Computers and Fluids xxx (xxxx) xxx

Contents lists available at ScienceDirect

Computers and Fluids

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

# A high-order stochastic Galerkin code for the compressible Euler and Navier-Stokes equations

Jakob Dürrwächter[a,*], Fabian Meyer[b], Thomas Kuhn[a], Andrea Beck[a], Claus-Dieter Munz[a], Christian Rohde[b]

[a] Institute of Aerodynamics and Gas Dynamics, University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany
[b] Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Pfaffenwaldring 57, 70569 Stuttgart, Germany

ARTICLE INFO

*Article history:*
Received 4 November 2019
Revised 5 May 2021
Accepted 2 June 2021

*Keywords:*
Stochastic Galerkin
Discontinuous Galerkin
Uncertainty quantification
Compressible Navier-Stokes equations
Unstructured grids
Computational fluid dynamics

ABSTRACT

We present a Stochastic Galerkin (SG) scheme for Uncertainty Quantification (UQ) of the compressible Navier-Stokes and Euler equations. For spatial discretization, we rely on the high-order Discontinuous Galerkin Spectral Element Method (DGSEM) in combination with an explicit time-stepping scheme. We simulate complex flow problems in two- and three-dimensional domains using curved unstructured hexahedral meshes. The dimension of the stochastic space can be arbitrarily large. In order to treat discontinuities and to ensure hyperbolicity, we employ a multi-element approach in combination with a hyperbolicity-preserving limiter in the stochastic domain and a Finite Volume subcell shock capturing scheme in physical space. Special emphasis is put on code performance and massive parallel scalability. We demonstrate the versatility and broad applicability of our code with various numerical experiments including the supersonic flow around a spacecraft geometry. By this, we wish to contribute to the path of the Stochastic Galerkin method towards practical applicability in research and industrial engineering.

## 1. Introduction

In addition to classical laboratory experiments, deterministic high-resolution numerical simulations have become a main pillar of research and development in academia and industry. In numerous cases deterministic numerical simulation results provide good agreement with experimental data. Though, whenever uncertainty about the experimental setting or uncertainties due to unknown model parameters enter the often non-linear governing equations, deterministic simulations fail to capture the whole picture. It is therefore necessary to quantify the influence of uncertain model input on numerical simulation results and derived quantities. Uncertainty Quantification (UQ) provides different mathematical tools to do so. It can thus help to uncover hidden regimes in the underlying systems.

A major class of UQ methods rely on the generalized Polynomial Chaos expansion (gPC) introduced by Xiu and Karniadakis [1], whose roots go back to the seminal work of Wiener [2]. In gPC the underlying random field is developed into a generalized Fourier series where the approximating polynomials are orthogonal with respect to an inner product induced by the probability density function of the uncertain input. The deterministic modes in this Fourier series representation can either be computed via numerical quadrature [3–5] or using a least-squares fit [6, Section 3.5]. These approaches are called non-intrusive because essentially deterministic versions of the random equation have to be solved. Hence, existing deterministic numerical solvers can readily be used. In contrast to non-intrusive methods, the intrusive spectral projection method, also known as Stochastic Galerkin (SG) approach, considers a weak formulation of the random equation with respect to the random variables and derives a modified deterministic systems of equations which is solved for the finite number of unknown modes. The adaption of existing numerical solvers to this modified system is in this case inevitable.

There exist different numerical schemes for the SG systems resulting from the compressible Euler and Navier-Stokes equations, cf. [6–16] and references therein. These works build on earlier applications to flow problems with different governing equations, cf. [3,17,18]. Due to its high computational cost, most applications of SG for compressible flow problems are restricted to spatially one-dimensional domains, or simplified equations. Moreover, the majority of existing numerical schemes for SG are based on basic finite volume solvers, relying on Cartesian grids and limited to first or second order accuracy and parallelization limited to a few cores. We would like to show that an SG exten-

[*] Corresponding author.
*E-mail address:* jd@iag.uni-stuttgart.de (J. Dürrwächter).

https://doi.org/10.1016/j.compfluid.2021.105039
0045-7930/© 2021 Elsevier Ltd. All rights reserved.

sion can also be introduced to a state-of-the-art industrially applied code with high order accuracy and massive parallel scalability. We aim to achieve this with our software framework SG-FLEXI, which is open source and can be retrieved from GitHub[1]. It relies on the SG method in combination with the Discontinuous Galerkin Spectral Element (DGSEM) method [19,20] and an explicit Runge-Kutta time-stepping scheme [21]. Thanks to its compact stencil and the explicit time-stepping scheme the DGSEM method allows for a massively parallel implementation and is therefore well-suited for applications on high-performance computing systems [19,20,22,23]. Our highly efficient implementation allows us to overcome the aforementioned limitations and perform UQ of complex flow problems in two and even three spatial dimensions. Moreover, our numerical scheme is able to handle an arbitrary number of random dimensions.

The rest of the paper is organized as follows: In Section 2 we present our equations of interest, the random Euler and random Navier-Stokes equations. Section 3 is concerned with the space, time and stochastic discretization of the governing equations. We present the SG approach and review the multi-element SG method, which decomposes the stochastic space into disjoint random elements and performs a local approximation on each random element. Furthermore, we present the DGSEM discretization of the resulting SG system. Our novel software framework SG-FLEXI is presented in Section 4, where we also evaluate code performance. Finally, in Section 5 we demonstrate the broad applicability of our framework by performing UQ of many different challenging flow problems for the compressible Euler and Navier-Stokes equations.

## 2. Random Navier-Stokes and Euler equations

We start with a description of our deterministic problem of interest, the three-dimensional compressible Navier-Stokes equations written in conservation form

$$\partial_t \boldsymbol{u} + \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{G}(\boldsymbol{u}) - \boldsymbol{H}(\boldsymbol{u}, \nabla_{\boldsymbol{x}} \boldsymbol{u})) = 0 \quad \text{in } D \times \mathbb{R}_+. \tag{1}$$

In (1) $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t) : D \times \mathbb{R}_+ \to \mathcal{U} \subset \mathbb{R}^{n_{\text{var}}}$ denotes the vector of the $n_{\text{var}} = 5$ conserved quantities and $\partial_t$ represents the derivative with respect to time, $\nabla_{\boldsymbol{x}}$ is the gradient operator in physical space $D \subset \mathbb{R}^3$. $\boldsymbol{G}$ and $\boldsymbol{H}$ describe the convective and viscous flux tensors, respectively, which we summarize for brevity as $\boldsymbol{F}(\boldsymbol{u}, \nabla_{\boldsymbol{x}} \boldsymbol{u}) := \boldsymbol{G}(\boldsymbol{u}) - \boldsymbol{H}(\boldsymbol{u}, \nabla_{\boldsymbol{x}} \boldsymbol{u})$. The deterministic conserved quantities are $\boldsymbol{u} = (\rho, m_1, m_2, m_3, \rho e)^T = (\rho, \rho v_1, \rho v_2, \rho v_3, \rho e)^T$, where $\rho$ is the mass density, $m_i$ and $v_i$ are the momentum and velocity in $i$-direction, respectively, and $e$ is the total energy. The $i$th column of the convective and viscous fluxes is given by

$$\boldsymbol{G}^i = \begin{pmatrix} \rho v_i \\ \rho v_1 v_i + \delta_{1i} p \\ \rho v_2 v_i + \delta_{2i} p \\ \rho v_3 v_i + \delta_{3i} p \\ \rho e v_i + p v_i \end{pmatrix} \quad \text{and} \quad \boldsymbol{H}^i = \begin{pmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} v_j - Q_i \end{pmatrix}, \qquad i = 1, 2, 3. \tag{2}$$

Here, $\delta_{ij}$ is the Kronecker Delta, $p = p(\boldsymbol{u})$ is the pressure given by the ideal gas equation

$$p = \rho R \mathcal{T} = (\gamma - 1)\rho \left( e - \frac{1}{2} \boldsymbol{v} \cdot \boldsymbol{v} \right), \tag{3}$$

with temperature $\mathcal{T}$. Furthermore, $\tau_{ij}$ denote the entries of the viscous stress tensor

$$\boldsymbol{\tau} = \mu \left( \nabla_{\boldsymbol{x}} \boldsymbol{v} + (\nabla_{\boldsymbol{x}} \boldsymbol{v})^T - \frac{2}{3} (\nabla_{\boldsymbol{x}} \cdot \boldsymbol{v}) \boldsymbol{I} \right), \tag{4}$$

[1] https://github.com/flexi-framework/sg-flexi

and $Q_i$ are the components of the heat flux vector $\boldsymbol{Q} = -k \nabla_{\boldsymbol{x}} \mathcal{T}$. Fluid dependent variables used above are the specific gas constant $R$, the adiabatic coefficient $\gamma$, the kinematic viscosity $\mu$ and the heat conductivity $k$. They are all assumed to be constant in space and time. By neglecting viscous effects and the heat flux i.e. $\boldsymbol{H} = 0$, we immediately obtain the Euler equations in conservation form from (1). The equations can easily be reduced to two space dimensions by setting all derivatives with respect to the third dimension as well as the third velocity component to zero.

In reality, due to measurement or modeling errors, it is often not possible to prescribe model parameters or initial and boundary conditions exactly. We model these uncertain parameters as random variables and propagate them through space and time using (1). To this end we consider a $M$-dimensional real-valued random vector $\boldsymbol{\xi} = (\xi^1, \ldots, \xi^M)$ defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ which represents various uncertainties, for example an uncertain viscosity, uncertain initial data, or uncertain boundary conditions. We denote the image of the random vector $\Gamma$, which will also be dubbed the stochastic domain in the following. We assume that the entries of the random vector are independent of each other. Consequently, the solution $\boldsymbol{u}$ depends not only on space and time, but also on $\boldsymbol{\xi}$, i.e.

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}) : D \times \mathbb{R}_+ \times \Gamma \to \mathcal{U} \subset \mathbb{R}^{n_{\text{var}}}. \tag{5}$$

Hence, our equations of interest are the following random Navier-Stokes equations in conservation form,

$$\partial_t \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}) + \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{G}(\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}), \boldsymbol{\xi}) - \boldsymbol{H}(\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}), \nabla_{\boldsymbol{x}} \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi})), \boldsymbol{\xi}) = 0, \tag{6}$$

or shortly

$$\partial_t \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}) + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{F}(\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}), \nabla_{\boldsymbol{x}} \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\xi}), \boldsymbol{\xi}) = 0. \tag{7}$$

We augment (7) with suitable (random) boundary and initial condition, which we denote by

$$\boldsymbol{u}(\boldsymbol{x}, 0, \boldsymbol{\xi}) = \boldsymbol{u}^0(\boldsymbol{x}, \boldsymbol{\xi}), \qquad B(\boldsymbol{u}) = g(\boldsymbol{x}, \boldsymbol{\xi}).$$

The boundary operator $B$, the boundary data $g$ and the initial condition $\boldsymbol{u}^0$ will be specified when we describe the numerical experiments in Section 5.

Our goal is to approximate the solution of (7) using a Stochastic Galerkin (SG) discretization in the stochastic domain and a Discontinuous Galerkin Spectral Element Method (DGSEM) method for the space-time discretization of (7). A detailed description of the resulting SG-DGSEM discretization is the content of the following section.

## 3. Space-time-stochastic discretization

In this section we describe the spatio-temporal and stochastic discretization of (7). In Section 3.1, we start with the SG method used for stochastic discretization, including details on elemental operations, the multi-element approach, and a hyperbolicity-preserving limiter. In Section 3.2, we briefly present the DGSEM method used for discretization in space and time, including remarks on Riemann solvers for the SG system.

