information, thus requiring more storage but are more efficient. See [14, 15, 23] for more details of these preconditioners.

3.4 Storage requirement for preconditioners

When a DG discretization is adopted, the degrees of freedoms are only coupled with those of the neighbouring elements and the number of nonzero blocks for each block row K in the Jacobian is equal to the number of elements surrounding the element K plus one ($=7$ for each interior hexahedral element). Neglecting the boundary elements, $7\times N$ nonzero blocks are stored for the block LU-SGS or ILU preconditioner, which leads to the overall amount of nonzeros storage

$$\text{Memory(LU-SGS) = Memory(ILU)}=7\times N\times\left(M\times N_{dof}\right)^2.$$

Since only the diagonal blocks are stored for the BD preconditioner, the nonzeros in this preconditioner is

$$\text{Memory(BD)}=N\times\left(M\times N_{dof}\right)^2.$$

In brief, the actual storage requirement of the preconditioners mentioned is $M_{\text{LUSGS}}\approx M_{\text{ILU}}\approx 7\times M_{\text{BD}}$, where M_x denotes the storage requirement of the preconditioner x.

4 Numerical results

In this section, a few examples, including both inviscid and viscous test cases, are presented to illustrate the high efficiency and robustness of the JFNK-DG method in terms of the storage requirement and computational CPU time. All of the computations are

performed on a HP Compaq dx7518 desktop computer (2.67GHz Q8400 CPU with 8Gbytes memory) with linux operating system except the last case which is performed on a HP Z620 workstation computer (3.6GHz Xeon CPU with 32Gbytes memory). A restarted GMRES method is used for all these test cases, with a maximum Krylov space of 60 vectors and a maximum number of 180 iterations (i.e., at most two restarts). The nonlinear convergence residual is set to be 10^{-10} and the maximum step is set to be 100 for each test case (i.e., non-convergence is assumed if this number of nonlinear iterations is exceeded). In order for robustness, $\Delta t = 0.1$ is used for the first time steps of all the computations, which is then increased sharply to $\Delta t \to \infty$ (i.e., one step of the Newton's method in the computation of steady state solutions) in the following time steps. Note that we do not propose this as an optimal strategy (the study of which is beyond the scope of this paper) however it does allow us to make a consistent comparison of the preconditioners considered.

4.1 An inviscid flow past a channel with a smooth bump

The inviscid flow past a channel with a smooth bump [31] is considered in this test case. This simple problem is chosen to assess the accuracy of the numerical solution obtained by the JFNK-DG method. The flow condition is given at a Mach number of 0.5 based on the freestream velocity. Three successive refined quadratic grids are used to obtain the convergence rate, with point distribution of $17\times8\times4$, $33\times15\times7$ and $65\times29\times13$, respectively. The L2-norm of entropy production is used as the error measurement

$$\|\varepsilon\|_{L_{2(\Omega)}} = \sqrt{\int_\Omega \varepsilon^2 d\Omega}, \tag{4.1}$$

where the entropy production is defined as

$$\varepsilon = \frac{S - S_\infty}{S_\infty} = \frac{p}{p_\infty}\left(\frac{\rho_\infty}{\rho}\right)^\gamma - 1. \tag{4.2}$$

Table 1 illustrates the computed rates of convergence of the JFNK-DG method. It can be clearly observed that the expected order of $\mathcal{O}(h^{p+1})$ can be obtained. Since the non-uniform grids are used for this test case with curved boundaries, the grid distribution may cause some error on the computed order. This can be the reason why the order between the second and third mesh is 2.88 for DG ($p=2$) results, which is not very close to 3.

Tables 2-4 shows the convergence results for the different preconditioners in terms of the number of non-linear Newton iterations, linear GMRES iterations and the executed CPU time. All the preconditioners are computationally efficient for this case and converged solutions can be obtained within 10 nonlinear steps although the number of grids increases. Although the LU-SGS and ILU preconditioners require more storage requirement, they are several times more efficient than the block diagonal preconditioner.

Table 1: Accuracy study for inviscid flow past a channel with a smooth bump.

Order	Mesh	L_2 error	L_2 order
DG, $p=1$	$17\times8\times4$	$1.77E-03$	—
	$33\times15\times7$	$4.61E-04$	1.94
	$65\times29\times13$	$1.17E-04$	1.98
DG, $p=2$	$17\times8\times4$	$1.85E-04$	—
	$33\times15\times7$	$2.17E-05$	3.09
	$65\times29\times13$	$2.49E-06$	2.88

Table 2: Convergence results of the inviscid channel case (336 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time (min).

