High-Order Navier-Stokes Simulations using a Sparse Line-Based Discontinuous Galerkin Method

Per-Olof Persson*

University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A.

We study some of the properties of a line-based discontinuous Galerkin (DG) scheme for the compressible Euler and Navier-Stokes equations. The scheme is based on fully unstructured meshes of quadrilateral or hexahedral elements, and it is closely related to the standard nodal DG scheme as well as several of its variants such as the collocation-based DG spectral element method (DGSEM) or the spectral difference (SD) method. However, our motivation is to maximize the sparsity of the Jacobian matrices, since this directly translates into higher performance in particular for implicit solvers. The scheme is based on applying one-dimensional DG solvers along each coordinate direction in a reference element, which reduces the number of connectivities drastically in a line-wise fashion. The resulting scheme is similar to the DGSEM and the SD methods, but it uses fully consistent integration along each 1-D coordinate direction. This gives the method almost the same accuracy and nonlinear stability properties as the standard nodal DG method, at a cost similar to that of the collocation-based methods. Also, the scheme uses solution points along each element face, which further reduces the number of connections with the neighboring elements. Second-order terms are handled by an LDG-type approach, with an upwind/downwind flux function based on a switch function at each element face. We demonstrate the accuracy of the method and compare it to the standard nodal DG method for model problems of an inviscid vortex problem, flow over a cylinder, and laminar flow around an airfoil. We also show how to use Newton-Krylov solvers without impairing the high sparsity of the matrices, by a splitting of the matrix-vector products and a block-Jacobi preconditioner. We integrate in time using a high-order diagonally implicit Runge-Kutta (DIRK) scheme, and apply it on a problem of transient laminar flow around an airfoil. Using a quasi-Newton approach, the leads to an implicit scheme with a computational cost comparable to that of an explicit one, without stability-based timestep restrictions.

I. Introduction

While it is clear that the discontinuous Galerkin (DG) and related methods[1,2,3,4,5,6,7] are getting sufficiently mature to handle realistic problems, their computational cost is still at least a magnitude higher than low-order methods or high-order finite difference methods on similar grids. For some problems, explicit time-stepping or matrix-free implicit methods can be employed, but for many real-world problems with unstructured meshes full Jacobian matrices are required for the solvers to be efficient. Here, nodal-based Galerkin methods have a fundamental disadvantage in that they connect all unknowns inside an element, as well as all neighboring face nodes, even for first-order derivatives. This leads to a stencil size that scales like p^D for polynomial degrees p in D spatial dimensions. As a contrast, a standard finite difference method only involves nodes along neighboring lines, which gives a stencil size proportional to Dp, which in three dimensions can be magnitudes smaller ever for moderate values of p.

Several high-order schemes for unstructured meshes have been proposed with a similar stencil-size reduction. In particular, the DG spectral element method[2] is a collocation-based method on a staggered grid which only uses information along each coordinate line for the discretized equations. Other closely related schemes have the same property, such as the spectral difference method[4] and the flux reconstruction method.[5,6] For

*Assistant Professor, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: persson@berkeley.edu. AIAA Member.

the special case of a linear one-dimensional problem, many of these method can be shown to be identical to the standard DG method,[8] but in general they define different schemes with varying properties.

In an attempt to further reduce the sizes of the Jacobians, and to ensure that the scheme is identical to the standard DG method along each line of nodes, we proposed a line-based DG scheme in Ref. 9. Like the DGSEM, our Line-DG scheme is derived by considering only the 1-D problems that arise along each coordinate direction. We apply a standard 1-D DG formulations for each of these sub-problems, and all integrals are computed fully consistently (to machine precision), which means in particular that the definition of the scheme makes no statement about flux points. We note that this can be done without introducing additional connectivities, since all nodes in the local 1-D problem are already connected by the shape functions. In addition, our scheme uses solution points along each element face, which further reduces the number of connectivities with the neighboring elements.

For the second-order terms in the Navier-Stokes equations, we use an LDG-type approach with upwind/downwind fluxes based on consistent switches along all global lines. Special care is required to preserve the sparsity of the resulting matrices, and we propose a simple but efficient Newton-Krylov solver which splits the matrix product into a series of separate ones, in order to avoid introducing additional matrix entries. Many options for preconditioning is possible, and in this work we use a standard block-Jacobi method.

In our numerical examples, we demonstrate optimal convergence for an inviscid Euler vortex and we show the accuracy for flow over a cylinder. We also compare the accuracy of the method to that of the standard nodal DG method, and we conclude that the differences are very small. For the Navier-Stokes equations, we show convergence of drag and lift forces for steady-state laminar flow around an airfoil, and finally we demonstrate our implicit time-integrators on a transient laminar flow problem.

II. Spatial discretization

Here we give a brief overview of the line-DG scheme, for more details we refer to Ref. 9.

II.A. Line-based discontinuous Galerkin

Our method maps a system of m first-order conservation laws

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

onto a reference element:

$$J\frac{\partial \boldsymbol{u}}{\partial t} + \nabla_{\boldsymbol{X}} \cdot \widetilde{\boldsymbol{F}}(\boldsymbol{u}) = \boldsymbol{0}, \tag{2}$$

by a diffeomorphism $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X})$, see Fig. 1. Here we have defined the mapping Jacobian $J = \det(\boldsymbol{G})$ and the contravariant fluxes $\widetilde{\boldsymbol{F}} = (\widetilde{\boldsymbol{f}}_1, \widetilde{\boldsymbol{f}}_2, \widetilde{\boldsymbol{f}}_3) = J\boldsymbol{G}^{-1}\boldsymbol{F}$, with the mapping deformation gradient $\boldsymbol{G} = \nabla_{\boldsymbol{X}}\boldsymbol{x}$.

The scheme now introduces a grid function $\boldsymbol{u}_{ijk} = \boldsymbol{u}(\boldsymbol{X}_{ijk})$ within each element, and considers each of the three spatial derivatives in (2) separately and approximates them numerically using one-dimensional discontinuous Galerkin formulations along each of the $3(p+1)$ curves defined by straight lines in the reference domain V, through each set of nodes along each space dimension. The Galerkin formulation for each line along the first dimension has the form: Find $\boldsymbol{r}_{jk}(\xi) \in \mathcal{P}_p([0,1])^m$ such that

$$\begin{aligned}\int_0^1 \boldsymbol{r}_{jk}(\xi)\cdot\boldsymbol{v}(\xi)\,d\xi &= \int_0^1 \frac{d\widetilde{\boldsymbol{f}_1}(\boldsymbol{u}_{jk}(\xi))}{d\xi}\cdot\boldsymbol{v}(\xi)\,d\xi \\ &= \widehat{\widetilde{\boldsymbol{f}_1}}(\boldsymbol{u}^+_{jk}(1),\boldsymbol{u}_{jk}(1))\cdot\boldsymbol{v}(1) - \widehat{\widetilde{\boldsymbol{f}_1}}(\boldsymbol{u}_{jk}(0),\boldsymbol{u}^-_{jk}(0))\cdot\boldsymbol{v}(0) - \int_0^1 \widetilde{\boldsymbol{f}_1}(\boldsymbol{u}_{jk}(\xi))\cdot\frac{d\boldsymbol{v}}{d\xi}\,d\xi,\end{aligned} \tag{3}$$