### 3.1. The stochastic Galerkin method

The Stochastic Galerkin (SG) method goes back to Ghanem and Spanos [24] and relies on the seminal work of [1,2] which states that the solution of (7) can be developed into a generalized Fourier series using a suitable orthogonal polynomial basis. We consider an approximation $\boldsymbol{u}^{\mathcal{K}}$ in a finite-dimensional ansatz space given by

$$\boldsymbol{u}^{\mathcal{K}}(\boldsymbol{x}, t, \boldsymbol{\xi}) := \sum_{\kappa \in \mathcal{K}} \widehat{\boldsymbol{u}}_{\kappa}(\boldsymbol{x}, t)\, \Psi_{\kappa}(\boldsymbol{\xi}) = \widehat{\boldsymbol{u}}^T \boldsymbol{\Psi}, \tag{8}$$

with coefficients $\widehat{\boldsymbol{u}} \in \mathbb{R}^{n_{\text{var}} \times |\mathcal{K}|}$. The ansatz functions $\{\Psi_\kappa(\boldsymbol{\xi})\}_{\kappa \in \mathcal{K}}$ in (8) are chosen to be orthonormal with respect to the joint probability density function of the uncertain input, i.e. $\langle \Psi_\kappa, \Psi_\iota \rangle = \delta_{\kappa,\iota}$ with the inner product

$$\langle g(\boldsymbol{\xi}), h(\boldsymbol{\xi}) \rangle = \int_{\boldsymbol{\xi} \in \Gamma} g(\boldsymbol{\xi}) h(\boldsymbol{\xi}) f_\Gamma(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi}, \tag{9}$$

where $f_\Gamma(\boldsymbol{\xi}) : \Gamma \to \mathbb{R}_+$ is the joint probability density function of the random vector. In the multivariate case $M > 1$, a set of one-dimensional orthonormal polynomials $\{\Psi_{\kappa_s}(\xi^s)\}_{\kappa_s=0}^{\infty}$ is considered for each random variable $\xi^s$, $s = 1, \dots, M$. The multivariate orthonormal basis functions $\Psi_i(\boldsymbol{\xi})$ are constructed as the product

$$\Psi_\kappa(\boldsymbol{\xi}) = \prod_{s=1}^{M} \Psi_{\kappa_s}(\xi^s). \tag{10}$$

We restrict the multivariate basis to polynomials with combined polynomial degree less or equal to $N_{\text{stoch}}$ as introduced in [24], i.e. we consider the complete polynomial space described by the following index set, cf. [18,25].

$$\kappa \in \mathcal{K} := \left\{ \kappa = (\kappa_1, \dots, \kappa_M)^T \in \mathbb{N}_0^M \;\middle|\; \sum_{s=1}^{M} \kappa_s \le N_{\text{stoch}} \right\}. \tag{11}$$

We suppose that the coefficients in (8) are determinied such that

$$\partial_t \left\langle \widehat{\boldsymbol{u}}^T \boldsymbol{\Psi}, \Psi_k \right\rangle + \nabla_{\boldsymbol{x}} \cdot \left\langle \boldsymbol{F}\left( \widehat{\boldsymbol{u}}^T \boldsymbol{\Psi}, \nabla_{\boldsymbol{x}} \widehat{\boldsymbol{u}}^T \boldsymbol{\Psi}, \boldsymbol{\xi} \right), \Psi_k \right\rangle = 0, \tag{12}$$

holds for all $k \in \mathcal{K}$, i.e., $\widehat{\boldsymbol{u}}_k$ is the Galerkin projection to span $\{\Psi_k\}$ in the stochastic space. Due to orthonormality of the polynomial basis, the first term reduces to $\partial_t \widehat{\boldsymbol{u}}_k$. If we write the entries of the rank-2 tensor $\widehat{\boldsymbol{u}}$ into a vector, we are able to write the whole discretized system as the following modified deterministic conservation law

$$\partial_t \widehat{\boldsymbol{u}} + \nabla_{\boldsymbol{x}} \cdot \widetilde{\boldsymbol{F}}(\widehat{\boldsymbol{u}}, \nabla_{\boldsymbol{x}} \widehat{\boldsymbol{u}}, \boldsymbol{\xi}) = 0. \tag{13}$$

Here, $\widehat{\boldsymbol{u}} \in \mathbb{R}^{n_{\text{var}} \cdot |\mathcal{K}|}$ and $\widetilde{\boldsymbol{F}} : \mathbb{R}^{n_{\text{var}} \cdot |\mathcal{K}|} \to \mathbb{R}^{n_{\text{var}} \cdot |\mathcal{K}|}$, where $|\mathcal{K}| = \binom{M+N_{\text{stoch}}}{M} - 1$ is the number of basis polynomials of (11). The modified system (13) is a strongly coupled system of conservation laws with $n_{\text{var}} \cdot |\mathcal{K}|$ unknowns, which can now be solved using the DGSEM method. The discretized system is deterministic in the sense that the dependence on the random vector $\boldsymbol{\xi}$ is eliminated by integration.

The unknowns are the stochastic moments $\widehat{\boldsymbol{u}}_k$ of the solution $\boldsymbol{u}$. Thus, mean and variance can be retrieved via

$$\mathbb{E}(\boldsymbol{u}) \approx \widehat{\boldsymbol{u}}_0 \quad \text{and} \quad \text{Var}(\boldsymbol{u}) \approx \sum_{\kappa \in \mathcal{K} \setminus (0,\dots,0)^T} \widehat{\boldsymbol{u}}_\kappa^2. \tag{14}$$

### 3.1.1. Elementary operations and the pseudo-spectral method

For highly nonlinear flux functions $\boldsymbol{F}(\boldsymbol{u}, \nabla_{\boldsymbol{x}} \boldsymbol{u}, \boldsymbol{\xi})$, the second integral in (12) may not be evaluated analytically or only at very high computational cost. In the pseudo-spectral approach [6,26–28], the nonlinear function $\boldsymbol{F}$ is split into elemental operations, which are individually projected onto the polynomial space with a Galerkin ansatz. As an example for this procedure, let us consider the first term of the second component of $\boldsymbol{G}^1$ in (2), which is $\rho v_1^2 := \alpha$. It can be expressed in terms of the conserved quantities $\boldsymbol{u}$ as $\alpha = m_1^2/\rho$. We split the evaluation of this term into the fraction $v_1 = m_1/\rho$ and the multiplication $\alpha = m_1 v_1$. We recall that in SG the input quantities to these operations, i.e. $\rho, m_1, v_1$, are represented as polynomials as in (8).

Consequently, multiplications and fractions are computed by performing an orthogonal projection, i.e. projecting products of two stochastic polynomials $\boldsymbol{g}^{\mathcal{K}}$ and $\boldsymbol{h}^{\mathcal{K}}$ yields

$$\left\langle \boldsymbol{g}^{\mathcal{K}} \boldsymbol{h}^{\mathcal{K}}, \Psi_k \right\rangle = \sum_{i,j} \widehat{g}_i \widehat{h}_j \left\langle \Psi_i \Psi_j, \Psi_k \right\rangle = \sum_{i,j} \widehat{g}_i \widehat{h}_j C_{ijk} = \widehat{\boldsymbol{g}}^\top C_k \widehat{\boldsymbol{h}}, \tag{15}$$

where $C_{ijk} = \langle \Psi_i \Psi_j, \Psi_k \rangle$ are the entries of a rank-3 tensor $\boldsymbol{C}$, which is often called the Galerkin multiplication tensor [6] and will be dubbed the multiplication tensor in the following. Computing an SG discretized product of two variables thus requires computation of a double sum of products. In Appendix A we present an optimized computation of the SG product for a stochastically one-dimensional product.

SG fractions of the form $r = g/h$ can be discretized by re-writing them to $rh = g$. This equation can be projected as in (15) resulting in a system of linear equations, which is solved for the moments $\widehat{\boldsymbol{r}}$ of $r$ (though at high computational cost), cf. [17,26,27].

Stochastic integrals arising from the projection of any nonlinear operation $\mathcal{N}(\boldsymbol{g}^{\mathcal{K}})$ with operands $\boldsymbol{g}^{\mathcal{K}}$, i.e. integrals of the form $\langle \mathcal{N}(\boldsymbol{g}^{\mathcal{K}}), \Psi_k \rangle$, can also be evaluated numerically, as introduced in [27] based on the works [29] and [3]. We use a Gauß quadrature with $(Q_\Gamma + 1) \in \mathbb{N}$ points and weights $(\boldsymbol{\xi}_\beta, \omega_\beta)$, $\beta = 0, \dots, Q_\Gamma$. For a multi-dimensional stochastic domain $\Gamma \subset \mathbb{R}^M$ we use tensor-product quadrature rules of one-dimensional quadrature rules, described by the multi-index set $\beta \in \mathcal{P} := \{\beta = (\beta_1, \dots, \beta_M)^\top \in \mathbb{N}_0^M \mid \beta_s \le Q_\Gamma, \quad s = 1, \dots, M\}$. Specifically, for a uniformly distributed random variable and Legendre basis functions, we use the Gauß-Legendre quadrature rule, for a normal distribution we use the Gauß-Hermite quadrature rule, cf. [1].

Hence, for any nonlinear operation $\mathcal{N}(\boldsymbol{g}^{\mathcal{K}})$ the operands are evaluated at quadrature points $\boldsymbol{\xi}_\beta$ in the stochastic space,

$$\boldsymbol{g}^{\mathcal{K}}(\boldsymbol{\xi}_\beta) = \sum_{\kappa \in \mathcal{K}} \widehat{g}_\kappa \Psi_\kappa(\boldsymbol{\xi}_\beta). \tag{16}$$

The nonlinear operation $\mathcal{N}$ is then carried out at these quadrature points $\mathcal{N}_\beta := \mathcal{N}(\boldsymbol{g}^{\mathcal{K}}(\boldsymbol{\xi}_\beta))$, and the output is projected back onto the orthonormal basis using numerical quadrature

$$\widehat{\mathcal{N}}_\kappa := \left\langle \mathcal{N}(\boldsymbol{g}^{\mathcal{K}}), \Psi_\kappa \right\rangle \approx \sum_{\beta \in \mathcal{P}} \mathcal{N}_\beta \, \Psi_\kappa(\boldsymbol{\xi}_\beta) \, \omega_\beta, \tag{17}$$

for all $\kappa \in \mathcal{K}$. This method introduces an additional error due to the non-exact integration and thus the Gaussian quadrature should include sufficiently many points.

### 3.1.2. The multi-element SG method

In order to improve accuracy, especially in the presence of discontinuities, a multi-element Stochastic Galerkin (ME-SG) approach introduced by Wan and Karniadakis [30,31] is employed. The idea is to divide the image of each random variable into intervals, which leads to a decomposition of the stochastic space into disjoint stochastic elements. On each stochastic element, an SG simulation is carried out. In order to derive the ME-SG method, new auxiliary random variables are introduced, as laid out in the following.

For ease of presentation, we restrict ourselves to the case where the image of each random variable is the interval [0,1], such that the stochastic domain becomes $\Gamma = \times_{s=1}^{M} [0,1]$. We let $0 = d_1 < d_2 < \ldots < d_{N_\Gamma+1} = 1$ be a decomposition of [0,1]. We further define the intervals $D_n := [d_n, d_{n+1})$, for $n = 1, \dots, N_\Gamma - 1$, and $D_{N_\Gamma} := [d_{N_\Gamma}, d_{N_\Gamma+1}]$. We introduce an index-set which assigns a scalar index to the multi-index of the sub-intervals of the random variables $\mathcal{L} := \{l = (l_1, \dots, l_M)^\top \in \mathbb{N}_0^M : l_s \le N_\Gamma, \quad s = 1 \dots, M\}$. This allows us to define for each index $l \in \mathcal{L}$ the multi-element $D_l := \times_{s=1}^{M} D_{l_s}$ as cartesian product of intervals $D_{l_s}$.

We now consider a new local random variable $\boldsymbol{\xi}_l : \boldsymbol{\xi}^{-1}(D_l) \to D_l$ on the local probability space $(\boldsymbol{\xi}^{-1}(D_l), \mathcal{F} \cap \boldsymbol{\xi}^{-1}(D_l), \mathbb{P}(\cdot | \boldsymbol{\xi}^{-1}(D_l)))$. Using Bayes' rule we are able to compute the local probability density function of $\boldsymbol{\xi}_l$ via

$$f_{D_l}(\boldsymbol{\xi}_l) := f_{D_l}(\boldsymbol{\xi}_l | \boldsymbol{\xi}^{-1}(D_l)) = \frac{f_\Gamma(\boldsymbol{\xi}_l)}{\mathbb{P}(\boldsymbol{\xi}^{-1}(D_l))}, \qquad \boldsymbol{\xi}_l \in D_l, \tag{18}$$

where $\mathbb{P}(\boldsymbol{\xi}^{-1}(D_l)) > 0$ for $l \in \mathcal{L}$ can be assumed w.l.o.g., due to the independence of the corresponding random variables. We use uniformly distributed random variables, where the local random variables are also uniformly distributed, so that the set of local basis polynomials is known. For other distributions, the basis polynomials can be computed numerically from a Gram-Schmidt orthogonalization or the procedure described in [31]. We parametrize the uncertain input in (7) using the local random variable $\boldsymbol{\xi}_l$ and consider a local approximation on every ME $D_l$. Due to the disjoint decomposition of the stochastic domain, the SG method from Section 3.1 can be applied to every ME independently. Let us note that this allows for a trivial parallelization strategy in the stochastic domain.

