Intrusive acceleration strategies for Uncertainty Quantification for hyperbolic systems of conservation laws

Jonas Kusch[a], Jannick Wolters[b], Martin Frank[c]

[a] *Karlsruhe Institute of Technology, Karlsruhe, jonas.kusch@kit.edu*

[b] *Karlsruhe Institute of Technology, Karlsruhe, jannick.wolters@kit.edu*

[c] *Karlsruhe Institute of Technology, Karlsruhe, martin.frank@kit.edu*

Abstract

Methods for quantifying the effects of uncertainties in hyperbolic problems can be divided into intrusive and non-intrusive techniques. Non-intrusive methods allow the usage of a given deterministic solver in a black-box manner, while being embarrassingly parallel. However, avoiding intrusive modifications of a given solver takes away the ability to use several inherently intrusive numerical acceleration tools. Moreover, intrusive methods are expected to reach a given accuracy with a smaller number of unknowns compared to non-intrusive techniques. This effect is amplified in settings with high dimensional uncertainty. A downside of intrusive methods is however the need to guarantee hyperbolicity of the resulting moment system. In contrast to stochastic-Galerkin (SG), the Intrusive Polynomial Moment (IPM) method is able to maintain hyperbolicity at the cost of solving an optimization problem in every spatial cell and every time step.
In this work, we propose several acceleration techniques for intrusive methods and study their advantages and shortcomings compared to the non-intrusive stochastic-Collocation method. When solving steady problems with IPM, the numerical costs arising from repeatedly solving the IPM optimization problem can be reduced by using concepts from PDE-constrained optimization. Integrating the iteration from the numerical treatment of the optimization problem into the moment update reduces numerical costs, while preserving local convergence. Additionally, we propose an adaptive implementation and efficient parallelization strategy of the IPM method. The effectiveness of the proposed adaptations is demonstrated for multi-dimensional uncertainties in fluid dynamics applications, resulting in the observation of requiring a smaller number of unknowns to achieve a given accuracy when using intrusive methods. Furthermore, using the proposed acceleration techniques, our implementation reaches a given accuracy faster than stochastic-Collocation.

Keywords: uncertainty quantification, conservation laws, hyperbolic, intrusive, stochastic-Galerkin, Collocation, Intrusive Polynomial Moment Method

1. Introduction

Hyperbolic equations play an important role in various research areas such as fluid dynamics or plasma physics. Efficient numerical methods combined with robust implementations are widely available for these problems, however they do not account for uncertainties which can arise in measurement data or modeling assumptions. Including the effects of uncertainties in differential equations has become an important topic in the last decades.

One strategy to represent the solution's dependence on uncertainties is to use a polynomial chaos (PC) expansion [1, 2], i.e. the uncertainty space is spanned by polynomial basis functions. The remaining task is then to determine adequate expansion coefficients, often called moments or PC coefficients. Numerical methods for approximating these coefficients can be divided into intrusive and non-intrusive techniques. A popular non-intrusive method is the stochastic-Collocation (SC) method, see e.g. [3, 4, 5], which computes

the moments with the help of a numerical quadrature rule. Commonly, SC uses sparse grids, since they posses a reduced number of collocation points for multi-dimensional problems. Since the solution at a fixed quadrature point can be computed by a standard deterministic solver, the SC method does not require a significant implementation effort. Furthermore, SC is embarrassingly parallel, since the required computations decouple and the workload can easily be distributed across several processors.

The main idea of intrusive methods is to derive a system of equations describing the time evolution of the moments which can then be solved with a deterministic numerical scheme. A popular approach to describe the moment system is the stochastic-Galerkin (SG) method [6], which chooses a polynomial basis ansatz of the solution and performs a Galerkin projection to derive a closed system of equations. One significant drawback of SG is, that its moment system is not necessarily hyperbolic [7]. A generalization of stochastic-Galerkin, which ensures hyperbolicity is the Intrusive Polynomial Moment (IPM) method [7]. Instead of performing the PC expansion on the solution, the IPM method represents the entropy variables with polynomials. Besides yielding a hyperbolic moment system, the IPM method has several advantages: Choosing a quadratic entropy yields the stochastic-Galerkin moment system, i.e. IPM generalizes different intrusive methods. Furthermore, at least for scalar problems, IPM is significantly less oscillatory compared to SG [8]. Also, as discussed in [7], when choosing a physically correct entropy of the deterministic problem, the IPM solution dissipates the expectation value of the entropy, i.e. the IPM method yields a physically correct entropy solution. Unfortunately, the desirable properties of IPM come along with significantly increased numerical costs, since IPM requires the repeated computation of the entropic expansion coefficients from the moment vector, which involves solving a convex optimization problem. However, IPM and minimal entropy methods in general are well suited for modern HPC architectures [9].

When studying hyperbolic equations, the moment approximations of various methods such as Stochastic Galerkin [10, 11], IPM [11, 12] and stochastic-Collocation [13, 14] tend to show incorrect discontinuities in certain regions of the physical space. These non-physical structures dissolve when the number of basis functions is increased [15, 16] or when artificial diffusion is added through either the spatial numerical method [16] or by filters [11]. Also, a multi-element approach which divides the uncertain domain into cells and uses piece-wise polynomial basis functions to represent the solution has proven to mitigate non-physical discontinuities [17, 18]. Non-intrusive Monte-Carlo methods [19, 20, 21], which randomly sample input uncertainties to compute quantities of interest are robust, but suffer from a slow rate of convergence while again lacking the ability to use adaptivity to their full extend. Discontinuous structures commonly arise on a small portion of the space-time domain. Therefore, intrusive methods seem to be an adequate choice since they are well suited for adaptive strategies. By locally increasing the polynomial order [22, 23, 24] or adding artificial viscosity [11] at certain spatial positions and time steps in which complex structures such as discontinuities occur, a given accuracy can be reached with significantly reduced numerical costs. In addition to that, the number of moments needed to obtain a certain order with intrusive methods is smaller than the number of quadrature points for SC. An additional downside of collocation methods are aliasing effects, which stem from the inexact approximation of integrals. Consequently, collocation methods typically require a higher number of unknowns than intrusive methods to reach a given accuracy [25, 26]. Therefore, one aim should be to accelerate intrusive methods, since they can potentially outperform non-intrusive methods in complex and high-dimensional settings.

In this paper, we propose acceleration techniques for intrusive methods and compare them against stochastic-Collocation. For steady and unsteady problems, we use adaptivity, for which intrusive methods provide a convenient framework:

- Since complex structures in the uncertain domain tend to arise in small portions of the spatial mesh, our aim is to locally increase the accuracy of the stochastic discretization in region that show a complex structure in the random domain, while choosing a low order method in the remainder. Such an adaptive treatment cannot be realized with non–intrusive methods, since one needs to break up the black-box approach. To guarantee an efficient implementation, we propose an adaptive discretization strategy for IPM.

A steady problem provides different opportunities to take advantage of features of intrusive methods:

- When using adaptivity, one can perform a large number of iterations to the steady state solution on a low number of moments and increase the maximal truncation order when the distance to the steady state has reached a specified barrier. Consequently, a large number of iterations will be performed by a cheap, low order method, i.e. we can reduce numerical costs.
- Perform an inexact map from the moments to the entropic expansion coefficients for IPM: Since the moments during the iteration process are inaccurate, i.e. they are not the correct steady state solution, we propose to not fully converge the dual iteration, which solves the IPM optimization problem. Consequently, the entropic expansion coefficients and the moments are converged simultaneously to their steady state, which is similar to the idea of One-Shot optimization in shape optimization [27].

The effectiveness of these acceleration ideas are tested by comparing results with stochastic-Collocation for the uncertain NACA test case as well as a bent shock tube problem. Our numerical studies show the following main results:

- In our test cases, the need to solve an optimization problem when using the IPM method leads to a significantly higher run time than SC and SG. However when using the discussed acceleration techniques, IPM requires the shortest time to reach a given accuracy.
- Comparing SG with IPM, one observes that for the same number of unknowns, SG yields more accurate expectation values, whereas IPM shows improved variance approximations.
- By studying aliasing effects, we show that SC requires a higher number of unknowns than intrusive methods (even for a one-dimensional uncertainty) to reach the same accuracy level.
- Using sparse grids for the IPM discretization when the space of uncertainty is multi-dimensional, the number of quadrature points needed to guarantee sufficient regularity of the Hessian matrix is significantly increased.