for all test functions $\boldsymbol{v}(\xi) \in \mathcal{P}_p([0,1])^m$. The numerical flux in the reference space $\widehat{\widetilde{\boldsymbol{f}_1}}(\boldsymbol{u}_R,\boldsymbol{u}_L)$ can be written

$$\widehat{\widetilde{\boldsymbol{f}_1}}(\boldsymbol{u}_R,\boldsymbol{u}_L) = \widehat{\widetilde{\boldsymbol{F}}\cdot\boldsymbol{N}_1^+}(\boldsymbol{u}_R,\boldsymbol{u}_L) = \widehat{\boldsymbol{F}\cdot\boldsymbol{n}_1^+}(\boldsymbol{u}_R,\boldsymbol{u}_L), \tag{4}$$

where $\boldsymbol{N}_1^+ = (1,0,0)$ and $\boldsymbol{n}_1^+ = J\boldsymbol{G}^{-T}\boldsymbol{N}_1^+$. We note that this has exactly the same form as the standard numerical fluxes used in finite volume and discontinuous Galerkin schemes. The left endpoint is handled in a similar way.

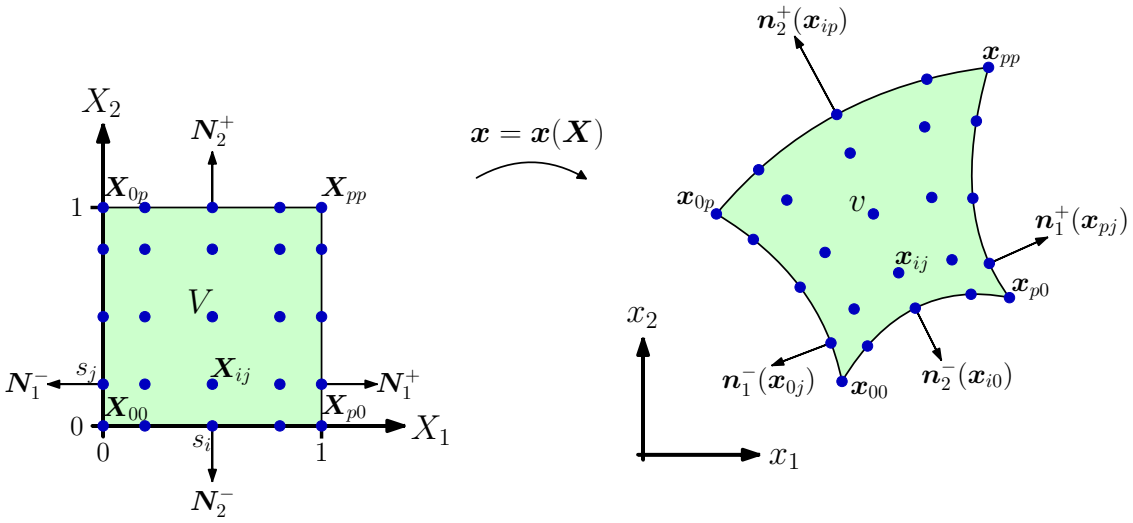


Figure 1. A two-dimensional illustration of the mapping from a reference element V to the actual curved element v, for the case $p = 4$.

A standard finite element procedure is used to solve for $\boldsymbol{r}_{jk}(\xi)$. We introduce nodal Lagrangian basis functions $\phi_i \in \mathcal{P}([0,1])$ such that $\phi_i(s_j) = \delta_{ij}$, for $i, j = 0, \ldots, p$:

$$\boldsymbol{u}_{jk}(\xi) = \sum_{i=0}^{p} \boldsymbol{u}_{ijk}\phi_i(\xi), \tag{5}$$

$$\boldsymbol{r}_{jk}(\xi) = \sum_{i=0}^{p} \boldsymbol{r}_{ijk}\phi_i(\xi). \tag{6}$$

Setting $\boldsymbol{v}(\xi)$ equal to each of the basis functions, we arrive at discrete equations in the form of a system $\boldsymbol{M}\boldsymbol{r}_{jk} = \boldsymbol{b}$, where $\boldsymbol{M}$ is the $(p+1)$-by-$(p+1)$ mass matrix in the reference element.

In an analogous way, we calculate coefficients $\boldsymbol{r}_{ijk}^{(2)}$ and $\boldsymbol{r}_{ijk}^{(3)}$ that approximate $\partial\widetilde{\boldsymbol{F}}_2/\partial X_2$ and $\partial\widetilde{\boldsymbol{F}}_3/\partial X_3$, respectively, at the grid points. The solution procedure involves the same mass matrix $\boldsymbol{M}$ and is identical to before. Using the calculated numerical approximations to each partial derivative in (2), we obtain our final semi-discrete formulation:

$$J_{ijk}\frac{d\boldsymbol{u}_{ijk}}{dt} + \sum_{n=1}^{3} \boldsymbol{r}_{ijk}^{(n)} = \boldsymbol{0}, \tag{7}$$

where $J_{ijk} = J(\boldsymbol{x}_{ijk})$.

II.B. Second-order terms

Viscous terms are handled by an LDG-type approach.[10] The system is split into a first order system

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q}) = \boldsymbol{0}, \tag{8}$$

$$\nabla \boldsymbol{u} = \boldsymbol{q}, \tag{9}$$

where we assume that the flux function $\boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q})$ is purely viscous to simplify the notation. This essentially has the form of our first-order system (1), and we can apply Line-DG to each solution component as described above. The change of variables from $\boldsymbol{x}$ to $\boldsymbol{X}$ transforms (8)-(9) into

$$J\frac{\partial \boldsymbol{u}}{\partial t} + \nabla_{\boldsymbol{X}} \cdot \widetilde{\boldsymbol{F}}(\boldsymbol{u}, \boldsymbol{q}) = \boldsymbol{0}, \tag{10}$$

$$\nabla_{\boldsymbol{X}} \cdot \widetilde{\boldsymbol{u}}(\boldsymbol{u}) = J\boldsymbol{q}, \tag{11}$$

where $\widetilde{\boldsymbol{u}} = (\widetilde{\boldsymbol{u}}_1, \widetilde{\boldsymbol{u}}_2, \widetilde{\boldsymbol{u}}_3) = \boldsymbol{u} \otimes J\boldsymbol{G}^{-1}$. The first-order scheme described above can be applied to this system, and it remains only to specify the numerical fluxes $\widehat{\boldsymbol{F} \cdot \boldsymbol{n}}$ and $\widehat{\boldsymbol{u}}$. We use an LDG-based approach,[10] and set

$$\widehat{\boldsymbol{F} \cdot \boldsymbol{n}}(\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{n}) = \{\{\boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q}) \cdot \boldsymbol{n}\}\} + C_{11}[\![\boldsymbol{u} \otimes \boldsymbol{n}]\!] + \boldsymbol{C}_{12}[\![\boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q}) \cdot \boldsymbol{n}]\!], \tag{12}$$

$$\widehat{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{n}) = \{\{\boldsymbol{u}\}\} - \boldsymbol{C}_{12} \cdot [\![\boldsymbol{u} \otimes \boldsymbol{n}]\!] + C_{22}[\![\boldsymbol{F}(\boldsymbol{u}, \boldsymbol{q}) \cdot \boldsymbol{n}]\!]. \tag{13}$$

