ELSEVIER
Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics xxx (2007) xxx–xxx
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Application of implicit–explicit high order Runge–Kutta methods to discontinuous-Galerkin schemes

Alex Kanevsky [a,*], Mark H. Carpenter [b], David Gottlieb [a], Jan S. Hesthaven [a]

[a] *Division of Applied Mathematics, Brown University, Box F, Providence, RI 02912, USA*
[b] *Aeronautics and Aeroacoustic Methods Branch, NASA Langley Research Center, Hampton, VA 23681-0001, USA*

Received 6 September 2006; received in revised form 9 February 2007; accepted 16 February 2007

Abstract

Despite the popularity of high-order explicit Runge–Kutta (ERK) methods for integrating semi-discrete systems of equations, ERK methods suffer from severe stability-based time step restrictions for very stiff problems. We implement a discontinuous Galerkin finite element method (DGFEM) along with recently introduced high-order implicit–explicit Runge–Kutta (IMEX-RK) schemes to overcome geometry-induced stiffness in fluid-flow problems. The IMEX algorithms solve the non-stiff portions of the domain using explicit methods, and isolate and solve the more expensive stiff portions using an L-stable, stiffly-accurate explicit, singly diagonally implicit Runge–Kutta method (ESDIRK). Furthermore, we apply adaptive time-step controllers based on the embedded temporal error predictors. We demonstrate in a number of numerical test problems that IMEX methods in conjunction with efficient preconditioning become more efficient than explicit methods for systems exhibiting high levels of grid-induced stiffness.

Keywords: High-order methods; Discontinuous Galerkin finite element method (DGFEM); Implicit–explicit Runge–Kutta (IMEX-RK) schemes; Navier–Stokes equations

1. Introduction

In this paper, we are interested in alleviating the severe stability-based time-step restrictions that affect explicit time integration schemes when applied to problems that exhibit high levels of geometry-induced stiffness. Geometry-induced stiffness, or scale-separation stiffness, is a result of attempting to simultaneously simulate a system that has geometric features of drastically varying scales, and is defined in Section 3.2.

One example of this effect in the field of computational electromagnetics (CEM) occurs when attempting to simulate EM scattering off of a jet fighter, whose very thin stealth coating is much smaller than the other aircraft dimensions. Such a stealth coating can be discretized using proportionately few high-order elements.

[*] Corresponding author.
E-mail addresses: kanevsky@dam.brown.edu (A. Kanevsky), Mark.H.Carpenter@nasa.gov (M.H. Carpenter), dig@dam.brown.edu (D. Gottlieb), Jan.Hesthaven@Brown.edu (J.S. Hesthaven).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.02.021

However, introducing these relatively small elements will result in a very high stiffness (on the order of 10^3) and a very small time step, since the stable time step for the scheme will be determined by the smallest-sized element. As a result, current algorithms in CEM can only handle purely harmonic (up to 10 GHz plane wave) scattering by fighter aircraft, which are assumed to be pure metallic shells, and cannot handle the inclusion of coatings, penetration into and radiation out of the aircraft.

Another important example can be found in computational fluid dynamics (CFD), where the elements used to discretize the boundary layer near an airfoil can often result in a geometry-induced stiffness on the order of 10^3–10^4 or greater depending on the Reynolds number, and will thus severely restrict the maximum stable time step. Mesh generation may also result in high stiffness if a small percentage of "poor" elements are considerably more skewed than the average element.

The basic form of time-dependent algorithms has not changed in the last 30–40 years. Explicit methods are the most efficient methods for long-time simulations of non-stiff systems, while implicit methods are more efficient for solving stiff systems. One approach that has been used to increase the efficiency of explicit methods for stiff equations is based on explicit local timestepping schemes (often called multi-rate integration) , where equations on individual cells or elements are integrated using different local time-steps. Osher and Sanders introduced a local time stepping method for one-dimensional conservation laws in [32]. Other examples of such schemes include [5,15,12,39,34].

A disadvantage with multi-rate methods is that they are generally implemented at 2nd-order (or lower) temporal accuracy. Methods higher than 2nd-order exist, but suffer increasing implementation complexity. Even 2nd-order multi-rate methods suffer difficulties contending with irregular unstructured engineering meshes for which elements can range in size by many orders of magnitude.

Implicit–explicit or IMEX algorithms were originally developed to solve the stiff term or operator of convection–diffusion–reaction (CDR) type equations implicitly and the nonstiff term explicitly [4]. A number of IMEX Runge–Kutta methods have been developed in recent times, such as [3,8,13,16,42,43], which combine ERK schemes with diagonally implicit Runge–Kutta (DIRK) schemes. However, these schemes have various drawbacks, such as lower-order coupling errors, coupling stability problems, no error control, and poor ERK or DIRK stability properties.

The recently-developed additive Runge–Kutta (ARK) methods in [26] can be used for the classical operator-based IMEX time-splitting or a geometric region-based IMEX time-splitting. They allow for integration of stiff terms by an L-stable, stiffly-accurate explicit, singly diagonally implicit Runge–Kutta method (ESDIRK), and integration of nonstiff terms by an explicit Runge–Kutta method (ERK). Furthermore, they provide extrapolation-based stage-value predictors as well as embedded schemes (one order lower) which allow for the use of automatic error-based time-step controllers, such as integral (I), proportional-integral (PI) and proportional-integral-derivative (PID) controllers, which are defined in Section 3.3.8. We implement the high-order implicit–explicit Runge–Kutta (IMEX-RK) methods of Kennedy and Carpenter [26] to overcome geometry-induced stiffness. IMEX algorithms solve the non-stiff portions of the domain using explicit methods, and isolate and solve the more expensive stiff portions (e.g. stealth coating or boundary layer) using implicit methods.

We follow the method of lines approach, and discretize space using a nodal discontinuous Galerkin spectral element method based on [21,22]. The discontinuous Galerkin method is a class of finite element methods using a completely discontinuous piecewise polynomial space for the numerical solution and the test functions. The first discontinuous Galerkin method was introduced in 1973 by Reed and Hill [36], in the framework of neutron transport (steady state linear hyperbolic equations).

Since then, the discontinuous Galerkin method has been applied in a number of fields, such as aeroacoustics, electro-magnetism, gas dynamics, granular flows, magneto-hydrodynamics, meteorology, modeling of shallow water, oceanography, oil recovery simulation, semiconductor device simulation,turbulent flows, viscoelastic flows and weather forecasting. For a detailed description of the method as well as its implementation and applications, we refer readers to the lecture notes [10] and the papers in Springer volume [11]. The discontinuous Galerkin finite element (DGFEM) method builds upon the strengths of the classical spectral element method introduced by Patera [33], and has a number of advantages over classical finite difference and finite volume methods. DGFEM methods are especially well suited for IMEX algorithms, since they allow for clean and easy decoupling of the stiff from the nonstiff regions of the domain. Furthermore, they are highly parallelizable and accurate, provide for simple treatment of boundary conditions, handle complicated geometries well, and can easily handle adaptivity.

This paper is organized as follows. In Section 2, we discuss the details of the spatial discretization scheme, which is based on a nodal discontinuous Galerkin finite element method (DGFEM). We review the properties and characteristics of implicit–explicit Runge–Kutta (IMEX-RK) time-integration methods in Section 3. Numerical results comparing IMEX-RK and ERK schemes for various test problems are presented in Section 4. Finally, we discuss all the results and give concluding remarks in Section 5.

2. Spatial discretization

2.1. Two-dimensional scheme

The nodal discontinuous Galerkin (DG) finite element spatial discretization is based on [21–23]. We now review the details for a two-dimensional spatial discretization, although a generalization to the three-dimensional case is fairly straightforward. Assume that we have a multi-dimensional wellposed conservation law

$$\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{u}(\mathbf{x},t)) = 0, \quad \mathbf{x} \in \mathbf{\Omega},\ t \geqslant 0 \tag{2.1}$$

with initial and boundary conditions

$$\begin{aligned}&\mathbf{u}(\mathbf{x},0) = f(\mathbf{x}), \quad \mathbf{x} \in \mathbf{\Omega}\\ &\mathbf{u}(\mathbf{x},t) = g(\mathbf{x}), \quad \mathbf{x} \in \delta\mathbf{\Omega},\ t \geqslant 0,\end{aligned}$$

where $\mathbf{u}$ is the state vector of unknown/s, and $\mathbf{F}(\mathbf{u})$ is the flux. We assume that our computational domain $\mathbf{\Omega}$ is composed of K non-overlapping d-simplices or elements

$$\mathbf{\Omega} = \bigcup_{k=1}^{K} \mathbf{D}^k. \tag{2.2}$$

In two dimensions, we will assume that the elements are 2-simplexes or triangles to allow for fully unstructured meshes. We also assume that the triangles have straight sides, which results in a constant transformation Jacobian for all elements, and greatly simplifies the scheme. The reference or standard triangle $\mathbf{I} \subset R^2$ has the three vertices

$$\mathbf{v}_{\mathrm{I}} = \begin{bmatrix}-1\\-1\end{bmatrix}, \quad \mathbf{v}_{\mathrm{II}} = \begin{bmatrix}1\\-1\end{bmatrix}, \quad \mathbf{v}_{\mathrm{III}} = \begin{bmatrix}-1\\1\end{bmatrix}, \tag{2.3}$$

while the physical simplex or subdomain $\mathbf{D}^k$ has the three corresponding vertices $\mathbf{v}_1^k$, $\mathbf{v}_2^k$, and $\mathbf{v}_3^k$ as can be seen in Fig. 1. Also, element $\mathbf{D}^k$ has physical coordinates $\mathbf{x} = (x,y)$, while the reference element $\mathbf{I}$ has coordinates $\boldsymbol{\xi} = (\xi,\eta)$. $\mathbf{D}^k$ and $\mathbf{I}$ are related through the linear, invertible map $\mathbf{\Psi}$

$$\mathbf{\Psi} : \mathbf{I} \to \mathbf{D} \Rightarrow \mathbf{\Psi}^{-1} : \mathbf{D} \to \mathbf{I}. \tag{2.4}$$

We construct the linear map $\mathbf{\Psi}$ given as

$$\mathbf{x} = \mathbf{\Psi}(\xi,\eta) = -\left(\frac{\xi+\eta}{2}\right)\mathbf{v}_1^k + \left(\frac{1+\xi}{2}\right)\mathbf{v}_2^k + \left(\frac{1+\eta}{2}\right)\mathbf{v}_3^k. \tag{2.5}$$

We assume that the solution in each subdomain $\mathbf{D}^k$ is well approximated by the local polynomial of degree p

$$\mathbf{u}^k(\mathbf{x},t) = \sum_{i=0}^{N} \mathbf{u}^k(\mathbf{x}_i^k,t) L_i^k(\mathbf{x}) = \sum_{i=0}^{N} \mathbf{u}_i^k(t) L_i^k(\mathbf{x}), \tag{2.6}$$

where $\mathbf{x}_i^k$ are the $N+1$ grid points in the kth element and $L_i^k(\mathbf{x})$ is the two-dimensional multivariate Lagrange polynomial based on these points

$$L_i(\mathbf{x}) \in P_{\mathrm{p}}^2 = \mathrm{span}\{x^i y^j; i,j \geqslant 0; i+j \leqslant p\}. \tag{2.7}$$

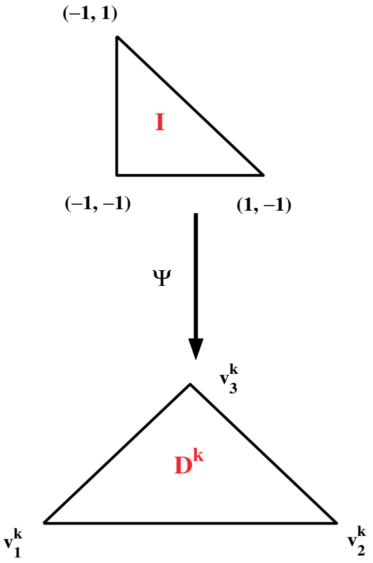


Fig. 1. Linear mapping $\mathbf{\Psi}$ from reference element $\mathbf{I}$ to element $\mathbf{D}^k$ in 2D.

Note that

$$N = \frac{(p+1)(p+2)}{2} - 1, \tag{2.8}$$

and $N+1$ is the total number of grid points necessary in 2D for polynomials of degree p. The physical flux $\mathbf{F}$ is approximated as

$$\mathbf{F}^k(\mathbf{u}^k) = \sum_{i=0}^{N} \mathbf{F}^k(\mathbf{u}^k(\mathbf{x}_i, t))L_i^k(\mathbf{x}). \tag{2.9}$$

We express the local polynomials in a more general framework

$$\mathbf{u}^k(\mathbf{x}, t) = \sum_{i=0}^{N} \mathbf{u}_i^k(t)L_i^k(\mathbf{x}) = \sum_{n=0}^{N} \hat{\mathbf{u}}_n^k(t)\phi_n(\mathbf{x}), \tag{2.10}$$

where $\phi_n(\mathbf{x})$ are the basis functions defined on the kth element, while $\hat{\mathbf{u}}_n^k(t)$ are the modal coefficients. A polynomial basis such as the multivariate monomials $\phi_{ij}(\mathbf{x}) = x^i y^j$ will result in a nearly dependent basis, and therefore a poorly conditioned Vandermonde matrix (grows exponentially with p). We choose an orthonormal basis that has been rediscovered on several occasions by Dubiner [14], Proriol [35] and Koornwinder [29]

$$\tilde{\phi}_{ij}(\xi, \eta) = P_i^{(0,0)}\left(\frac{2(\xi+1)}{(1-\eta)} - 1\right)\left(\frac{1-\eta}{2}\right)^i P_j^{(2i+1,0)}(\eta), \tag{2.11}$$

$$\phi_{ij}(\xi, \eta) = \frac{\tilde{\phi}_{ij}(\xi, \eta)}{\sqrt{\gamma_{ij}}}, \quad \gamma_{ij} = \left(\frac{2}{2i+1}\right)\left(\frac{1}{i+j+1}\right), \tag{2.12}$$

where $P_n^{(\alpha,\beta)}$ is the Jacobi polynomial of order n, which are orthogonal on $\mathbf{I}$, and γ_{ij} is the orthonormalizing weight.

We choose the grid points x_j^k, $j = 0, 1, \ldots, N$, computed as the steady state, minimum energy solution to an electrostatics problem on an equilateral triangle by Hesthaven in [20]. The distribution is illustrated in Fig. 2 for polynomial degrees p ranging from 2 to 12 on the reference element $\mathbf{I}$. Note that this grid distribution becomes the Legendre–Gauss–Lobatto distribution along the edges of the triangle.