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	7	263	0.10	5	54	0.02	6	48	0.05
$p=2$	6	277	0.78	6	96	0.35	5	41	0.28

Table 3: Convergence results of the inviscid channel case (2688 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time (min).

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	7	515	1.40	7	151	0.52	6	72	0.40
$p=2$	6	399	8.57	6	126	3.27	5	66	3.23

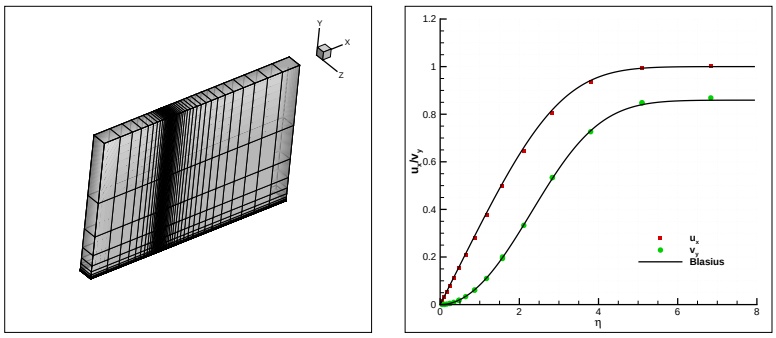
Table 4: Convergence results of the inviscid channel case (21504 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time (min).

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	8	778	16.78	9	247	6.70	8	169	8.83
$p=2$	8	881	142.88	6	215	43.70	6	126	65.10

4.2 Laminar flow past a flat plate

A laminar flow over an adiabatic flat plate is chosen to study the performance of the different preconditioners when using JFNK-DG method. In this case, the free-stream flow condition is given as $Ma_\infty = 0.5$ and $Re_\infty = 100{,}000$. The problem is solved using a hierarchical basis on three successive refined grids with 960, 1920 and 3840 quadratic elements, respectively. Fig. 1 shows a portion of the coarse grid and the corresponding $p=2$ solution of velocity components on this grid. The non-dimensional variables are given as $u_x = u/u_\infty$, $v_y = v\times\sqrt{Re_x}/u_\infty$ versus $\eta = y\times\sqrt{Re_x}/x$.

Solutions of $p=1,2$ are obtained, where the low-order converged solutions are used as an initial guess of the higher-order computations. Tables 5-7 show the convergence results of the different preconditioners. It can be observed from the computational results



(a) Grid (b) Results of order $p=2$

Figure 1: Computational grid and solution of velocity profiles on coarse grid (960 elements, $p=2$) for laminar flow past a flat plate.

that the block diagonal preconditioner is not efficient for this convection-dominated laminar flow problem and the convergence results can't be derived within 100 nonlinear iterations on the medium and fine grids. The LU-SGS preconditioner performs notably better than the block diagonal preconditioner for this laminar flow case, however it is significantly inferior to the ILU preconditioner. Recall that the Reynolds number of this test case is 100000, which means that the contribution of the convection term to the Jacobian dominates. The LU-SGS preconditioners does not capture this contribution so well-hence it gives a poorer approximation to the Jacobian than the ILU preconditioner.

4.3 Laminar flow past a sphere

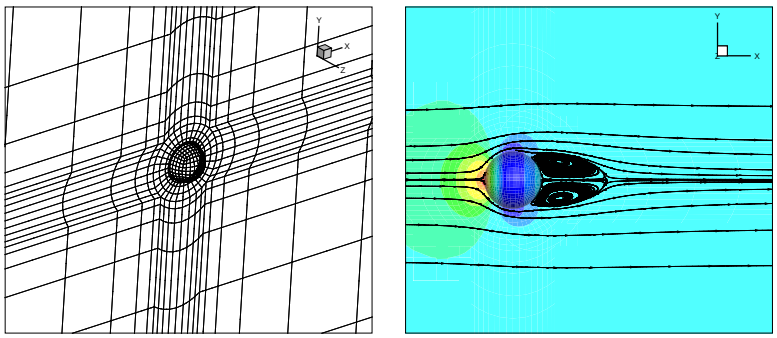
A viscous flow past a sphere is considered in this case. The computation is performed at a Mach number of 0.5 and Reynolds number of 118 based on the diameter of the sphere.

Table 5: Convergence results of the laminar flat plate case (960 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time (min).

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	45	7726	8.43	18	2062	2.43	13	150	0.27
$p=2$	30	4970	27.73	8	956	5.72	6	123	1.02

Table 6: Convergence results of the laminar flat plate case (1920 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time(min). "$*$" denotes cases which did not convergence in maximum nonlinear iterations.