Here, $\{\{\cdot\}\}$ denotes the averaging operator and $[\![\cdot]\!]$ denotes the jump operator over a face. We set $C_{12} = S_i^{\pm}\boldsymbol{n}/2$, for some *switch function* $S_i^{\pm} \in \{-1, 1\}$ at each side of each element face i. We set $C_{11} = C_{22} = 0$ everywhere, except for Dirichlet-type boundary edges where we impose appropriate fluxes. For more details, we refer to Refs. 10, 1, 11, 9.

With the fluxes defined, our final semi-discrete formulation for (10)-(11) gets the form

$$\frac{d\boldsymbol{u}_{ijk}}{dt} + \frac{1}{J_{ijk}} \sum_{n=1}^{3} \boldsymbol{r}_{ijk}^{(n)} = \boldsymbol{0}, \tag{14}$$

$$\frac{1}{J_{ijk}} \sum_{n=1}^{3} \boldsymbol{d}_{ijk}^{(n)} = \boldsymbol{q}_{ijk}, \tag{15}$$

which we solve either for a steady-state solution, or integrate in time using the method of lines as described in section III.

II.C. Stencil size and sparsity pattern

To illustrate the drastic reduction of the number of entries in the Jacobian matrices for the Line-DG method, consider the $(p+1)^3$ nodes in an (interior) element and its six neighboring elements. We note that a standard nodal DG formulation will in general produce full block matrices, that is, each degree of freedom will depend on all the other ones within the element. In addition, the face integrals will produce dependencies on each node along the face of each neighboring element, for a total of up to $(p+1)^3 + 6(p+1)^2$ connections. This illustrates why matrix-based DG methods are considered memory intensive and expensive even at modest values of p.

As a contrast, in our line-based method each node will only connect to other nodes within the same lines, and to only one node in each neighboring element, for a total of $(3p+1) + 6 = 3p + 7$ connectivities. This is similar to that of the DGSEM/SD methods, although with Gauss-Legendre solution points these schemes also connects entire lines of nodes in the neighboring element, giving a total of $(3p+1) + 6(p+1) = 9p + 7$ connectivities. These numbers are tabulated for a range of degrees p in three dimensions in table 1. The sparsity patterns are illustrated in figure 2 for two-dimensional quadrilateral elements, for all three methods. The connectivities are shown both by a nodal plot, with bold nodes corresponding to the dependencies of the single red node, and by sparsity plots of the Jacobian matrices.

We note that in three dimensions, for $p = 3$ the Line-DG method is 10 times sparser than nodal DG, and for $p = 10$ it is more than 50 times sparser. This reduction in stencil size translates into cheaper assembly times and drastically lower storage requirements for matrix-based solvers.

	Polynomial order p	1	2	3	4	5	6	7	8	9	10
2-D	Line-DG connectivities	7	9	11	13	15	17	19	21	23	25
	Spectral Difference connectivities	11	17	23	29	35	41	47	53	59	65
	Nodal-DG connectivities	12	21	32	45	60	77	96	117	140	165
3-D	Line-DG connectivities	10	13	16	19	22	25	28	31	34	37
	Spectral Difference connectivities	16	25	34	43	52	61	70	79	88	97
	Nodal-DG connectivities	32	81	160	275	432	637	896	1215	1600	2057

Table 1. The number of connectivities per node for 3-D hexahedral elements with the Line-DG, the spectral difference, and the nodal DG methods.