The IPM and SG calculations use a semi-intrusive numerical method, meaning that the discretization allows recycling a given deterministic code to generate the IPM solver. While facilitating the task of implementing general intrusive methods, this framework reduces the number of operations required to compute numerical fluxes. Also, it provides the ability to base the intrusive method on the same deterministic solver as used in the implementation of a black-box fashion stochastic-Collocation code, which allows for, what we believe, a fair comparison between intrusive and non-intrusive methods. The code is publicly available to allow reproducibility [28].

The paper is structured as follows: After the introduction, we present the discussed methods in more detail in section 2. The numerical discretization as well as the implementation and structure of the semi-intrusive method is introduced in section 3. In section 4, we discuss the idea of not converging the dual iteration. Section 5 extends the presented numerical framework to an algorithm making use of adaptivity. Implementation and parallelization details are given in section 6. A comparison of results computed with the presented methods is then given in section 7, followed by a summary and outlook in section 8.

2. Background

In the following, we briefly introduce the notation and methods used in this work. A general hyperbolic set of equations with random initial data can be written as

$$\partial_t \boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi}) + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi})) = \boldsymbol{0} \quad \text{in } D, \tag{1a}$$

$$\boldsymbol{u}(t=0,\boldsymbol{x},\boldsymbol{\xi}) = \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x},\boldsymbol{\xi}), \tag{1b}$$

where the solution $\boldsymbol{u} \in \mathbb{R}^m$ depends on time $t \in \mathbb{R}^+$, spatial position $\boldsymbol{x} \in D \subseteq \mathbb{R}^d$ as well as a vector of random variables $\boldsymbol{\xi} \in \Theta \subseteq \mathbb{R}^p$ with given probability density functions $f_{\Xi,i}(\xi_i)$ for $i = 1, \cdots, p$. Hence, the

probability density function of $\boldsymbol{\xi}$ is $f_\Xi(\boldsymbol{\xi}) := \prod_{i=1}^p f_{\Xi,i}(\xi_i)$. The physical flux is given by $\boldsymbol{f} : \mathbb{R}^m \to \mathbb{R}^{d\times m}$. To simplify notation, we assume that only the initial condition is random, i.e. $\boldsymbol{\xi}$ enters through the definition of $\boldsymbol{u}_{IC}$. Equations (1) are usually supplemented with boundary conditions, which we will specify later for the individual problems.

Due to the randomness of the solution, one is interested in determining the expectation value or the variance, i.e.

$$\mathrm{E}[\boldsymbol{u}] = \langle \boldsymbol{u} \rangle, \qquad \mathrm{Var}[\boldsymbol{u}] = \langle (\boldsymbol{u} - \mathrm{E}[\boldsymbol{u}])^2 \rangle,$$

where we use the bracket operator $\langle \cdot \rangle := \int_\Theta \cdot f_\Xi(\boldsymbol{\xi}) d\xi_1 \cdots d\xi_p$. To approximate quantities of interest (such as expectation value, variance or higher order moments), the solution is spanned with a set of polynomial basis functions $\varphi_i : \Theta \to \mathbb{R}$ such that for the multi-index $i = (i_1, \cdots, i_p)$ we have $|i| \leq M$. Note that this yields

$$N := \binom{M+p}{p} \tag{2}$$

basis functions when defining $|i| := \sum_{n=1}^p |i_n|$. Commonly, these functions are chosen to be orthonormal polynomials [1] with respect to the probability function, i.e. $\langle \varphi_i \varphi_j \rangle = \prod_{n=1}^p \delta_{i_n j_n}$. The generalized polynomial chaos (gPC) expansion [2] approximates the solution by

$$\mathcal{U}(\hat{\boldsymbol{u}}; \boldsymbol{\xi}) := \sum_{|i|\leq M} \hat{\boldsymbol{u}}_i \varphi_i(\boldsymbol{\xi}) = \hat{\boldsymbol{u}}^T \boldsymbol{\varphi}(\boldsymbol{\xi}), \tag{3}$$

where the deterministic expansion coefficients $\hat{\boldsymbol{u}}_i \in \mathbb{R}^m$ are called moments. To allow a more compact notation, we collect the N moments for which $|i| \leq M$ holds in the moment matrix $\hat{\boldsymbol{u}} := (\hat{\boldsymbol{u}}_i)_{|i|\leq M} \in \mathbb{R}^{N\times m}$ and the corresponding basis functions in $\boldsymbol{\varphi} := (\varphi_i)_{|i|\leq M} \in \mathbb{R}^N$. In the following, the dependency of $\mathcal{U}$ on $\boldsymbol{\xi}$ will occasionally be omitted for sake of readability. The solution ansatz (3) is L^2-optimal, if the moments are chosen to be the Fourier coefficients $\hat{\boldsymbol{u}}_i \equiv \langle \boldsymbol{u}\varphi_i \rangle \in \mathbb{R}^m$. One can also use the ansatz (3) to compute the quantities of interest as

$$\mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})] = \hat{\boldsymbol{u}}_0, \quad \mathrm{Var}[\mathcal{U}(\hat{\boldsymbol{u}})] = \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})^2] - \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}})]^2 = \left(\sum_{i=1}^N \hat{u}_{\ell i}^2 \right)_{\ell=1,\cdots,m}.$$

The core idea of the stochastic-Collocation method is to compute the moments in the gPC expansion with a quadrature rule. Given a set of Q quadrature weights w_k and quadrature points $\boldsymbol{\xi}_k$, the moments are approximated by

$$\hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i \rangle \approx \sum_{k=1}^Q w_k \boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}_k) \varphi_i(\boldsymbol{\xi}_k) f_\Xi(\boldsymbol{\xi}_k).$$

Hence, the moments can be computed by running a given deterministic solver for the original problem at each quadrature point. To reduce numerical costs in multi-dimensional settings, SC commonly uses sparse grids as quadrature rule: While tensorized quadrature sets require $O(M^p)$ quadrature points to integrate polynomials of maximal degree M exactly, sparse grids are designed to integrate polynomials of total degree M, for which they only require $O(M(\log_2(M)^{p-1}))$ quadrature points, see e.g. [29].

Intrusive methods derive a system which directly describes the time evolution of the moments: Plugging the solution ansatz (3) into the set of equations (1) and projecting the resulting residual to zero yields the stochastic-Galerkin moment system

$$\partial_t \hat{\boldsymbol{u}}_i(t, \boldsymbol{x}) + \nabla \cdot \langle \boldsymbol{f}(\mathcal{U}(\hat{\boldsymbol{u}})) \varphi_i \rangle = \boldsymbol{0}, \tag{4a}$$

$$\hat{\boldsymbol{u}}_i(t = 0, \boldsymbol{x}) = \langle \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x}, \cdot) \varphi_i \rangle, \tag{4b}$$

with $|i| \leq M$. As already mentioned, the SG moment system is not necessarily hyperbolic. To ensure hyperbolicity, the IPM method uses a solution ansatz which minimizes a given entropy under a moment constraint instead of a polynomial expansion (3). For a given convex entropy $s : \mathbb{R}^m \to \mathbb{R}$ for the original problem (1), the IPM solution ansatz is given by

$$\mathcal{U}(\hat{\boldsymbol{u}}) = \arg\min_{\boldsymbol{u}} \langle s(\boldsymbol{u}) \rangle \quad \text{subject to } \hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i \rangle \text{ for } |i| \leq M. \tag{5}$$

Rewritten in its dual form, (5) is transformed into an unconstrained optimization problem. Defining the variables $\boldsymbol{\lambda}_i \in \mathbb{R}^m$, where i is again a multi index, gives the unconstrained dual problem

$$\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}) := \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^{N \times m}} \left\{ \langle s_*(\boldsymbol{\lambda}^T \boldsymbol{\varphi}) \rangle - \sum_{|i| \leq M} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \right\}, \tag{6}$$

where $s_* : \mathbb{R}^m \to \mathbb{R}$ is the Legendre transformation of s, and $\hat{\boldsymbol{\lambda}} := (\hat{\boldsymbol{\lambda}}_i)_{|i| \leq M} \in \mathbb{R}^{N \times m}$ are called the dual variables. The solution to (5) is then given by

$$\mathcal{U}(\hat{\boldsymbol{u}}) = (\nabla_{\boldsymbol{u}} s)^{-1} (\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}})^T \boldsymbol{\varphi}). \tag{7}$$

When plugging this ansatz into the original equations (1) and projecting the resulting residual to zero again yields the moment system (4), but with the ansatz (7) instead of (3).