We define the vectors of nodal and modal values on $\mathbf{D}^k$ as

$$\mathbf{u}_N^k = [\mathbf{u}_0^k, \ldots, \mathbf{u}_N^k]^{\mathrm{T}}, \quad \hat{\mathbf{u}}_N^k = [\hat{\mathbf{u}}_0^k, \ldots, \hat{\mathbf{u}}_N^k]^{\mathrm{T}}, \tag{2.13}$$

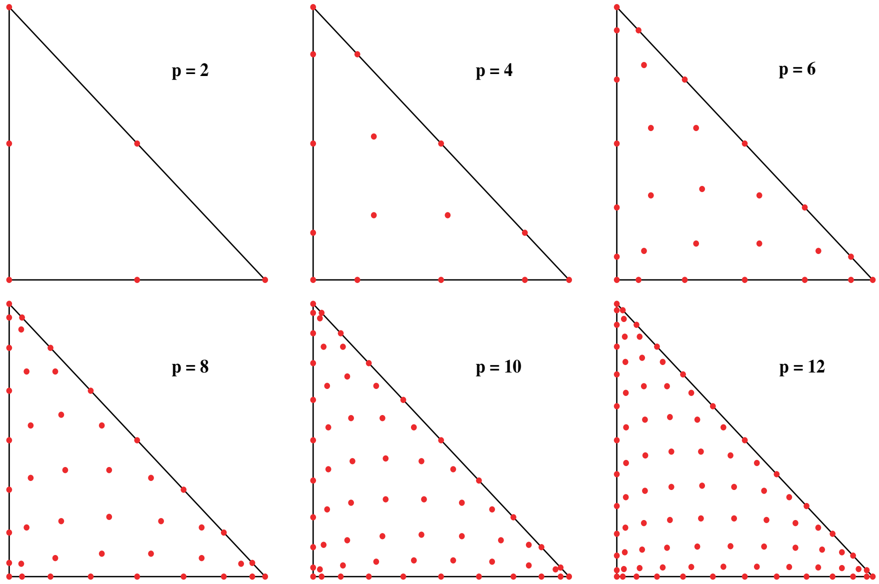


Fig. 2. Electrostatic node distribution [20] on **I**.

and the vectors of local Lagrange polynomials and basis functions on $\mathbf{D}^k$ as

$$\mathbf{L}_N^k = [L_0^k, \ldots, L_N^k]^{\mathrm{T}}, \quad \boldsymbol{\phi}_N^k = [\phi_0^k, \ldots, \phi_N^k]^{\mathrm{T}}. \tag{2.14}$$

Let us simplify our notation for ϕ_{ij} by defining a new index $\alpha \in [0, N]$ that represents a reordering of (i, j) and rewrite $\phi_\alpha = \phi_{ij}$. The Vandermonde matrix is defined to be

$$\mathbf{V}_{i\alpha} = \phi_\alpha(x_i). \tag{2.15}$$

This implies that

$$\mathbf{u}_N^k = \mathbf{V}\hat{\mathbf{u}}_N^k, \quad \hat{\mathbf{u}}_N^k = \mathbf{V}^{-1}\mathbf{u}_N^k, \quad \mathbf{V}^T\mathbf{L}_N^k = \boldsymbol{\phi}_N^k. \tag{2.16}$$

We implement a Galerkin projection methodology and integrate

$$\frac{\partial \mathbf{u}_N^k}{\partial t} + \nabla \cdot \mathbf{F}_N^k = 0 \tag{2.17}$$

against a sequence of $N+1$ test functions $L_i(\mathbf{x})$. After integrating by parts twice, we get the final form of the scheme

$$\int_{\mathbf{D}^k} \left(\frac{\partial \mathbf{u}_N^k}{\partial t} + \nabla \cdot \mathbf{F}_N^k \right) L_i^k(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \oint_{\delta \mathbf{D}^k} L_i^k(\mathbf{x}) \hat{\mathbf{n}} \cdot [\mathbf{F}_N^k - \mathbf{F}_N^*] \, \mathrm{d}\mathbf{x}. \tag{2.18}$$

The numerical flux is the local Lax–Friedrichs flux [30,31]

$$\mathbf{F}_N^* = \mathbf{F}_N^*(\mathbf{u}^-, \mathbf{u}^+) = \frac{\mathbf{F}_N(\mathbf{u}^+) + \mathbf{F}_N(\mathbf{u}^-)}{2} - \frac{|\lambda|}{2}(\mathbf{u}^+ - \mathbf{u}^-), \tag{2.19}$$

where $\mathbf{u}^-$ refers to the local solution, $\mathbf{u}^+$ refers to the neighboring solution/s, and λ is the maximum local eigenvalue of the flux Jacobian.

The mass and stiffness matrices on **I** are

$$\mathbf{M}_{ij} = (L_i(\xi), L_j(\xi))_{\mathbf{I}} = \int_{\mathbf{I}} L_i(\xi) L_j(\xi) \, \mathrm{d}\xi, \tag{2.20}$$

$$\mathbf{S}_{ij} = (\mathbf{S}_{ij}^\xi, \mathbf{S}_{ij}^\eta)_{\mathbf{I}} = (L_i(\xi), \nabla L_j(\xi))_{\mathbf{I}} = \int_{\mathbf{I}} L_i(\xi) \nabla L_j(\xi) \, \mathrm{d}\xi, \tag{2.21}$$

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

and are computed in two dimensions as

$$\mathbf{M} = (\mathbf{V}^{-1})^T(\mathbf{V}^{-1}), \tag{2.22}$$

$$\mathbf{S}_\xi = (\mathbf{V}^{-1})^T\mathbf{W}^\xi(\mathbf{V}^{-1}), \quad \mathbf{W}^\xi_{ij} = \int_\mathbf{I} \phi_i(\xi)\frac{\partial\phi_j(\xi)}{\partial\xi}\,\mathrm{d}\xi, \tag{2.23}$$

$$\mathbf{S}_\eta = (\mathbf{V}^{-1})^T\mathbf{W}^\eta(\mathbf{V}^{-1}), \quad \mathbf{W}^\eta_{ij} = \int_\mathbf{I} \phi_i(\xi)\frac{\partial\phi_j(\xi)}{\partial\eta}\,\mathrm{d}\xi. \tag{2.24}$$

It is important to mention that for higher order equations (having spatial derivatives of order greater than 1), such as the Navier–Stokes equations, we follow the approach of Bassi and Rebay [6] by introducing an additional variable (for formulation only) so that we may rewrite the higher order equation

$$\frac{\partial\mathbf{u}}{\partial t} + \nabla\cdot\mathbf{F} = \nabla\cdot(v\nabla\mathbf{u}) \tag{2.25}$$

as a system of first order equations

$$\frac{\partial\mathbf{u}}{\partial t} + \nabla\cdot(\mathbf{F}-\mathbf{p}) = 0, \tag{2.26}$$

$$v\nabla\mathbf{u} = \mathbf{p}. \tag{2.27}$$

We then use the same approach and seek an approximation as

$$\int_{\mathbf{D}^k}\left(\frac{\partial\mathbf{u}_N}{\partial t} + \nabla\cdot(\mathbf{F}_N - \mathbf{p}_N)\right)L_i(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{2.28}$$

$$= \oint_{\delta\mathbf{D}^k} L_i(\mathbf{x})\hat{\mathbf{n}}\cdot\left[\mathbf{F}_N - \mathbf{F}^*_N - (\mathbf{p}_N - \mathbf{p}^*_N)\right]\,\mathrm{d}\mathbf{x}, \tag{2.29}$$

$$\int_{\mathbf{D}^k}(\mathbf{p}_N - v\nabla\mathbf{u}_N)L_i(\mathbf{x})\,\mathrm{d}\mathbf{x} = \oint_{\delta\mathbf{D}^k} L_i(\mathbf{x})\hat{\mathbf{n}}\cdot[\mathbf{u}_N - \mathbf{u}^*_N]\,\mathrm{d}\mathbf{x}. \tag{2.30}$$

A central flux is used for $\mathbf{u}^*_N$ and the local Lax–Friedrichs flux is used for $\mathbf{F}^*_N$ and $\mathbf{p}^*_N$.

To stabilize the scheme, we apply a modal filter to the numerical approximation at regular intervals

$$\mathbf{F}_\mathcal{N}\mathbf{u}_N(x,t) = \sum_{n=0}^{N}\sigma\left(\frac{n}{N}\right)\hat{\mathbf{u}}_n(t)\phi_n(x), \tag{2.31}$$

where $\sigma(\eta)$ is the filter kernel. Two commonly used filters, which are implemented in Section 4, are the exponential and the sharp-cutoff filters [24].

3. Time integration schemes

3.1. Explicit Runge–Kutta (ERK) methods

We have a semi-discrete scheme, which we will integrate in time using a high-order Runge–Kutta method. Let us write the system of ordinary differential equations (ODEs) as the initial value problem (IVP)

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathbf{F}(t,\mathbf{U}(t)), \quad \mathbf{U}(t_0) = \mathbf{U}_0, \tag{3.1}$$

where $\mathbf{U}$ is a vector of length m, and m is the number of ODEs resulting from the spatial discretization of the given PDE. To compute $\mathbf{U}(t+\Delta t) = \mathbf{U}^{(n+1)}$ with an s-stage RK method

$$\mathbf{U}^{(i)} = \mathbf{U}^{(n)} + \Delta t\sum_{j=1}^{s}a_{ij}\mathbf{F}(t^{(n)} + c_j\Delta t, \mathbf{U}^{(j)}), \quad 1\leqslant i\leqslant s, \tag{3.2}$$

$$\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)} + \Delta t\sum_{i=1}^{s}b_i\mathbf{F}(t^{(n)} + c_i\Delta t, \mathbf{U}^{(i)}), \tag{3.3}$$

where $\mathbf{U}^{(n)} = \mathbf{U}(t^{(n)})$, $\mathbf{U}^{(i)} = \mathbf{U}(t^{(n)} + c_i\Delta t)$, and the fixed scalar coefficients $a_{i,j}$, b_j and c_i determine all of the accuracy, stability and efficiency properties of the given RK scheme. The nodes c_i are the RK intermediate time levels, the coefficients of the RK-matrix $\mathbf{A}$, $a_{i,j}$, are the weights for the ith RK stage, and b_i are the weights of the final stage.

Following Butcher [7], we write the RK scheme in a tabular format known as the Butcher tableau

$$\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_i \end{array}$$

Fully-explicit Runge–Kutta schemes, commonly referred to as ERK schemes, have zeros on the main diagonal and above the main diagonal of $\mathbf{A}$, e.g. $a_{ij} = 0,\ j \geqslant i$.

We implement an efficient and accurate 5-stage, 4th-order low-storage ERK scheme [9] in order to minimize memory storage. Carpenter and Kennedy [9] derive a 2N-storage scheme which is competitive with the classical 4th-order high-storage method, where N is the dimension of the ODE system. Given the coefficients A_j, B_j, and c_j [9], the algorithm to compute $\mathbf{U}(t + \Delta t)$ requires the storage and overwriting of only 2 vectors $\mathbf{U}_j$ and $d\mathbf{U}_j$

$$d\mathbf{U}_j = A_j d\mathbf{U}_{j-1} + \Delta t\mathbf{F}(\mathbf{U}_j), \quad j = 1, \ldots, s, \tag{3.4}$$

$$\mathbf{U}(t + \Delta t) = \mathbf{U}_j = \mathbf{U}_{j-1} + B_j d\mathbf{U}_j. \tag{3.5}$$

Williamson [41] demonstrated that the connection between the 2N-scheme and the general RK scheme

$$\begin{aligned} &B_j = a_{j+1,j}, \quad j \neq s, \\ &B_\mathrm{s} = b_\mathrm{s}, \\ &A_j = (b_{j-1} - B_{j-1})/b_j, \quad j \neq 1,\ b_j \neq 0, \\ &A_j = (a_{j+1,j-1} - c_j)/B_j, \quad j \neq 1,\ b_j = 0. \end{aligned}$$

Although fully-explicit time-integration schemes are simple to implement and the most efficient methods for low levels of stiffness, they are at the mercy of the stability-based time-step restriction (CFL condition), especially for problems that have high levels of geometry-induced or physics/operator-induced stiffness. For this reason, we implement implicit–explicit RK methods, which we discuss in Section 3.3.

3.2. Geometry-induced stiffness

We now define two measures of geometry-induced stiffness, $\mathscr{S}$, which will be referred to as "stiffness" throughout this paper, unless specified otherwise. For the one-dimensional case, the definition of geometry-induced stiffness is straightforward, since the system eigenvalues will scale just as the ratio of element lengths. We define the grid-induced stiffness as the ratio of the minimum element length in the explicit set, $\mathbf{\Omega}_{[\mathbf{ex}]}$, to that of the minimum element length in the implicit set, $\mathbf{\Omega}_{[\mathbf{im}]}$ (i.e. ratio of minimum element length of all elements integrated with ARK-ERK to that of minimum element length of all elements integrated with ARK-ESDIRK)

$$\mathscr{S}^{1\mathrm{D}} = \frac{\min_{\mathbf{D}^k \in \mathbf{\Omega}_{[\mathbf{ex}]}}(l)}{\min_{\mathbf{D}^k \in \mathbf{\Omega}_{[\mathbf{im}]}}(l)}, \tag{3.6}$$

where l represents the element length. The two-dimensional grid-induced stiffness is defined to be the ratio of the minimum element (triangle) chord length in the explicit set, $\mathbf{\Omega}_{[\mathbf{ex}]}$, to that of the minimum element (triangle) chord length in the implicit set, $\mathbf{\Omega}_{[\mathbf{im}]}$ (i.e. ratio of minimum element chord length of all elements integrated with ARK-ERK to that of minimum element chord length of all elements integrated with ARK-ESDIRK)

$$\mathscr{S}^{2\mathrm{D}} = \frac{\min_{\mathbf{D}^k \in \mathbf{\Omega}_{[\mathbf{ex}]}}(c)}{\min_{\mathbf{D}^k \in \mathbf{\Omega}_{[\mathbf{im}]}}(c)}, \tag{3.7}$$

where c represents the element chord length. The two-dimensional stiffness may also be based on other measures, such as the triangles' inscribed radius.