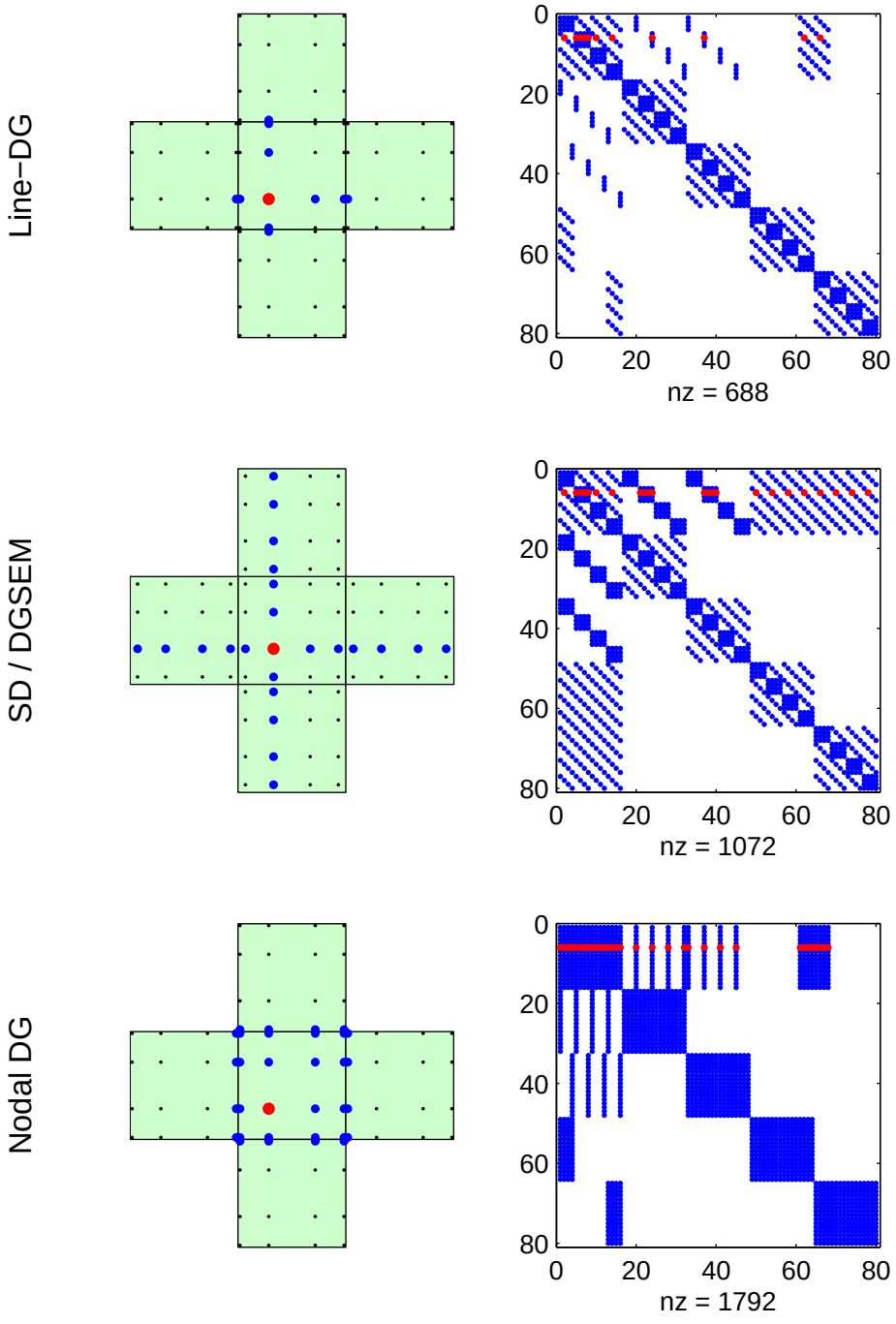


Figure 2. The connectivities (blue circles) to a single node (red circle) for the Line-DG method, the nodal DG method, and the spectral difference method (2-D quadrilateral elements).

III. Temporal discretization and nonlinear solvers

Our final semi-discrete scheme (14)-(15) can be written in the form

$$\frac{d\boldsymbol{U}}{dt} = \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{Q}), \tag{16}$$

$$\boldsymbol{Q} = \boldsymbol{D}(\boldsymbol{U}), \tag{17}$$

for solution vectors $\boldsymbol{U}, \boldsymbol{Q}$ and residual functions $\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{Q})$, $\boldsymbol{D}(\boldsymbol{U})$ defined by the spatial discretization scheme. Eliminating $\boldsymbol{Q}$ from the system, we obtain the primal form

$$\frac{d\boldsymbol{U}}{dt} = \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{D}(\boldsymbol{U})) \equiv \boldsymbol{L}(\boldsymbol{U}). \tag{18}$$

This is clearly in the preferred form for explicit time-stepping, where we use a standard fourth-order Runge-Kutta solver to integrate (18) in time.

III.A. Implicit Newton-Krylov solvers

For implicit time-stepping or steady-state solution, we use Diagonally Implicit Runge-Kutta (DIRK) schemes with Newton's method for the nonlinear systems. In particular, we use the following L-stable, three-stage, third-order accurate method:[12]

$$\boldsymbol{K}_i = \boldsymbol{L}\left(\boldsymbol{U}_n + \Delta t \sum_{j=1}^{s} a_{ij}\boldsymbol{K}_j\right), \quad i = 1, \ldots, s \tag{19}$$

$$\boldsymbol{U}_{n+1} = \boldsymbol{U}_n + \Delta t \sum_{j=1}^{s} b_j \boldsymbol{K}_j, \tag{20}$$

with $s = 3$ and the coefficients given by the Runge-Kutta tableaux:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|ccc} \alpha & \alpha & 0 & 0 \\ \tau_2 & \tau_2 - \alpha & \alpha & 0 \\ 1 & b_1 & b_2 & \alpha \\ \hline & b_1 & b_2 & \alpha \end{array} \qquad \begin{array}{l} \alpha = 0.435866521508459 \\ \tau_2 = (1+\alpha)/2 \\ b_1 = -(6\alpha^2 - 16\alpha + 1)/4 \\ b_2 = (6\alpha^2 - 20\alpha + 5)/4 \end{array}$$

When applied to the reduced system (18), these require the solution of s nonlinear systems of equations (19). Using a Newton method, we then need to solve linear systems of equations of the form

$$(\boldsymbol{I} - \alpha\Delta t \boldsymbol{A})\Delta\boldsymbol{K}_i = \boldsymbol{G} \tag{21}$$

for some vector $\boldsymbol{G}$, where

$$\boldsymbol{A} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{U}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}}\frac{\partial \boldsymbol{D}}{\partial \boldsymbol{U}} = \boldsymbol{K}_{11} + \boldsymbol{K}_{12}\boldsymbol{K}_{21}. \tag{22}$$

The problem with this form is that for second-order systems, the product $\boldsymbol{K}_{12}\boldsymbol{K}_{21}$ is in general much less sparse than the individual matrices $\boldsymbol{K}_{11}, \boldsymbol{K}_{12}, \boldsymbol{K}_{21}$. This is expected due to the repeated differentiation along two different directions, but it requires special solvers to avoid explicitly forming the denser matrix $\boldsymbol{A}$. We also points out that most methods suffer from sparsity reduction for second-order systems, including standard finite different methods as well as nodal-DG methods.[13]

In this work, we use a simple but efficient approach to solve the systems (21). In a restarted GMRES method,[14] we need to perform two operations: Multiplication of a vector $\boldsymbol{p}$ by the matrix $(\boldsymbol{I} - \alpha\Delta t\boldsymbol{A})$, and approximate solution of $(\boldsymbol{I} - \alpha\Delta t\boldsymbol{A})\boldsymbol{x} = \boldsymbol{b}$ for the preconditioning. The matrix-vector product is straightforward, by keeping the individual matrix in a separated form and nesting the products:

$$(\boldsymbol{I} - \alpha\Delta t\boldsymbol{A})\boldsymbol{p} = \boldsymbol{p} - \alpha\Delta t\left(\boldsymbol{K}_{11}\boldsymbol{p} + \boldsymbol{K}_{12}(\boldsymbol{K}_{21}\boldsymbol{p})\right). \tag{23}$$

This can be done without explicitly forming the matrix $\boldsymbol{A}$, and the cost per matrix-vector product is proportional to the number of entries in the matrices $\boldsymbol{K}_{11}, \boldsymbol{K}_{12}, \boldsymbol{K}_{21}$.

For preconditioning, we use a block-Jacobi approach. The diagonal blocks of $\boldsymbol{A}$ corresponding to each element are computed and factorized using dense linear solvers. We note that this does produce fill in our original sparsity pattern, however, since it only involves the diagonal blocks the total number of entries are comparable to that of $\boldsymbol{A}$. This will likely not be the case for 3-D problems, where more sophisticated preconditioners should be considered, such as p-multigrid, block-ILU,[15] subiterations, etc.

In our implementation, we store all matrices using a general purpose compressed column storage.[16] We also point out that the matrix $\boldsymbol{K}_{21}$ can be handled very efficiently, since it is a discrete gradient operator and therefore (a) linear, (b) constant in time, and (c) equal for all solution components (except possibly at the boundaries).

III.B. Quasi-Newton solver

In the solution of the nonlinear equations (19), it is often more computationally expensive to form the matrix $\boldsymbol{I} - \alpha \Delta t \boldsymbol{A}$ than to solve the linear system (21). This is especially true for time-accurate integration where the timesteps Δt are relatively small (but still large enough to motivate the use of implicit solvers). A standard procedure is to then use *quasi-Newton* methods, which allows for inconsistent Jacobian matrices at the cost of slower convergence. A natural choice is to keep a previously generated matrix for as long as possible, as determined by a control mechanism that monitors the Newton convergence.