3. Discretization of the IPM system

3.1. Finite Volume Discretization

In the following, we discretize the moment system in space and time according to [8]. Due to the fact, that stochastic-Galerkin can be interpreted as IPM with a quadratic entropy, it suffices to only derive a discretization of the IPM moment system. Hence, we discretize the system (4) with the more general IPM solution ansatz (7). Omitting initial conditions and assuming a one-dimensional spatial domain, we can write the IPM system as

$$\partial_t \hat{\boldsymbol{u}} + \partial_x \boldsymbol{F}(\hat{\boldsymbol{u}}) = \boldsymbol{0}$$

with the flux $\boldsymbol{F} : \mathbb{R}^{N \times m} \to \mathbb{R}^{N \times m}$, $\boldsymbol{F}(\hat{\boldsymbol{u}}) = \langle \boldsymbol{f}(\mathcal{U}(\hat{\boldsymbol{u}}))\boldsymbol{\varphi}^T \rangle^T$. Note that the inner transpose represents a dyadic product and therefore the outer transpose is applied to a matrix. Due to hyperbolicity of the IPM moment system, one can use a finite-volume method to approximate the time evolution of the IPM moments. We choose the discrete unknowns which represent the solution to be the spatial averages over each cell at time t_n, given by

$$\hat{\boldsymbol{u}}_{ij}^n \simeq \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \hat{\boldsymbol{u}}_i(t_n, x) dx.$$

If a moment vector in cell j at time t_n is denoted as $\hat{\boldsymbol{u}}_j^n = (\hat{\boldsymbol{u}}_{ij}^n)_{|i| \leq M} \in \mathbb{R}^{N \times m}$, the finite-volume scheme can be written in conservative form with the numerical flux $\boldsymbol{G} : \mathbb{R}^{N \times m} \times \mathbb{R}^{N \times m} \to \mathbb{R}^{N \times m}$ as

$$\hat{\boldsymbol{u}}_j^{n+1} = \hat{\boldsymbol{u}}_j^n - \frac{\Delta t}{\Delta x} \left(\boldsymbol{G}(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) - \boldsymbol{G}(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n) \right) \tag{8}$$

for $j = 1, \cdots, N_x$ and $n = 0, \cdots, N_t$. Here, the number of spatial cells is denoted by N_x and the number of time steps by N_t. The numerical flux is assumed to be consistent, i.e. $\boldsymbol{G}(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}) = \boldsymbol{F}(\hat{\boldsymbol{u}})$.

When a consistent numerical flux $\boldsymbol{g} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$, $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{u}_\ell, \boldsymbol{u}_r)$ is available for the original problem (1), then for the IPM system we can simply take the numerical flux

$$\tilde{\boldsymbol{G}}(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) = \langle \boldsymbol{g}(\mathcal{U}(\hat{\boldsymbol{u}}_j^n), \mathcal{U}(\hat{\boldsymbol{u}}_{j+1}^n))\boldsymbol{\varphi}^T \rangle^T$$

in (8). Commonly, this integral cannot be evaluated analytically and therefore needs to be approximated by a quadrature rule

$$\langle h \rangle \approx \langle h \rangle_Q := \sum_{k=1}^{Q} w_k h(\boldsymbol{\xi}_k) f_\Xi(\boldsymbol{\xi}_k).$$

The approximated numerical flux then becomes

$$\boldsymbol{G}(\hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n) = \langle \boldsymbol{g}(\mathcal{U}(\hat{\boldsymbol{u}}_j^n), \mathcal{U}(\hat{\boldsymbol{u}}_{j+1}^n)) \boldsymbol{\varphi}^T \rangle_Q^T. \tag{9}$$

Note that the numerical flux requires evaluating the ansatz $\mathcal{U}(\hat{\boldsymbol{u}}_j^n)$. To simplify notation, we define $\boldsymbol{u}_s : \mathbb{R}^p \to \mathbb{R}^p$,

$$\boldsymbol{u}_s(\boldsymbol{\Lambda}) := \left(\nabla_{\boldsymbol{u}} s\right)^{-1}(\boldsymbol{\Lambda}),$$

meaning that the IPM ansatz (7) at cell j in timestep n can be written as

$$\mathcal{U}(\hat{\boldsymbol{u}}_j^n) = \boldsymbol{u}_s(\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_j^n)^T \boldsymbol{\varphi}).$$

The computation of the dual variables $\hat{\boldsymbol{\lambda}}_j^n := \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_j^n)$ requires solving the dual problem (6) for the moment vector $\hat{\boldsymbol{u}}_j^n$. Therefore, to determine the dual variables for a given moment vector $\hat{\boldsymbol{u}}$, the cost function

$$L(\boldsymbol{\lambda}; \hat{\boldsymbol{u}}) := \langle s_*(\boldsymbol{\lambda}^T \boldsymbol{\varphi}) \rangle_Q - \sum_{i \leq M} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \tag{10}$$

needs to be minimized. Hence, one needs to find the root of

$$\nabla_{\boldsymbol{\lambda_i}} L(\boldsymbol{\lambda}; \hat{\boldsymbol{u}}) = \langle \nabla s_*(\boldsymbol{\lambda}^T \boldsymbol{\varphi}) \boldsymbol{\varphi}^T \rangle_Q^T - \hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}_s(\boldsymbol{\lambda}^T \boldsymbol{\varphi}) \boldsymbol{\varphi}^T \rangle_Q^T - \hat{\boldsymbol{u}}_i,$$

where we used $\nabla s_* \equiv \boldsymbol{u}_s$. The root is usually determined by using Newton's method. For simplicity, let us define the full gradient of the Lagrangian to be $\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}; \hat{\boldsymbol{u}}) \in \mathbb{R}^{N \cdot m}$, i.e. we store all entries in a vector. Newton's method uses the iteration function $\boldsymbol{d} : \mathbb{R}^{N \times m} \times \mathbb{R}^{N \times m} \to \mathbb{R}^{N \times m}$,

$$\boldsymbol{d}(\boldsymbol{\lambda}, \hat{\boldsymbol{u}}) := \boldsymbol{\lambda} - \boldsymbol{H}(\boldsymbol{\lambda})^{-1} \cdot \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}; \hat{\boldsymbol{u}}), \tag{11}$$

where $\boldsymbol{H} \in \mathbb{R}^{N \cdot n \times N \cdot m}$ is the Hessian of (10), given by

$$\boldsymbol{H}(\boldsymbol{\lambda}) := \langle \nabla \boldsymbol{u}_s(\boldsymbol{\lambda}^T \boldsymbol{\varphi}) \otimes \boldsymbol{\varphi} \boldsymbol{\varphi}^T \rangle_Q^T.$$

The function $\boldsymbol{d}$ will in the following be called dual iteration function. Now, the Newton iteration l for spatial cell j is given by

$$\boldsymbol{\lambda}_j^{(l+1)} = \boldsymbol{d}(\boldsymbol{\lambda}_j^{(l)}, \hat{\boldsymbol{u}}_{\boldsymbol{j}}). \tag{12}$$

The exact dual state is then obtained by computing the fixed point of $\boldsymbol{d}$, meaning that one converges the iteration (12), i.e. $\hat{\boldsymbol{\lambda}}_j^n := \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_j^n) = \lim_{l \to \infty} \boldsymbol{d}(\boldsymbol{\lambda}_j^{(l)}, \hat{\boldsymbol{u}}_j^n)$. To obtain a finite number of iterations for the iteration in cell j, the stopping criterion

$$\sum_{i=0}^{m} \left\| \nabla_{\boldsymbol{\lambda_i}} L(\boldsymbol{\lambda}_j^{(l)}; \hat{\boldsymbol{u}}_j^n) \right\| < \tau \tag{13}$$

is used.