3.3. Implicit–explicit Runge–Kutta (IMEX-RK) methods

In order to alleviate geometry-induced stiffness, we implement the recently introduced additive Runge–Kutta schemes by Kennedy and Carpenter [26], which are a class of implicit–explicit Runge–Kutta or IMEX-RK methods. IMEX algorithms solve the nonstiff terms using explicit methods, and isolate and solve the more expensive stiff terms using implicit methods. The N-Additive Runge–Kutta (ARK-N) schemes [9] are used to integrate equations of the form

$$\frac{d\mathbf{U}}{dt} = \mathbf{F}(t, \mathbf{U}(t)) = \sum_{\nu=1}^{N} \mathbf{F}^{[\nu]}(t, \mathbf{U}(t)), \quad \mathbf{U}(t_0) = \mathbf{U}_0, \tag{3.8}$$

and are given by the s-stage RK scheme

$$\mathbf{U}^{(i)} = \mathbf{U}^{(n)} + \Delta t \sum_{\nu=1}^{N} \sum_{j=1}^{s} a_{ij}^{[\nu]} \mathbf{F}^{[\nu]}(t^{(n)} + c_j \Delta t, \mathbf{U}^{(j)}), \quad 1 \leqslant i \leqslant s, \tag{3.9}$$

$$\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)} + \Delta t \sum_{\nu=1}^{N} \sum_{i=1}^{s} b_i^{[\nu]} \mathbf{F}^{[\nu]}(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)}), \tag{3.10}$$

where $\mathbf{U}^{(n)} = \mathbf{U}(t^{(n)})$, $\mathbf{U}^{(n+1)} = \mathbf{U}(t^{(n+1)})$, and $\mathbf{U}^{(i)} = \mathbf{U}(t^{(n)} + c_i \Delta t)$. We shall order $\mathbf{U}$ in the following way:

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_{[\mathrm{ex}]} \\ \mathbf{U}_{[\mathrm{im}]} \end{pmatrix}, \tag{3.11}$$

where $\mathbf{U}_{[\mathrm{ex}]}$ corresponds to the $m_{[\mathrm{ex}]}$ ordinary differential equations resulting from the spatial discretization of the partial differential equation on the explicit set of elements, $\mathbf{\Omega}_{[\mathrm{ex}]}$, and $\mathbf{U}_{[\mathrm{im}]}$ corresponds to the $m_{[\mathrm{im}]}$ ordinary differential equations resulting from the spatial discretization of the partial differential equation on the explicit set of elements, $\mathbf{\Omega}_{[\mathrm{im}]}$. Note that $m = m_{[\mathrm{ex}]} + m_{[\mathrm{im}]}$. We define $\mathbf{F} = \mathbf{F}^{[\mathrm{ex}]} + \mathbf{F}^{[\mathrm{im}]} = \mathbf{F}^{[1]} + \mathbf{F}^{[2]}$, where

$$\mathbf{F}^{[1]} \begin{pmatrix} \mathbf{U}_{[\mathrm{ex}]} \\ \mathbf{U}_{[\mathrm{im}]} \end{pmatrix} = \begin{pmatrix} \mathbf{F}(\mathbf{U}_{[\mathrm{ex}]}) \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{F}^{[2]} \begin{pmatrix} \mathbf{U}_{[\mathrm{ex}]} \\ \mathbf{U}_{[\mathrm{im}]} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{F}(\mathbf{U}_{[\mathrm{im}]}) \end{pmatrix}, \tag{3.12}$$

and the coefficient matrices $\mathbf{A}^{[\nu]}$ and vectors $\mathbf{b}^{[\nu]}$ are

$$\mathbf{A}^{[1]} = \mathbf{A}^{[\mathrm{ERK}]}, \quad \mathbf{A}^{[2]} = \mathbf{A}^{[\mathrm{ESDIRK}]} \tag{3.13}$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[2]} = \mathbf{b}, \tag{3.14}$$

where $\mathbf{A}^{[\mathrm{ERK}]}$, $\mathbf{A}^{[\mathrm{ESDIRK}]}$, and $\mathbf{b}$ are given in Appendix A. We now write the scheme as

$$\mathbf{U}_{[\mathrm{ex}]}^{(i)} = \mathbf{U}_{[\mathrm{ex}]}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij}^{[1]} \mathbf{F}(t^{(n)} + c_j \Delta t, \mathbf{U}_{[\mathrm{ex}]}^{(j)}), \quad 1 \leqslant i \leqslant s, \tag{3.15}$$

$$\mathbf{U}_{[\mathrm{im}]}^{(i)} = \mathbf{U}_{[\mathrm{im}]}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij}^{[2]} \mathbf{F}(t^{(n)} + c_j \Delta t, \mathbf{U}_{[\mathrm{im}]}^{(j)}), \quad 1 \leqslant i \leqslant s, \tag{3.16}$$

$$\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)} + \Delta t \sum_{i=1}^{s} b_i \mathbf{F}(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)}), \tag{3.17}$$

since $b_i^{[1]} = b_i^{[2]}$. This set of RK schemes allows for great flexibility in the sense that the implicit–explicit partition can be based on the operator or on the grid point/geometric region. In this paper, we reduce the N-Additive RK scheme to a 2-Additive scheme, which is given by an explicit–implicit partition. In other words, we choose to perform the time-splitting by geometric region.

The ARK schemes can be expressed in the following Butcher tableau format, which is similar to the basic tableau, but has one extra set of coefficients $\tilde{b}_i$. The coefficients $\tilde{b}_i$ provide a scheme of one order lower than the main scheme based on the coefficient weights b_i. Such schemes are referred to as embedded schemes.

$$\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_i \\ \hline & \tilde{b}_i \end{array}$$

Note that the two fourth-order schemes in Table A.1 are coupled through the nodes c_i, e.g. $c_i^{[\mathrm{ERK}]} = c_i^{[\mathrm{ESDIRK}]}$ so that the corresponding RK times $t^{(n)} + c_i\Delta t$ will be the same for both schemes at each RK stage, and also through the weights b_i, e.g. $b_i^{[\mathrm{ERK}]} = b_i^{[\mathrm{ESDIRK}]}$. Also, the embedded scheme will be used to compute the temporal error after every time step, which will be fed into a time-step controller to adaptively control the time-step (refer to Section 3.3.8).

The coupling between the explicit and implicit regions is straightforward. At each RK stage, the explicit grid points are integrated to find $\mathbf{U}_{[\mathrm{ex}]}^{(i)}$, and then the implicit grid points are integrated to find $\mathbf{U}_{[\mathrm{im}]}^{(i)}$, using the explicit regions as boundary conditions.

3.3.1. Newton–Krylov methods: Newton methods (outer iteration)

Let us assume for generality that we are solving a nonlinear conservation law, such as the Navier–Stokes equations. To integrate the semi-discrete system forward in time with an implicit Runge–Kutta scheme, we must solve a nonlinear system of equations at the ith RK stage if the ith row of $\mathbf{A}$ has at least one entry a_{ij} that is nonzero for $j \geqslant i$.

For example, for the second stage of the ARK4(3)-ESDIRK ($i = 2$) scheme, we need to solve for $\mathbf{U}^{(2)}$

$$\mathbf{U}^{(i)} = \mathbf{U}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij}\mathbf{G}(\mathbf{U}^{(j)}), \tag{3.18}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(n)} + \Delta t \sum_{j=1}^{6} a_{2,j}\mathbf{G}(\mathbf{U}^{(j)}), \tag{3.19}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(n)} + \Delta t(a_{2,1}\mathbf{G}(\mathbf{U}^{(1)}) + a_{2,2}\mathbf{G}(\mathbf{U}^{(2)})), \tag{3.20}$$

$$\mathbf{U}^{(2)} = \mathbf{U}^{(n)} + \Delta t\left(\frac{1}{4}\mathbf{G}(\mathbf{U}^{(1)}) + \frac{1}{4}\mathbf{G}(\mathbf{U}^{(2)})\right). \tag{3.21}$$

We choose to solve for $\mathbf{U}^{(2)}$ using a modified Newton–Krylov method [25]. Let us assume that $\mathbf{U} = \mathbf{U}^{(2)}$. Eq. (3.19) becomes

$$\mathbf{U} = \mathbf{U}^{(n)} + \Delta t\left(\frac{1}{4}\mathbf{G}(\mathbf{U}^{(1)}) + \frac{1}{4}\mathbf{G}(\mathbf{U})\right). \tag{3.22}$$

We rewrite the system as

$$\begin{aligned} \mathbf{F}(\mathbf{U}) &= \mathbf{U} - \mathbf{U}^{(n)} - \Delta t\left(\frac{1}{4}\mathbf{G}(\mathbf{U}^{(1)}) + \frac{1}{4}\mathbf{G}(\mathbf{U})\right) && (3.23)\\ &= \left(\mathbf{U} - \frac{\Delta t}{4}\mathbf{G}(\mathbf{U})\right) + \mathbf{H}(\mathbf{U}^{(n)}, \mathbf{U}^{(1)}) && (3.24)\\ &= 0, && (3.25) \end{aligned}$$

where $\mathbf{H}(\mathbf{U}^{(n)}, \mathbf{U}^{(1)}) = -\mathbf{U}^{(n)} - \frac{\Delta t}{4}\mathbf{G}(\mathbf{U}^{(1)})$. A multivariate Taylor expansion about the current iterate of the solution $\mathbf{U}^k$ gives us

$$\mathbf{F}(\mathbf{U}^{k+1}) = \mathbf{F}(\mathbf{U}^{k}) + \mathbf{F}'(\mathbf{U}^{k})(\mathbf{U}^{k+1} - \mathbf{U}^{k}) \tag{3.26}$$

$$+ \mathbf{F}''(\mathbf{U}^{k})(\mathbf{U}^{k+1} - \mathbf{U}^{k})^2 + \cdots \tag{3.27}$$

Neglecting the higher order terms $\mathcal{O}(\mathbf{U}^{k+1} - \mathbf{U}^{k})^2$, we arrive at Newton's method

$$\mathbf{U}^{k+1} = \mathbf{U}^{k} + \delta\mathbf{U}^{k}, \quad k = 0, 1, \ldots, \tag{3.28}$$

$$\mathbf{J}(\mathbf{U}^{k})\delta\mathbf{U}^{k} = -\mathbf{F}(\mathbf{U}^{k}), \tag{3.29}$$

where $\mathbf{J} = \mathbf{F}'$ is the Jacobian matrix.

3.3.2. MFNK method

The above method is a strict Newton method and requires the formation and storage of the Jacobian matrix for each nonlinear solve (each implicit RK stage). This can be a very expensive and perhaps unfeasible task for large-scale problems. For these reasons, we implement a modified Jacobian-free Newton–Krylov method (JFNK) [27], which is referred to as the MFNK method by Knoll and McHugh in [28]. The MFNK method is not exactly JFNK, due to the fact that some Jacobians are computed and stored, and differs from the modified Newton–Krylov method (MNK) in that MNK holds both the preconditioner and the action of the Jacobian (Eq. (3.32)) constant over a number of Newton iterations, while MFNK only holds the Jacobian-based Preconditioner constant. For this reason, MFNK has stronger nonlinear convergence properties than MNK. We also note that the very expensive formation and storage of the Jacobian is performed infrequently.

3.3.3. Krylov methods: inner iteration

Each Newton iteration involves solving a sequence of linear systems

$$\mathbf{J}\delta\mathbf{U} = -\mathbf{F}(\mathbf{U}) \tag{3.30}$$

for $\delta\mathbf{u}$. Due to the nature of the DGFEM spatial discretization discussed in Section 2, the Jacobians are sparse and therefore lead to extremely sparse linear systems, since elements communicate only with "adjacent" neighboring elements that share a common point in one-dimension, edge in two-dimensions, and face in three-dimensions. The Jacobian matrix $\mathbf{J}$ for the ARK schemes may be found by differentiating Eq. (3.22) with respect to $\mathbf{U}$

$$\mathbf{J} = \frac{d\mathbf{F}(\mathbf{U})}{d\mathbf{U}} = \mathbf{I} - a_{ii}\Delta t\frac{d\mathbf{G}(\mathbf{U})}{d\mathbf{U}}, \quad 2 \leqslant i \leqslant s, \tag{3.31}$$

where the Jacobian $\frac{d\mathbf{G}}{d\mathbf{U}}$ may be computed analytically (note: this is not true for all equations) and the factor a_{ii} for the ARK-ESDIRK schemes is constant for all RK stages ($i > 1$) since the schemes are SDIRK for $i > 1$, or singly diagonally implicit Runge–Kutta.

Iterative methods are particularly well-suited for solving extremely-sparse, unsymmetric linear systems [37]. (Iterative methods are indirect, as opposed to direct methods such as Gaussian elimination, and require a certain criteria to end the iterations.) For these reasons, we solve these sparse linear systems using two popular Krylov subspace methods [37]: the generalized minimum residual method, commonly referred to as GMRES, and the Bi-Conjugate Gradient STABilized method also known as BiCGSTAB.

The success of an iterative linear solver largely depends on an effective preconditioner [37], which efficiently clusters the eigenvalues of the iteration matrix, and results in a speed-up of the Krylov method. We apply right preconditioning, which leaves the right-hand side of (3.28) unchanged

$$(\mathbf{J}\mathbf{P}^{-1})(\mathbf{P}\delta\mathbf{U}) = -\mathbf{F}(\mathbf{U}), \tag{3.32}$$

where $\mathbf{P}$ represents the preconditioning matrix. Solving the preconditioned system above involves two main steps.

(a) Firstly, we define $\mathbf{z} = \mathbf{P}\delta\mathbf{U}$ and solve

$$\mathbf{J}\mathbf{P}^{-1}\mathbf{z} = -\mathbf{F}(\mathbf{U}) \tag{3.33}$$

for $\mathbf{z}$ using a Krylov solver and the Frechet derivative

$$\mathbf{J}\mathbf{P}^{-1}\mathbf{r}_0 \approx [\mathbf{F}(\mathbf{u}+\epsilon\mathbf{P}^{-1}\mathbf{r_0})-\mathbf{F}(\mathbf{u})]/\epsilon, \quad \epsilon \ll 1. \tag{3.34}$$

(b) Secondly, we solve for $\delta\mathbf{U}$ using a linear solver

$$\mathbf{P}\delta\mathbf{U}=\mathbf{z} \Rightarrow \delta\mathbf{U}=\mathbf{P}^{-1}\mathbf{z}. \tag{3.35}$$

The Newton–Krylov algorithm only requires the action of $\mathbf{P}^{-1}$ on vector $\mathbf{v}$ (matrix–vector product $\mathbf{P}^{-1}\mathbf{v}$). Thus, only the matrix elements required for the action of $\mathbf{P}^{-1}$ are formed. This may be done at every single Newton iteration or periodically when required (MFNK, MNK). We form the Jacobian once every k time steps ($k = 20, 50, 100, \ldots$) and reuse the "frozen" Jacobian as the preconditioner for the next k steps. However, even though we reuse the old Jacobian for preconditioning, we compute the current action of the Jacobian (current matrix–vector multiply $\mathbf{J}\mathbf{P}^{-1}\mathbf{r}_0$) using forward differencing (3.32).

It is also important to mention that GMRES involves only one matrix–vector multiply per Krylov iteration versus BiCGSTAB's two, which becomes an increasingly important consideration for increasingly stiff systems when using preconditioned Newton–Krylov algorithms.

Note that all of the Newton–Krylov algorithms applied in the numerical tests in Section 4 are based on C.T. Kelley's nsoli algorithm, which is a Newton–Krylov solver using inexact Newton-Armijo iteration, an Eisenstat–Walker forcing term and parabolic line search via 3-point interpolation [25]. The code is available from SIAM at the URL: http://www.siam.org/books/fa01/.