If we let $\{\Psi^l_\kappa(\boldsymbol{\xi}_l)\}_{\kappa\in\mathcal{K}}$ be the orthonormal polynomials with respect to the conditional probability density function (18), we may consider the local gPC approximation in element $D_l$,

$$\boldsymbol{u}^l(\boldsymbol{x},t,\boldsymbol{\xi}_l) \approx \sum_{\kappa\in\mathcal{K}} \widehat{\boldsymbol{u}}^l_\kappa(\boldsymbol{x},t)\Psi^l_\kappa(\boldsymbol{\xi}_l), \tag{19}$$

for all $l \in \mathcal{L}$. The global approximation (8) can then be written as

$$\boldsymbol{u}(\boldsymbol{x},t,\boldsymbol{\xi}) = \sum_{l\in\mathcal{L}} \boldsymbol{u}^l(\boldsymbol{x},t,\boldsymbol{\xi}) \approx \sum_{l\in\mathcal{L}}\sum_{\kappa\in\mathcal{K}} \widehat{\boldsymbol{u}}^l_\kappa(\boldsymbol{x},t)\Psi^l_\kappa(\boldsymbol{\xi})\chi_{D_l}(\boldsymbol{\xi}), \tag{20}$$

where $\chi_{D_l}$ is the indicator function of $D_l$.

In a stochastic post-processing step the SG approximations on each ME are combined into expectation and variance of the quantity of interest for the original set of uncertain variables. The expected value and variance of $\boldsymbol{u}$ are obtained as weighted sum of local mean and variance, i.e.

$$\begin{aligned}\mathbb{E}(\boldsymbol{u}) &\approx \int_\Gamma \sum_{l\in\mathcal{L}}\sum_{\kappa\in\mathcal{K}} \widehat{\boldsymbol{u}}^l_\kappa \Psi^l_\kappa(\boldsymbol{\xi})\chi_l(\boldsymbol{\xi}) f_\Gamma(\boldsymbol{\xi})\ \mathrm{d}\boldsymbol{\xi} \\ &= \sum_{l\in\mathcal{L}}\sum_{\kappa\in\mathcal{K}} \widehat{\boldsymbol{u}}^l_\kappa \int_{D_l} \Psi^l_\kappa(\boldsymbol{\xi})\Psi^l_0(\boldsymbol{\xi})\mathbb{P}(\boldsymbol{\xi}^{-1}(D_l)) f_{D_l}(\boldsymbol{\xi}_l)\ \mathrm{d}\boldsymbol{\xi}_l \\ &= \sum_{l\in\mathcal{L}} \mathbb{P}(\boldsymbol{\xi}^{-1}(D_l))\widehat{\boldsymbol{u}}^l_0,\end{aligned}$$

$$\mathrm{Var}(\boldsymbol{u}) \approx \sum_{l\in\mathcal{L}} \left(\mathrm{Var}(\boldsymbol{u}^l) + (\widehat{\boldsymbol{u}}^l_0 - \mathbb{E}(\boldsymbol{u}))^2\right)\mathbb{P}(\boldsymbol{\xi}^{-1}(D_l)),$$

where the local variance $\mathrm{Var}(\boldsymbol{u}^l)$ is given by

$$\mathrm{Var}(\boldsymbol{u}^l) \approx \int_{D_l} \Big(\sum_{\kappa\in\mathcal{K}} \widehat{\boldsymbol{u}}^l_\kappa\Big)^2 f_{D_l}(\boldsymbol{\xi}_l)\ \mathrm{d}\boldsymbol{\xi}_l - (\widehat{\boldsymbol{u}}^l_0)^2 = \sum_{\kappa\in\mathcal{K}\setminus(0,\dots,0)^T} (\widehat{\boldsymbol{u}}^l_\kappa)^2.$$

Whenever we employ the SG method in combination with the ME ansatz, we call the method ME-SG method.

### 3.1.3. A hyperbolicity-preserving limiter

In [32] it has been proven that for non-scalar and non-symmetric hyperbolic systems like the Euler equations, the SG-discretized system (13) may lose its hyperbolicity i.e. the flux Jacobian of (13) has some complex eigenvalues. To circumvent this problem a hyperbolicity-preserving limiter has been developed in [33] and advanced by the authors of [9] to a limiter for the SG-DGSEM method. The limiting strategy ensures that the space-stochastic cell mean $\bar{\boldsymbol{u}}$ generates a hyperbolic SG-system after one Euler forward time-step. A point-wise (in spatial and stochastic domain) application of a slope-limiter, which limits the solution towards the admissible cell-mean, which is guaranteed to generate a hyperbolic SG-system, induces a hyperbolic system. We employ this limiter in our software framework and refer for further information concerning the limiter to [9].

## 3.2. Discontinuous Galerkin spectral element method

For the space-time discretization of the SG system (13), we use a high-order Discontinuous Galerkin Spectral Element method (DGSEM) in combination with a Runge–Kutta time-stepping. We restrict ourselves to a short summary of the method, further details can be found for example in [34–36].

We partition the physical domain $D$ into $N_e \in \mathbb{N}$ disjoint curved hexahedral elements. The basic idea of DG is to discretize each of these elements with the Galerkin method (in the case of DGSEM with a Lagrange polynomial solution basis on each element) and to allow for discontinuities between the elements, which are treated using Riemann solvers.

For reducing computational complexity, all elements are transformed from physical space onto the reference element $E(\boldsymbol{\eta}) = [-1,1]^3$, such that for each element

$$\partial_t J\widehat{\boldsymbol{u}} + \nabla_\eta \cdot \widetilde{\boldsymbol{F}}(\widehat{\boldsymbol{u}}, \nabla_\eta \cdot \widehat{\boldsymbol{u}}, \boldsymbol{\xi}) = 0, \tag{21}$$

where $J$ is the Jacobian determinant of the mapping $\boldsymbol{x}(\boldsymbol{\eta})$. Multiplication with a test function $\phi$ and integration over the reference element $E$ yields

$$\partial_t \int_E J\widehat{\boldsymbol{u}}\phi\, d\boldsymbol{\eta} + \int_E \nabla_\eta \cdot \widetilde{\boldsymbol{F}}(\boldsymbol{u}, \nabla_{\boldsymbol{x}}\boldsymbol{u}, \boldsymbol{\xi})\phi\, d\boldsymbol{\eta} = 0. \tag{22}$$

Analogously to SG, this step corresponds to a projection onto the test function. To avoid confusion, we do not use the same short hand scalar product notation $\langle\cdot,\cdot\rangle$ as in SG for the DG projection, despite the parallels between both operations. Integration by parts finally leads to the weak form of (1):

$$\partial_t \int_E J\widehat{\boldsymbol{u}}\phi\, d\boldsymbol{\eta} + \oint_{\partial E} (\widetilde{\boldsymbol{F}}\cdot\boldsymbol{n})^*\phi\, d\boldsymbol{\eta} - \int_E \widetilde{\boldsymbol{F}}(\widehat{\boldsymbol{u}}, \nabla_{\boldsymbol{x}}\widehat{\boldsymbol{u}}, \boldsymbol{\xi})\cdot\nabla_\eta\phi\, d\boldsymbol{\eta} = 0, \tag{23}$$

where $\boldsymbol{n}$ is the outward pointing unit normal vector of the reference element surface. At element interfaces, discontinuities are allowed, such that the flux $\widetilde{\boldsymbol{F}}$ is not uniquely defined. We use Riemann solvers (further detailed in Section 3.2.1) to approximate the convective flux and the arithmetic mean of element traces for the viscous flux. This approximation is indicated by the superscript $^*$. Components of the solution vector are approximated on each element by a tensor product of one-dimensional polynomials of order $N \in \mathbb{N}$. Lagrange polynomials are employed as both basis and test functions. We use a collocation approach following [34] using the $(N+1)^3$ Gauß-Legendre or Gauß-Lobatto quadrature points for interpolation and integration.

In (13), the gradient of the solution $\nabla_{\boldsymbol{x}}\widehat{\boldsymbol{u}}$ is needed as input to the viscous terms of the flux function. In the context of a DGSEM discretization, we obtain this gradient with a procedure called lifting as in [36]. To this end, we re-write (13) into the extended system of equations

$$\widehat{\boldsymbol{q}} = \nabla_{\boldsymbol{x}}\widehat{\boldsymbol{u}}, \tag{24}$$

$$\partial_t\widehat{\boldsymbol{u}} + \nabla_{\boldsymbol{x}}\cdot(\widetilde{\boldsymbol{G}}(\widehat{\boldsymbol{u}}) - \widetilde{\boldsymbol{H}}(\widehat{\boldsymbol{u}}, \widehat{\boldsymbol{q}})) = 0. \tag{25}$$

We discretize the components $\widehat{\boldsymbol{q}}_i = \partial_{x_i}\widehat{\boldsymbol{u}}$ of the first equation with DGSEM analogously to the discretization shown above, leading to the weak form

$$\int_E J\widehat{\boldsymbol{q}}_i\phi\, d\boldsymbol{\eta} - \oint_{\partial E} (\widehat{\boldsymbol{u}}\cdot\boldsymbol{n})^{**}\phi\, d\boldsymbol{\eta} + \int_E \widehat{\boldsymbol{u}}\cdot\nabla_\eta\phi\, d\boldsymbol{\eta} = 0. \tag{26}$$

In order to gain the input $\widehat{\boldsymbol{q}}$ for the viscous flux in (23), we solve in each time step the lifting equations (26). We employ a lifting scheme called BR1 [36], where the arithmetic mean is used at discontinuities at the element interfaces $(a)^{**} = (a^+ + a^-)/2$.

To advance the solution in time, we use a high-order explicit Runge-Kutta time integration scheme of Carpenter&Kennedy [21], which is implemented in a low storage version. The time-stepping scheme is constrained by a CFL type condition of the form

$$\Delta t \le \min\left\{\frac{h_{\min}}{\lambda^c_{\max}(2N+1)}, \left(\frac{h_{\min}}{\lambda^\nu_{\max}(2N+1)}\right)^2\right\}. \tag{27}$$

In (27) $h_{\min}$ denotes the smallest characteristic length among all spatial cells, $\lambda^c_{\max} := \left((|v_1| + c) + (|v_2| + c)\right)$ is an estimate for the absolute value of the largest eigenvalue of the convective flux Jacobian, $c := \sqrt{\gamma \frac{p}{\rho}}$ being the speed of sound. Moreover, $\lambda^v_{\max} := \left(\max\left(\frac{4}{3\rho}, \frac{\gamma}{p}\right)\frac{\mu}{Pr}\right)$ is an estimate for the largest eigenvalue of the diffusion matrix of the viscous flux, $Pr = \frac{c_p \mu}{k}$ being the Prandtl number. Following [37], we estimate the largest eigenvalues $\lambda^c_{\max}$ and $\lambda^v_{\max}$ by computing the largest eigenvalue of the flux Jacobians evaluated at quadrature points in stochastic space. We describe this procedure in (30).

If shock waves or other discontinuities are present in the physical domain, the DGSEM scheme is prone to non-physical oscillations of the high-order polynomials. Several approaches exist to address this issue, one of which is a shock capturing method based on Finite-Volume (FV) sub-cells [38]. Here, an indicator function such as [39,40] detects cells in the physical domain where discontinuities are present. High-order cells containing shocks are subdivided into equidistant sub-cells, where the number of sub-cells corresponds to the number of interpolation points, i.e. $(N+1)^3$. A second order total variation diminishing reconstruction procedure improves accuracy while ensuring stability.

We note that the DGSEM method can be derived equivalently for conservation laws in two space dimensions.