We now write down the entire scheme: To obtain a more compact notation, we define

$$\boldsymbol{c}(\boldsymbol{\lambda}_\ell, \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_r) := \langle \boldsymbol{u}_s(\boldsymbol{\lambda}_c^T \boldsymbol{\varphi}) \boldsymbol{\varphi}^T \rangle_Q^T - \frac{\Delta t}{\Delta x} \left(\langle \boldsymbol{g}(\boldsymbol{u}_s(\boldsymbol{\lambda}_c^T \boldsymbol{\varphi}), \boldsymbol{u}_s(\boldsymbol{\lambda}_r^T \boldsymbol{\varphi})) \boldsymbol{\varphi}^T \rangle_Q^T - \langle \boldsymbol{g}(\boldsymbol{u}_s(\boldsymbol{\lambda}_\ell^T \boldsymbol{\varphi}), \boldsymbol{u}_s(\boldsymbol{\lambda}_c^T \boldsymbol{\varphi})) \boldsymbol{\varphi}^T \rangle_Q^T \right). \tag{14}$$

The moment iteration is then given by

$$\hat{\boldsymbol{u}}_j^{n+1} = \boldsymbol{c}\left(\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_{j-1}^n), \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_j^n), \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_{j+1}^n)\right), \tag{15}$$

where the map from the moment vector to the dual variables, i.e. $\boldsymbol{\lambda}(\hat{\boldsymbol{u}}_j^n)$, is obtained by iterating

$$\boldsymbol{\lambda}_j^{(l+1)} = \boldsymbol{d}(\boldsymbol{\lambda}_j^{(l)}; \hat{\boldsymbol{u}}_j^n). \tag{16}$$

until condition (13) is fulfilled. This gives Algorithm 1.

Algorithm 1 IPM algorithm

for $j = 0$ to $N_x + 1$ **do**
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x,\cdot)\boldsymbol{\varphi}\rangle_Q dx$
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\boldsymbol{\lambda}_j^{(0)} \leftarrow \hat{\boldsymbol{\lambda}}_j^n$
    while (13) is violated **do**
      $\boldsymbol{\lambda}_j^{(l+1)} \leftarrow \boldsymbol{d}(\boldsymbol{\lambda}_j^{(l)}; \hat{\boldsymbol{u}}_j^n)$
      $l \leftarrow l + 1$
    $\hat{\boldsymbol{\lambda}}_j^{n+1} \leftarrow \boldsymbol{\lambda}_j^{(l)}$
  for $j = 1$ to N_x **do**
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}(\hat{\boldsymbol{\lambda}}_{j-1}^{n+1}, \hat{\boldsymbol{\lambda}}_j^{n+1}, \hat{\boldsymbol{\lambda}}_{j+1}^{n+1})$

3.2. Properties of the kinetic flux

A straight-forward implementation is ensured by the choice of the numerical flux (9). This choice of the numerical flux is common in the field of transport theory, where it is called the *kinetic flux* or *kinetic scheme*, see e.g. [30, 31, 32, 33]. By simply taking moments of a given numerical flux for the deterministic problem, the method can easily be applied to various physical problems whenever an implementation of $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{u}_\ell, \boldsymbol{u}_r)$ is available. Therefore, we call the proposed numerical method *semi-intrusive*.

Intrusive numerical methods which compute arising integrals analytically and therefore directly depend on the moments (i.e. they do not necessitate the evaluation of the gPC expansion on quadrature points) can be constructed by performing a gPC expansion on the system flux directly [34]. Examples can be found in [35, 36, 37, 38] for the computation of numerical fluxes and sources. While the analytic computation of arising integrals is not always more efficient [39, Section 6], it can also complicate recycling a deterministic solver. See Appendix A for a comparison of numerical costs when using Burgers' equation. However, when not using a quadratic entropy in the IPM method or when the physical flux of the deterministic problem is not a polynomial, it is not clear how many quadrature points the numerical quadrature rule requires to guarantee a sufficiently small quadrature error. We will study the approximation properties of IPM with different quadrature orders in Section 7.1.

4. One-Shot IPM

In the following section we only consider steady state problems, i.e. equation (1a) reduces to

$$\nabla \cdot \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\xi})) = \boldsymbol{0} \quad \text{in } D \tag{17}$$

with adequate boundary conditions. A general strategy for computing the steady state solution to (17) is to introduce a pseudo-time and numerically treat (17) as an unsteady problem. A steady state solution is then obtained by iterating in pseudo-time until the solution remains constant. It is important to point out that

the time it takes to converge to a steady state solution is crucially affected by the chosen initial condition and its distance to the steady state solution. Similar to the unsteady case (1), we can again derive a moment system for (17) given by

$$\nabla \cdot \langle \boldsymbol{f}(\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\xi}))\boldsymbol{\varphi}^T \rangle^T = \boldsymbol{0} \quad \text{in } D \tag{18}$$

which is again needed for the construction of intrusive methods. By introducing a pseudo-time and using the IPM closure, we obtain the same system as in (4), i.e. Algorithm 1 can be used to iterate to a steady state solution. Note that now, the time iteration is not performed for a fixed number of time steps N_t, but until the condition

$$\sum_{j=1}^{N_x} \Delta x_j \|\hat{\boldsymbol{u}}_j^n - \hat{\boldsymbol{u}}_j^{n-1}\| \leq \varepsilon \tag{19}$$

is fulfilled. Since one is generally interested in low order moments such as the expectation value, this residual can be modified by only accounting for the zero order moments.

In this section we aim at breaking up the inner loop in the IPM Algorithm 1, i.e. to just perform one iteration of the dual problem in each time step. Consequently, the IPM reconstruction given by (5) is not done exactly, meaning that the reconstructed solution does not minimize the entropy while not fulfilling the moment constraint. However, the fact that the moment vectors are not yet converged to the steady solution seems to permit such an inexact reconstruction. Hence, we aim at iterating the moments to steady state and the dual variables to the exact solution of the IPM optimization problem (5) simultaneously. By successively performing one update of the moment iteration and one update of the dual iteration, we obtain

$$\boldsymbol{\lambda}_j^{n+1} = \boldsymbol{d}(\boldsymbol{\lambda}_j^n, \boldsymbol{u}_j^n) \quad \text{for all j} \tag{20a}$$

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{c}\left(\boldsymbol{\lambda}_{j-1}^{n+1}, \boldsymbol{\lambda}_j^{n+1}, \boldsymbol{\lambda}_{j+1}^{n+1}\right). \tag{20b}$$

This yields Algorithm 2.

Algorithm 2 One-Shot IPM implementation

for $j = 0$ to $N_x + 1$ **do**
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot)\boldsymbol{\varphi} \rangle_Q dx$
while (19) is violated **do**
  for $j = 1$ to N_x **do**
    $\boldsymbol{\lambda}_j^{n+1} \leftarrow \boldsymbol{d}(\boldsymbol{\lambda}_j^n; \hat{\boldsymbol{u}}_j^n)$
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}(\boldsymbol{\lambda}_{j-1}^{n+1}, \boldsymbol{\lambda}_j^{n+1}, \boldsymbol{\lambda}_{j+1}^{n+1})$
  $n \leftarrow n + 1$

We call this method One-Shot IPM, since it is inspired by One-Shot optimization, see for example [27], which uses only a single iteration of the primal and dual step in order to update the design variables. Note that the dual variables from the One-Shot iteration are written without a hat to indicate that they are not the exact solution of the dual problem.

In the following, we will show that this iteration converges, if the chosen initial condition is sufficiently close to the steady state solution. For this we take an approach commonly chosen to prove local convergence properties of Newton's method: In Theorem 1, we show that the iteration function is contractive at its fixed point and conclude in Theorem 2 that this yields local convergence. Hence, we preserve the convergence property of the original IPM method, which uses Newton's method and therefore only converges locally as well.