3.3.4. Newton–Krylov termination criteria

Iterative methods will continue iterating until a prescribed stopping or termination criteria is met. We use the following termination conditions for the Newton (outer) and Krylov (inner) iterations.

The outer Newton iteration will stop when

$$\|\mathbf{F}(\mathbf{U}^{k+1})\|_2 < atol + rtol\|\mathbf{F}(\mathbf{U}^0)\|_2, \tag{3.36}$$

where $atol$ and $rtol$ are the user-specified absolute and relative tolerances respectively. Typically, $atol = rtol = 1\mathrm{E}-03$ for most numerical tests in Section 4.

The inner Krylov iteration will stop when the relative linear residual

$$\|\mathbf{r}^{k+1}\|_2 < \eta_{\max}\|\mathbf{F}(\mathbf{U}^{k+1})\|_2, \tag{3.37}$$

where $\eta_{\max} = .9$.

3.3.5. Preconditioning

We conduct several numerical tests comparing the performance of the preconditioners discussed in this section. We perform the tests on the nozzle flow with shock test case from Section 4 for polynomials of degree $p = 4$ ($\Delta t_{\mathrm{mean}} = 1/50$) and $p = 8$ ($\Delta t_{\mathrm{mean}} = 1/142$). The tests are run until final time $T = 1$, which is much earlier than the time for which the shock begins to develop (roughly $T = 20$). The reason for this is because the times steps in this region are still fairly large with respect to the ERK case, and the results for this case are therefore more meaningful and important as far as IMEX-RK schemes are concerned. Please refer to Section 4 for all other parameters and details on the nozzle flow problem.

The results for the Jacobi, block (subdomain) Jacobi, ILU(0) and ILUT(τ) preconditioners are summarized in Tables 1 and 2, and are plotted in Fig. 3. The tests were conducted on three different grids having geometry-induced stiffnesses of 12.6, 96.4 and 928.6, in order to study how the various preconditioners respond to geometry-induced stiffness. The preconditioners were formed and stored once every physical unit of time (once every $t = 1$ or once every 50 time steps for $p = 4$, and once every 142 time steps for $p = 8$). We used the GMRES Krylov scheme with no restarts as part of the MFNK method, and used a Newton tolerance of 1E−03 to stop the iterations. We tested the ILUT(τ) preconditioner for three values of τ, namely for $\tau = 1\mathrm{E}{-}02$, $\sqrt{10}\mathrm{E}{-}03$ and 1E−03.

First, let us clarify that the term "failed" in Tables 1 and 2 signifies stagnating or repeating failures of the MFNK, which resulted from ill-conditioned preconditioners. We can see from Table 1 that for polynomials of degree $p = 4$, the Jacobi, ILU(1E−02) and the ILU(0) preconditioners are not robust and result in repeated

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

Table 1
2D preconditioner tests, $p = 4$, $T = 1$

Preconditioner	Stiffness ($\mathscr{S}$)	Avg. GMRES iter. per Δt	CPU time
None	12.6	99	1.67E+02
Jacobi		946	1.19E+03
Block Jacobi		56	1.30E+02
ILU(1E−03)		7	2.06E+02
ILU($\sqrt{10}$E−03)		10	2.60E+02
ILU(1E−02)		24	2.99E+02
ILU(0)		Singular	Singular
None	96.4	850	1.58E+03
Jacobi		Failed	Failed
Block Jacobi		158	3.35E+02
ILU(1E−03)		12	6.31E+02
ILU($\sqrt{10}$E−03)		41	1.35E+03
ILU(1E−02)		Failed	Failed
ILU(0)		Singular	Singular
None	928.6	1054	2.19E+04
Jacobi		Failed	Failed
Block Jacobi		454	8.00E+02
ILU(1E−03)		25	8.50E+02
ILU($\sqrt{10}$E−03)		196	3.85E+03
ILU(1E−02)		Failed	Failed
ILU(0)		Singular	Singular

Table 2
2D preconditioner tests, $p = 8$, $T = 1$

Preconditioner	Stiffness ($\mathscr{S}$)	Avg. GMRES iter. per Δt	CPU time
None	12.6	121	7.25E+02
Jacobi		410	2.41E+03
Block Jacobi		43	2.01E+03
ILU(1E−03)		7	3.81E+03
ILU($\sqrt{10}$E−03)		12	4.02E+03
ILU(1E−02)		68	8.48E+03
ILU(0)		Singular	Singular
None	96.4	832	5.30E+03
Jacobi		Failed	Failed
Block Jacobi		113	4.94E+03
ILU(1E−03)		16	1.65E+04
ILU($\sqrt{10}$E−03)		371	1.78E+05
ILU(1E−02)		Failed	Failed
ILU(0)		Singular	Singular
None	928.6	6064	3.80E+04
Jacobi		Failed	Failed
Block Jacobi		350	1.23E+04
ILU(1E−03)		101	5.62E+04
ILU($\sqrt{10}$E−03)		Failed	Failed
ILU(1E−02)		failed	failed
ILU(0)		Singular	Singular

failures, especially as the stiffness increases. The ILU(0) factorization produced singular factors in all cases (and therefore repeating failures) and was the least robust method.

However, the ILU($\sqrt{10}$E−03), ILU(1E−03) and the block Jacobi preconditioners were consistently robust, even for high levels of stiffness, and are plotted in Fig. 3a for this reason. As expected, the ILU($\sqrt{10}$E−03) forms the factors faster than the ILU(1E−03). It is evident from Fig. 3 that preconditioning helps increase the efficiency of the MFNK and therefore the IMEX-RK scheme. The preconditioners help alleviate the

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

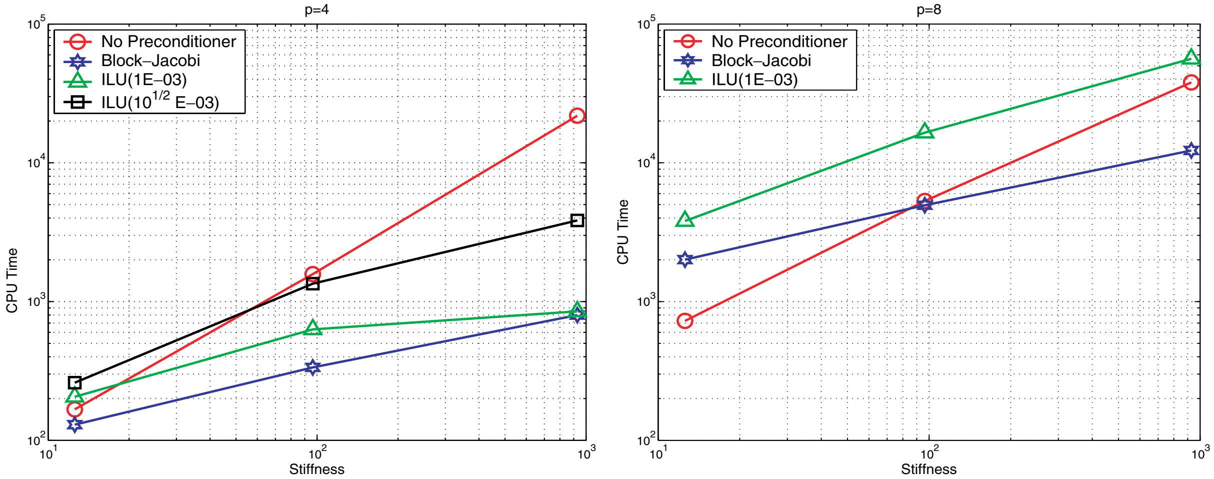


Fig. 3. 2D preconditioner tests for $p = 4$ (left) and $p = 8$ (right), $T = 1$.

CPU time versus stiffness slope. The ILU(1E−03) resulted in the flattest curve (smallest CPU time versus stiffness slope), but was slower than the block Jacobi preconditioner for all three test cases. The ILU(1E−03) may become more efficient than the block Jacobi for extremely high levels of stiffness (i.e. >1E+03). In terms of speed, storage, formation time, practicality (if we want to form the preconditioner at more frequent intervals) and implementation, the block Jacobi is the clear winner of this group, especially for the levels of stiffness that were tested.

Similarly, Table 2 for polynomials of degree $p = 8$ shows that the Jacobi, ILU(1E−02), ILU($\sqrt{10}\mathrm{E} - 03$) and the ILU(0) fail repeatedly for increasing stiffness. We plot the ILU(1E−03) and the block Jacobi results in Fig. 3b. Again, the block Jacobi preconditioner is more efficient than the ILU(1E−03). It is interesting to note that the preconditioned MFNK only starts to pay off for stiffness levels greater than roughly two orders of magnitude. This result may seem counterintuitive since preconditioned Implicit-RK methods are typically implemented for very large levels of stiffness where the preconditioner increases efficiency. However, if the stiffness level is low enough, the preconditioner may not increase the efficiency of the method.

3.3.6. Stability: explicit Runge–Kutta methods

We now analyze the domain of absolute stability (linear stability envelope) for a general ERK scheme. In order to determine the region of absolute stability, we apply the RK scheme to the scalar test equation

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathbf{F}(\mathbf{U}) = \lambda\mathbf{U}, \tag{3.38}$$

where λ is a complex constant that generally represents an eigenvalue of a matrix. Since Runge–Kutta schemes are one-step methods, we can write the numerical solution $\mathbf{U}^{(n+1)}$ at time $t^{(n+1)}$ as the product of an amplification factor $R(z)$ and the numerical solution $\mathbf{U}^{(n)}$ at time $t^{(n)}$

$$\mathbf{U}^{(n+1)} = R(z)\mathbf{U}^{(n)}, \tag{3.39}$$

where the complex number $z = \lambda h$ and $h = \Delta t$ is the time-step. The region of absolute stability occurs when $|\mathbf{U}^{(n+1)}| \leqslant |\mathbf{U}^{(n)}|$ or when $|R(z)| \leqslant 1$.

For an s-stage ERK of order p, the amplification factor $R(z)$ is given as [2]

$$R(z) = 1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \mathbf{b}^{\mathrm{T}}\mathbf{A}^{j-1}\mathbf{1}, \tag{3.40}$$

where the vectors $\mathbf{1} = [1, \ldots, 1]^{\mathrm{T}}$ and $\mathbf{b} = [b_1, \ldots, b_s]^{\mathrm{T}}$.

We plot the regions of absolute stability for ERK methods with $s = p \leqslant 4$, which includes the classical fourth-order, 4-stage method, the 5-stage, fourth-order low-storage 2N ((5,4)-2N ERK) scheme, and the 6-

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

stage, fourth-order ARK4(3)-ERK scheme in Fig. 4. We note that the $s = 6$ ARK4(3)-ERK and the $s = 5$ low-storage (5,4)-2N ERK schemes have the largest stability regions in the left-hand plane. The ARK4(3)-ERK scheme has the largest extent along the imaginary axis, while the (5,4)-2N ERK scheme has the largest extent along the real axis. However, we can see that for all of the explicit RK schemes, the values of $z = \lambda h$ necessary for stability are confined by the envelope regions. For stiff problems, the eigenvalues may become very large, thus squeezing the maximum allowable time step h to very small values. For this reason, we consider semi-implicit methods, such as the ARK4(3) scheme, which couples an explicit RK scheme to an implicit RK scheme, and therefore extends the stability region of purely explicit RK methods.

3.3.7. Stability: implicit Runge–Kutta methods

Let us discuss the stability of implicit RK methods. For explicit RK schemes, the amplification function is a polynomial. However, for implicit RK schemes, the amplification function $R(z)$ is not a polynomial, but a rational function that may be expressed as the quotient of two polynomials (by definition)

$$\begin{aligned} R(z) &= 1 + z\mathbf{b}^T(\mathbf{I} - z\mathbf{A})^{-1}\mathbf{1} && (3.41)\\ &= \frac{N(z)}{D(z)} && (3.42)\\ &= \frac{\det(\mathbf{I} + z(\mathbf{1}\mathbf{b}^T - \mathbf{A}))}{\det(\mathbf{I} - z\mathbf{A})}. && (3.43) \end{aligned}$$

Let us review a couple of important definitions regarding stability. A numerical method is A-stable if its region of absolute stability includes the entire left half-plane of $z = h\lambda$, i.e. $|R(z)| \leqslant 1$, for all z s.t. $Re(z) < 0$. A numerical method is L-stable if it is A-stable and $R(z = \infty) = 0$. Also, a scheme that has a nonsingular coefficient matrix $\mathbf{A}$ for which $a_{sj} = b_j,\ j = 1, \ldots, s$, is stiffly-accurate. Note that stiffly accurate methods have stiff decay. Methods with stiff decay have the property that as the real part of z goes to negative infinity ($Re(z) \to -\infty$), the amplification factor tends to 0 ($R(z) \to 0$).

We note that the ARK-ESDIRK [26] family of schemes are implicit RK methods ranging from third to fifth-order accurate. The three schemes are designed for the integration of stiff terms $|z| \to \infty$, and have many desirable properties with respect to stability. They are L-stable and stiffly-accurate with vanishing stability functions for very large eigenvalues $z \to -\infty$.

3.3.8. Time-step control

In order to control both accuracy and stability, it is important to choose a time-step controller which is a function of both criteria. The basic idea behind embedded time-integration schemes is to provide an additional scheme that is one order lower than the main scheme in order to allow for the computation of the temporal error.

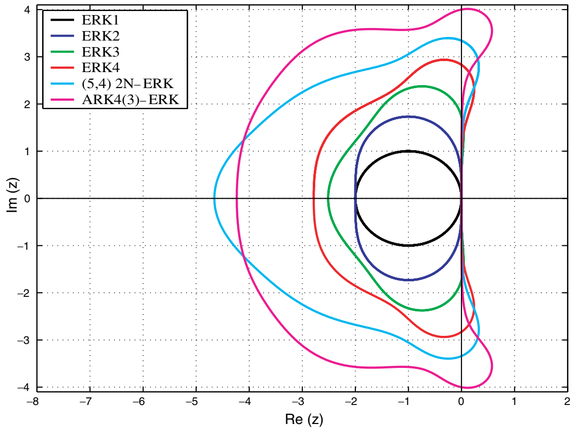


Fig. 4. The regions of absolute stability for various ERK schemes.

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

For example, the ARK5(4), ARK4(3) and ARK3(2) [26] schemes are of design orders 5, 4, and 3 respectively with embedded schemes of orders 4, 3, and 2 respectively. The computed temporal error may be fed into a controller such as an I, PI, or a PID controller, in order to automatically and adaptively control the time step Δt.