### 3.2.1. Riemann solvers for the SG discretized system

To approximate the normal convective flux $\widetilde{\boldsymbol{G}} \cdot \boldsymbol{n}$ at the elements interfaces, we solve one-dimensional Riemann problems along the direction of the normal vector of the corresponding faces. In the DGSEM method, discontinuities are small in smooth regions, so in contrast to low-order finite volume methods, these Riemann solvers only have a minor impact on numerical dissipation and numerical error. For the SG discretized system (13), we employ a computationally inexpensive Lax-Friedrichs Riemann solver or an HLLE flux evaluated at quadrature points. The Lax-Friedrichs numerical flux reads

$$\widetilde{\boldsymbol{G}}^*(\widehat{\boldsymbol{u}}^-, \widehat{\boldsymbol{u}}^+) = \frac{1}{2}\left(\widetilde{\boldsymbol{G}}(\widehat{\boldsymbol{u}}^-) + \boldsymbol{G}(\widehat{\boldsymbol{u}}^+) - \widetilde{\boldsymbol{\lambda}}_{\max}(\widehat{\boldsymbol{u}}^+ - \widehat{\boldsymbol{u}}^-)\right), \tag{28}$$

where $\widehat{\boldsymbol{u}}^-$ and $\widehat{\boldsymbol{u}}^+$ are the states at the element interface from the two adjacent elements. $\widetilde{\boldsymbol{\lambda}}_{\max}$ is the largest eigenvalue of the system matrices, which we denote by eig.

$$\widetilde{\boldsymbol{\lambda}}_{\max} := \max\left(\left\|\widetilde{\boldsymbol{\lambda}}(\widehat{\boldsymbol{u}}^-)\right\|_\infty, \left\|\widetilde{\boldsymbol{\lambda}}(\widehat{\boldsymbol{u}}^+)\right\|_\infty\right) \quad \text{with} \quad \widetilde{\boldsymbol{\lambda}}(\widehat{\boldsymbol{u}}) := \text{eig}\left(\frac{\partial \widetilde{\boldsymbol{G}}(\widehat{\boldsymbol{u}})}{\partial \widehat{\boldsymbol{u}}}\right). \tag{29}$$

Here, $\|\cdot\|_\infty$ denotes the $L^\infty$-norm of a vector. We estimate the maximal absolute interface normal signal speed from an evaluation at quadrature points $\boldsymbol{\xi}_\beta$, $\beta \in \mathcal{P}$ in the stochastic domain, i.e.

$$\left\|\widetilde{\boldsymbol{\lambda}}(\widehat{\boldsymbol{u}})\right\|_\infty \approx \max_\beta\left(\left\|\boldsymbol{\lambda}(\boldsymbol{u}(\boldsymbol{\xi}_\beta))\right\|_\infty\right) \quad \text{with} \quad \boldsymbol{\lambda}(\boldsymbol{u}) := \text{eig}\left(\frac{\partial \boldsymbol{G}(\boldsymbol{u})}{\partial \boldsymbol{u}}\right). \tag{30}$$

Note that the evaluation at quadrature points might in some cases lead to an underestimation of the maximal signal speeds. We have not encountered adverse effects due to this, which might be due to the overall highly dissipative nature of the Lax-Friedrichs flux.

Alternatively, we choose the HLLE numerical flux as in [41], but we evaluate the numerical flux at quadrature points $\{\boldsymbol{\xi}_\beta\}_{\beta\in\mathcal{P}}$ and project the numerical fluxes back onto the orthonormal basis using the procedure in (17).

$$\boldsymbol{G}^*(\boldsymbol{u}^-(\boldsymbol{\xi}_\beta), \boldsymbol{u}^+(\boldsymbol{\xi}_\beta)) := \frac{b^+(\boldsymbol{\xi}_\beta)\boldsymbol{G}(\boldsymbol{u}^-(\boldsymbol{\xi}_\beta)) - b^-(\boldsymbol{\xi}_\beta)\boldsymbol{G}(\boldsymbol{u}^-(\boldsymbol{\xi}_\beta))}{b^+(\boldsymbol{\xi}_\beta) - b^-(\boldsymbol{\xi}_\beta)} + \frac{b^+(\boldsymbol{\xi}_\beta)b^-(\boldsymbol{\xi}_\beta)}{b^+(\boldsymbol{\xi}_\beta) - b^-(\boldsymbol{\xi}_\beta)}(\boldsymbol{u}^+(\boldsymbol{\xi}_\beta) - \boldsymbol{u}^-(\boldsymbol{\xi}_\beta)), \tag{31}$$

with signal speed estimates

$$b^-(\boldsymbol{\xi}_\beta) := \{\boldsymbol{\lambda}_{\min}(\bar{\boldsymbol{u}}(\boldsymbol{\xi}_\beta)), \boldsymbol{\lambda}_{\min}(\boldsymbol{u}^-(\boldsymbol{\xi}_\beta)), 0\},$$
$$b^+(\boldsymbol{\xi}_\beta) := \{\boldsymbol{\lambda}_{\max}(\bar{\boldsymbol{u}}(\boldsymbol{\xi}_\beta)), \boldsymbol{\lambda}_{\max}(\boldsymbol{u}^+(\boldsymbol{\xi}_\beta)), 0\},$$

where $\boldsymbol{\lambda}_{\min}(\boldsymbol{u}^-(\boldsymbol{\xi}_\beta)), \boldsymbol{\lambda}_{\max}(\boldsymbol{u}^+(\boldsymbol{\xi}_\beta)$ are the smallest and largest eigenvalue of $\frac{\partial \boldsymbol{G}(\boldsymbol{u})}{\partial \boldsymbol{u}}$ and $\bar{\boldsymbol{u}}(\boldsymbol{\xi}_\beta)$ is the corresponding Roe mean value, cf[41].

## 4. Employed software: SG-FLEXI

Based on the open source DGSEM flow solver FLEXI [19] (open source on the FLEXI homepage[2]), the code SG-FLEXI for the 3D compressible Navier-Stokes equations has been developed, which is also open source and can be retrieved from GitHub.[3] FLEXI is an unstructured, arbitrary order DGSEM solver designed for high performance computing in computational fluid dynamics. It is written in Fortran, with a focus on performance and massive scalability. The SG extension of the code can handle an arbitrary number of uncertain variables with uniform or normal distribution. Uncertainty can be introduced via initial and boundary conditions, uncertain source terms and uncertain viscosity.

Expanding a code towards Stochastic Galerkin requires the following changes:

- The stochastic operators have to be set up. This includes functions for the stochastic basis polynomials, stochastic qudrature points, the multiplication tensor, and pseudo-spectral substitutes for products, fractions and potentially other nonlinear operations in the flux functions (For example, a routine `SG_Product` is introduced, which takes two vectors with $|\mathcal{K}|$ entries and returns their SG projected product as a vector of length $|\mathcal{K}|$. The same is done for fractions and other non-linear operations).
- Solution arrays and all other arrays with previously $n_{\text{var}}$ entries are extended to $n_{\text{var}} \times |\mathcal{K}|$ entries. All non-linear operations in the flux calculation are replaced by their SG projected equivalents. Riemann solvers suitable for SG discretized systems Section 3.2.1 are introduced instead of conventional ones.
- Projection routines are introduced to project random initial and boundary conditions as well as random source terms onto the polynomial basis.
- Output and post-processing is adapted to include stochastic evaluation. In FLEXI, we use HDF5 for output and an in-house plugin to ParaView®for visualization.

### 4.1. Performance

The calculation of double sums (15) with the sparse multiplication tensor $\boldsymbol{C}$ consumes the largest proportion of computational effort in the SG method for the compressible Euler and Navier-Stokes equations. Its efficiency is therefore crucial to the code. The multiplication tensor is sparse, symmetrical in all directions and solution-independent, and its entries can be pre-computed and stored a priori. Analytical formulae for the tensor entries for uniform and normal distributions can be found in [42]

Both sparsity and symmetry should be exploited to improve computational and memory efficiency. In particular, each index set $i, j, k$ of the double sum should only be looped over if $C_{ijk} \neq 0$. Also, each non-zero value occurring in the tensor should only be stored

[2] www.flexi-project.org
[3] https://github.com/flexi-framework/sg-flexi

once. In the case of one random variable ($M = 1$), the entries and their indices can be addressed directly by nested loops. Details are laid out in Appendix A. In the case of several variables, index arrays are used. For the case of one random variable, index arrays were observed to perform worse than nested loops, as also shown in detail below.

The number of random variables and the stochastic polynomial degree can be prescribed statically at compile time. They are the only variables that the loop controls in the function depend on, which means that all loop index limits are constant and known at compile time. Nonetheless, it was observed that these loops are not unrolled by the investigated compilers. As the employed loops are mostly very short, a considerable share of computational effort is spent for loop control, which is undesirable. Another option has therefore been investigated: A script was incorporated into the compile process, where the operations within the loops are explicitly converted to a list of statements, which then replace the original source code. This can be interpreted as explicit custom loop unrolling. An example pseudo source code generated with this method is shown in Appendix B. Minor further optimizations (no initialization with zero, no multiplication with tensor entries equal to one) are incorporated. In this case, the array for the multiplication tensor $C$ is filled with the same loop layout as in the source code generation script. Note that a scalar index was introduced for sorting of the multi-index $\kappa$.

For the treatment of SG discretized fractions, both methods outlined in Section 3.1.1 are implemented, i.e. both solving a linear equation system (called exact evaluation in the following) and an evaluation at quadrature points. In order to efficiently solve the linear equation system, the diagonal pivoting routine DSYSV from the LAPACK library is used. It solves symmetric positive definite matrices efficiently and exactly. Symmetry of the matrix follows directly from the symmetry of the multiplication tensor. Positive definiteness follows from Theorem 2 in [37] if the denominator is positive in the whole stochastic domain. For the uncertain Navier-Stokes equations, all fractions have the same denominator $\rho(\boldsymbol{\xi})$, which is ensured to be positive by the hyperbolicity preserving limiter described in Section 3.1.3.

For efficient parallelization, the strategy of the baseline solver FLEXI [19] is used without considerable changes. The DGSEM method allows for an outstanding parallel performance, since the high order stencils are compact and only values at cell interfaces have to be communicated. The domain is partitioned into parts of equal size with a space filling curve, where each DG element is handled entirely by one process. At the interfaces of the thread-local subdomains, master and slave sides are determined. The solution from the neighbouring cell which is needed for the numerical flux approximation is communicated from the slave side to the master side, where the flux is computed and sent back to the slave side. Latency hiding is used in order to improve parallel performance. The stochastic polynomial coefficients are not parallelized, since they are strongly coupled. In ME-SG, several independent SG simulations are carried out, so stochastic multi-elements can be handled by different processes without additional communication. Nonetheless, in the current implementation the runs for each multi-element are all incorporated into one parallel run of SG-FLEXI. This facilitates organized parallel output to one common file based on the HDF5 library. Parallel performance generally improves with stochastic dimensionality, as the amount of communicated data increases less than the computational effort on each core. However, each process has to handle an entire DG element, which can lead to a high minimum load per core for many stochastic degrees of freedom and high DG polynomial degree.

We investigate the effects of the performance optimizations and of the parallelization strategy. We first consider thread level performance in Fig. 1a and b. To this end, the freestream (stochastically, spatially and temporally constant state)

$$\boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi}) = \begin{cases} \rho(\boldsymbol{x}) & = 1, \\ v_1(\boldsymbol{x}) & = 0, \\ v_2(\boldsymbol{x}) & = 0, \\ p(\boldsymbol{x}) & = 1 \end{cases} \tag{32}$$

is considered. Dummy stochastic variables are introduced, where only the mean is initialized with a non-zero value. The physical polynomial degree is chosen as $N = 10$ and a grid of $4 \times 4$ DG-elements with periodic boundary conditions is used. The numerical error for this example is determined by machine accuracy. In all computations, the stochastic polynomial degree is given statically at compile time for better comparability. Computations were carried out in a single thread on a Intel i5-6300U CPU. Results were averaged over five repeated runs. Fig. 1a shows the performance index

$$\mathrm{PID} = \frac{\text{wall time}}{n_{\mathrm{DOF,phys}} \times n_{\mathrm{RKStages}}}, \tag{33}$$

where wall time refers to the wall time for the whole simulation (excluding initialization routines and file I/O, which do not contribute considerably), $n_{\mathrm{DOF,phys}}$ is the number of physical degrees of freedom and $n_{\mathrm{RKStages}}$ is the number of stages of the employed Runge-Kutta time integration scheme multiplied with the computed number of time steps. The performance index is plotted versus the number of stochastic degrees of freedom (equal to $N_{\mathrm{stoch}} + 1$) for computations with different numbers of uncertain variables $M$. The conventional version using index vectors was used for the routine for computation of stochastic products in all computations in Fig. 1a. The computation time and therefore also the PID increases with increasing $n_{\mathrm{DOF,stoch}}$, and the slope increases as well, as parts of the computation cost independent of $n_{\mathrm{DOF,stoch}}$ dominate for low $n_{\mathrm{DOF,stoch}}$.

Figure 1 b shows the speed-up achieved with the two optimizations for SG product evaluation discussed above, namely the version with nested loops described in Appendix A (only applicable for $M = 1$) and the version with manual loop unrolling. Speed-up is computed as the ratio of the baseline PID (using index vectors, as shown in the previous subplot) and the respective optimized PID. The optimized nested loop layout yields significant improvements except for stochastic polynomial degrees of two to four. For these computations, it is assumed that the stochastic polynomial degree is so high that the compiler does not remove any loops as obsolete, but low enough that loop control takes a substantial share of computational cost. The more significant savings are however achieved by the manually scripted source code generation in the compile process. For $M = 1$ and high stochastic polynomial degrees, overall computation time is reduced by more than a factor of two. The speed-up somewhat decreases with increasing stochastic dimension, which can be probably attributed to the fact that numerical evaluation of fractions requires tensor product quadrature, which is very costly for high stochastic dimensions and thus increasingly dominates computation time.