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	*	*	*	29	4087	9.42	14	271	0.8
$p=2$	*	*	*	13	1887	23.01	8	184	3.1



(a) Grid (b) Solutions of streamlines

Figure 2: Computational grid and solution of streamlines on pressure contours for laminar flow past a sphere (4216 elements, $p=2$).

This test case is chosen to demonstrate the robustness of the current solver in the computation of viscous flows around a curved geometry. Since the model and flow is symmetric, we consider only half of the configuration. Fig. 2 shows a portion of the grid with 4216 quadratic hexahedral elements and the corresponding $p=2$ solution of velocity streamlines on the pressure contours, which agrees well with the results in [33] using Reconstructed Discontinuous Galerkin (RDG) method. Table 8 gives the convergence results for this case. Similar to the results of the flat plate case, the ILU0 preconditioner is much more efficient than the other preconditioners, though the lower Reynolds number in this example means the superiority over LU-SGS is a little less marked.

4.4 Subsonic flow past a Delta Wing

A laminar flow at a high angle of attack around a delta wing with a sharp leading edge and a blunt trailing edge is considered. This is a benchmark in the 3rd international workshop on high-order CFD methods [32] and the flow condition is given as $Ma_\infty=0.3$ and $Re_\infty=4000$ with an angle of attack $\alpha=12.5^\circ$. This case is chosen to compare the performance of the Jacobian-Free method with the Jacobian based method in storage requirement and computational efficiency. 7159 quadratic hexahedral elements are used for computation and the ILU0 decomposition approach is adopted as the precondition-

Table 7: Convergence results of the laminar flat plate case (3840 elements). nni: numbers of non-linear iterations, nli: total numbers of linear iterations, time: total run time (min). "$*$" denotes cases which did not convergence in maximum nonlinear iterations.

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	*	*	*	33	4734	22.38	14	463	2.43
$p=2$	*	*	*	12	1709	41.08	8	386	8.51

Table 8: Convergence results of the laminar flow past a sphere test case (4216 elements). nni: total numbers of non-linear iterations, nli: numbers of linear iterations, time: total run time (min).

	BD			LU-SGS			ILU		
	nni	nli	time	nni	nli	time	nni	nli	time
$p=1$	30	2135	13.5	19	866	5.95	13	249	2.25
$p=2$	63	4131	126.8	27	1581	53.4	16	746	23.01

Table 9: Memory requirements of different solvers for delta wing case (7159 elements).

	Jacobian based (Mb)	Jacobian free (Mb)	Saving(%)
$p=1$	584.0	439.4	24.8%
$p=2$	2477.5	1573.8	36.5%

Table 10: Convergence results of different solvers for delta wing case (7159 elements). nni: total numbers of non-linear iterations, nli: numbers of linear iterations, work units: normalized CPU time.

	Jacobian based method			Jacobian free method		
	nni	nli	work units	nni	nli	work units
$p=1$	6	172	25.22	6	174	25.84
$p=2$	9	240	217.17	9	280	244.91

er. The memory saving achieved for different polynomial approximation is reported in Table 9. It can be seen that a significant saving in storage is achieved with the JFNK-DG method, which increases with the degree of polynomial approximation, from 24% for $p=1$ to 36% for $p=2$, which demonstrates the superiority of the Jacobian-Free approach.

Table 10 gives the convergence results of both methods. The computational CPU time is normalized by TauBench time, which is 10.75 seconds for this desktop computer. The computed lift and drag coefficients of both methods with order $p=2$ are $C_l=0.35456$ and $C_d=0.16919$, which is very close to the reference value of $C_l^{ref}=0.347$ and $C_d^{ref}=0.17148$ from the 3rd workshop. From the results it can be observed that compared with the Jacobian base approach, the JFNK-DG method can substantially reduce the storage requirement without significantly compromising accuracy and efficiency.

4.5 Subsonic flow past a DLR-F6 wing body transport configuration

A subsonic flow past a DLR-F6 wing body transport configuration is chosen to test the reliability and robustness of the current JFNK-DG solver. The flow condition is given as $Ma_\infty=0.3$ with an angle of attack $\alpha=1^\circ$. Solutions have been computed up to $p=3$ polynomial approximation for this case on a grid with 33448 quadratic hexahedral elements. The total degrees of freedoms for different orders ranging from $p=1$ to $p=3$ are 668960, 1672400 and 3344800, respectively. ILU0 preconditioner is used for this case and the resulting Mach number contours of order $p=3$ are illustrated in Fig. 3 which is converged to a residual of 10^{-6}. The computed lift and drag coefficients are $C_l=0.52705$ and $C_d=0.02386$ for order $p=3$. Table 11 shows the convergence results in terms of the num-