Our algorithm is simple and straight-forward. We recompute the matrix and the preconditioner in the following cases:

1. First Newton step of the first timestep (since no prior Jacobian exists)
2. If the residual norm is 10 times larger than at the previous Newton iteration (divergence)
3. If the number of Newton steps equals 15 (too slow convergence)

Clearly this simple strategy can be much improved on, but as we show in our numerical experiments this is sufficient to allow for a reuse of the Jacobian for several Newton steps, DIRK stages, and timesteps at a time.

IV. Numerical Results

IV.A. Euler Vortex

First we consider the Euler equations, and a model problem of a compressible vortex in a rectangular domain.[17] The vortex is initially centered at (x_0, y_0) and is moving with the free-stream at an angle θ with respect to the x-axis. The analytic solution at (x, y, t) is given by

$$u = u_\infty \left(\cos\theta - \frac{\epsilon((y - y_0) - \bar{v}t)}{2\pi r_c} \exp\left(\frac{f(x,y,t)}{2} \right) \right), \tag{24}$$

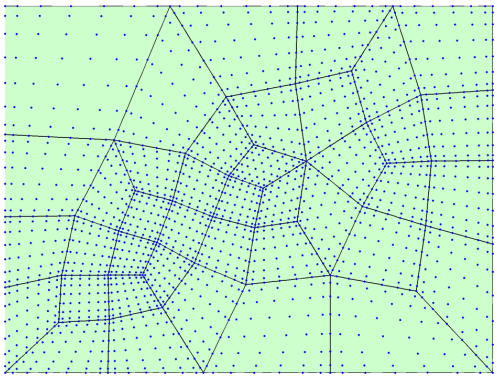
$$v = u_\infty \left(\sin\theta + \frac{\epsilon((x - x_0) - \bar{u}t)}{2\pi r_c} \exp\left(\frac{f(x,y,t)}{2} \right) \right), \tag{25}$$

$$\rho = \rho_\infty \left(1 - \frac{\epsilon^2(\gamma - 1)M_\infty^2}{8\pi^2} \exp\left(f(x,y,t)\right) \right)^{\frac{1}{\gamma - 1}}, \tag{26}$$

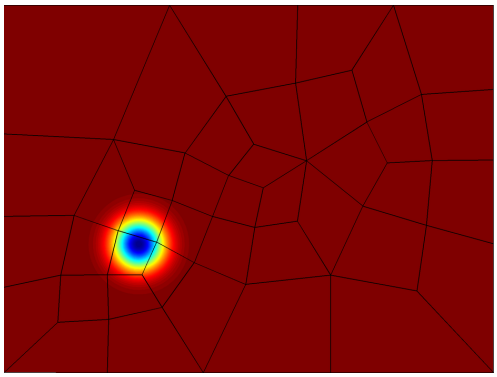
$$p = p_\infty \left(1 - \frac{\epsilon^2(\gamma - 1)M_\infty^2}{8\pi^2} \exp\left(f(x,y,t)\right) \right)^{\frac{\gamma}{\gamma - 1}}, \tag{27}$$

where $f(x,y,t) = (1 - ((x - x_0) - \bar{u}t)^2 - ((y - y_0) - \bar{v}t)^2)/r_c^2$, M_∞ is the Mach number, $\gamma = c_p/c_v$, and u_∞, p_∞, ρ_∞ are free-stream velocity, pressure, and density. The Cartesian components of the free-stream velocity are $\bar{u} = u_\infty \cos\theta$ and $\bar{v} = u_\infty \sin\theta$. The parameter ϵ measures the strength of the vortex and r_c is its size.

We use a domain of size 20-by-15, with the vortex initially centered at $(x_0, y_0) = (5, 5)$ with respect to the lower-left corner. The Mach number is $M_\infty = 0.5$, the angle $\theta = \arctan 1/2$, and the vortex has



Coarsest mesh, with degree $p = 7$



Solution (density)

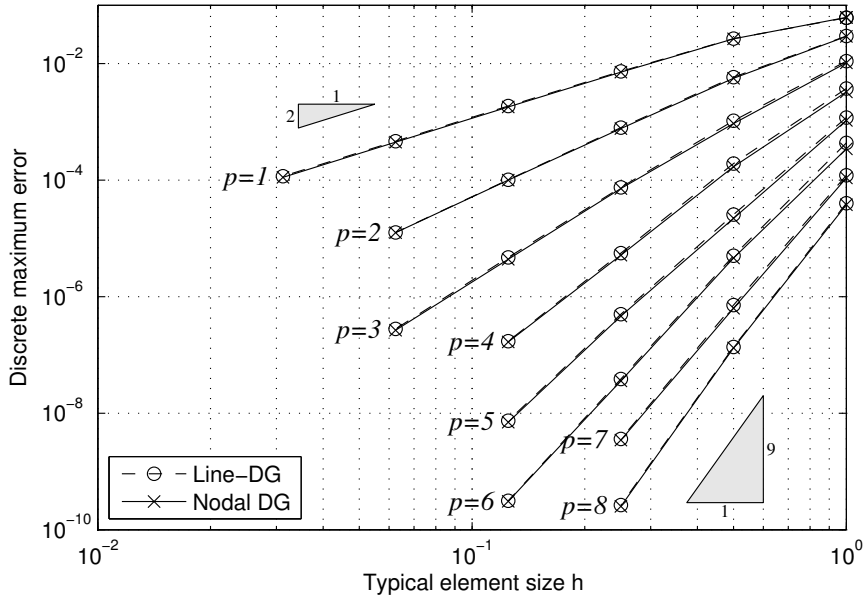


Figure 3. Convergence test for an Euler vortex test problem using the Line-DG method and the nodal-DG method. The results show optimal order of convergence $\mathcal{O}(h^{p+1})$ with very small differences between the two methods.

the parameters $\epsilon = 0.3$ and $r_c = 1.5$. We use characteristic boundary conditions and integrate until time $t_0 = \sqrt{10^2 + 5^2}/10$, when the vortex has moved a relative distance of $(1, 1/2)$.

We write the Euler equations as a first-order system of conservation laws (1), in the conserved variables $(\rho, \rho u, \rho v, \rho E)$. The scheme (7) is implemented in a straight-forward way, and we use Roe's method for the numerical fluxes (4).[18] The time-integration is done explicitly with the form (18) using the RK4 solver and a timestep Δt small enough so that all truncation errors are dominated by the spatial discretization. We start from a coarse unstructured quadrilateral mesh (figure 3, top left), which we refine uniformly a number of times for a range of polynomial degrees p. The top right plot also shows the density field for a sample solution.

