ARTICLE IN PRESS

Computers and Mathematics with Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method. Part III: Hybridized discontinuous Galerkin (HDG) formulation

Alexander Jaust [a], Balthasar Reuter [b], Vadym Aizinger [c,b,*], Jochen Schütz [a], Peter Knabner [b]

[a] Hasselt University, Faculty of Sciences, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium
[b] Friedrich-Alexander University of Erlangen-Nürnberg, Department of Mathematics, Cauerstraße 11, 91058 Erlangen, Germany
[c] Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research, Am Handelshafen 12, 27570 Bremerhaven, Germany

ARTICLE INFO

Article history:
Received 11 August 2017
Accepted 18 March 2018
Available online xxxx

Keywords:
MATLAB
GNU Octave
Hybridized discontinuous Galerkin (HDG) method
Vectorization
Open source
Diagonally implicit Runge–Kutta method (DIRK)

ABSTRACT

The third paper in our series on open source MATLAB/GNU Octave implementation of the discontinuous Galerkin (DG) method(s) focuses on a hybridized formulation. The main aim of this ongoing work is to develop rapid prototyping techniques covering a range of standard DG methodologies and suitable for small to medium sized applications. Our FESTUNG package relies on fully vectorized matrix/vector operations throughout, and all details of the implementation are fully documented. Once again, great care is taken to maintain a direct mapping between discretization terms and code routines as well as to ensure full compatibility to GNU Octave. The current work formulates a hybridized DG scheme for a linear advection problem, describes hybrid approximation spaces on the mesh skeleton, and compares the performance of this discretization to the standard (element-based) DG method for different polynomial orders.

1. Introduction

The discontinuous Galerkin (DG) method first introduced in the early 70s in [1] went on to have an illustrious career as one of the most popular numerical methods especially (but not exclusively) for fluid simulation and engendered a whole family of numerical schemes (see, e.g. [2,3] and the references therein). The reasons for this success are many [4,5]: an extremely flexible framework easily lending itself to many different types of equations, stability and conservation properties comparable to those of the finite volume method, natural support for high order discretizations and various types of adaptivity (h-, p-, r-) as well as for complex domain geometries, etc. Due to a favorable computation-to-communication ratio, the method also fits extremely well into the popular parallel and hybrid computational paradigms [6].

* Corresponding author at: Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research, Am Handelshafen 12, 27570 Bremerhaven, Germany.
E-mail addresses: alexander.jaust@uhasselt.be (A. Jaust), reuter@math.fau.de (B. Reuter), aizinger@math.fau.de, vadym.aizinger@awi.de (V. Aizinger), jochen.schuetz@uhasselt.be (J. Schütz), knabner@math.fau.de (P. Knabner).

https://doi.org/10.1016/j.camwa.2018.03.045
0898-1221/

Please cite this article in press as: A. Jaust, et al., FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method. Part III: Hybridized discontinuous Galerkin (HDG) formulation, Computers and Mathematics with Applications (2018), https://doi.org/10.1016/j.camwa.2018.03.045.

However, one aspect of the DG methodology places it at a clear disadvantage compared to the classical finite element and finite volume methods: a large number of degrees of freedom with corresponding memory requirements. This drawback becomes even more restrictive in time-implicit or stationary numerical solvers that rely on matrix assembly, where increases in the lengths of vectors of unknowns become quadratically compounded in the sizes of corresponding matrix blocks. One idea proposed to speed up the solution of linear systems resulting from DG discretizations exploits the specifics of DG approximation spaces. Since DG basis functions are usually element-local (and often also hierarchical), the linear system can be trivially split into parts corresponding to different polynomial orders resulting in a scheme somewhat inspired by multigrid solvers, where different polynomial approximations on a fixed mesh play the role of fine and coarse mesh solutions of the classical multigrid. This approach can be carried out for each polynomial degree as in the *p*-multigrid method [7–9] or using a two scale technique as in the hierarchical scale separation (HSS) method or variations thereof [10–13]. Another common way to deal with this issue – and in many cases even to speed up the time-to-solution [14–16] – is to use *hybridization*. Roughly speaking, this means that the discretized PDE is augmented with an unknown – let us call it λ_h – supported on the skeleton of the mesh consisting of element edges in 2D and element faces in 3D. Using static condensation on the algebraic system level produces a significantly smaller system than that obtained for an unhybridized DG method at the price of additional cell-wise (small and uncoupled) linear systems that have to be solved alongside the globally coupled system on the mesh skeleton. Since all local solves are element-local and fully decoupled, the parallel communication cost of this algorithm part is zero.