Weak scaling is investigated to evaluate the parallelization strategy of the code. The same problem is simulated, but on a three-dimensional domain and with adapted physical polynomial degrees of four and eight. The number of elements per core is kept constant at one or four, and the number of cores is increased proportionally to the number of DG elements. One uncertain variable $M = 1$ is considered. Computations were carried out on the Cray XC40 (Hazel Hen) at the High-Performance Computing Center Stuttgart (HLRS). The baseline computation is carried out on one node (24 cores, 24 threads), the largest computation is on 512 nodes (12288 cores). The results are shown in Fig. 1c. Weak scaling is excellent, with a parallel efficiency of more than 74% in all

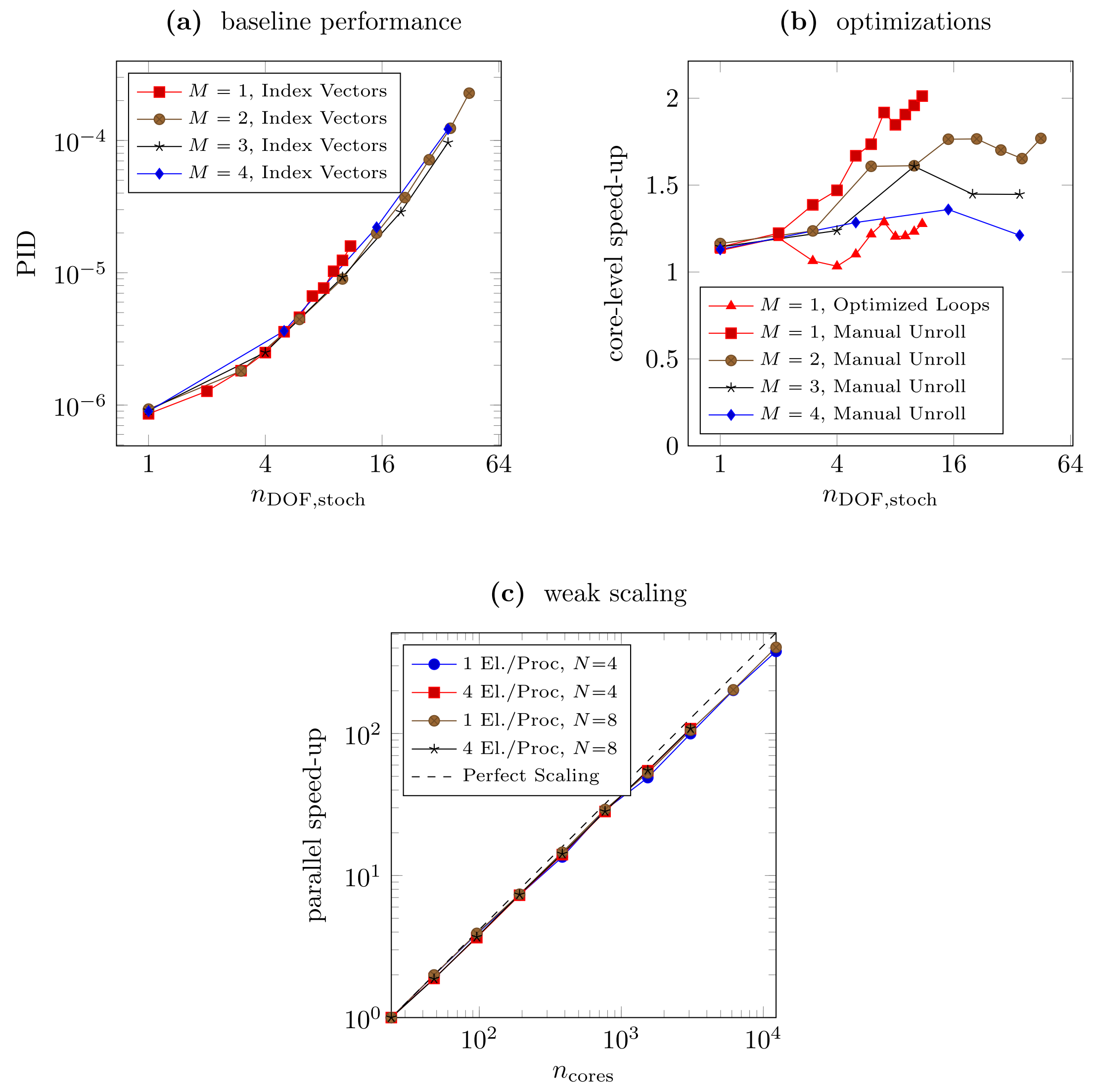

**Fig. 1.** Performance of SG-FLEXI. (a) performance index plotted over stochastic degrees of freedom using index vectors for SG product evaluation; (b) core-level speed-up for different optimizations; (c) weak scaling (speed-up over number of cores).

cases and no sign of speed-up flattening out at the largest number of cores.

## 5. Numerical results

In this section we present various numerical results for our (ME-)SG-DGSEM scheme SG-FLEXI. As a first numerical experiment in Section 5.1 we consider the convergence towards a benchmark solution of the Euler equations which is obtained with the method of manufactured solutions. The second numerical experiment in Section 5.2 considers the viscous flow over a NACA 0012 airfoil, where we let the dynamic viscosity and the angle of attack be uncertain. In Section 5.3 we consider a forward facing step problem with uncertain Mach number. This problem poses a challenge to the ME-SG-DGSEM scheme because of the possible loss of hyperbolicity due to strong shocks in the flow field. Finally, in Section 5.4 we simulate the inviscid flow around a 3D spacecraft model. We demonstrate that our code is capable of simulating flow in complex three-dimensional domain geometries in conjunction with strong shocks.

In the following sections we denote the number of spatial cells by $N_e$, the corresponding mesh-width by $h$, the DG polynomial degree by $N$ and the degree of the SG polynomial by $N_{\mathrm{stoch}}$. The corresponding experiments can be reproduced using the open source code.

### 5.1. Convergence

To test the convergence of the SG-DGSEM scheme for the random Euler equations, we employ the method of manufactured solutions. To this end we insert a source term into (7), such that the solution is given by

$$\boldsymbol{u}(\boldsymbol{x},t,\boldsymbol{\xi}) = \begin{pmatrix} \rho(\boldsymbol{x},t,\boldsymbol{\xi}) \\ m_1(\boldsymbol{x},t,\boldsymbol{\xi}) \\ m_2(\boldsymbol{x},t,\boldsymbol{\xi}) \\ m_3(\boldsymbol{x},t,\boldsymbol{\xi}) \\ \rho e(\boldsymbol{x},t,\boldsymbol{\xi}) \end{pmatrix} = \begin{pmatrix} 2+\xi_2\cos(2\pi(x_1-\xi_1 t)) \\ 2+\xi_2\cos(2\pi(x_1-\xi_1 t)) \\ 0 \\ 0 \\ (2+\xi_2\cos(2\pi(x_1-\xi_1 t)))^2 \end{pmatrix}. \tag{34}$$

Here we consider two uniformly distributed random variables $\boldsymbol{\xi} = (\xi_1, \xi_2)$. More specifically we let $\xi_1 \sim \mathcal{U}(0.1, 0.5)$ and for uniform spatial mesh refinement, i.e. $h$-refinement, we consider $\xi_2 \sim \mathcal{U}(0.1, 0.3)$, whereas for $N_{\mathrm{stoch}}$-refinement we consider $\xi_2 \sim \mathcal{U}(0.1, 1.0)$. Therefore, we use Gauß-Legendre orthonormal polynomials in the stochastic space. The solution is computed up to $T = 1$ on the spatial domain $D = (-1, 1)^2$, where we consider periodic boundary conditions. As numerical flux we choose the Lax-

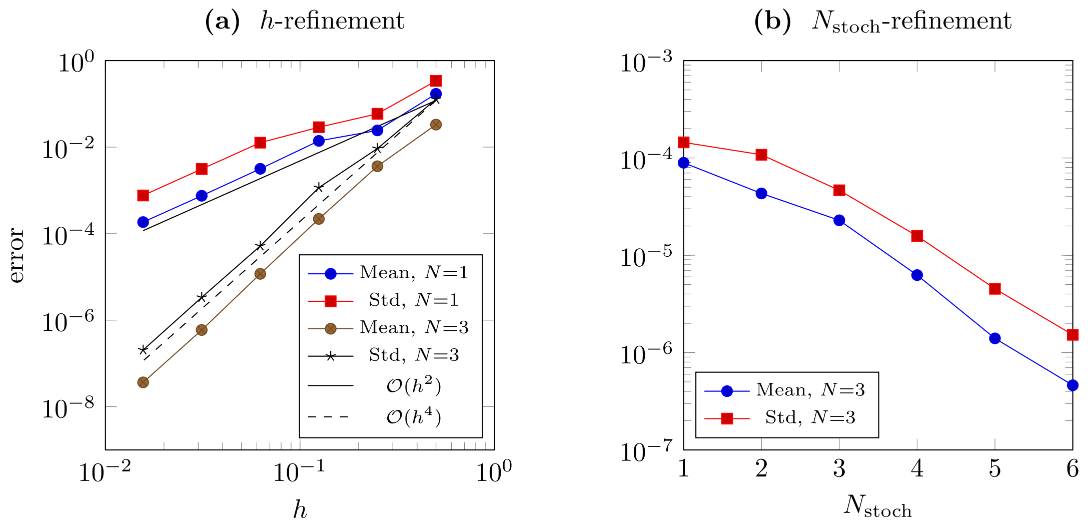

**Fig. 2.** Convergence for exact solution (34) under spatial mesh refinement ($N_e$-refinement) and stochastic refinement ($N_{\text{stoch}}$-refinement). Example (5.1).

Friedrichs flux (28). The error is measured in mean and standard deviation (std) of density at time $T = 1$ in the $L^2(D)$-norm which we approximate by a tensor product Gauß quadrature rule with 10 points (in one dimension) in every spatial cell. For the stochastic quadrature we use 20 points (in one dimension).

In Fig. 2a we consider the error vs. $h$-refinement for two different DG polynomial degrees. We observe that both the error in mean and std converge with the expected order of convergence, which is $N + 1$. Figure 2b shows the error vs. stochastic polynomial degree $N_{\text{stoch}}$ in a semi-log plot for a DG polynomial degree $N = 3$ and $N_e = 32 \times 32$ spatial cells. For the regular solution (34), the error clearly exhibits spectral convergence.

### 5.2. 2D flow around a NACA 0012 airfoil

In this example we consider the viscous flow over a NACA 0012 airfoil, which is described by the compressible Navier-Stokes equations. We consider two operational uncertainties, namely the dynamic viscosity of the fluid and the angle of attack of the airfoil. More specifically, we let $\mu \sim \mathcal{N}(0.001, 0.0001^2)$ and $\alpha \sim \mathcal{N}(3^\circ, (0.5^\circ)^2)$. The normal distributions with their infinite support are strictly speaking not suited for these parameters, as they can lead to negative viscosity and an infinitely rotated airfoil. The probabilities of such events are however very small and numerically irrelevant, such that these distributions are acceptable approximations for this demonstrative example. The initial configuration for this problem reads as follows:

$$\boldsymbol{u}(t=0,\boldsymbol{x},\boldsymbol{\xi}) = \begin{cases} \rho(\boldsymbol{x}) & = 1.0, \\ v_1(\boldsymbol{x},\boldsymbol{\xi}) & = \cos(\alpha), \\ v_2(\boldsymbol{x},\boldsymbol{\xi}) & = \sin(\alpha), \\ p(\boldsymbol{x}) & = 4.464. \end{cases} \tag{35}$$

We set the ideal gas constant $R = 2.857$, the Prandtl number $Pr = 0.72$ and the isentropic coefficient in (3) to $\gamma = 1.4$. The solution is computed up to $T = 5$.