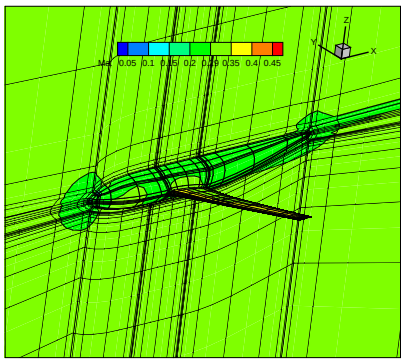


Figure 3: Solutions of Mach number contours for DRL-F6 wing body configuration ($p=3$).

ber of non-linear Newton iterations, linear GMRES iterations and the executed CPU time. In order to speed up, a preconditioner-freezing approach is tested in this complex flow case in which the preconditioner is re-evaluated every 5 nonlinear iterations. Since the process of forming preconditioner is especially costly for DG method, the preconditioner-freezing approach shows a great superiority for this case.

Table 11: Convergence results of the subsonic flow past a DLR-F6 wing body configuration (33448 elements). nni: total numbers of non-linear iterations, nli: numbers of linear iterations, time: total run time (min).

	without preconditioner–freezing			preconditioner–freezing		
	nni	nli	time	nni	nli	time
$p=1$	6	158	32.3	6	149	5.35
$p=2$	8	272	117.9	8	280	38.96

5 Conclusions

A fully implicit Jacobian-Free Newton-Krylov discontinuous Galerkin method has been presented to solve the compressible Euler and Navier-Stokes equations on hexahedral grids. Several preconditioners are investigated and compared in both storage requirement and computational efficiency in the framework of the JFNK-DG solver. A variety of three dimensional test cases have been conducted to demonstrate the efficiency and robustness of the developed JFNK-DG solver. The contribution of this paper is to show that, in comparison with with the Jacobian based method, the Jacobian-Free approach can substantially reduce the storage requirement without significantly compromising accuracy and efficiency. Furthermore, our numerical experiments clearly demonstrate that, across a wide range of test cases, the ILU0 preconditioner is a good choice considering both storage requirement and computational efficiency, especially for complex flow simulations. Finally, we note that current trends in computer hardware (e.g., many-core chips) suggest that the cost of CPU cycles relative to memory capacity will decrease rapidly over

the coming years. We suggest therefore that the importance of low memory algorithms is likely to grow, even where this is at the expense of additional numerical operations.

Acknowledgements

Thanks to the National Natural Science Foundation of China (No. 11272152), Aeronautical Science Foundation of China (No. 20151452021) and the Open Project of Key Laboratory of Aerodynamic Noise Control.

References

[1] B. COCKBURN AND C. W. SHU, *The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: multidimensional systems,* J. Comput. Phys., 141 (1998), pp. 199–224.
[2] B. COCKBURN AND C. W. SHU, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems,* SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463.
[3] F. BASSI AND S. REBAY, *High-order accurate discontinuous finite element solution of the 2d Euler equations,* J. Comput. Phys., 138 (1997), pp. 251–285.
[4] R. HARTMANN, J. HELD AND T. LEICHT, *Adjoint-based error estimation and adaptive mesh refinement for the RANS and $k-\omega$ turbulence model equations,* J. Comput. Phys., 230 (2011), pp. 4268–4284.
[5] LAIPING ZHANG, MING LI AND WEI LIU ET AL., *An implicit algorithm for high-order DG/FV schemes for compressible flows on 2D arbitrary grids,* Commun. Comput. Phys., 17 (2015), pp. 287–316.
[6] WANGLONG QIN, HONGQIANG LU AND YIZHAO WU, *High-order discontinuous Galerkin solution of N-S equations on hybrid mesh,* Chinese J. Theoret. Appl. Mech., 45 (2013), pp. 987–991.
[7] H. LU AND Q. SUN, *A straightforward hp-adaptivity strategy for shock-capturing with high-order discontinuous Galerkin methods,* Adv. Appl. Math. Mech., 6 (2014), pp. 135–144.
[8] H. LU, YIDA XU AND YUKUN GAO ET AL., *A high-order discontinuous Galerkin method for the two-dimensional time-domain Maxwell's equations on curved mesh,* Adv. Appl. Math. Mech., 8 (2016), pp. 104–116.
[9] Z. H. JIANG, C. YAN AND J. YU, *Implicit high-order discontinuous Galerkin method with H-WENO type limiters for steady viscous flow simulations*, Acta Mech. Sinica, 29 (2013), pp. 526–533.
[10] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM J. Sci. Comput., 7 (1986), pp. 856–869.
[11] Z. J. WANG, *High-order methods for the Euler and Navier-Stokes equations on unstructured grids,* Progress Aerospace Sci., 40 (2002), pp. 1969–1978.
[12] B. LANDMANN, M. KESSLER AND S. WAGNER ET AL., *A parallel, high-order discontinuous Galerkin code for laminar and turbulent flows,* Comput. Fluids, 37 (2008), pp. 427–438.
[13] S. AHMED, C. E. GOODYER AN D P. K. JIMACK, *An efficient preconditioned iterative solution of fully-coupled elastohydrodynamic lubrication problems,* Appl. Numer. Math., 62 (2012), pp. 649–663.
[14] L. T. DIOSADY AND D. L. DARMOFAL, *Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations,* J. Comput. Phys., 228 (2009), pp. 3917–3935.