The idea of using hybridization can be traced back to the 60s [17], it has subsequently been used in the context of mixed methods [18,19]. In those works, λ_h was not only considered an implementation feature but also as a way to obtain a more accurate solution via postprocessing. Based on the work in [20], Cockburn and coworkers introduced the hybridized discontinuous Galerkin method in a unifying framework in [21]. Subsequently, the method has been extended to various types of equations such as the Stokes [22,23] and Darcy–Stokes equations [24], the incompressible and compressible Navier–Stokes equations [25–28], the Maxwell equations [29], and – particularly relevant for this work – the convection(-diffusion) equation [30–32]. For an interesting unification framework, we would also like to draw reader's attention to a recent publication [33].

The current work applies and extends to the hybridized schemes the framework and design principles introduced in [34,35] for unhybridized DG formulations and epitomized in our MATLAB / GNU Octave toolbox *FESTUNG* (*F*inite *E*lement *S*imulation *T*oolbox for *UN*structured *G*rids) available at [36,37]. Citing from [34], we aim to

1. Design a general-purpose software package using the DG method for a range of standard applications and provide this toolbox as a research and learning tool in the open source format.
2. Supply a well-documented, intuitive user-interface to ease adoption by a wider community of application and engineering professionals.
3. Relying on the vectorization capabilities of MATLAB / GNU Octave, optimize the computational performance of the toolbox components and demonstrate these software development strategies.
4. Maintain throughout full compatibility with GNU Octave to support users of open source software.

For details about basic data structures and a general overview of the solver structure, we refer the interested reader to our first publication [34], which applies the local discontinuous Galerkin (LDG) method to the diffusion operator. A DG discretization of the same model problem as in this work combined with higher-order explicit time stepping schemes and arbitrary order vertex-based slope limiters [38,39] is presented in the second paper in series [35]. The implementation presented in the current publication makes use of a new solver structure tailored towards improved readability and maintainability of the code and designed to ease coupling of different solvers. A detailed description of this new structure with an outline of the coupling capabilities is in preparation [40].

The rest of the paper is structured as follows: The model problem is introduced in the next section accompanied by a detailed description of the space and time discretization in Section 3. Important aspects of the implementation including local mappings, numerical quadrature, and the assembly of the system matrix are presented in Section 4. In Section 5, the code is verified using analytical convergence tests, and the numerical results are compared to those of the unhybridized DG implementation of the model problem from our previous publication [35]. Section 6 lists the routines mentioned in this article, and Section 7 contains some conclusions and a brief outlook of future tasks.

2. Model problem

Let $J := (0, t_{\text{end}})$ be a finite time interval, and let $\Omega \subset \mathbb{R}^2$ be a polygonally bounded Lipschitz domain with boundary $\partial\Omega$. We solve the *linear advection equation*

$$\partial_t c\ (t, \mathbf{x}) + \nabla \cdot \mathbf{f}\left(t, \mathbf{x}, c\,(t, \mathbf{x})\right) = \xi\,(t, \mathbf{x}) \quad \text{in } J \times \Omega\,, \tag{1}$$

where the scalar quantity $c : J \times \Omega \to \mathbb{R}$ is unknown. The flux function $\mathbf{f} : J \times \Omega \times \mathbb{R} \to \mathbb{R}^2$ determines the type of the problem and may depend on time t and space coordinate $\mathbf{x}$. Within the context of this work, we assume

$$\mathbf{f}\left(t, \mathbf{x}, c\ (t, \mathbf{x})\right) := \left[u^1(t, \mathbf{x}), \quad u^2(t, \mathbf{x})\right]^{\mathrm{T}} c\,(t, \mathbf{x}) \tag{2}$$

Please cite this article in press as: A. Jaust, et al., FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method. Part III: Hybridized discontinuous Galerkin (HDG) formulation, Computers and Mathematics with Applications (2018), https://doi.org/10.1016/j.camwa.2018.03.045.