In the bottom plot of figure 3, we graph the maximum errors (discretely at the solution nodes) for all simulation cases, both for the Line-DG method and the standard nodal DG method. The results clearly show the optimal order of convergence $\mathcal{O}(h^{p+1})$ for element size h for both methods, and that the Line-DG errors are in all cases very close to those of the nodal-DG method.

IV.B. Inviscid flow over a cylinder

Next we study a problem with steady-state simulation and curved boundaries, and solve the Euler equations for the inviscid flow over a half-cylinder with radius 1 at a Mach number of 0.3. Structured quadrilateral

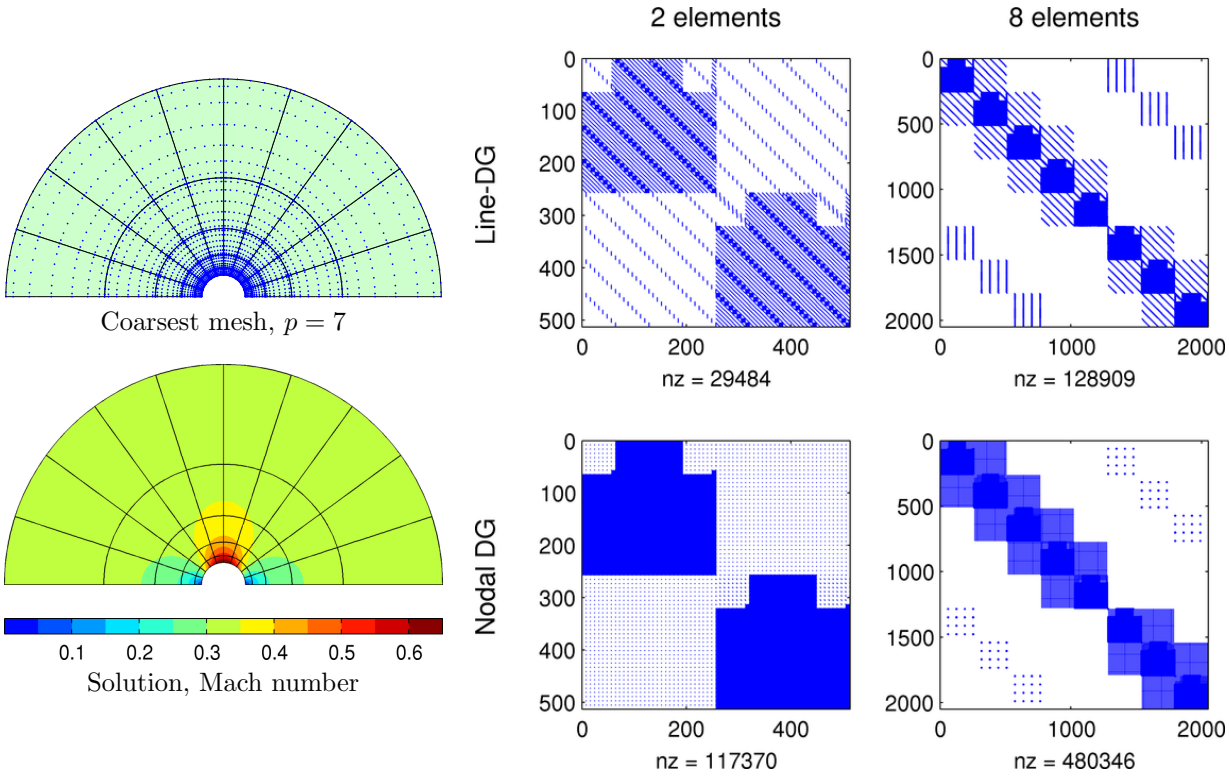


Figure 4. Inviscid flow over a cylinder. The plots show the coarsest grid used in the convergence study and the nodes for polynomial degree $p = 7$ (top left), the corresponding solution as Mach number color plot (bottom left), and a sparsity plot of the Jacobian matrices for both Line-DG and nodal DG (right).

meshes are used, with strong element size grading to better resolve the region close to the cylinder (see figure 4, top left). The outer domain boundary is a half-cylinder with radius 10, where characteristics boundary conditions are imposed. Standard slip wall/symmetry conditions are used at the cylinder and at the symmetry plane.

The steady-state solutions are found using a fully consistent Newton method, applied directly to the equations (7), with the linear systems solved using a direct sparse solver.[16] Starting the iterations from an approximate analytical solution, derived from a potential flow model, the solver converges to machine precision in 4 to 6 iterations. The solution is shown in the bottom left of figure 4, and to the right portions of the Jacobian matrices are shown for both the Line-DG and the nodal DG method. This illustrates again the reduced sparsity of the Line-DG scheme, with about a factor of 4 fewer entries than nodal DG already in two space dimensions.

To evaluate the accuracy and convergence of the scheme, in figure 5 we plot the errors in the lift coefficient C_L (left) and the maximum errors in the entropy (right). These plots again confirm the convergence of the schemes as well as the small variations between the Line-DG and the nodal DG schemes.

IV.C. Laminar flow around airfoil

An example of a steady-state viscous computation is shown in figure 6. The compressible Navier-Stokes equations are solved at Mach 0.2 and Reynolds number 5,000, for a flow around an SD7003 airfoil. The quadrilateral mesh is fully unstructured except for a structured graded boundary layer region, with a total of 461 elements for the coarse mesh and 1,844 elements for the refined mesh. With approximating polynomials of degree $p = 7$, this gives a total number of high-order nodes of 29,504 and 118,016, respectively.

We find the steady-state solution with 10 digits of accuracy in the residual using a consistent Newton's method, with pseudo-timestepping for regularization. A solution is shown in figure 6 (top right), for the coarse mesh with $p = 7$. In the bottom plots, we show the convergence of the drag and the lift coefficients, for increasing values of p on the two meshes. While it is hard to asses the order of convergence from these numbers, it is clear that the coefficients appear to converge with increasing degrees p.

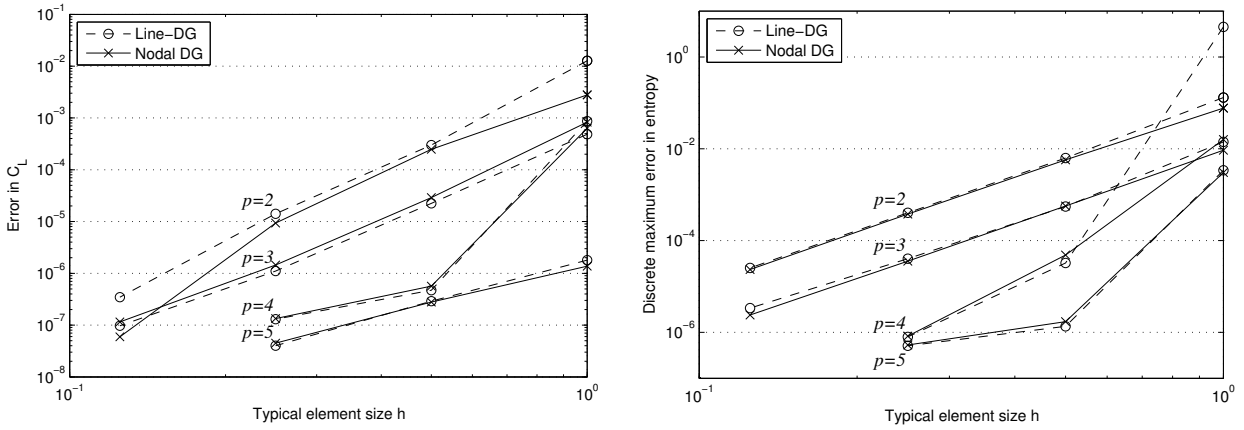


Figure 5. The convergence of the lift coefficient C_L (left) and the entropy difference (right) for the inviscid flow over cylinder problem. The plots show a series of results for varying polynomial degrees and number of refinements, for the two methods Line-DG and nodal DG.