Theorem 1. *Assume that the classical IPM iteration is contractive at its fixed point $\hat{\boldsymbol{u}}^*$. Then the Jacobi matrix $\boldsymbol{J}$ of the One-Shot IPM iteration* (20) *has a spectral radius $\rho(\boldsymbol{J}) < 1$ at the fixed point $(\boldsymbol{\lambda}^*, \hat{\boldsymbol{u}}^*)$.*

Proof. First, to understand what contraction of the classical IPM iteration implies, we rewrite the moment iteration (15) of the classical IPM scheme: When defining the update function

$$\tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_c, \hat{\boldsymbol{u}}_r\right) := \boldsymbol{c}\left(\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_\ell), \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_c), \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_r)\right)$$

we can rewrite the classical moment iteration as

$$\hat{\boldsymbol{u}}_j^{n+1} = \tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}_{j-1}^n, \hat{\boldsymbol{u}}_j^n, \hat{\boldsymbol{u}}_{j+1}^n\right). \tag{21}$$

Since we assume that the classical IPM scheme is contractive at its fixed point, we have $\rho(\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}}(\hat{\boldsymbol{u}}^*)) < 1$ with $\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}} \in \mathbb{R}^{N\cdot N_x \times N\cdot N_x}$ defined by

$$\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}} = \begin{pmatrix} \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_1 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_1 & 0 & 0 & \dots \\ \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_2 & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_2 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_2 & 0 & \dots \\ 0 & \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_3 & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_3 & \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_3 & \\ \vdots & & & \ddots & \\ 0 & \cdots & 0 & \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_{N_x} & \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_{N_x} \end{pmatrix},$$

where we define $\tilde{\boldsymbol{c}}_j := \tilde{\boldsymbol{c}}\left(\hat{\boldsymbol{u}}_{j-1}^*, \hat{\boldsymbol{u}}_j^*, \hat{\boldsymbol{u}}_{j+1}^*\right)$ for all j. Now for each term inside the matrix $\nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}}$ we have

$$\partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{\lambda}}_\ell}\frac{\partial \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_{j-1}^*)}{\partial \hat{\boldsymbol{u}}}, \;\; \partial_{\hat{\boldsymbol{u}}_c}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{\lambda}}_c}\frac{\partial \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_j^*)}{\partial \hat{\boldsymbol{u}}}, \;\; \partial_{\hat{\boldsymbol{u}}_r}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \hat{\boldsymbol{\lambda}}_r}\frac{\partial \hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}_{j+1}^*)}{\partial \hat{\boldsymbol{u}}}. \tag{22}$$

We first wish to understand the structure of the terms $\partial_{\hat{\boldsymbol{u}}}\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}})$. For this, we note that the exact dual variables fulfill

$$\hat{\boldsymbol{u}} = \langle \boldsymbol{u}_s(\hat{\boldsymbol{\lambda}}^T\boldsymbol{\varphi})\boldsymbol{\varphi}^T\rangle^T =: \boldsymbol{h}(\hat{\boldsymbol{\lambda}}), \tag{23}$$

which is why we have the mapping $\hat{\boldsymbol{u}} : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$, $\hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}}) = \boldsymbol{h}(\hat{\boldsymbol{\lambda}})$. Since the solution of the dual problem for a given moment vector is unique, this mapping is bijective and therefore we have an inverse function

$$\hat{\boldsymbol{\lambda}} = \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}})). \tag{24}$$

Now we differentiate both sides w.r.t. $\hat{\boldsymbol{\lambda}}$ to get

$$\boldsymbol{I}_d = \frac{\partial \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}}))}{\partial \hat{\boldsymbol{u}}}\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}})}{\partial \hat{\boldsymbol{\lambda}}}.$$

We multiply with the matrix inverse of $\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}})}{\partial \hat{\boldsymbol{\lambda}}}$ to get

$$\left(\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}})}{\partial \hat{\boldsymbol{\lambda}}}\right)^{-1} = \frac{\partial \boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}}))}{\partial \hat{\boldsymbol{u}}}.$$

Note that on the left-hand-side we have the inverse of a matrix and on the right-hand-side, we have the inverse of a multi-dimensional function. By rewriting $\boldsymbol{h}^{-1}(\hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}}))$ as $\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}})$ and simply computing the term $\frac{\partial \hat{\boldsymbol{u}}(\hat{\boldsymbol{\lambda}})}{\partial \hat{\boldsymbol{\lambda}}}$ by differentiating (23) w.r.t. $\hat{\boldsymbol{\lambda}}$, one obtains

$$\partial_{\hat{\boldsymbol{u}}}\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}) = \langle \nabla \boldsymbol{u}_s(\hat{\boldsymbol{\lambda}}^T\boldsymbol{\varphi})\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-T}. \tag{25}$$

Now we begin to derive the spectrum of the *One-Shot IPM* iteration (20). Note that in its current form this iteration is not really a fixed point iteration, since it uses the time updated dual variables in (20b). To obtain a fixed point iteration, we plug the dual iteration step (20a) into the moment iteration (20b) to obtain

$$\begin{aligned}\boldsymbol{\lambda}_j^{n+1} &= \boldsymbol{d}(\boldsymbol{\lambda}_j^n, \hat{\boldsymbol{u}}_j^n) \quad \text{for all j} \\ \hat{\boldsymbol{u}}_j^{n+1} &= \boldsymbol{c}\left(\boldsymbol{d}(\boldsymbol{\lambda}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n), \boldsymbol{d}(\boldsymbol{\lambda}_j^n, \hat{\boldsymbol{u}}_j^n), \boldsymbol{d}(\boldsymbol{\lambda}_{j+1}^n, \hat{\boldsymbol{u}}_{j+1}^n)\right).\end{aligned}$$

The Jacobian $\boldsymbol{J} \in \mathbb{R}^{2N\cdot N_x \times 2N\cdot N_x}$ has the form

$$\boldsymbol{J} = \begin{pmatrix} \partial_{\boldsymbol{\lambda}}\boldsymbol{d} & \partial_{\hat{\boldsymbol{u}}}\boldsymbol{d} \\ \partial_{\boldsymbol{\lambda}}\boldsymbol{c} & \partial_{\hat{\boldsymbol{u}}}\boldsymbol{c} \end{pmatrix}, \tag{26}$$

where each block has entries for all spatial cells. We start by looking at $\partial_{\boldsymbol{\lambda}}\boldsymbol{d}$. For the columns belonging to cell j, we have

$$\begin{aligned}\partial_{\boldsymbol{\lambda}}\boldsymbol{d}(\boldsymbol{\lambda}_j^n, \hat{\boldsymbol{u}}_j^n) &= \boldsymbol{I}_d - \boldsymbol{H}(\boldsymbol{\lambda})^{-1} \cdot \langle \nabla \boldsymbol{u}_s(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^T - \partial_{\boldsymbol{\lambda}}\boldsymbol{H}(\boldsymbol{\lambda})^{-1} \cdot \left(\langle \boldsymbol{u}_s(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}^T\rangle^T - \hat{\boldsymbol{u}}\right) \\ &= -\partial_{\boldsymbol{\lambda}}\boldsymbol{H}(\boldsymbol{\lambda})^{-1} \cdot \left(\langle \boldsymbol{u}_s(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}^T\rangle^T - \hat{\boldsymbol{u}}\right).\end{aligned}$$

Recall that at the fixed point $(\boldsymbol{\lambda}^*, \hat{\boldsymbol{u}}^*)$, we have $\langle \boldsymbol{u}_s(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}^T\rangle^T = \hat{\boldsymbol{u}}$, hence one obtains $\partial_{\boldsymbol{\lambda}}\boldsymbol{d} = \boldsymbol{0}$. For the block $\partial_{\hat{\boldsymbol{u}}}\boldsymbol{d}$, we get

$$\partial_{\hat{\boldsymbol{u}}}\boldsymbol{d}(\boldsymbol{\lambda}_j^n, \hat{\boldsymbol{u}}_j^n) = \boldsymbol{H}(\boldsymbol{\lambda})^{-1},$$

hence $\partial_{\hat{\boldsymbol{u}}}\boldsymbol{d}$ is a block diagonal matrix. Let us now look at $\partial_{\boldsymbol{\lambda}}\boldsymbol{c}$ at a fixed spatial cell j:

$$\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\frac{\partial \boldsymbol{d}(\boldsymbol{\lambda}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n)}{\partial \boldsymbol{\lambda}} = \boldsymbol{0},$$

since we already showed that by the choice of $\boldsymbol{H}(\boldsymbol{\lambda})^{-1}$ the term $\partial_{\boldsymbol{\lambda}}\boldsymbol{d}$ is zero. We can show the same result for all spatial cells and all inputs of $\boldsymbol{c}$ analogously, hence $\partial_{\boldsymbol{\lambda}}\boldsymbol{c} = \boldsymbol{0}$. For the last block, we have that

$$\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\frac{\partial \boldsymbol{d}(\boldsymbol{\lambda}_{j-1}^n, \hat{\boldsymbol{u}}_{j-1}^n)}{\partial \hat{\boldsymbol{u}}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\boldsymbol{H}(\boldsymbol{\lambda})^{-1} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\langle \nabla \boldsymbol{u}_s(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_{j-1}^n)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-T} = \partial_{\hat{\boldsymbol{u}}_\ell}\tilde{\boldsymbol{c}}_j$$

by the choice of $\boldsymbol{H}(\boldsymbol{\lambda})^{-1}$ as well as (22) and (25). We obtain an analogous result for the second and third input. Hence, we have that $\partial_{\hat{\boldsymbol{u}}}\boldsymbol{c} = \nabla_{\hat{\boldsymbol{u}}}\tilde{\boldsymbol{c}}$, which only has eigenvalues between -1 and 1 by the assumption that the classical IPM iteration is contractive. Since $\boldsymbol{J}$ is an upper triangular block matrix, the eigenvalues are given by $\lambda\left(\partial_{\boldsymbol{\lambda}}\boldsymbol{d}\right) = 0$ and $\lambda\left(\partial_{\hat{\boldsymbol{u}}}\boldsymbol{c}\right) \in (-1, 1)$, hence the One-Shot IPM is contractive around its fixed point. □