Let us the derive the I-based controller in order to gain a deeper understanding of time-step controller design in general. In order to compute the temporal error δ, we subtract the solution based on the embedded scheme of order p from the solution based on the main scheme of order $p+1$

$$\begin{aligned}\delta &= \mathbf{U} - \widehat{\mathbf{U}} && (3.44)\\ &= (\mathbf{U}_{\text{exact}} + \mathcal{O}((\Delta t)^{p+1})) - (\mathbf{U}_{\text{exact}} + \mathcal{O}((\Delta t)^{p})) && (3.45)\\ &= \mathcal{O}((\Delta t)^{p}) && (3.46)\\ &= C(\Delta t)^{p}, && (3.47)\end{aligned}$$

where C is a constant. Therefore, our computed temporal error is of order p. We now compare the time errors $\delta^{(n+1)}$ and $\delta^{(n)}$ for 2 different time steps, $(\Delta t)^{(n+1)}$ and $(\Delta t)^{(n)}$

$$\begin{aligned}\frac{\delta^{(n+1)}}{\delta^{(n)}} &= \frac{C((\Delta t)^{(n+1)})^{p}}{C((\Delta t)^{(n)})^{p}} && (3.48)\\ &= \left(\frac{(\Delta t)^{(n+1)}}{(\Delta t)^{(n)}}\right)^{p}, && (3.49)\end{aligned}$$

where $(\Delta t)^{(n+1)}$ is the time-step we want to determine and $\delta^{(n+1)}$ is the temporal error that will occur for this step. Let us specify the time error we want to commit for this step and call it $\epsilon = \delta^{(n+1)}$. Substituting ϵ for $\delta^{(n+1)}$ gives us

$$\frac{\epsilon}{\delta^{(n)}} = \left(\frac{(\Delta t)^{(n+1)}}{(\Delta t)^{(n)}}\right)^{p}, \qquad (3.50)$$

and solving for $(\Delta t)^{(n+1)}$

$$(\Delta t)^{(n+1)} = (\Delta t)^{(n)}\left(\frac{\epsilon}{\delta^{(n)}}\right)^{\frac{1}{p}}. \qquad (3.51)$$

Finally, we add a factor of safety κ

$$(\Delta t)^{(n+1)} = \kappa(\Delta t)^{(n)}\left(\frac{\epsilon}{\delta^{(n)}}\right)^{\frac{1}{p}} \qquad (3.52)$$

Two common controllers are given below (refer to [17,18,38])

$$\begin{aligned}(\Delta t)_{\mathrm{I}}^{(n+1)} &= \kappa(\Delta t)^{(n)}\left(\frac{\epsilon}{\|\delta^{(n)}\|_{\infty}}\right)^{\frac{1}{p}}, && (3.53)\\ (\Delta t)_{PID}^{(n+1)} &= \kappa(\Delta t)^{(n)}\left(\frac{\epsilon}{\|\delta^{(n)}\|_{\infty}}\right)^{\alpha}\left(\frac{\|\delta^{(n-1)}\|_{\infty}}{\epsilon}\right)^{\beta}\left(\frac{\epsilon}{\|\delta^{(n-2)}\|_{\infty}}\right)^{\gamma}, && (3.54)\end{aligned}$$

where $\kappa \approx .9$ is a factor of safety, ϵ is a specified tolerance for the controlled parameter (e.g. temporal error, ...), and p is the order of accuracy of the embedded scheme. δ is a measure of temporal error and is defined as

$$\begin{aligned}\delta^{(n+1)} &= \mathbf{U}^{(n+1)} - \hat{\mathbf{U}}^{(n+1)} && (3.55)\\ &= \Delta t\sum_{i=1}^{s} b_i\mathbf{F}(\mathbf{U}^{(i)}) - \Delta t\sum_{i=1}^{s}\hat{b}_i\mathbf{F}(\mathbf{U}^{(i)}) && (3.56)\\ &= \Delta t\sum_{i=1}^{s}(b_i - \hat{b}_i)\mathbf{F}(\mathbf{U}^{(i)}). && (3.57)\end{aligned}$$

We follow [26] and select the PID controller with the following fixed controller gains

$$k_I = 0.25, \quad k_P = 0.14, \quad k_D = 0.10, \quad \omega_n = 1, \tag{3.58}$$

where

$$p\alpha = \left[k_I + k_P + \left(\frac{2\omega_n}{1+\omega_n}\right)k_D\right], \quad p\beta = [k_P + 2\omega_n k_D], \tag{3.59}$$

$$p\gamma = \left(\frac{2\omega_n^2}{1+\omega_n}\right)k_D. \tag{3.60}$$

and

$$\omega_n = \frac{(\Delta t)^{(n)}}{(\Delta t)^{(n-1)}}. \tag{3.61}$$

Therefore,

$$\alpha = \frac{.49}{p}, \quad \beta = \frac{.34}{p}, \quad \gamma = \frac{.10}{p}. \tag{3.62}$$

We demonstrate the responsiveness and time-step control of the PID-controller for the one-dimensional Burgers equation

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial (u^2)}{\partial x} = \epsilon \frac{\partial^2 u}{\partial x^2} \tag{3.63}$$

with a perturbation at the inflow $x = -.5$ given as

$$u(-.5,t) = \left(-a\tanh\left(a\frac{x-ct}{2\epsilon}\right) + c\right)\cdot(1 + A(\sin(ft))^4) \tag{3.64}$$

$$= 1 + .1(\sin(100t))^4, \tag{3.65}$$

since $a = 1$, wave speed $c = 0$, $\epsilon = 1\text{E}{-03}$, $f = 100$ and amplitude $A = .1$. The perturbation is designed to test the time-controller's responsiveness (refer to Section 4 for details for this test case). The time-step history is illustrated in Fig. 5. The red curve is the inflow $\sin^4$ perturbation function scaled so that both the time-step history curve (black) and the inflow function (red) can easily be visually compared. We can see that the PID controller responds well to the oscillations of the inflow perturbation function, thereby having a frequency that appears to match the frequency of (3.63), which is $100/\pi$, quite well. Also, note that the time-step history is a smooth function, indicating the proper behavior for the PID controller (since the problem is smooth and is spatially resolved).

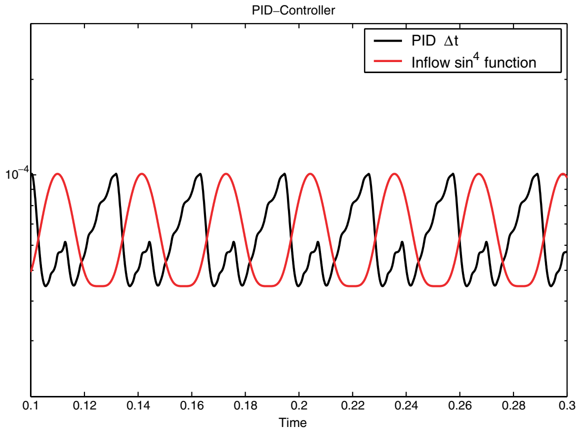


Fig. 5. PID-control for 1-D Burgers equation with perturbation at inflow.

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

Finally, the time step Δt is chosen as the minimum of the stability-based time-step and the time-accurate controller-based time step

$$\Delta t = \min(\Delta t_{\text{Stable}}, \Delta t_{\text{Controller}}). \tag{3.66}$$

4. Numerical tests

In this section, we carry out numerical experiments in 1D and 2D. We implement both ERK and IMEX-RK schemes to solve several test problems, such as nozzle flows modeled by the Euler equations, and compare the efficiency of the methods. Note that the ERK method used for all 1D test cases is the classical fourth-order ERK4 scheme, while the ERK method used for the 2D test cases is the low-storage (5,4)-2N ERK scheme. The IMEX method used is always the ARK4(3) IMEX-RK scheme, unless specified otherwise, and selects time steps using a PID time step controller (refer to Section 3.3.8).

4.1. Viscous burgers equation

The one-dimensional viscous Burgers equation is the classical one-dimensional analog of the multidimensional viscous Navier–Stokes equations

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial(u^2)}{\partial x} = \epsilon\frac{\partial^2 u}{\partial x^2}, \quad -1 \leqslant x \leqslant 1, \ t \geqslant 0. \tag{4.1}$$

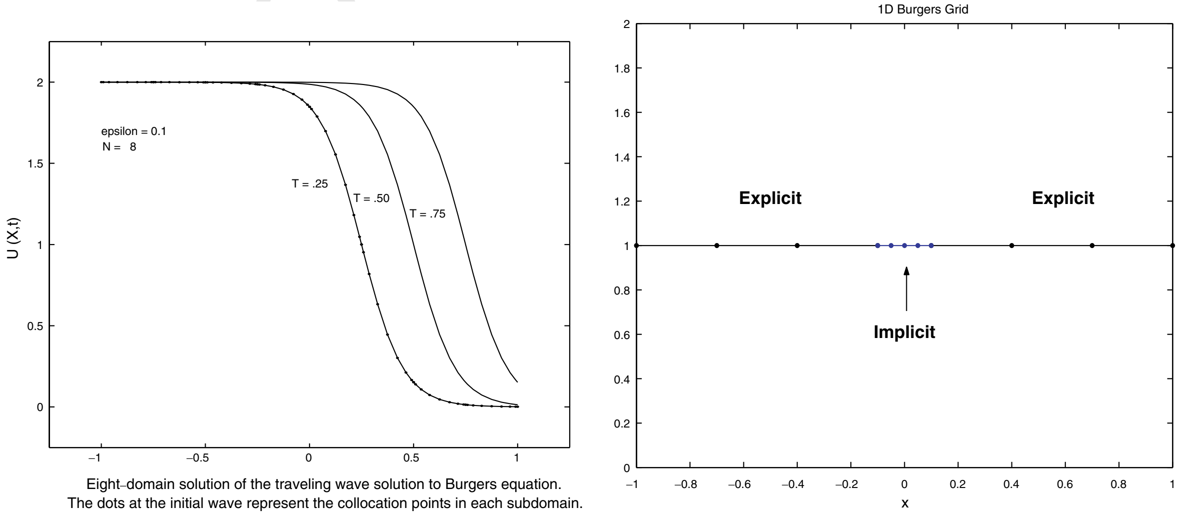
We set the initial condition to be a hyperbolic tangent wave so that the exact solution to Eq. (4.1) is a rightward traveling hyperbolic tangent wave with velocity equal to c, and initial condition $u(x,0)$:

$$u(x,t) = -a\tanh\left(a\frac{x-ct}{2\epsilon}\right) + c, \quad u(x,0) = -a\tanh\left(a\frac{x}{2\epsilon}\right) + c. \tag{4.2}$$

The wave-speed c, and the constant a are:

$$c = \frac{u_{-\infty} + u_{\infty}}{2}, \quad a = \frac{u - \infty - u_{\infty}}{2}. \tag{4.3}$$

The numerical solutions to Eq. (4.1) are shown in Fig. 6a. The grid used is displayed in Fig. 6b, where the elements in the blue region are solved using an IMEX-RK method, while the elements in the black region are solved using the ERK scheme. The results for both $\epsilon = .01$ and $\epsilon = .001$ are shown in Fig. 7 and are sum-



Eight–domain solution of the traveling wave solution to Burgers equation.
The dots at the initial wave represent the collocation points in each subdomain.

Fig. 6. The traveling wave solutions to Burgers equation (left), and the mesh used (right).

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

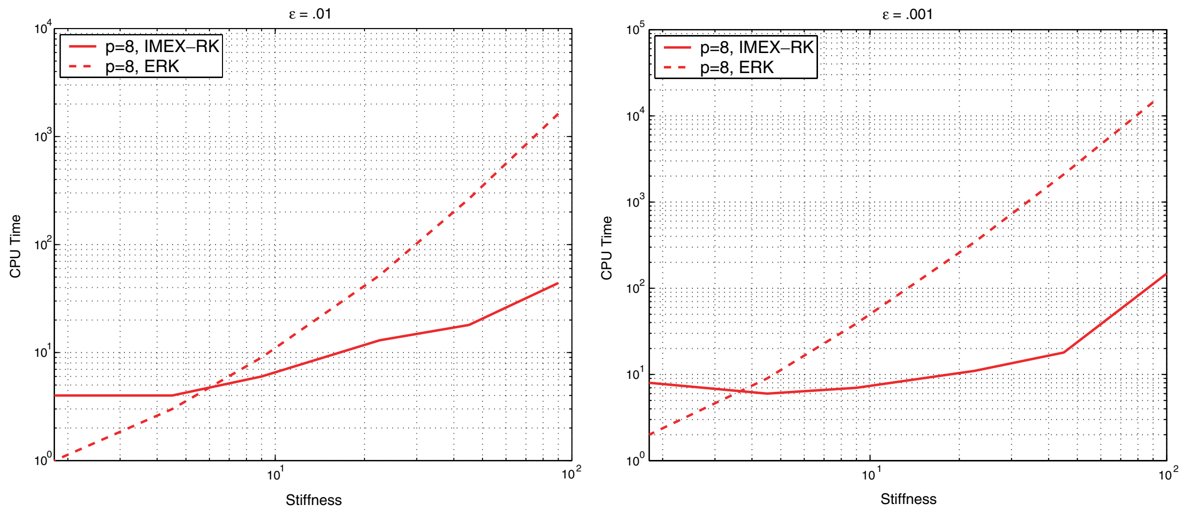


Fig. 7. Comparison of IMEX-RK and ERK results for 1D traveling wave solution to Burgers equation for $\epsilon = .01$ (left) and $\epsilon = .001$ (right).

Table 3
ERK and IMEX-RK results for 1D Burgers equation

ϵ	Stiffness ($\mathscr{S}$)	Avg. Δt (ERK)	Avg. Δt (IMEX)	Avg. GMRES iter. per Δt	$\mathrm{CPU_{ERK}}/\mathrm{CPU_{IMEX}}$
.01	90.0	5.03E−06	1.98E−03	4	36.96
	45.0	1.78E−05	1.98E−03	3	14.72
	22.5	5.76E−05	1.98E−03	3	4.00
	9.0	2.30E−04	1.98E−03	3	1.50
	4.5	5.26E−04	1.98E−03	2	.75
	1.8	1.69E−03	1.98E−03	3	.25
.001	90.0	5.71E−07	6.38E−04	3	128.88
	45.0	2.25E−06	3.19E−03	6	116.39
	22.5	8.74E−06	3.19E−03	6	31.18
	9.0	5.03E−05	3.19E−03	5	5.57
	4.5	1.78E−04	3.19E−03	5	1.50
	1.8	8.23E−04	3.19E−03	4	.25

marized in Table 3. We can see the same type of pattern appear as for the previous two cases. At a certain critical stiffness level, $\mathscr{S}_*$, the IMEX scheme starts to beat the ERK scheme. $\mathscr{S}_* \approx 6$ for $\epsilon = .01$, and $\mathscr{S}_* \approx 3$ for $\epsilon = .001$.