IV.D. Transient flow around airfoil at Re=20,000

In our last example, we demonstrate time-accurate implicit solution of transient flow around an SD7003 airfoil at Re=20,000. The mesh is highly resolved in the boundary layer, however, for this Reynolds number it is coarser than the flow features in much of the domain and the computations should therefore be considered an under-resolved ILES-type model.[19]

The Mach number is 0.1 and the angle of attack is 30 degrees to force flow separation at the leading edge. We use the three-stage DIRK scheme (19)-(20), solved with the quasi-Newton method described in section III.B. The mesh has 1,122 quadrilateral elements with polynomial degrees $p = 4$. The mesh and a solution at the normalized time of $t = 0.6$ are shown in the top plots of figure 7.

We have not performed an accuracy study for this problem, but instead we investigate the performance of the implicit scheme. In the bottom graphs of figure 7, we plot the number of iterations in the quasi-Newton scheme for each nonlinear system (one for each stage of the DIRK scheme, or three for each timestep). The horizontal line at 15 iterations shows where the Jacobian is recalculated (rule number 3 in the control algorithm of section III.B). Rule number 2 is only in effect for a few iterations in the beginning. In particular, we note that Jacobians are computed only once in about every 5th solve, leading to a negligible total cost of matrix assembly.

To illustrate the performance of the preconditioned Krylov solver, the bottom plot in figure 7 shows the number of GMRES iterations used for each linear solve, which is always between 10 and 25. Because of the sparsity of the matrices and the splitting (23), these iterations are relatively inexpensive compared to residual evaluation. This has the consequence that a majority of the solution time is spent in the residual evaluations, which means our scheme gives the benefits of a fully implicit scheme (in particular the ability to handle stiff systems without timestep restrictions) at a cost similar to that of an explicit scheme.

V. Conclusions

We have presented the Line-DG method for high-order Navier-Stokes simulations, and studied a number of problems ranging from benchmark to more realistic once, although currently only in two space dimensions. The main difference of the scheme compared to the standard nodal DG method is a fundamentally different sparsity structure, which we used to develop efficient matrix-based implicit solvers. We showed that the accuracy of the discretizations are very similar to the standard DG method, and we demonstrated essentially explicit performance for a high-order DIRK time-integration scheme with quasi-Newton solvers, with computational cost dominated by the residual evaluations.

A number of further studies are currently being performed, in particular the development of more efficient preconditioners, 3-D and parallel implementations of the schemes, and nonlinear stabilization for shocks and other under-resolved features.

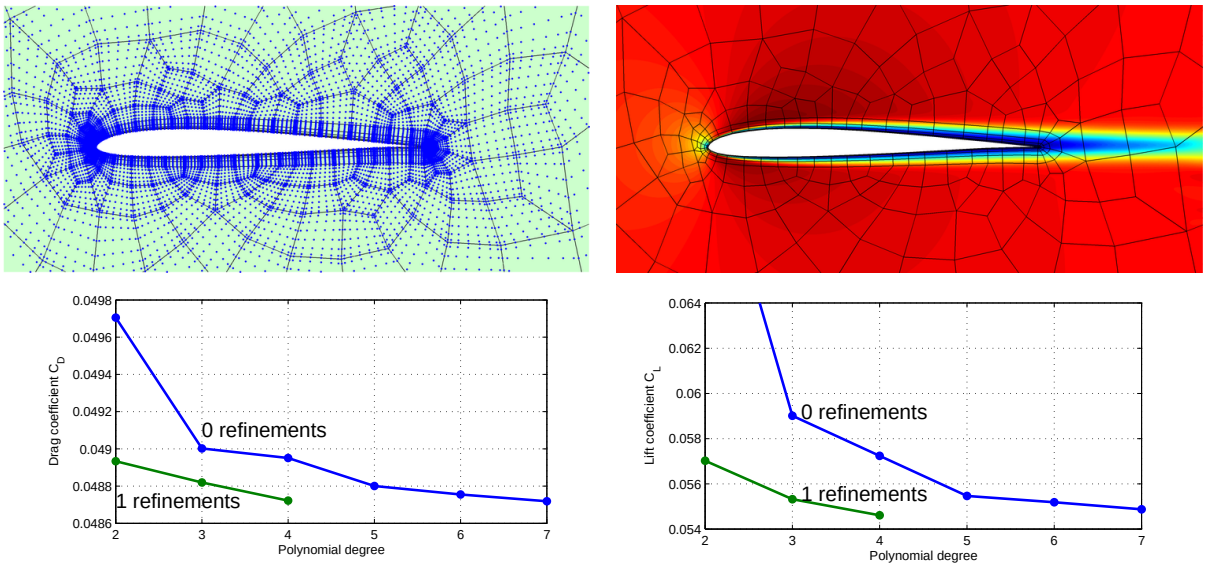


Figure 6. Stationary flow around an SD7003 airfoil (top left: mesh, top right: Mach number), computed with the Line-DG method with $p = 7$, at free-stream Mach 0.2, zero angle of attack, and Reynolds number 5,000. The bottom plots show the convergence of C_D and C_L, for a range of polynomial degrees and with 0 or 1 uniform mesh refinements.

References

[1]Cockburn, B. and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *J. Sci. Comput.*, Vol. 16, No. 3, 2001, pp. 173–261.

[2]Kopriva, D. A. and Kolias, J. H., "A conservative staggered-grid Chebyshev multidomain method for compressible flows," *J. Comput. Phys.*, Vol. 125, No. 1, 1996, pp. 244–261.

[3]Hesthaven, J. S. and Warburton, T., *Nodal discontinuous Galerkin methods*, Vol. 54 of *Texts in Applied Mathematics*, Springer, New York, 2008, Algorithms, analysis, and applications.

[4]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.

[5]Huynh, H., "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods," *18th AIAA Computational Fluid Dynamics Conference, Miami, Florida*, 2007, AIAA-2007-4079.