Theorem 2. *With the assumptions from Theorem 1, the One-Shot IPM converges locally, i.e. there exists a $\delta > 0$ s.t. for all starting points $(\boldsymbol{\lambda}^0, \hat{\boldsymbol{u}}^0) \in B_\delta(\boldsymbol{\lambda}^*, \hat{\boldsymbol{u}}^*)$ we have*

$$\|(\boldsymbol{\lambda}^n, \hat{\boldsymbol{u}}^n) - (\boldsymbol{\lambda}^*, \hat{\boldsymbol{u}}^*)\| \to 0 \qquad \text{for } n \to \infty.$$

Proof. By Theorem 1, the One-Shot scheme is contractive at its fixed point. Since we assumed convergence of the classical IPM scheme, we can conclude that all entries in the Jacobian $\boldsymbol{J}$ are continuous functions. Furthermore, the determinant of $\tilde{\boldsymbol{J}} := \boldsymbol{J} - \lambda \boldsymbol{I}_d$ is a polynomial of continuous functions, since

$$\det(\tilde{\boldsymbol{J}}) = \sum_\sigma \operatorname{sgn}(\sigma) \prod_{i=1}^{2N_x N} \tilde{J}_{\sigma(i),i}.$$

Since the roots of a polynomial vary continuously with its coefficients, the eigenvalues of $\boldsymbol{J}$ are continuous w.r.t $(\boldsymbol{\lambda}, \hat{\boldsymbol{u}})$. Hence there exists an open ball with radius δ around the fixed point in which the eigenvalues remain in the interval $(-1, 1)$. □

5. Adaptivity

The following section presents the adaptivity strategy used in this work. Since stochastic hyperbolic problems generally experience shocks in a small portion of the space-time domain, the idea is to perform arising computations on a high accuracy level in this small area, while keeping a low level of accuracy in the remainder. The idea is to automatically select the lowest order moment capable of approximating the solution with given accuracy, i.e. the same error is obtained while using a significantly reduced number of unknowns in most parts of the computational domain and thus boost the performance of intrusive methods.

In the following, we discuss the building blocks of the IPM method for refinement levels $\ell = 1, \cdots, N_{\text{ad}}$, where level 1 uses the coarsest discretization and level N_{ad} uses the finest discretization of the uncertain domain. At a given refinement level ℓ, the total degree of the basis function is given by M_ℓ with a corresponding number of moments N_ℓ. The number of quadrature points at level ℓ is denoted by Q_ℓ. To determine the refinement level of a given moment vector $\hat{\boldsymbol{u}}$ we choose techniques used in discontinuous Galerkin (DG) methods. Adaptivity is a common strategy to accelerate this class of methods and several indicators to determine the smoothness of the solution exist. Translating the idea of the so-called discontinuity sensor which has been defined in [40] to uncertainty quantification, we define the polynomial approximation at refinement level ℓ as

$$\tilde{\boldsymbol{u}}_\ell := \sum_{|i| \leq M_\ell} \hat{\boldsymbol{u}}_i \varphi_i.$$

Now the indicator for a moment vector at level ℓ is defined as

$$\boldsymbol{S}_\ell := \frac{\langle (\tilde{\boldsymbol{u}}_\ell - \tilde{\boldsymbol{u}}_{\ell-1})^2 \rangle}{\langle \tilde{\boldsymbol{u}}_\ell^2 \rangle}, \tag{27}$$

where divisions and multiplications are performed element-wise. Note that a similar indicator has been used in [23] for intrusive methods in uncertainty quantification. In this work, we use the first entry in $\boldsymbol{S}_\ell$ to determine the refinement level, i.e. in the case of gas dynamics, the regularity of the density is chosen to indicate an adequate refinement level. If the moment vector in a given cell and at a certain timestep is initially at refinement level ℓ, this level is kept if the error indicator (27) lies in the interval $I_\delta := [\delta_-, \delta_+]$. Here $\delta_\pm$ are user determined parameters. If the indicator is smaller than δ_-, the refinement level is decreased to the next lower level, if it lies above δ_+, it is increased to the next higher level.

Now we need to specify how the different building blocks of IPM can be modified to work with varying truncation orders in different cells. Let us first add dimensions to the notation of the dual iteration function $\boldsymbol{d}$, which has been defined in (11). Now, we have $\boldsymbol{d}_\ell : \mathbb{R}^{N_\ell \times m} \times \mathbb{R}^{N_\ell \times m} \to \mathbb{R}^{N_\ell \times m}$, given by

$$\boldsymbol{d}_\ell(\boldsymbol{\lambda}, \hat{\boldsymbol{u}}) := \boldsymbol{\lambda} - \boldsymbol{H}_\ell^{-1}(\boldsymbol{\lambda}) \cdot \left(\langle \boldsymbol{u}_s(\boldsymbol{\lambda}^T \boldsymbol{\varphi}_\ell) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T - \hat{\boldsymbol{u}} \right), \tag{28}$$

where $\boldsymbol{\varphi}_\ell \in \mathbb{R}^{N_\ell}$ collects all basis functions with total degree smaller or equal to M_ℓ. The Hessian $\boldsymbol{H}_\ell$ is given by

$$\boldsymbol{H}_\ell(\boldsymbol{\lambda}) := \langle \nabla \boldsymbol{u}_s(\boldsymbol{\lambda}^T \boldsymbol{\varphi}_\ell) \otimes \boldsymbol{\varphi}_\ell \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T.$$

An adaptive version of the moment iteration (14) is denoted by $\boldsymbol{c}_\ell^{\boldsymbol{\ell}'} : \mathbb{R}^{N_{\ell_1'} \times m} \times \mathbb{R}^{N_{\ell_2'} \times m} \times \mathbb{R}^{N_{\ell_3'} \times m} \to \mathbb{R}^{N_\ell \times m}$ and given by

$$\begin{aligned} \boldsymbol{c}_\ell^{\boldsymbol{\ell}'}(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) :=& \langle \boldsymbol{u}_s(\boldsymbol{\lambda}_2^T \boldsymbol{\varphi}_{\ell_2'}) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T \\ &- \frac{\Delta t}{\Delta x} \left(\langle \boldsymbol{g}(\boldsymbol{u}_s(\boldsymbol{\lambda}_2^T \boldsymbol{\varphi}_{\ell_2'}), \boldsymbol{u}_s(\boldsymbol{\lambda}_3^T \boldsymbol{\varphi}_{\ell_3'})) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T - \langle \boldsymbol{g}(\boldsymbol{u}_s(\boldsymbol{\lambda}_1^T \boldsymbol{\varphi}_{\ell_1'}), \boldsymbol{u}_s(\boldsymbol{\lambda}_2^T \boldsymbol{\varphi}_{\ell_2'})) \boldsymbol{\varphi}_\ell^T \rangle_{Q_\ell}^T \right). \end{aligned} \tag{29}$$

Hence, the index vector $\boldsymbol{\ell}' \in \mathbb{N}^3$ denotes the refinement levels of the stencil cells, which are used to compute the time updated moment vector at level ℓ.

The strategy now is to perform the dual update for a set of moment vectors $\hat{\boldsymbol{u}}_j^n$ at refinement levels ℓ_j^n for $j = 1, \cdots, N_x$. Thus, the dual iteration makes use of the iteration function (28) at refinement level ℓ_j^n. After that, the refinement level at the next time step ℓ_j^{n+1} is determined by making use of the smoothness indicator (27). The moment update then computes the moments at the time updated refinement level ℓ_j^{n+1}, utilizing the the dual states at the old refinement levels $\boldsymbol{\ell}' = (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$.