4.2. Compressible Navier–Stokes equations

We review the compressible, nondimensional Navier–Stokes equations in conservation form, which will be used to test the RK schemes described in this paper. Consider the three-dimensional Navier–Stokes equations given in Cartesian coordinates

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \frac{1}{Re_{\mathrm{ref}}}(\nabla \cdot \mathbf{F}_v), \quad t > 0. \tag{4.4}$$

The state vector $\mathbf{q}$ and the flux vector $\mathbf{F}(\mathbf{q})$ are given as

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

$$\mathbf{q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{bmatrix},\quad \mathbf{F}(\mathbf{q})=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ (E+p)u\end{bmatrix}\hat{\imath}+\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho vw\\ (E+p)v\end{bmatrix}\hat{\jmath}+\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ (E+p)w\end{bmatrix}\hat{k}, \tag{4.5}$$

where ρ is density, u, v and w are the Cartesian velocity components, E is the total energy, and p is the pressure. The total energy

$$E=\rho\left(T+\frac{1}{2}(u^2+v^2+w^2)\right). \tag{4.6}$$

The pressure and temperature are related through the ideal gas law

$$p=(\gamma-1)\rho T, \tag{4.7}$$

where T is the temperature and $\gamma=c_\mathrm{p}/c_\mathrm{v}$ is the ratio between the constant pressure (c_p) and constant volume (c_v) heat capacities. $\gamma=1.4$ for air. The viscous vector is

$$\mathbf{F}_v=\begin{bmatrix}0\\ \tau_{xx}\\ \tau_{yx}\\ \tau_{zx}\\ \tau_{xx}u+\tau_{yx}v+\tau_{zx}w+\frac{\gamma k}{Pr}\frac{\partial T}{\partial x}\end{bmatrix}\hat{\imath}+\begin{bmatrix}0\\ \tau_{xy}\\ \tau_{yy}\\ \tau_{zy}\\ \tau_{xy}u+\tau_{yy}v+\tau_{zy}w+\frac{\gamma k}{Pr}\frac{\partial T}{\partial y}\end{bmatrix}\hat{\jmath} \tag{4.8}$$

$$+\begin{bmatrix}0\\ \tau_{xz}\\ \tau_{yz}\\ \tau_{zz}\\ \tau_{xz}u+\tau_{yz}v+\tau_{zz}w+\frac{\gamma k}{Pr}\frac{\partial T}{\partial z}\end{bmatrix}\hat{k}. \tag{4.9}$$

Note that the Cartesian coordinates $(x,y,z)=(x_1,x_2,x_3)$. We assume that the fluid is Newtonian, for which the stress tensor is defined as

$$\tau_{x_ix_j}=\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)+\delta_{ij}\lambda\sum_{k=1}^{3}\frac{\partial u_k}{\partial x_k}, \tag{4.10}$$

where μ is the dynamic viscosity, λ is the coefficient of Bulk viscosity for the fluid, and k is the coefficient of thermal conductivity. We use Sutherland's law to relate the dynamic viscosity to the temperature

$$\frac{\mu(T)}{\mu_\mathrm{s}}=\left(\frac{T}{T_\mathrm{s}}\right)^{\frac{3}{2}}\frac{T_\mathrm{s}+S}{T+S}, \tag{4.11}$$

where $\mu_\mathrm{s}=1.716\times10^{-5}$ kg/m s , $T_\mathrm{s}=273$ K, $S=111$ K and the Prandtl number $Pr=.72$ for atmospheric air. Stokes hypothesis gives us $\lambda=-\frac{2}{3}\mu$.

We normalize Eq. (4.4) using reference values $u_\mathrm{ref}=u_0$, $\rho_\mathrm{ref}=\rho_0$, $p_\mathrm{ref}=\rho_0u_0^2$, $T_\mathrm{ref}=u_0^2/c_\mathrm{v}$ and L as the reference length. Therefore, the reference Reynolds number $Re_\mathrm{ref}=\frac{\rho_0u_0L}{\mu_0}$ and the Prandtl number $Pr=\frac{c_\mathrm{p}\mu_0}{k_0}$.

4.3. Euler equations: two-dimensional nozzle flows

Consider the two-dimensional Euler equations given in conservation form

$$\frac{\partial\mathbf{q}}{\partial t}+\nabla\cdot\mathbf{F}(\mathbf{q})=0. \tag{4.12}$$

The state vector $\mathbf{q}$ and the flux vector $\mathbf{F}(\mathbf{q})$ are given in Section 4.2 for the three-dimensional Euler equations. For the two-dimensional Euler equations, the state vector is

$$\mathbf{q} = [\rho, \rho u, \rho v, E]. \tag{4.13}$$

We consider the flow in a two-dimensional duct (rectangular cross-section) or nozzle, modeled using the Euler equations. We solve the two-dimensional compressible Euler equations using both ERK and IMEX-RK time-stepping schemes and compare the accuracy and efficiency of both schemes. The converging–diverging nozzle (Fig. 8) has an area $A(x)$ given by

$$A(x) = \begin{cases} 1.75 - .75\cos((.2x - 1.0)\pi), & 0 \leqslant x \leqslant 5, \\ 1.25 - .25\cos((.2x - 1.0)\pi), & 5 \leqslant x \leqslant 10. \end{cases} \tag{4.14}$$

This is a classic one-dimensional steady (steady-state), inviscid compressible flow problem that has an analytic solution [1] on the centerline at $y = 0$. The initial condition is a linear profile that connects the exact (analytic) boundary conditions at $x = 0$ and $x = 10$.

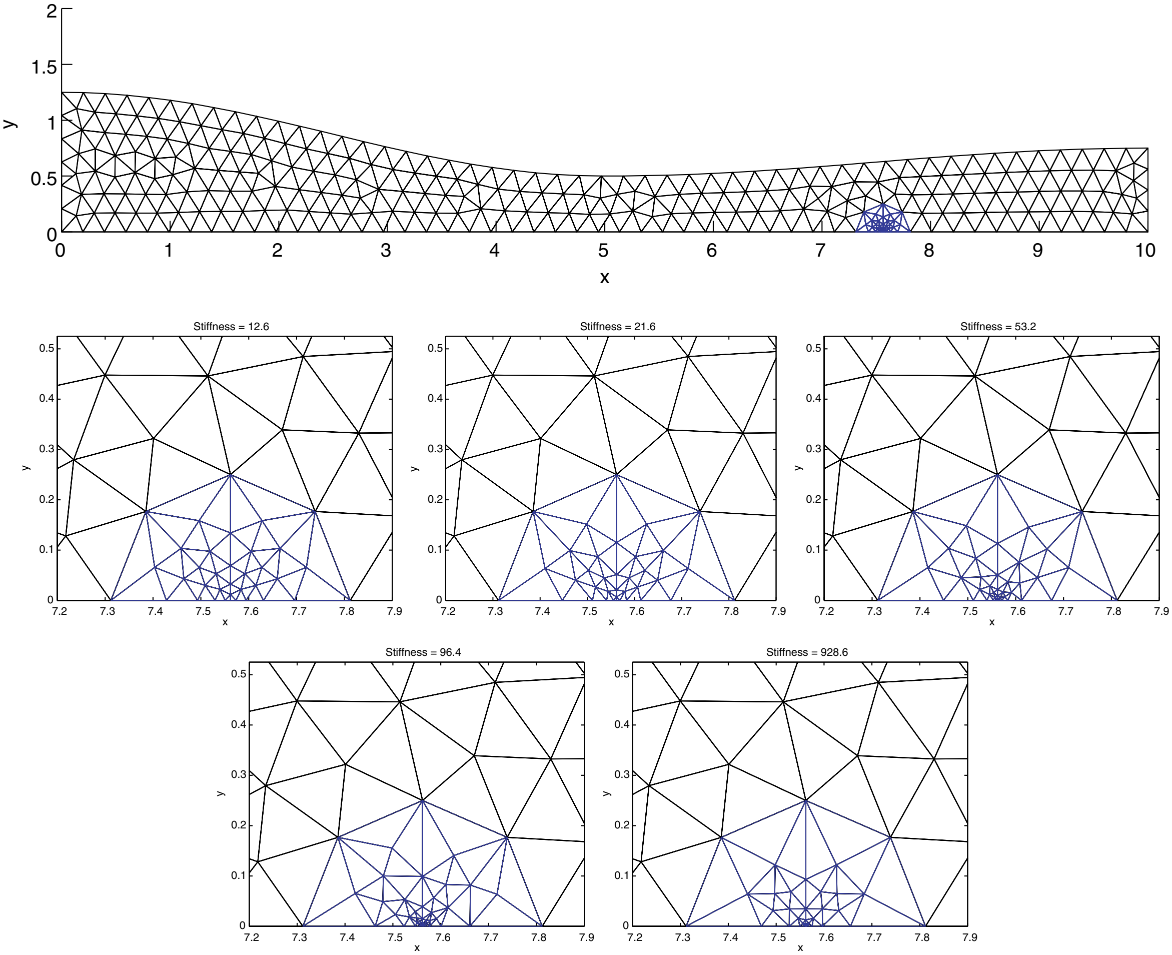


Fig. 8. The set of five grids used for the nozzle flow tests. $Nozzle_a$ (top-left) has a stiffness ≈12.6, $Nozzle_b$ (top-middle) has a stiffness ≈21.6, $Nozzle_c$ (top-right) has a stiffness ≈53.2, $Nozzle_d$ (bottom-left) has a stiffness ≈96.4, and $Nozzle_e$ (bottom-right) has a stiffness ≈928.6. The blue regions are solved implicitly when using the IMEX-RK scheme, and explicitly when using the ERK scheme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3.1. Nozzle flow with normal shock

A ratio between the stagnation pressure and the back pressure of .75 (back pressure/stagnation pressure) results in a choked flow with a stationary normal shock in the divergent part of the nozzle at $x \cong 7.5623$. The Mach number $M = 1.0$ and the stagnation temperature $T = 300$ K as the flow is choked. The inflow Mach number $M = .240$ and the outflow Mach number $M = .501$. The inflow values of the conserved variables are $(\rho_i, \rho u_i, \rho v_i, E_i) = (1.5331E+00, 4.0000E-01, 0, 3.3001E+00)$, while the outflow values are $(\rho_o, \rho u_o, \rho v_o, E_o) = (1.2427E+00, 6.6668E-01, 0, 2.7141E+00)$. A sample numerical solution for the Mach number and pressure contours at time $T = 40$ is shown in Fig. 9 (for $p = 4$). We compare ERK and IMEX results at final time $T = 1$, since the flow is still smooth in this regime. The shock begins to develop roughly at $T = 20$, after which the PID controller (based on L_∞ norm) drives the time steps to very small values (about the same as for ERK), and the computational advantage of the IMEX scheme disappears. We perform the nozzle tests on a set of five different grids illustrated in Fig. 8: $Nozzle_a$ has a stiffness $\mathscr{S} \approx 12.6$, $Nozzle_b$ has a stiffness $\mathscr{S} \approx 21.6$, $Nozzle_c$ has a stiffness $\mathscr{S} \approx 53.2$, $Nozzle_d$ has a stiffness $\mathscr{S} \approx 96.4$, and $Nozzle_e$ has a stiffness $\mathscr{S} \approx 928.6$. All of the grids have roughly the same number of elements (56–72) in the implicit set, $\mathbf{\Omega}_{[\mathbf{im}]}$, so that we can measure the effects of changing stiffness with roughly the same system sizes (for constant order p). Furthermore, the grids are clustered near the location of the shock ($x \cong 7.56$) on the centerline ($y = 0$), which is the axis along which we compare the numerical solution to the one-dimensional analytic solution (away from the walls).

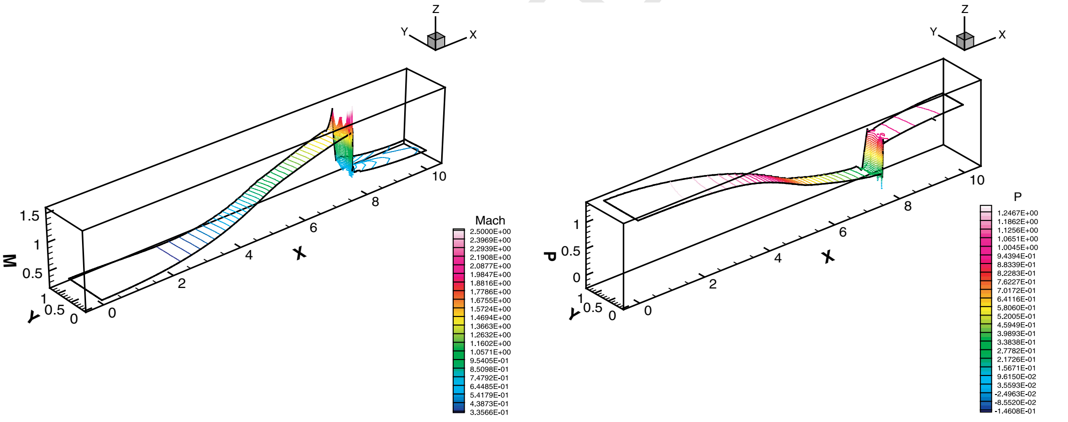


Fig. 9. Nozzle flow with shock: left plot is Mach contour, right is pressure.

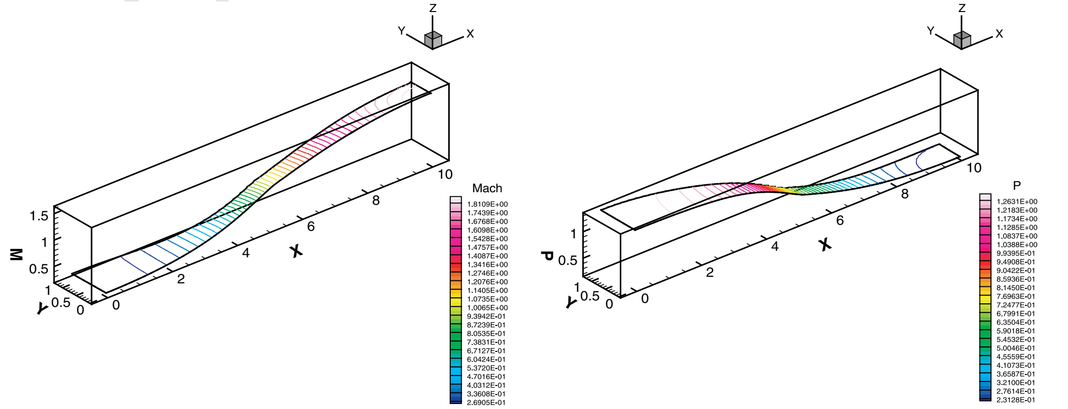


Fig. 10. Supersonic nozzle flow: left plot is Mach contour, right is pressure.