For SG-DGSEM we consider $N_{\text{stoch}} = 10$ and a DG polynomial degree of $N = 3$. As numerical flux we use the Lax-Friedrichs numerical flux (28). To validate our result we compute mean and std with the non-intrusive Stochastic Collocation method (cf. [25]) using the deterministic DGSEM code FLEXI. We dub this method SC-DGSEM. The DG polynomial degree for SC-DGSEM is also $N = 3$. For the two-dimensional stochastic domain we use a tensor-product of Gauß-Hermite quadrature points as collocation points, where we use 11 points in each stochastic dimension. The spatial computational grid uses an unstructured mesh with 652 quadrilateral elements and a third order curved boundary surface. For the boundary condition at inflow and outflow we employ Dirichlet boundary conditions specified by (35). Around the airfoil we prescribe an isothermal wall with a no-slip condition, where the temperature $\mathcal{T}$ is computed using the ideal gas law (3).

In Fig. 3 mean and std of momentum in $x_1$-direction are plotted, i.e. $m_1$ at final time $T = 5$ obtained with SG-DGSEM and SC-DGSEM. We observe an excellent agreement in mean, cf. Figure 3a and b. The std obtained with SG-DGSEM is slightly smaller than for SC-DGSEM, however the main flow structure of std coincides very good with the result of SC-DGSEM. Due to the random viscosity the highest std can be seen downstream of the airfoil, because in this area the main flow is significantly dominated by viscous effects around the airfoil. For SG product evaluations, the baseline method using index vectors was used in this case, as the optimized loop layout Appendix A is not applicable for $M > 1$ and the high stochastic polynomial degree leads to 9191 non-zero multiplication tensor entries, which is excessive for manual loop unrolling.

### 5.3. Uncertain forward facing step

In this numerical example we consider a classical forward facing step problem from Woodward and Colella [43] modeled by the Euler equations. The computational domain is given by $D = [0, 3] \times [0, 1] \backslash [0.6, 3] \times [0, 0.2]$. We let the Mach number of the flow be uncertain. The initial data in primitive form read as

$$\boldsymbol{u}(t=0,\boldsymbol{x},\boldsymbol{\xi}) = \begin{cases} \rho(\boldsymbol{x}) & = 1.4, \\ v_1(\boldsymbol{x}) & = 3, \\ v_2(\boldsymbol{x}) & = 0, \\ p(\boldsymbol{x},\boldsymbol{\xi}) & = 1 + \xi_1. \end{cases} \tag{36}$$

where $\xi_1 \sim \mathcal{U}(-0.2, 0.2)$. The solution is computed up to $T = 3$ and we let $\gamma = 1.4$.

For ME-SG-DGSEM we consider $N_\Gamma = 8$ MEs and let $N_{\text{stoch}} = 3$. We use ME-SG for this example as discontinuities are present in the physical solution of this problem. The DG polynomial degree is $N = 4$ and the spatial mesh consists of $N_e = 60 \times 100$ cells in front of the step and $N_e = 240 \times 80$ cells above the step. As a reference solution we compute mean and variance using the ME-SC method. We use a piecewise linear stochastic approximation in combination with a fine grid both in the stochastic and in the physical domain. This is to create an accurate, non-oscillating reference solution even in the presence of discontinuities. In the stochastic space, the linear interpolation corresponds to $N_{\text{stoch}} = 1$. In the physical space, a piecewise linear solution approximation is achieved by us-

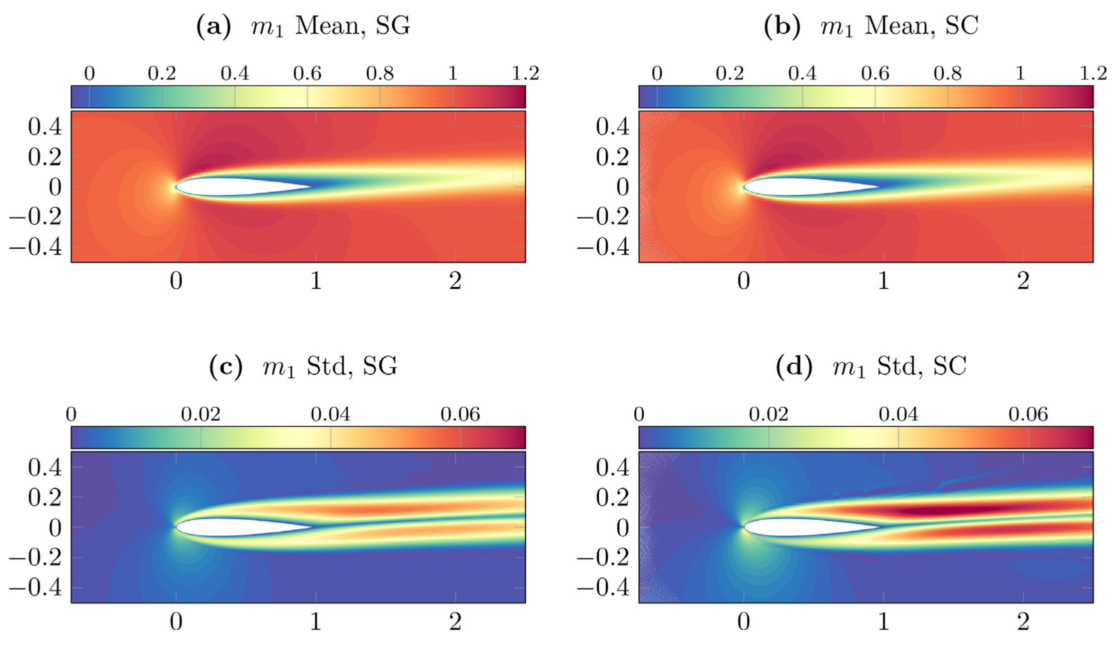

**Fig. 3.** Comparison of mean and std for $m_1$ obtained with SG and SC. Example (5.2).

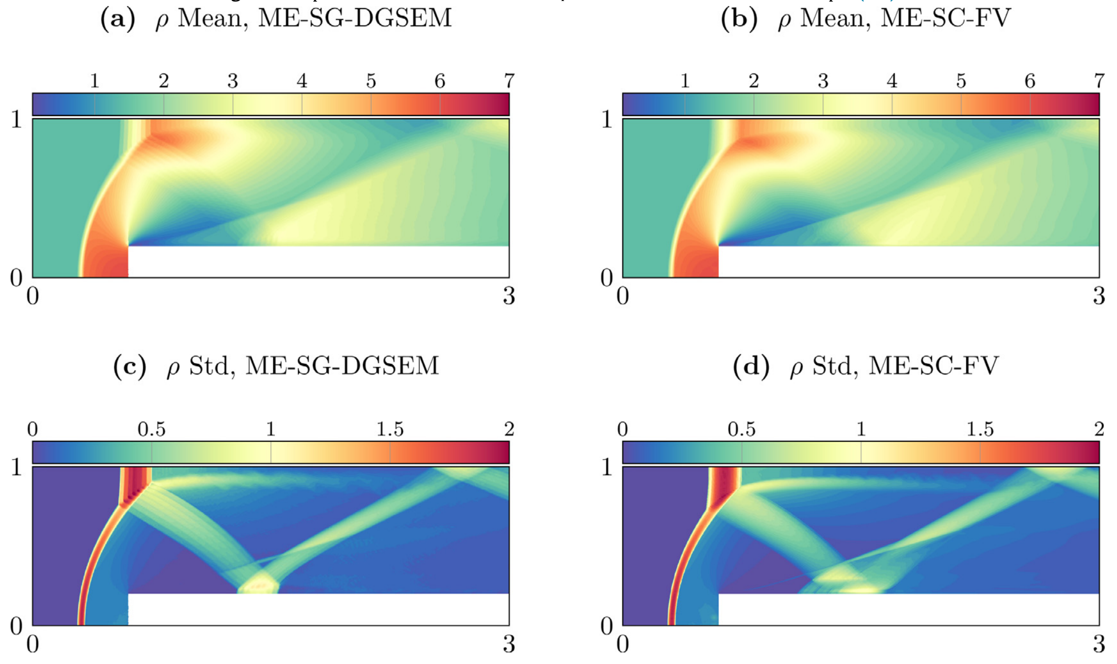

**Fig. 4.** Comparison of mean and std of density $\rho$ at time $T = 3$ obtained with ME-SG-DGSEM and ME-SC-FV. Example 5.3.

ing a purely Finite-Volume method with first-order reconstruction, which is a submodule of the standard FLEXI code. We dub this method ME-SC-FV in the following. Specifically, we use $N_\Gamma = 20$ stochastic MEs. The spatial mesh consists of $N_e = 200 \times 280$ cells in front of the step and $N_e = 600 \times 200$ cells above the step. For both ME-SG and ME-SC-FV we use the HLLE numerical flux (31) at physical element interfaces. At the inflow we prescribe Dirichlet boundary condition given by (36) and we prescribe outflow boundary conditions at the end of the computational domain. The remaining walls are reflective walls, i.e. we employ a slip condition.

Figure 4 shows mean and std of density at time $T = 3$. Overall we see a very good agreement both in mean and std for ME-SG-DGSEM and ME-SC-FV. Due to the high-order of ME-SG-DGSEM many small-scale flow features in std can be observed, although the spatial mesh is much coarser than for ME-SC-FV. From Fig. 4c and d we can deduce that the uncertain Mach number has a strong influence on the position and intensity of the shock upstream of the step. It also influences the flow structure after the step, but not as much as in front of the step.

Performance of the optimizations discussed in Section 4.1 was also investigated. Compared to the baseline version using index vectors, the optimized loop layout yielded a speed-up of 1.07, which is in line with the value in Fig. 1b for $M = 1$ and $N_{\text{stoch}} = 3$. Manual loop unrolling yielded a speed-up of 1.26, which is lower than the speed-up encountered in Fig. 1b. This might be due to other factors such as finite-volume sub-cells or parallel communication dominating computation time. Strong scaling for this example was investigated on the HPE Apollo system Hawk at the High Performance Computing Center Stuttgart (HLRS) using AMD EPYC 7742 CPUs. The system is different from the one mentioned above, as the original system was not available anymore at the time of this investigation. Only 99 of the available 128 CPUs were used on

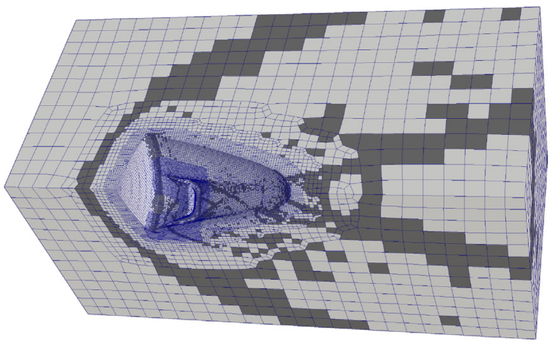
**Fig. 5.** Spacecraft: Mesh and DG/FV distribution. Example 5.4.

each node to allow for an even distribution of DG elements across cores, which prevents idle times. Compared to a reference computation on one node, parallel efficiency of a computation on 256 nodes (25344 cores) was 94%.

### 5.4. 3D flow around a spacecraft

In order to demonstrate the methods capability to simulate completely three-dimensional problems, the inviscid flow around a space capsule with an uncertain Mach number $Ma \sim \mathcal{U}(2.8, 3.2)$ is considered.

The underlying CAD model geometry is based on a simplified model of the SpaceX Dragon spacecraft. A quarter model with symmetry boundary conditions was used as the geometry of the spacecraft is in good approximation symmetrical along two orthogonal planes. The CAD model was created in SolidWorks®, transformed to an STL surface mesh using ANSA®and a mesh was created from this using NUMECA HEXPRESS®. The mesh, shown in Fig. 5, has 71,819 hexahedral elements and uses non-conforming mortar interfaces for refinement near the spacecraft surface.

In order to save computational cost, the simulation was split into three stages. Each stage was run until the solution converged to a steady state. In the first stage, a deterministic simulation with a finite volume space discretization (with $2 \times 2 \times 2$ FV cells per DG mesh cell) was run with $Ma = 3$, which corresponds to the farfield condition $(\rho, v_1, v_2, v_3, p)^T = (1, 0, 1, 0, 0.079367)^T$. In the second stage, the aforementioned uncertainty was gradually introduced by altering $v_2$ in the inflow boundary condition at constant speed of sound, which corresponds to $v_2 \sim \mathcal{U}(0.0741, 0.08466)$. A stochastic polynomial degree of $N_{\text{stoch}} = 3$ was used. ME-SG was not used for this example in order to reduce computational cost. In the third stage, space discretization was changed to the mixed DG/FV scheme described above, and the spatial polynomial degree was increased to 3. Simulations were carried out on 4800 cores at the Cray XC40 system Hazel Hen at the High Performance Computing Center Stuttgart (HLRS). The wall time totalled to approximately three hours for all three stages.