[15] P. O. PERSSON AND J. PERAIRE, *Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations,* SIAM J. Sci. Comput., 30 (2008), pp. 2709–2733.
[16] H. LUO, J. D. BAUM AND R. LÖHNER, *A fast, matrix-free implicit method for compressible flows on unstructured grids,* J. Comput. Phys., 146 (1998), pp. 664–690.
[17] F. BASSI, A. CRIVELLINI AND S. REBAY ET AL., *Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and $k-w$ turbulence model equations,* Comput. Fluids, 34 (2005), pp. 507–540.
[18] A. CRIVELLINI AND F. BASSI, *An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations*, Comput. Fluids, 50 (2011), pp. 81–93.
[19] P. J. CAPON AND P. K. JIMACK, *An inexact Newton method for systems arising from the finite element method,* Appl. Math. Lett., 10 (1997), pp. 9–12.
[20] P. RASETARINERA AND M. Y. HUSSAINI, *An efficient implicit discontinuous spectral Galerkin method,* J. Comput. Phys., 172 (2001), pp. 718–738.
[21] P. K. JIMACK AND M. A. WALKLEY, *Asynchronous parallel solvers for linear systems arising in computational engineering*, Comput. Tech. Rev., 3 (2011), pp. 1–20.
[22] D. A. KNOLL AND D. E. KEYES, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications,* J. Comput. Phys., 193 (2004), pp. 357–397.
[23] Y. XIA, H. LUO AND R. NOURGALIEV, *An implicit Hermite WENO reconstruction-based discontinuous Galerkin method on tetrahedral grids,* Comput. Fluids, 96 (2014), pp. 406–421.
[24] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations,* SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.
[25] A. C. HINDMARSH, P. N. BROWN AND K. E. GRANT ET AL., *SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers,* ACM Trans. Math. Software (TOMS), 31 (2005), pp. 363–396.
[26] M. PERNICE AND H. F. WALKER, *NITSOL: A Newton iterative solver for nonlinear systems,* SIAM J. Sci. Comput., 19 (1998), pp. 302–318.
[27] E. F. TORO, M. SPRUCE AND W. SPEARES, *Restoration of the contact surface in the HLL-Riemann solver*, Shock Waves, 4 (1994), pp. 25–34.
[28] P. L. ROE, *Approximate Riemann solvers, parameter vectors and difference schemes,* J. Comput. Phys., 43 (1981), pp. 357–372.
[29] D. N. ARNOLD, F. BREZZI AND B. COCKBURN ET AL., *Unified analysis of discontinuous Galerkin methods for elliptic problems,* SIAM J. Numer. Anal., 39 (2002), pp. 1749–1779.
[30] J. PERAIRE AND P. O. PERSSON, *The compact discontinuous Galerkin (CDG) method for elliptic problems,* SIAM J. Sci. Comput., 43 (2007), pp. 1–41.
[31] WANGLONG QIN, HONGQIANG LU AND YIZHAO WU ET AL., *Implicit discontinuous Galerkin method on agglomerated high-order grids for 3D simulations,* Chinese J. Aeronautics, 29 (2016), pp. 1496–1505.
[32] `http://www.as.dlr.de/hiocfd/case_c2.4.html`.
[33] Y. XIA, H. LUO AND R. NOURGALIEV, *An implicit reconstructed discontinuous Galerkin method based on automatic differentiation for the compressible flows on tetrahedral grids,* (2013), Report No. AIAA-2013-0687.
[34] Q. WEI, H. J. LU AND J. J. FAN, *Wirelss sensor network localization technology based on internet of things*, Command Information System and Technology, 5 (2014), pp. 22–26.