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

Table 4
ERK and IMEX-RK Results, Nozzle flow with shock, $T = 1$

Stiffness ($\mathscr{S}$)	p	Avg. Δt (ERK)	Avg. Δt (IMEX)	Avg. GMRES Iter. per Δt	$\text{CPU}_{\text{ERK}}/\text{CPU}_{\text{IMEX}}$
12.6	4	1.66E−03	2.00E−02	56	.97
	6	8.87E−04	1.06E−02	46	.78
	8	4.99E−04	7.04E−03	43	.62
96.4	4	2.16E−04	2.00E−02	158	2.99
	6	1.16E−04	1.06E−02	126	2.21
	8	6.50E−05	7.04E−03	113	1.99
928.6	4	2.25E−05	2.00E−02	454	11.96
	6	1.20E−05	1.06E−02	391	8.51
	8	6.75E−06	7.04E−03	350	7.67

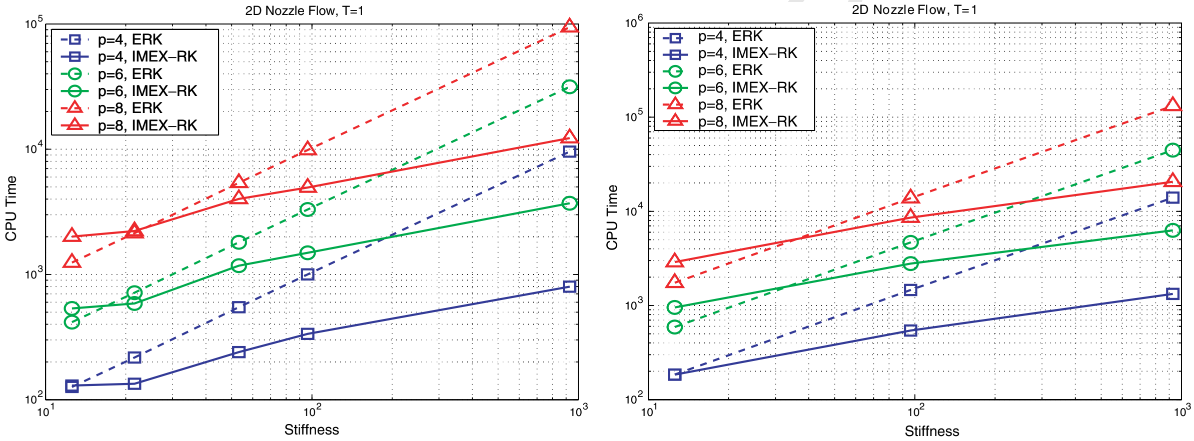


Fig. 11. 2D nozzle flow with shock (left) and supersonic (right) CPU time vs. stiffness ($\mathscr{S}$), $T = 1$.

The ERK and IMEX-RK results are summarized in Table 4, and are plotted in Fig. 11a for polynomials of degree $p = 4, 6, 8$. The IMEX method becomes more efficient than the ERK method at roughly a stiffness level of $\mathscr{S} = 10$ for the $p = 4$ case, while it does so at roughly a stiffness of $\mathscr{S} = 20$ for $p = 8$.

Fig. 12 compares the time-step histories (a), the L_2 norm residuals of ρ (b) defined as

$$\text{residual}(t + \Delta t) = \frac{\|\rho(t + \Delta t) - \rho(t)\|^2}{\|\rho(t)\|^2}, \tag{4.15}$$

and the CPU time versus the physical time (c) for the ERK and the IMEX-RK schemes for $Nozzle_c$ which has a stiffness level of approximately 22. We can see that before the shock develops ($t < 10$), the ratio of the IMEX time-steps to that of the ERK time-steps is roughly equal to the stiffness level. From the point when the shock begins to develop, the PID-controller takes charge and reduces the magnitude of the IMEX time-steps. This translates into a loss of computational efficiency as far as the IMEX results are concerned, since the original time-step ratio ≈22 shrinks to levels of $\mathcal{O}(1)$. The effect of this can be seen in Fig. 12c, where we plot CPU time versus physical time. Initially, the slope of CPU to physical time is lower for the IMEX scheme, but starts to catch up after the shock develops. We can see that both methods result in a decrease of the residual with time, although the ERK residual decreases more smoothly due to the smaller time-steps.

Finally, we plot the number of Newton and Krylov iterations versus time in Fig. 13a, and the temporal error (based on ρ) vs. time based on the embedded scheme in Fig. 13b.

4.3.2. Supersonic nozzle flow

A ratio between the stagnation pressure and the back pressure of .16 (back pressure/stagnation pressure) results in supersonic nozzle flow (no normal shock).

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

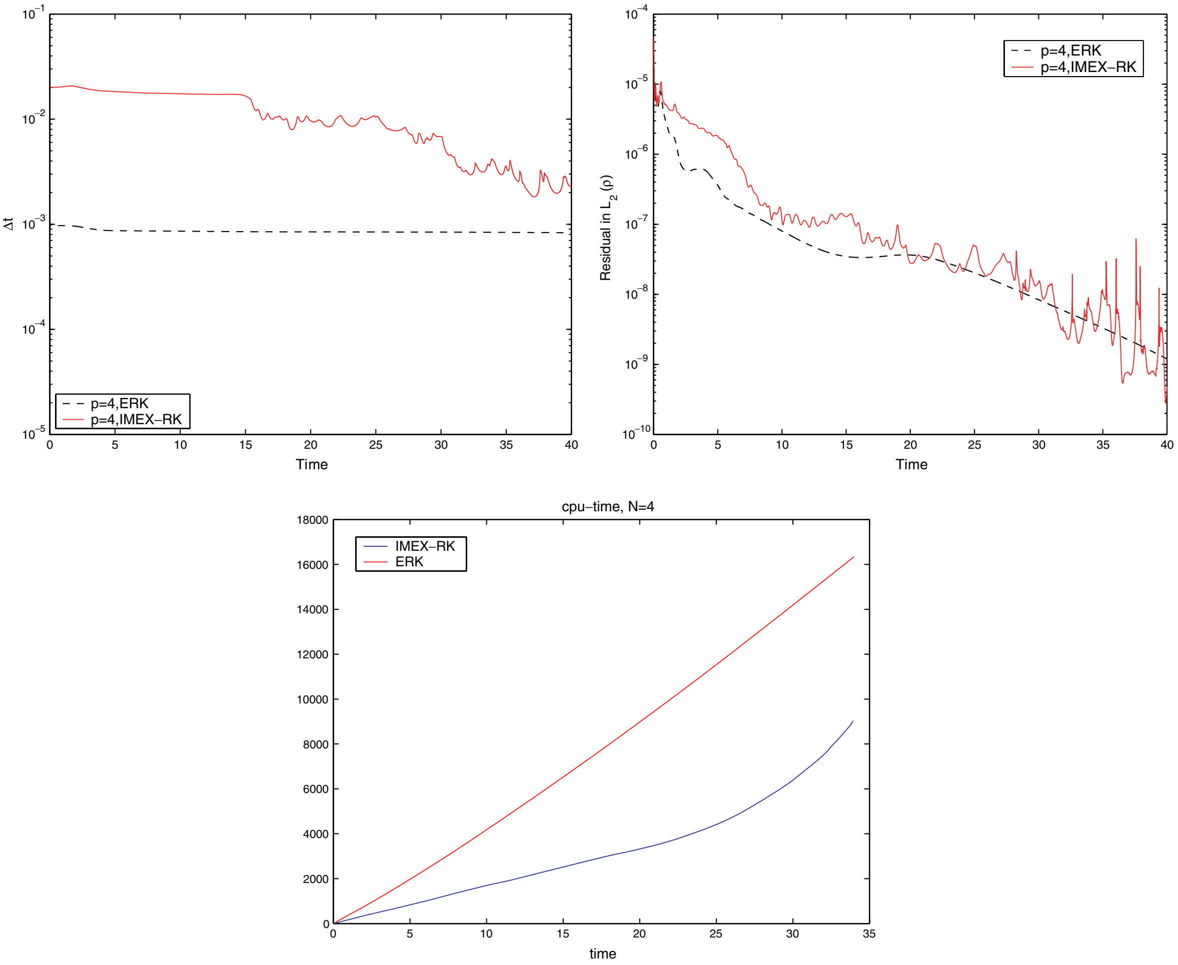


Fig. 12. Comparison of time-step histories (top-left), density residuals (L_2, top-right) and CPU time versus physical time (bottom).

The inflow values of the conserved variables are $(\rho_i, \rho u_i, \rho v_i, E_i) = (1.5331E+00, 4.0000E-01, 0, 3.3001E+00)$, while the outflow values are $(\rho_o, \rho u_o, \rho v_o, E_o) = (4.2639E-01, 6.6667E-01, 0, 1.0626E+00)$. A sample numerical solution for the Mach number and pressure contours at time $T = 40$ is shown in Fig. 10 (for $p = 4$).

The ERK and IMEX-RK results are summarized in Table 5, respectively, and are plotted in Fig. 11b for polynomials of degree $p = 4, 6, 8$. The results are quite similar to those of the normal shock case and will not be discussed further.

4.3.3. Navier–Stokes equations: cylinder flow

Consider the two-dimensional Navier–Stokes equations given in conservation form

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \frac{1}{Re_{\text{ref}}}(\nabla \cdot \mathbf{F}_v). \tag{4.16}$$

The state vector $\mathbf{q}$ and the flux vector $\mathbf{F}(\mathbf{q})$ are given in Section 4.2 for the three-dimensional Navier–Stokes equations. For the two-dimensional Navier–Stokes equations, the state vector is

$$\mathbf{q} = [\rho, \rho u, \rho v, E]. \tag{4.17}$$

Two-dimensional flow around a cylinder predicted by the 2D NS equations has good agreement with experimental results up to Reynolds numbers of roughly $Re = 180$. For $Re > 180$, three-dimensional effects take place, and numerical results can no longer be validated against experimental results. We perform calculations

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

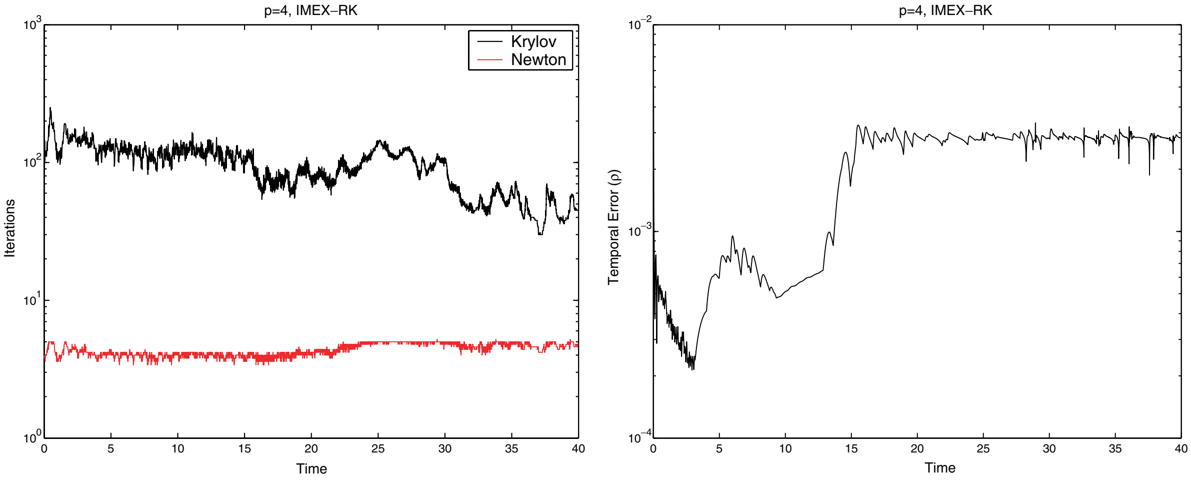


Fig. 13. Plot of Newton and Krylov iterations vs. time (left) and temporal errors vs. time (right) generated by the embedded IMEX-RK scheme.

Table 5
ERK and IMEX-RK results, supersonic nozzle flow, $T = 1$

Stiffness ($\mathscr{S}$)	p	Avg. Δt (ERK)	Avg. Δt (IMEX)	Avg. GMRES iter. per Δt	$\text{CPU}_{\text{ERK}}/\text{CPU}_{\text{IMEX}}$
12.6	4	1.19E−03	1.33E−02	59	1.00
	6	6.33E−04	7.14E−03	49	.62
	8	3.56E−04	4.69E−03	45	.60
96.4	4	1.55E−04	1.33E−02	177	2.70
	6	8.25E−05	7.14E−03	147	1.68
	8	4.64E−05	4.69E−03	137	1.61
928.6	4	1.61E−05	1.33E−02	510	10.52
	6	8.56E−06	7.14E−03	447	7.12
	8	4.82E−06	4.69E−03	415	6.40

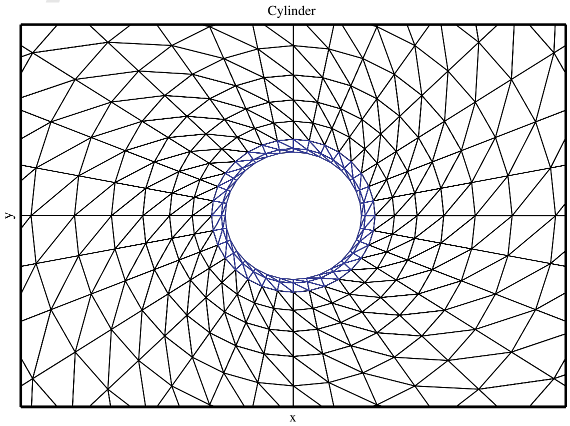


Fig. 14. The mesh used for the cylinder flow tests. The blue region is solved implicitly when using the IMEX-RK scheme, and explicitly when using the ERK scheme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at $Re = 75$, +100 and +125, and compare the Strouhal numbers for these flows versus experimental data by Williamson [40] and numerical results by Hesthaven [19]. The Strouhal number is the nondimensional shedding frequency and is defined as $St = \omega L/u_0$.

We run the tests with polynomials of degree $p = 4$ until time $T = 100$–150, by which periodic vortex shedding is well established. The computational domain is a disk with radius equal to approximately 20 cylinder diameters. The mesh used is shown in Fig. 14. The black elements are solved explicitly in time, while the two rows of elements in the blue region are solved implicitly. The ratio of number of elements in the implicit region to those in the explicit region is 128/1408.

We plot contours of density, pressure, vorticity and Mach number for $Re = 100$ in Fig. 15a–d, and the velocity streamlines in Fig. 15e. Table 6 compares the Strouhal numbers computed numerically using the IMEX scheme to those from Williamson's experimental results and Hesthaven's computations, and the comparison is very good. It is important to note that we use the sharp-cutoff filter with $N_c = N - 1$ for this test, the Newton tolerance is 1E−03, and the Krylov solver is BiCGSTAB without preconditioning. Also, the stiffness $\mathscr{S} \approx 3$, and the CPU time for the IMEX is roughly the same as that for an ERK scheme.