[6]Vincent, P. E., Castonguay, P., and Jameson, A., "A new class of high-order energy stable flux reconstruction schemes," *J. Sci. Comput.*, Vol. 47, No. 1, 2011, pp. 50–72.

[7]Peraire, J. and Persson, P.-O., *Adaptive High-Order Methods in Computational Fluid Dynamics*, Vol. 2 of *Advances in CFD*, chap. 5 – High-Order Discontinuous Galerkin Methods for CFD, World Scientific Publishing Co., 2011.

[8]Haga, T., Gao, H., and Wang, Z., "A High-Order Unifying Discontinuous Formulation for 3D Mixed Grids," *48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida*, 2010, AIAA-2010-540.

[9]Persson, P.-O., "A Sparse Line-Based Discontinuous Galerkin Method," in review.

[10]Cockburn, B. and Shu, C.-W., "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM J. Numer. Anal.*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[11]Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., "Unified analysis of discontinuous Galerkin methods for elliptic problems," *SIAM J. Numer. Anal.*, Vol. 39, No. 5, 2001/02, pp. 1749–1779.

[12]Alexander, R., "Diagonally implicit Runge-Kutta methods for stiff o.d.e.'s," *SIAM J. Numer. Anal.*, Vol. 14, No. 6, 1977, pp. 1006–1021.

[13]Peraire, J. and Persson, P.-O., "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM J. Sci. Comput.*, Vol. 30, No. 4, 2008, pp. 1806–1824.

[14]Saad, Y. and Schultz, M. H., "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, Vol. 7, No. 3, 1986, pp. 856–869.

[15]Persson, P.-O. and Peraire, J., "Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations," *SIAM J. Sci. Comput.*, Vol. 30, No. 6, 2008, pp. 2709–2733.

[16]Davis, T. A., *Direct methods for sparse linear systems*, Vol. 2 of *Fundamentals of Algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[17]Erlebacher, G., Hussaini, M. Y., and Shu, C.-W., "Interaction of a shock with a longitudinal vortex," *J. Fluid Mech.*, Vol. 337, 1997, pp. 129–153.

[18]Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Comput. Phys.*, Vol. 43, No. 2, 1981, pp. 357–372.

[19]Uranga, A., Persson, P.-O., Drela, M., and Peraire, J., "Implicit Large Eddy Simulation of transition to turbulence at low Reynolds numbers using a Discontinuous Galerkin method," *Int. J. Num. Meth. Eng.*, Vol. 87, No. 1-5, 2011, pp. 232–261.

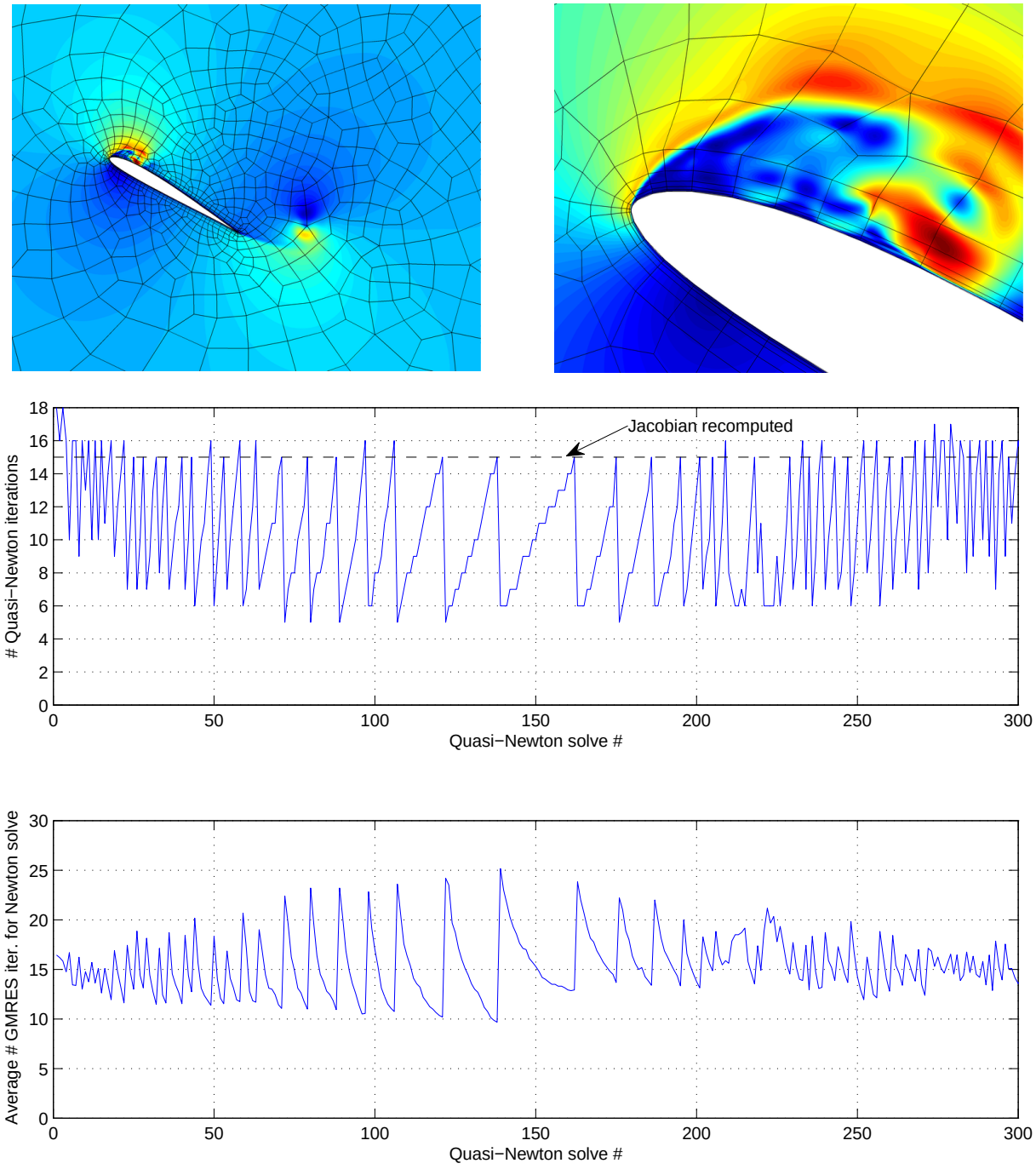


Figure 7. Implicit transient simulation of flow around an SD7003 airfoil, at 30 degrees angle of attack, Reynolds number 20,000, and Mach number 0.2. A three stage, third order accurate DIRK scheme is used for time integration, and the nonlinear systems are solved with a quasi-Newton-Krylov solver. The bottom graphs show the number of iterations per Newton solve, and the average number of Krylov iterations for each Newton solve. Since the Jacobians are reused for many iterations, and the matrix-vector products are inexpensive due to the sparsity, the majority of the computational cost is due to the residual evaluations (similar to an explicit solver).