Figure 6 shows mean and standard deviation of the field solution. Mach number is plotted on the spacecraft surface while pressure is depicted in a slice through the flow field. The solution is reflected along the symmetry planes for visualization. In front of the capsule, a bow shock can be observed. Another set of shocks reaches from the rear of the capsule to the far field. Furthermore, shock patterns form around the thruster outlets and along the thruster edges. Standard deviation is highest were shock positions vary with changing inflow Mach number. At the end of the computation, around 20% of the elements were converted to FV. The results show that high order SG simulations are feasible for complex three-dimensional geometries, in the presence of strong shocks.

In this example, the optimized loop layout led to a speed-up of 1.14 as compared to the baseline index vector implementation. Manual loop unrolling provided a speed-up of 1.67, which even exceeds the values presented in Fig. 1b for $M = 1$ and $N_{\text{stoch}} = 3$. Strong scaling for this example was also investigated on the Hawk system at HLRS. 71 CPUs were used on each node. Parallel efficiency of a computation on 256 nodes (18176 cores) was 101% (again compared to a reference computation on one node), which means that caching effects more than offset the communication times, resulting in a slightly super-linear speed-up.

## 6. Conclusion

We have presented a high-order Stochastic Galerkin code based on a Discontinuous Galerkin Spectral element spatial discretization. It is able solve the two- and three-dimensional compressible Navier-Stokes Equations and Euler equations with an arbitrary number of stochastic dimensions. For the discretization in the stochastic space we use a pseudo-spectral approach and incorporated a multi-element ansatz in combination with a hyperbolicity preserving limiter to deal with the possible loss of hyperbolicity of

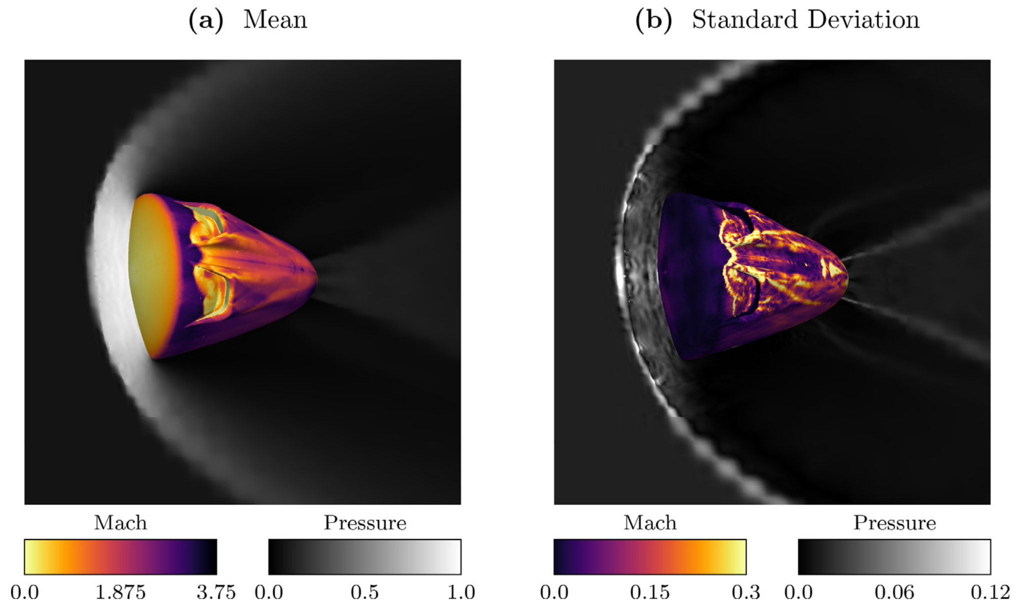

**Fig. 6.** Spacecraft: mean and standard deviation; Mach number on spacecraft surface, pressure in slice through flow field. Example 5.4.

the Euler equations. The Discontinuous Galerkin Spectral Element Method is used for spatial discretization and the semi-discrete scheme is advanced in time with an explicit time-stepping scheme. Our method can handle curved, unstructured hexahedral elements and it uses a lifting procedure for the treatment of parabolic terms and features shock capturing with a finite volume subcell method. Performance of the most computationally expensive routine was optimized and a post-processing environment was developed.

Our numerical results show the expected convergence order for refinement, both in the physical and the stochastic space. The described performance optimizations yield an overall speed-up by a factor of up to two as compared to standard approaches, depending on the setup. Investigated validations and numerical examples include the flow around a NACA0012 airfoil with uncertain Reynolds number and angle of attack, supersonic flow over a forward facing step with an uncertain inflow Mach number as well as flow around a three-dimensional spacecraft geometry with uncertain inflow Mach number.

Though not discussed in this paper, we want to mention an inherent limitation of the SG method regarding time-dependent problems: The so-called long-term degeneration effect, cf. [14], causes some SG simulations to deteriorate after a certain simulation time. This renders, for example, scale-resolving simulations such as large eddy simulations or direct numerical simulations with the SG method unfeasible.

In order to simulate turbulent flows, one of the obvious next steps is to incorporate a Reynolds-averaged Navier Stokes (RANS) turbulence model into the method. Efficiency can be further increased, for example by introducing adaptivity regarding discretization of the stochastic domain. These steps will advance our code and the underlying method even further on its path towards industrial applicability.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors would like to thank the Baden-Württemberg Stiftung for funding via the project “SEAL”. The authors would also like to thank the High-Performance Computing Center Stuttgart (HLRS) for the provided computing resources through the project “SEAL”.

## Appendix A. Optimized computation of SG products with one random variable

In order to compute the SG discretized product of $g$ and $h$, a double sum is to be calculated for each of the coefficients of $gh$, i.e.

$$(\widehat{gh})_k = \sum_{i,j} \widehat{g}_i \widehat{h}_j C_{ijk} \quad \text{with} \quad C_{ijk} = \langle \Psi_i \Psi_j, \Psi_k \rangle. \tag{A.1}$$

The naive implementation of this requires three loops as shown in Algorithm 1.

The tensor $C$ is however sparse. An obvious choice for optimization is to build a one-dimensional array with all non-zero entries of $C$ and an according index array for the indices $i$, $j$ and $k$ belonging to each entry. This implementation is relatively slow. A different approach is therefore laid out below. The resulting pseudo-code is given in Algorithm 3. It is derived as follows:

A set of for-loops can be derived which directly address only the indices $(i, j, k)$ belonging to the non-zero entries of $C$. These non-zero entries are stored in a one-dimensional array `C1`. After the use of an entry of `C1`, a scalar index `s` for `C1` is manually incremented to jump to the next entry of `C1(s)`. In an initialization phase, the values for $C$ are first stored in the original three-dimensional array and are subsequently transferred to a one-dimensional array. To this end, the same for-loops as for the later computation of the product are used, but with operations of type `C1(s)=C(i,j,k)`.

We use Algorithm 1 as a starting point. Firstly, the symmetry of the tensor can be exploited to save storage space and bandwidth for the tensor entries. This can be achieved by defining $i \geq j \geq k$. The operations belonging to different permutations of an index set can be carried out using the same tensor entry and can thus be grouped together. Four cases concerning the indices can be distinguished:

A) $i > j > k$; index set $(ijk)$; six permutations
B) $i = j > k$; index set $(iik)$; three permutations
C) $i > j = k$; index set $(ijj)$; three permutations
D) $i = j = k$; index set $(iii)$; one permutation

The cases are now treated separately. Case A) requires three nested for-loops, cases B) and C) require two for-loops and case D) one for-loop. The intermediate semi-optimized pseudo-code is shown in Algorithm 2. Note that in many cases, the upper limit of a for loop index is indeed lower than the lower limit. In these cases, the loop is not accessed at all.

Additionally, the sparsity of the tensor can be exploited. Orthogonality of the basis polynomials leads to $C_{ijk} = 0$ for $i > j + k$ [42]. This yields the following changes to the index limits for cases A) and C):

**Algorithm 1** SG product; naive implementation.

```
# Input: M, C, g, h
# Output: gh

gh(:) = 0.
for k = 0:M
  for j = 0:M
    for i = 0:M
      gh(k) = gh(k) + g(i)*h(j)*C(i,j,k)
    end for
  end for
end for
```

**Algorithm 2** SG product; auxiliary, intermediate version (semi-optimized).

```
# Input: M, C1, g, h
# Output: gh

# pre-compute A
gh(:) = 0.
s = 1

# case A (ijk)
for i = 0:M
  for j = 0:i-1
    for k = 0:j-1
      gh(k) = gh(k) + g(i)*h(j)*C1(s)
      gh(k) = gh(k) + g(j)*h(i)*C1(s)
      gh(j) = gh(j) + g(i)*h(k)*C1(s)
      gh(j) = gh(j) + g(k)*h(i)*C1(s)
      gh(i) = gh(i) + g(j)*h(k)*C1(s)
      gh(i) = gh(i) + g(k)*h(j)*C1(s)
      s = s + 1
    end for
  end for
end for

# case B (iik)
for i = 0:M
  for k = 0:i-1
    gh(i) = gh(i) + g(i)*h(k)*C1(s)
    gh(i) = gh(i) + g(k)*h(i)*C1(s)
    gh(k) = gh(k) + g(i)*h(i)*C1(s)
    s = s + 1
  end for
end for

# case C (ijj)
for i = 0:M
  for j = 0:i-1
    gh(i) = gh(i) + g(j)*h(j)*C1(s)
    gh(j) = gh(j) + g(i)*h(j)*C1(s)
    gh(j) = gh(j) + g(j)*h(i)*C1(s)
    s = s + 1
  end for
end for

# case D (iii)
for i = 0:M
  gh(i) = gh(i) + g(i)*h(i)*C1(s)
  s = s + 1
end for
```

A) $k \geq i - j$
C) $k \geq i/2$.

These changed index limits are marked green in Algorithm 3. The index limits for case B) remain unchanged, as $i \leq i + k$ always.

For even probability density functions, basis polynomials with even (odd) indices are also purely even (odd). Consequently, an odd sum $i + j + k$ yields an odd polynomial for the product $\Psi_i \Psi_j \Psi_k$ and thus also $C_{ijk} = 0$ [42]. This leads to the following indices

A) The first index of $k$ is $i - j$. In this case, $i + j + k = i + j + (i - j) = 2i$, which is even. Setting the step size of k to 2 thus only loops over even index sums.
B) The sum $i + i + k$ is even iff $k$ is even, which is achieved with a step size of two in the inner loop.

**Algorithm 3** SG product; optimized version for one random variable.

```
# Input: M, C1, g, h
# Output: gh

# pre-compute A
for i = 0:M
  for j = 0:i-1
    A(j,i) = g(i)*h(j) + g(j)*h(i)
  end for
  A(i,i) = g(i)*h(i)
end for

gh(:) = 0.
s = 1
for i = 0:M
  for j = 0:i-1
    for k = i-j:j-1:2
      # case A (ijk)
      gh(k) = gh(k) + A(j,i)*C1(s)
      gh(j) = gh(j) + A(k,i)*C1(s)
      gh(i) = gh(i) + A(k,j)*C1(s)
      s = s + 1
    end for
  end for
  for k = 0:i-1:2
    # case B (iik)
    gh(i) = gh(i) + A(k,i)*C1(s)
    gh(k) = gh(k) + A(i,i)*C1(s)
    s = s + 1
  end for
end for
for i = 0:M:2
  for j = i/2:i-1
    # case C (ijj)
    gh(i) = gh(i) + A(j,j)*C1(s)
    gh(j) = gh(j) + A(j,i)*C1(s)
    s = s + 1
  end for
  # case D (iii)
  gh(i) = gh(i) + A(i,i)*C1(s)
  s = s + 1
end for
```

C) The sum $i+j+j$ is even iff $i$ is even, which is achieved with a step size of two in the outer loop.
D) The sum $i+i+i$ is even iff $i$ is even, which is achieved with a step size of two in the single loop.

It becomes obvious that we can join the outer loops of cases A) and B) as well as C) and D). This leads to the nested loop layout in Algorithm 3. The changed index step sizes are marked red.

In a last optimization step, the operations belonging to two respective permutations of different indices of $g$ and $h$ can be joined. Since $g_i h_j C_{ijk} + g_j h_i C_{ijk} = (g_i h_j + g_j h_i) C_{ijk}$, we pre-compute the upper diagonal of a matrix $A$ with $A_{ij} = g_i h + g_j h_i$, which reduces the two operations

```
gh(k) = gh(k) + g(i)*h(j)*C1(s)
gh(k) = gh(k) + g(j)*h(i)*C1(s)
```

to the single operation

```
gh(k) = gh(k) + A(j,i)*C1(s)
```

By this, the number of operations in each innermost iteration is reduced to 3 for case A) and to 2 for cases B) and C). We also use the main diagonal of this matrix for equal indices $A_{ii} = g_i h_i$.