Note that we use nested quadrature rules, which facilitate the task of evaluating the quadrature in the moment update (29). Assume that we want to compute the moment update in cell j with refinement level ℓ_j where a neighboring cell $j-1$ has refinement level ℓ_{j-1}. Now if $\ell_{j-1} \geq \ell_j$, the solution of cell $j-1$ is known at all Q_ℓ quadrature points, hence the integral inside the moment update can be computed. Vice versa, if $\ell_{j-1} \leq \ell_j$, we need to evaluate the neighboring cell at the finer quadrature level ℓ_j. Except from this, increasing or decreasing the refinement level does not lead to additional costs.

The IPM algorithm with adaptivity results in Algorithm 3.

Algorithm 3 Adaptive IPM implementation

for $j = 0$ to $N_x + 1$ **do**
  $\ell_j^0 \leftarrow$ choose initial refinement level
  $\boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot) \boldsymbol{\varphi}_{\ell_j^0} \rangle_{Q_{\ell_j^0}} dx$
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\boldsymbol{\lambda}_j^{(0)} \leftarrow \hat{\boldsymbol{\lambda}}_j^n$
    while (13) is violated **do**
      $\boldsymbol{\lambda}_j^{(l+1)} \leftarrow \boldsymbol{d}_{\ell_j^n}(\boldsymbol{\lambda}_j^{(l)}; \hat{\boldsymbol{u}}_j^n)$
      $l \leftarrow l + 1$
    $\hat{\boldsymbol{\lambda}}_j^{n+1} \leftarrow \boldsymbol{\lambda}_j^{(l)}$
    $\ell_j^{n+1} \leftarrow \mathrm{DetermineRefinementLevel}\left(\hat{\boldsymbol{\lambda}}_j^{n+1}\right)$
  for $j = 1$ to N_x **do**
    $\boldsymbol{\ell}' \leftarrow (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$
    $\hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}_{\ell_j^{n+1}}^{\boldsymbol{\ell}'}(\hat{\boldsymbol{\lambda}}_{j-1}^{n+1}, \hat{\boldsymbol{\lambda}}_j^{n+1}, \hat{\boldsymbol{\lambda}}_{j+1}^{n+1})$

Adaptivity can be used for intrusive methods in general as well as for steady and unsteady problems. In the case of steady problems, we can make use of a strategy, which we call *refinement retardation*. Recall that the convergence to an admissible steady state solution is expensive and a high accuracy and desirable solution properties are only required at the end of this iteration process. Hence, we propose to iteratively increase the maximal refinement level whenever the residual (19) lies below a certain tolerance ε. For a given set of maximal refinement levels ℓ_l^* and a set of tolerances ε_l^* at which the refinement level must be increased, we can now perform a large amount of the required iterations on a lower, but cheaper refinement level. The same strategy can be applied for One-Shot IPM. In this case, the algorithm is given by Algorithm 4.

Algorithm 4 Adaptive One-Shot IPM implementation with refinement retardation

1: **for** $j = 0$ to $N_x + 1$ **do**
2: $\quad \boldsymbol{u}_j^0 \leftarrow \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_{\mathrm{IC}}(x, \cdot) \boldsymbol{\varphi} \rangle_Q dx$
3: **while** (19) is violated **do**
4: $\quad$ **for** $j = 1$ to N_x **do**
5: $\quad\quad \boldsymbol{\lambda}_j^{n+1} \leftarrow \boldsymbol{d}_{\ell_j^n}(\boldsymbol{\lambda}_j^n; \hat{\boldsymbol{u}}_j^n)$
6: $\quad\quad \ell_j^{n+1} \leftarrow \max\{\mathrm{DetermineRefinementLevel}\left(\boldsymbol{\lambda}_j^{n+1}\right), \ell_l^*\}$
7: $\quad\quad \boldsymbol{\ell}' \leftarrow (\ell_{j-1}^n, \ell_j^n, \ell_{j+1}^n)^T$
8: $\quad\quad \hat{\boldsymbol{u}}_j^{n+1} \leftarrow \boldsymbol{c}_{\ell_j^{n+1}}^{\boldsymbol{\ell}'}(\boldsymbol{\lambda}_{j-1}^{n+1}, \boldsymbol{\lambda}_j^{n+1}, \boldsymbol{\lambda}_{j+1}^{n+1})$
9: $\quad n \leftarrow n + 1$
10: $\quad$ **if** the residual (19) lies below ε_l^* **then**
11: $\quad\quad l \leftarrow l + 1$

6. Parallelization and Implementation

It remains to discuss the parallelization of the presented algorithms. In order to minimize the parallelization overhead, our goal is to minimize the communication between processors. Note that the dual problem (line 8 in Algorithm 3 and line 5 in Algorithm 4) does not require communication, i.e. it suffices to distribute the spatial cells between processors. In contrast to that, the finite volume update (line 14 in Algorithm 3 and line 8 in Algorithm 4) requires communication, since values at neighboring cells need to be evaluated. Hence, distributing the spatial mesh between processors will yield communication overhead since data needs to be sent whenever a stencil cell lies on a different processor. Therefore, we choose to parallelize the quadrature points, which minimizes the computational time spend on communication. As mentioned in Appendix A, we first compute the solution at stencil cells for all quadrature points. I.e. we determine $\boldsymbol{u}_k^{(j)} \in \mathbb{R}^m$ and the corresponding stencil cells for $k = 1, \dots, Q$ by

$$\boldsymbol{u}_k^{(j-1)} := \boldsymbol{u}_s(\boldsymbol{\lambda}_{j-1}^T \boldsymbol{\varphi}_{\ell_1'}(\boldsymbol{\xi}_k)), \quad \boldsymbol{u}_k^{(j)} := \boldsymbol{u}_s(\boldsymbol{\lambda}_j^T \boldsymbol{\varphi}_{\ell_2'}(\boldsymbol{\xi}_k)), \quad \boldsymbol{u}_k^{(j+1)} := \boldsymbol{u}_s(\boldsymbol{\lambda}_{j+1}^T \boldsymbol{\varphi}_{\ell_3'}(\boldsymbol{\xi}_k)).$$

Thus, the finite volume update function (29) can be written as

$$\boldsymbol{c}_\ell^{\boldsymbol{\ell}'}(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) = \sum_{k=1}^{Q} w_k \left[\boldsymbol{u}_k^{(j)} - \frac{\Delta t}{\Delta x} \left(\boldsymbol{g}(\boldsymbol{u}_k^{(j)}, \boldsymbol{u}_k^{(j+1)}) - \boldsymbol{g}(\boldsymbol{u}_k^{(j-1)}, \boldsymbol{u}_k^{(j)}) \right) \right] \boldsymbol{\varphi}_\ell(\boldsymbol{\xi}_k)^T. \tag{30}$$

Instead of distributing the spatial mesh on the different processors, we now distribute the quadrature set, i.e. the sum in (30) can be computed in parallel. Now, after having performed the dual update, the dual variables are send to all processors. With these variables, each processor computes the solution on its portion of the quadrature set and then computes its part of the sum in (30) on all spacial cells. All parts from the different processors are then added together and the full time-updated moments are distributed to all processors. From here, the dual update can again be performed. The standard IPM Algorithm 1 and One-Shot IPM Algorithm 2 use this parallelization strategy accordingly. Again, we point out that stochastic-Galerkin is a variant of IPM, i.e. all presented techniques for IPM can also be used for SG. The SC algorithm that we use to compare intrusive with non-intrusive methods uses a given deterministic solver as a black box. Here, we distribute the quadrature set between all processors. Note that both, SC and IPM are based on the same deterministic solver, i.e. we use the same deterministic numerical flux $\boldsymbol{g}$. To our best knowledge, this allows a fair comparison of the different intrusive and non-intrusive techniques.

7. Results

7.1. 2D Euler equations with a one dimensional uncertainty

We start by quantifying the effects of an uncertain angle of attack $\phi \sim U(0.75, 1.75)$ for a NACA0012 airfoil computed with different methods. The stochastic Euler equations in two dimensions are given by

$$\partial_t \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho e \end{pmatrix} + \partial_{x_1} \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ v_1(\rho e + p) \end{pmatrix} + \partial_{x_2} \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ v_2(\rho e + p) \end{pmatrix} = \mathbf{0}.$$

These equations determine the time evolution of the conserved variables $(\rho, \rho \boldsymbol{v}, \rho e)$, i.e. density, momentum and energy. A closure for the pressure p is given by

$$p = (\gamma - 1)\rho \left(e - \frac{1}{2}(v_1^2 + v_2^2) \right).$$

Here, the heat capacity ratio γ is chosen to be 1.4. The spatial mesh discretizes the flow domain around the airfoil. At the airfoil boundary, we use the Euler slip condition $\boldsymbol{v}^T \boldsymbol{n} = 0$, where $\boldsymbol{n}$ denotes the surface normal. At a sufficiently large distance away from the airfoil, we assume a far field flow with a given Mach number $Ma = 0.8$, pressure $p = 101\,325$ Pa and a temperature of 273.15 K. Now the angle of attack ϕ is uniformly distributed in the interval of $[0.75, 1.75]$ degrees, i.e. we choose $\phi(\xi) = 1.25 + 0.5\xi$ where $\xi \sim U(-1, 1)$. As commonly done, the initial condition is equal to the far field boundary values. Consequently, the wall condition at the airfoil is violated initially and will correct the flow solution.