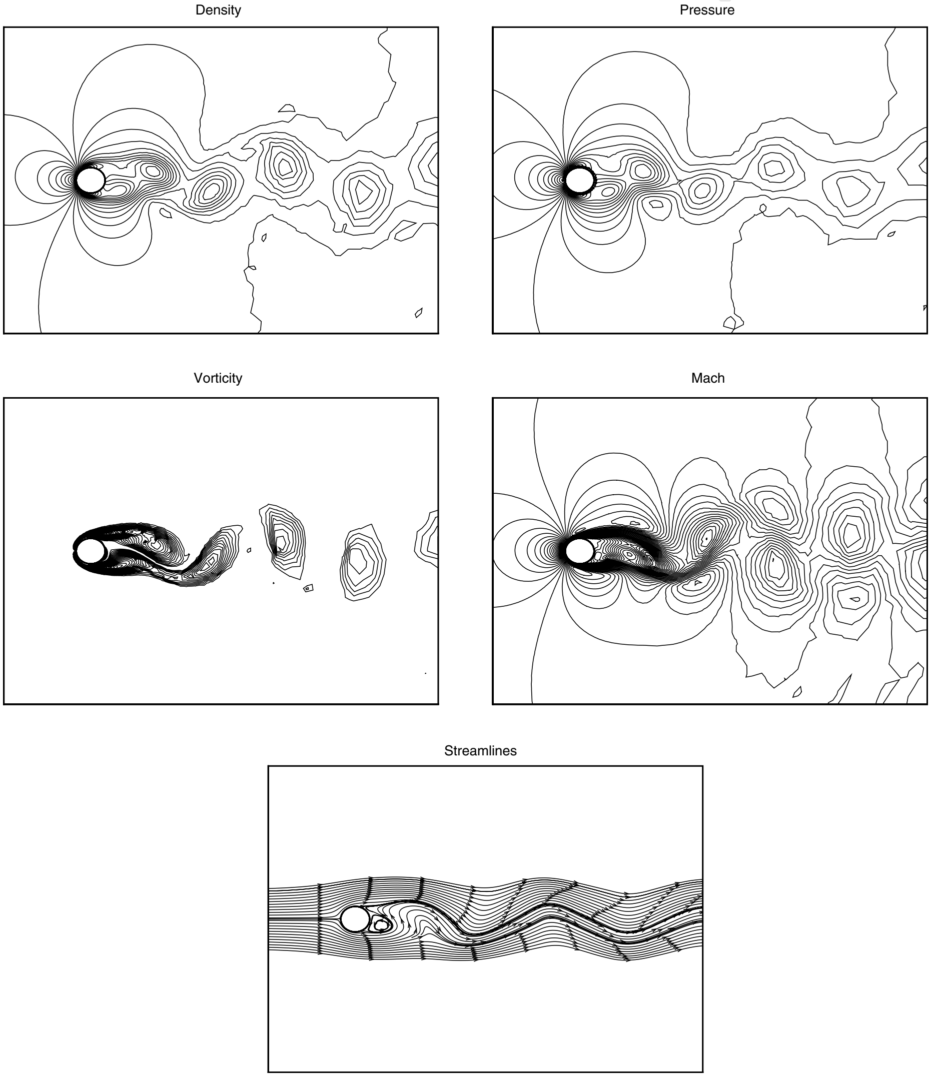


Fig. 15. The density, pressure, vorticity, and Mach number contour plots for the IMEX-RK simulation of cylinder vortex shedding at $Re = 100$ and time $T = 100$ ($p = 4$). The bottom plot shows the velocity streamlines for this flow.

Table 6
Strouhal numbers from experiment and computations at $Re = 100$

Re	*St* computed	*St* computed [19]	*St* experiment [40]
75	.151	.149	.149
100	.166	.165	.164
125	.177	.177	.175

5. Conclusions

In this paper, we introduced, discussed, tested and compared explicit (ERK) and implicit–explicit (IMEX-RK) Runge–Kutta time integration schemes. The main motivation for considering implicit RK methods is geometry-induced stiffness, which is a result of computing on grids that are composed of elements having drastically varying length scales. Geometry-induced stiffness leads to severe time-step restrictions in the context of ERK schemes, which have been the most popular vehicles for time-integration up to the present day.

Fig. 4 shows the regions of absolute stability for various explicit methods. The complex product $z = \lambda h = \lambda \Delta t$ must lie within this region for each respective ERK scheme to guarantee stability (amplification factor is bounded by 1). However, for problems for which the eigenvalues are driven towards infinity due to the presence of geometry-induced or physics/operator-induced stiffness, the maximum stable time-step Δt_{ST} is driven towards zero. This stability-based time step restriction is the Achilles heel of ERK methods in general. Explicit Runge–Kutta methods are at the mercy of the "smallest" element in the mesh. Explicit methods that allow integrating elements with variable local time-steps (depending on the size of each element), such as local timestepping or multi-rate methods [32], have been developoed, but are typically second-order accurate and suffer difficulties contending with irregular unstructured meshes.

Our approach for overcoming geometry-induced stiffness is to apply IMEX-RK schemes based on [26]. We divide a given mesh into two main sets or regions: the first containing the "explicit" elements which we integrate in time using an ERK scheme, and the second containing the "implicit" elements which are integrated in time with an implicit SDIRK scheme. The sets are divided in such a way so that the explicit set contains the "largest" elements (based on length in 1D, chord of triangle or other measure of length in 2D), while the implicit set contains the "smallest" elements which are responsible for constraining the maximum stable time step in purely ERK schemes. Thus, we alleviate the time-step restriction (to a degree) by integrating the small elements using an implicit scheme. With IMEX methods the problem of contending with irregular unstructured meshes that may have a combination of very small and highly distorted anisotropic elements is transferred over to that of building an adequate preconditioner for these strange cells.

All of the numerical test case results lead to a similar conclusion with regard to IMEX schemes. IMEX-RK schemes become more efficient than ERK schemes at a certain level of stiffness, even without the use of preconditioning. However, the application of efficient preconditioners in conjunction with IMEX MFNK schemes is critical to increasing the robustness and efficiency of IMEX methods, leading to even greater gains in computational efficiency for IMEX versus ERK methods. As the stiffness level $\mathscr{S}$ increases, efficient preconditioning becomes more important to speed-up the MFNK method. Effective preconditioning will decrease the CPU time versus stiffness slope. Also, out tests indicate that as stiffness levels increase, the preconditioned GMRES method becomes the Krylov method of choice (as compared to preconditioned BiCGSTAB), since it involves only one matrix–vector product per Krylov iteration versus BiCGSTAB's two. Adaptive controller-based time-stepping is very important in conjunction with IMEX schemes to control temporal errors. However, we found that L_∞-based time-step controllers are not suitable for problems with shocks.

Acknowledgments

The work of A.K. was partly supported by NASA Graduate Student Researchers Program (GSRP) Fellowship NGT-1-01024 and by the NSF VIGRE Program. The work of M.H.C. was partially funded under NASA fellowship 23847923847. The work of D.G. was . . . The work of J.S.H. was partly supported by NSF Career Award DMS-0132967 and by the Alfred P. Sloan Foundation through a Sloan Research Fellowship.

Appendix A

The fourth-order ARK4(3) scheme.

Table A.1
The fourth-order ARK4(3) scheme consists of two coupled RK schemes: a six-stage, fourth-order ERK scheme (top) and a six-stage fourth-order explicit singly diagonally implicit Runge–Kutta (ESDIRK) scheme (bottom)

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
$\frac{83}{250}$	$\frac{13861}{62500}$	$\frac{6889}{62500}$	0	0	0	0
$\frac{31}{50}$	$-\frac{116923316275}{2393684061468}$	$-\frac{2731218467317}{15368042101831}$	$\frac{9408046702089}{11113171139209}$	0	0	0
$\frac{17}{20}$	$-\frac{451086348788}{2902428689909}$	$-\frac{2682348792572}{7519795681897}$	$\frac{12662868775082}{11960479115383}$	$\frac{3355817975965}{11060851509271}$	0	0
1	$\frac{647845179188}{3216320057751}$	$\frac{73281519250}{8382639484533}$	$\frac{552539513391}{3454668386233}$	$\frac{3354512671639}{8306763924573}$	$\frac{4040}{17871}$	0
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
$\hat{b}_i$	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$-\frac{1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$-\frac{654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$-\frac{71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
$\hat{b}_i$	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

References

[1] J.D. Anderson, Modern Compressible Flow, McGraw-Hill, New York, 2002.

[2] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations, SIAM, 1998.

[3] U.M. Ascher, S.J. Ruuth, R.J. Spiteri, Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations, Appl. Numer. Math. 25 (1997) 151–167.

[4] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit–explicit methods for time-dependent partial differential equations, SIAM J. Numer. Anal. 32 (1995) 797–823.

[5] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484512.

[6] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (1997) 267–279.

[7] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, second ed., Wiley, Chichester, England, 2003.

[8] M.P. Calvo, J. de Frutos, J. Novo, Linearly implicit Runge–Kutta methods for advection–reaction–diffusion equations, Appl. Numer. Math. 37 (4) (2001) 535–549.

[9] M.H. Carpenter, C.A. Kennedy, Fourth-Order 2N-Storage Runge–Kutta Schemes, NASA-TM-109112, 1994, pp. 1–24.

[10] B. Cockburn, Discontinuous Galerkin methods for convection-dominated problems, in: T.J. Barth, H. Deconinck (Eds.), High-Order Methods for Computational Physics, Lecture Notes in Computational Science and Engineering, vol. 9, Springer, Berlin, 1999, pp. 69–224.

[11] B. Cockburn, G.E. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, 2000, Computing 16 (2001) 173–261.

[12] C.N. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (2001) 22562281.

[13] T.A. Driscoll, A composite Runge–Kutta method for the spectral solution of semilinear PDE, 2001, unpublished.

[14] M. Dubiner, Spectral methods on triangles and other domains, J. Sci. Comput. 6 (1991) 345–390.

[15] J.E. Flaherty, R.M. Loy, M.S. Shephard, B.K. Szymanski, J.D. Teresco, L.H. Ziantz, Adaptive local refinement with octree local-balancing for the parallel solution of three-dimensional conservation laws, J. Parallel Distributed Comput. 47 (1997) 139152.

[16] P. Fritzen, J. Wittekindt, Numerical solution of viscoplastic constitutive equations with internal state variables, Part I: Algorithms and implementation, Math. Meth. Appl. Sci. 20 (16) (1997) 1411–1425.

[17] K. Gustafsson, Control theoretic techniques for stepsize selection in Runge–Kutta methods, ACM Trans. Math. Soft. 17 (4) (1991) 533–554.

[18] K. Gustafsson, Control theoretic techniques for stepsize selection in implicit Runge–Kutta methods, ACM Trans. Math. Soft. 20 (4) (1994) 496–517.

[19] J.S. Hesthaven, A stable penalty method for the compressible Navier–Stokes equations: II. One-dimensional domain decomposition schemes, SIAM J. Sci. Comput. 18 (3) (1997) 658–685.

[20] J.S. Hesthaven, From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex, SIAM J. Numer. Anal. 35 (2) (1998) 655–676.

[21] J.S. Hesthaven, T. Warburton, Nodal high-order methods on unstructured grids I: Time-domain solution of Maxwell's equations, J. Comput. Phys. 181 (2002) 186–221.

[22] J.S. Hesthaven, T. Warburton, Discontinuous Galerkin methods for the time-domain Maxwell's equations: An introduction, ACES Newsletter 19 (2004) 12–30.

[23] A. Kanevsky, High-order implicit–explicit Runge–Kutta time integration schemes and time-consistent filtering in spectral Methods, Ph.D. Thesis, Brown University, 2006, pp. 1–138.

[24] A. Kanevsky, M.H. Carpenter, J.S. Hesthaven, Idempotent filtering in spectral and spectral element methods, J. Comput. Phys. 220 (1) (2006) 41–58.

[25] C.T. Kelley, Solving Nonlinear Equations with Newton's Method, SIAM, Philadelphia, 2003.

[26] C.A. Kennedy, M.H. Carpenter, Additive Runge–Kutta schemes for convection–diffusion–reaction equations, Appl. Numer. Math. 44 (2003) 139–181.

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

[28] D.A. Knoll, P.R. McHugh, Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow, SIAM J. Sci. Comput. 19 (1998) 291–301.

[29] T. Koornwinder, Two-variable analogues of the classical orthogonal polynomials, in: R.A. Askey (Ed.), Theory and Application of Special Functions, Academic Press, New York, 1975, pp. 435–495.

[30] R.J. LeVeque, Numerical Methods for Conservation Laws, Birkhauser Verlag, Basel, 1990.

[31] R.J. LeVeque, Finite-Volume Methods for Hyperbolic Problems, Cambridge University Press, 2002.

[32] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comp. 41 (1983) 321336.

[33] A.T. Patera, A. Spectral, Element method for fluid dynamics: laminar flow in a channel expansion, J. Comput. Phys. 54 (1984) 468–488.

[34] S. Piperno, Symplectic local time-stepping in non-dissipative DGTD methods applied to wave propagation problems, ESAIM:M2AN 40 (2006) 815–841.

Please cite this article in press as: A. Kanevsky et al., Application of implicit–explicit high order Runge–Kutta methods..., J. Comput. Phys. (2007), doi:10.1016/j.jcp.2007.02.021

[35] J. Proriol, Sur une famille de polynomes deux variables orthogonaux dans un triangle, C. R. Acad. Sci. Paris 257 (1957) 2459–2461.

[36] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Report LA-UR-73-479, 1973.

[37] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Co., 1996.

[38] G. Soderlind, Automatic control and adaptive time-stepping, Numerical methods for ordinary differential equations (Auckland, 2001), Numer. Algorithms 31 (1–4) (2002) 281–310.

[39] Z. Tan, Z. Zhang, Y. Huang, Tao Tang, Moving mesh methods with locally varying time steps, J. Comput. Phys. 200 (2004) 347–367.

[40] C.H.K. Williamson, Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, J. Fluid Mech. 206 (1989) 579–627.

[41] J.H. Williamson, Low-storage Runge–Kutta schemes, J. Comput. Phys. 35 (1980) 48.

[42] J.J.-I. Yoh, X. Zhong, Semi-implicit Runge–Kutta schemes for stiff multi-dimensional reacting flows, AIAA Paper 97-0803, AIAA, Aerospace Sciences Meeting and Exhibit, 35th, Reno, NV, January 6–9, 1997.

[43] X. Zhong, New high-order semi-implicit Runge–Kutta schemes for computing transient nonequilibrium hypersonic flows, AIAA Paper 95-2007, AIAA, Thermophysics Conference, 30th, San Diego, CA, June 19–22, 1995.