## Appendix B. SG product source code generated for a prescribed stochastic dimensionality and polynomial degree

**Algorithm 4** SG product; script-generated source code for $M = 2$ and $N_{\text{stoch}} = 2$.

```
# Input: M, C1, g, h
# Output: gh

A(1)  =  g(1) * h(0)  +  g(0) * h(1)
A(2)  =  g(1) * h(1)
A(3)  =  g(2) * h(1)  +  g(1) * h(2)
A(4)  =  g(2) * h(0)  +  g(0) * h(2)
A(5)  =  g(2) * h(2)
A(6)  =  g(3) * h(0)  +  g(0) * h(3)
A(7)  =  g(3) * h(3)
A(8)  =  g(4) * h(3)  +  g(3) * h(4)
A(9)  =  g(4) * h(1)  +  g(1) * h(4)
A(10) =  g(3) * h(1)  +  g(1) * h(3)
A(11) =  g(4) * h(0)  +  g(0) * h(4)
A(12) =  g(4) * h(4)
A(13) =  g(4) * h(2)  +  g(2) * h(4)
A(14) =  g(5) * h(3)  +  g(3) * h(5)
A(15) =  g(5) * h(4)  +  g(4) * h(5)
A(16) =  g(5) * h(0)  +  g(0) * h(5)
A(17) =  g(5) * h(5)

gh(0)  =  g(0) * h(0)
gh(1)  =                       A(1)
gh(0)  =  gh(0)  +             A(2)
gh(1)  =  gh(1)  +  C1(1)  *   A(3)
gh(2)  =            C1(1)  *   A(2)
gh(2)  =  gh(2)  +             A(4)
gh(0)  =  gh(0)  +             A(5)
gh(2)  =  gh(2)  +  C1(2)  *   A(5)
gh(3)  =                       A(6)
gh(0)  =  gh(0)  +             A(7)
gh(1)  =  gh(1)  +             A(8)
gh(3)  =  gh(3)  +             A(9)
gh(4)  =                       A(10)
gh(4)  =  gh(4)  +             A(11)
gh(0)  =  gh(0)  +             A(12)
gh(4)  =  gh(4)  +  C1(3)  *   A(13)
gh(2)  =  gh(2)  +  C1(3)  *   A(12)
gh(3)  =  gh(3)  +  C1(4)  *   A(14)
gh(5)  =            C1(4)  *   A(7)
gh(4)  =  gh(4)  +  C1(5)  *   A(15)
gh(5)  =  gh(5)  +  C1(5)  *   A(12)
gh(5)  =  gh(5)  +             A(16)
gh(0)  =  gh(0)  +             A(17)
gh(5)  =  gh(5)  +  C1(6)  *   A(17)
```

## References

[1] Xiu D, Karniadakis GE. The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J Sci Comput 2002;24(2):619–44.
[2] Wiener N. The homogeneous chaos. Amer J Math 1938;60(4):897–936.
[3] Le Maître OP, Reagan MT, Najm HN, Ghanem RG, Knio OM. A stochastic projection method for fluid flow. II. Random process. J Comput Phys 2002;181(1):9–44. doi:10.1006/jcph.2002.7104.
[4] Xiu D. Stochastic collocation methods: a survey. In: Handbook of uncertainty quantification. Vol. 1, 2, 3. Springer, Cham; 2017. p. 699–716.
[5] Kuhn T, Dürrwächter J, Meyer F, Beck A, Rohde C, Munz C-D. Uncertainty quantification for direct aeroacoustic simulations of cavity flows. J Theor Comput Acoust 2019;27(1):1850044,20. doi:10.1142/S2591728518500445.
[6] Le Maître OP, Knio OM. Spectral methods for uncertainty quantification. Scientific Computation. Springer, New York; 2010. doi:10.1007/978-90-481-3520-2. ISBN 978-90-481-3519-6. With applications to computational fluid dynamics
[7] Pettersson MP, Iaccarino G, Nordström J. Polynomial chaos methods for hyperbolic partial differential equations. Mathematical Engineering. Springer, Cham; 2015. doi:10.1007/978-3-319-10714-1. ISBN 978-3-319-10713-4; 978-3-319-10714-1. Numerical techniques for fluid dynamics problems in the presence of uncertainties
[8] Pettersson P, Nordström J, Doostan A. A well-posed and stable stochastic Galerkin formulation of the incompressible Navier-Stokes equations with random data. J Comput Phys 2016;306:92–116. doi:10.1016/j.jcp.2015.11.027.
[9] Dürrwächter J, Kuhn T, Meyer F, Schlachter L, Schneider F. A hyperbolicity-preserving discontinuous stochastic Galerkin scheme for uncertain hyperbolic systems of equations. J Comput Appl Math 2020;370:112602. doi:10.1016/j.cam.2019.112602.
[10] Tryoen J, Le Maître OP, Ndjinga M, Ern A. Intrusive Galerkin methods with upwinding for uncertain nonlinear hyperbolic systems. J Comput Phys 2010;229(18):6485–511. doi:10.1016/j.jcp.2010.05.007.
[11] Poëtte G, Després B, Lucor D. Uncertainty quantification for systems of conservation laws. J Comput Phys 2009;228(7):2443–67.
[12] Kusch J, McClarren RG, Frank M. Filtered stochastic Galerkin methods for hyperbolic equations. J Comput Phys 2020;403:109073. doi:10.1016/j.jcp.2019.109073.
[13] Wu K, Tang H, Xiu D. A stochastic Galerkin method for first-order quasilinear hyperbolic systems with uncertainty. J Comput Phys 2017;345:224–44. doi:10.1016/j.jcp.2017.05.027.
[14] Wan X, Karniadakis GE. Long-term behavior of polynomial chaos in stochastic flow simulations. Comput Methods Appl Mech Engrg 2006;195(41–43):5582–96. doi:10.1016/j.cma.2005.10.016.

[15] Mathelin L, Hussaini MY, Zang TA. Stochastic approaches to uncertainty quantification in CFD simulations. Numer Algorithms 2005;38(1–3):209–36. doi:10.1007/s11075-004-2866-z.
[16] Kantarakias KD, Chatzimanolakis M, Asouti VG, Giannakoglou KC. On the development of the 3D Euler equations using intrusive PCE for uncertainty quantification. In: Proceedings of UNCECOMP - 2nd ECCOMAS thematic conference on uncertainty quantification in computational sciences and engineering; 2017.
[17] Le Maître O, Reagan MT, Debusschere B, Najm HN, Ghanem RG, Knio OM. Natural convection in a closed cavity under stochastic non-boussinesq conditions. SIAM J Sci Comput 2006:375–94. doi:10.1137/S106482750342285.
[18] Reagan MT, Najm HN, Pébay PP, Knio OM, Ghanem RG. Quantifying uncertainty in chemical systems modeling. Int J Chem Kinet 2005;37(6):368–82. doi:10.1002/kin.20081.
[19] Krais N, Beck A, Bolemann T, Frank H, Flad D, Gassner G, Hindenlang F, Hoffmann M, Kuhn T, Sonntag M, Munz C-D. FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws. Comput Math Appl 2021;81:186–219. doi:10.1016/j.camwa.2020.05.004.
[20] Hindenlang F, Gassner GJ, Altmann C, Beck A, Staudenmaier M, Munz C-D. Explicit discontinuous Galerkin methods for unsteady problems. Computers & Fluids 2012;61:86–93.
[21] Carpenter M, Kennedy C. Fourth-order 2N-storage Runge-Kutta schemes. NASA TM 109112 1994:1–25.
[22] Beck AD, Bolemann T, Flad D, Frank H, Gassner GJ, Hindenlang F, Munz C-D. High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. Internat J Numer Methods Fluids 2014;76(8):522–48. doi:10.1002/fld.3943.
[23] Chalmers N, Agbaglah G, Chrust M, Mavriplis C. A parallel hp-adaptive high order discontinuous Galerkin method for the incompressible Navier-Stokes equations. J Comput Phys X 2019;2:100023. doi:10.1016/j.jcpx.2019.100023.
[24] Ghanem RG, Spanos PD. Stochastic finite elements: a spectral approach. New York: Springer-Verlag; 1991. doi:101007/978-1-4612-3094-6. ISBN 0-387-97456-3
[25] Xiu D, Hesthaven JS. High-order collocation methods for differential equations with random inputs. SIAM J Sci Comput 2005;27(3):1118–39.
[26] Debusschere BJ, Najm HN, Matta A, Knio OM, Ghanem RG, Le Maître OP. Protein labeling reactions in electrochemical microchannel flow: numerical simulation and uncertainty propagation. Phys Fluids 2003;15(8):2238–50. doi:10.1063/1.1582857.
[27] Debusschere BJ, Najm HN, Pébay PP, Knio OM, Ghanem RG, Le Maître OP. Numerical challenges in the use of polynomial chaos representations for stochastic processes. SIAM J Sci Comput 2006:698719.
[28] Lacor C, Dinescu C, Hirsch C, Smirnov S. Implementation of intrusive polynomial chaos in CFD codes and application to 3D Navier-Stokes. In: Uncertainty quantification in computational fluid dynamics. In: Lect. Notes Comput. Sci. Eng., vol. 92. Springer, Heidelberg; 2013. p. 193–223. doi:10.1007/978-3-319-00885-1_5.
[29] Ghanem R, Asce M, Red-horse JR, Sarkar A, Asce AM. Modal properties of a space-frame with localized system uncertainties. In: 8th ASCE Specialty conference of probabilistic mechanics and structural reliability, ASCE; 2000.
[30] Wan X, Karniadakis GE. An adaptive multi-element generalized polynomial chaos method for stochastic differential equations. J Comput Phys 2005;209(2):617–42.
[31] Wan X, Karniadakis GE. Multi-element generalized polynomial chaos for arbitrary probability measures. SIAM J Sci Comput 2006;28(3):901–28.
[32] Després B, Poëtte G, Lucor D. Robust uncertainty propagation in systems of conservation laws with the entropy closure method. In: Uncertainty quantification in computational fluid dynamics. In: Lect. Notes Comput. Sci. Eng., vol. 92. Springer, Heidelberg; 2013. p. 105–49. doi:10.1007/978-3-319-00885-1_3.
[33] Schlachter L, Schneider F. A hyperbolicity-preserving stochastic Galerkin approximation for uncertain hyperbolic systems of equations. J Comput Phys 2018;375:80–98.
[34] Kopriva DA. Implementing spectral methods for partial differential equations. Scientific Computation. Springer, Berlin; 2009. doi:10.1007/978-90-481-2261-5. ISBN 978-90-481-2260-8. Algorithms for scientists and engineers
[35] Hesthaven JS, Warburton T. Nodal discontinuous Galerkin methods. Texts in Applied Mathematics, vol. 54. Springer, New York; 2008. doi:10.1007/978-0-387-72067-8. ISBN 978-0-387-72065-4. Algorithms, analysis, and applications
[36] Bassi F, Rebay S. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J Comput Phys 1997;131(2):267–79.
[37] Sonday BE, Berry RD, Najm HN, Debusschere BJ. Eigenvalues of the Jacobian of a Galerkin-projected uncertain ODE system. SIAM J Sci Comput 2011:1212–33. doi:10.1137/100785922.
[38] Sonntag M, Munz C-D. Efficient parallelization of a shock capturing for discontinuous Galerkin methods using finite volume sub-cells. J Sci Comput 2017;70(3):1262–89. doi:10.1007/s10915-016-0287-5.
[39] Persson P-O, Peraire J. Sub-cell shock capturing for discontinuous Galerkin methods. 44th AIAA Aerospace sciences meeting and exhibit; 2006. doi:102514/62006-112.
[40] Jameson A, Schmidt W, Turkel E. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In: 14th Fluid and plasma dynamics conference; 1981. https://doi.org/10.2514/6.1981-1259.
[41] Einfeldt B, Munz C-D, Roe PL, Sjögreen B. On Godunov-type methods near low densities. J Comput Phys 1991;92(2):273–95. doi:10.1016/0021-9991(91)90211-3.
[42] Pettersson P, Iaccarino G, Nordström J. A stochastic Galerkin method for the Euler equations with Roe variable transformation. J Comput Phys 2014:481–500. doi:10.1016/j.jcp.2013.10.011.
[43] Woodward P, Colella P. The numerical simulation of two-dimensional fluid flow with strong shocks. J Comput Phys 1984;54(1):115–73. doi:10.1016/0021-9991(84)90142-6.