The computational domain is a circle with a diameter of 40 meters. In the center, the NACA0012 airfoil with a length of one meter is located. The spatial mesh is composed of a coarsely discretized far field and a finely resolved region around the airfoil, since we are interested in the flow solution at the airfoil. Altogether, the mesh consists of 22361 triangular elements.

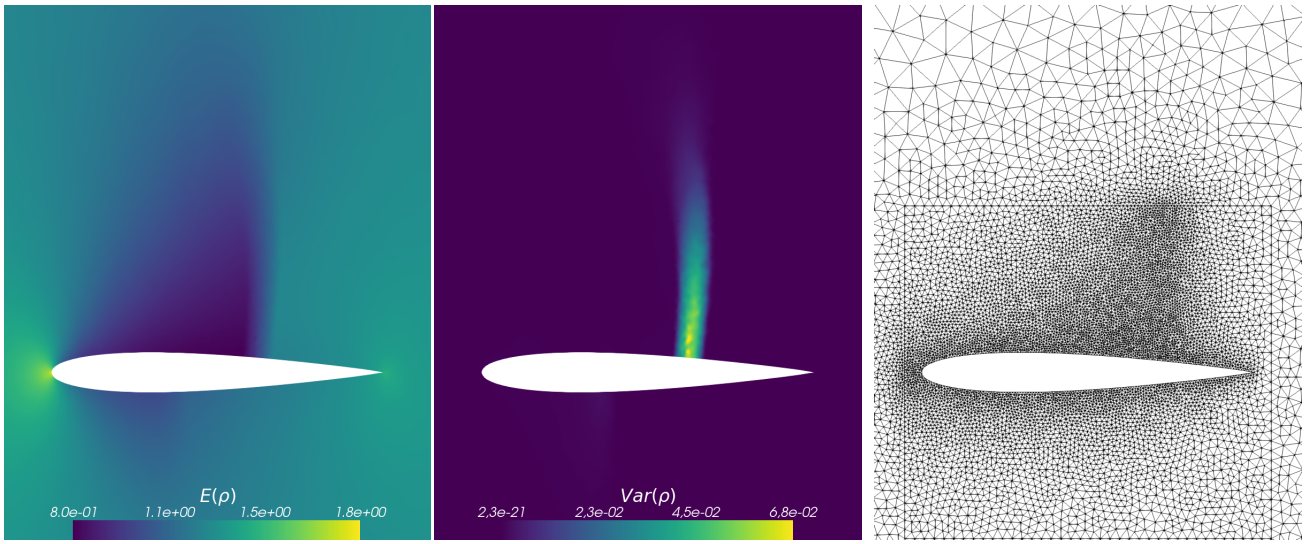


Figure 1: Reference solution E[ρ] and Var[ρ] and the mesh close to the airfoil which is used in the computation of all presented methods.

The aim is to quantify the effects arising from the one-dimensional uncertainty ξ and to investigate its effects on the solution with different methods. To be able to measure the quality of the obtained solutions, we

compute a reference solution using stochastic-Collocation with 100 Gauss-Legendre quadrature points, which can be found in Figure 1. In the following, we investigate the L^2-error of the variance and the expectation value. The L^2-error of the discrete quantity $\boldsymbol{e}_\Delta = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{N_x})^T$, where $\boldsymbol{e}_j$ is the cell average of the quantity $\boldsymbol{e}$ in spatial cell j, is denoted by

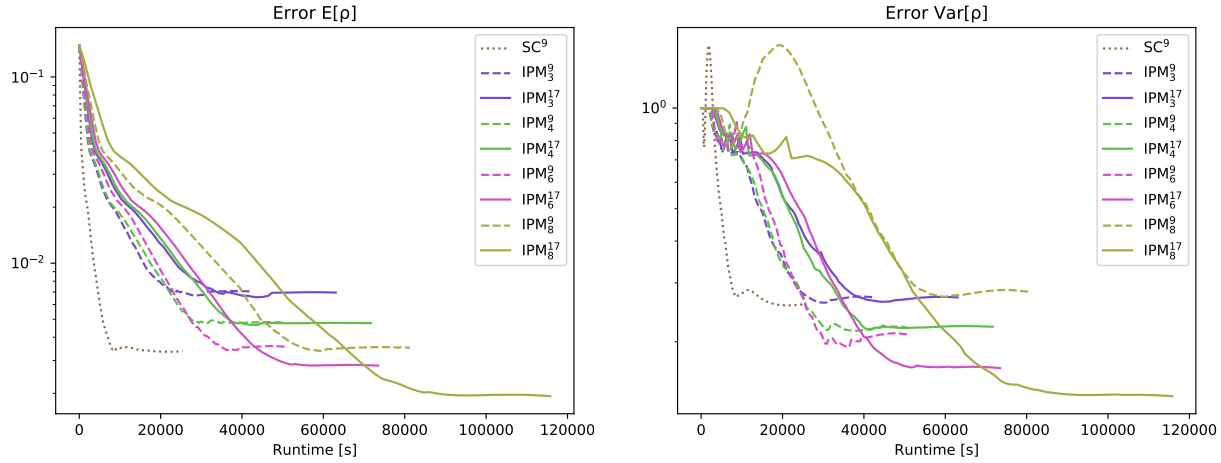
$$\|\boldsymbol{e}_\Delta\|_\Delta := \sqrt{\sum_{j=1}^{N_x} \Delta x_j \boldsymbol{e}_j^2}\,.$$

Hence, when denoting the reference solution by $\boldsymbol{u}_\Delta$ and the moments obtained with the numerical method by $\hat{\boldsymbol{u}}_\Delta$, we investigate the relative error

$$\frac{\|\mathrm{E}[\boldsymbol{u}_\Delta] - \mathrm{E}[\mathcal{U}(\hat{\boldsymbol{u}}_\Delta)]\|_\Delta}{\|\mathrm{E}[\boldsymbol{u}_\Delta]\|_\Delta} \quad \text{and} \quad \frac{\|\mathrm{Var}[\boldsymbol{u}_\Delta] - \mathrm{Var}[\mathcal{U}(\hat{\boldsymbol{u}}_\Delta)]\|_\Delta}{\|\mathrm{Var}[\boldsymbol{u}_\Delta]\|_\Delta}.$$

The error is computed inside a box of one meter height and 1.1 meters length around the airfoil to prevent small fluctuations in the coarsely discretized far field from affecting the error.

The quantities of interests are now computed with the different methods. All methods in this section have been computed using five MPI threads. For more information on the chosen entropy and the resulting solution ansatz for IPM, see Appendix B.



(a) (b)

Figure 2: Relative L^2-error with different quadrature levels for IPM in comparison with SC. The subscript denotes the moment order, the superscript denotes the number of Clenshaw-Curtis quadrature points.

Recall that the numerical flux (9) uses a quadrature rule to approximate integrals. We start by investigating the effects this quadrature has on the solution accuracy. For this, we run the IPM method with a moment order ranging from 3 to 7 using a Clenshaw-Curtis quadrature rule with level three (i.e. 9 quadrature points) and level four (i.e. 17 quadrature points). A comparison of the error obtained with these two quadrature levels is given in Figure 2. To denote the number of quadrature points, we use a superscript and the moment order is denoted by a subscript. We observe the following:

- When for example comparing the error obtained with IPM_8^9 and IPM_8^{17} (i.e. the polynomial order is 8, meaning that 9 moments are used and the computation is done using 9 and 17 quadrature points) it can be seen that the error stagnates when the chosen quadrature is not sufficiently accurate. Hence, the accuracy level is dominated by aliasing effects that result from an inaccurate quadrature rule in the numerical flux and not the truncation error of the moment system.

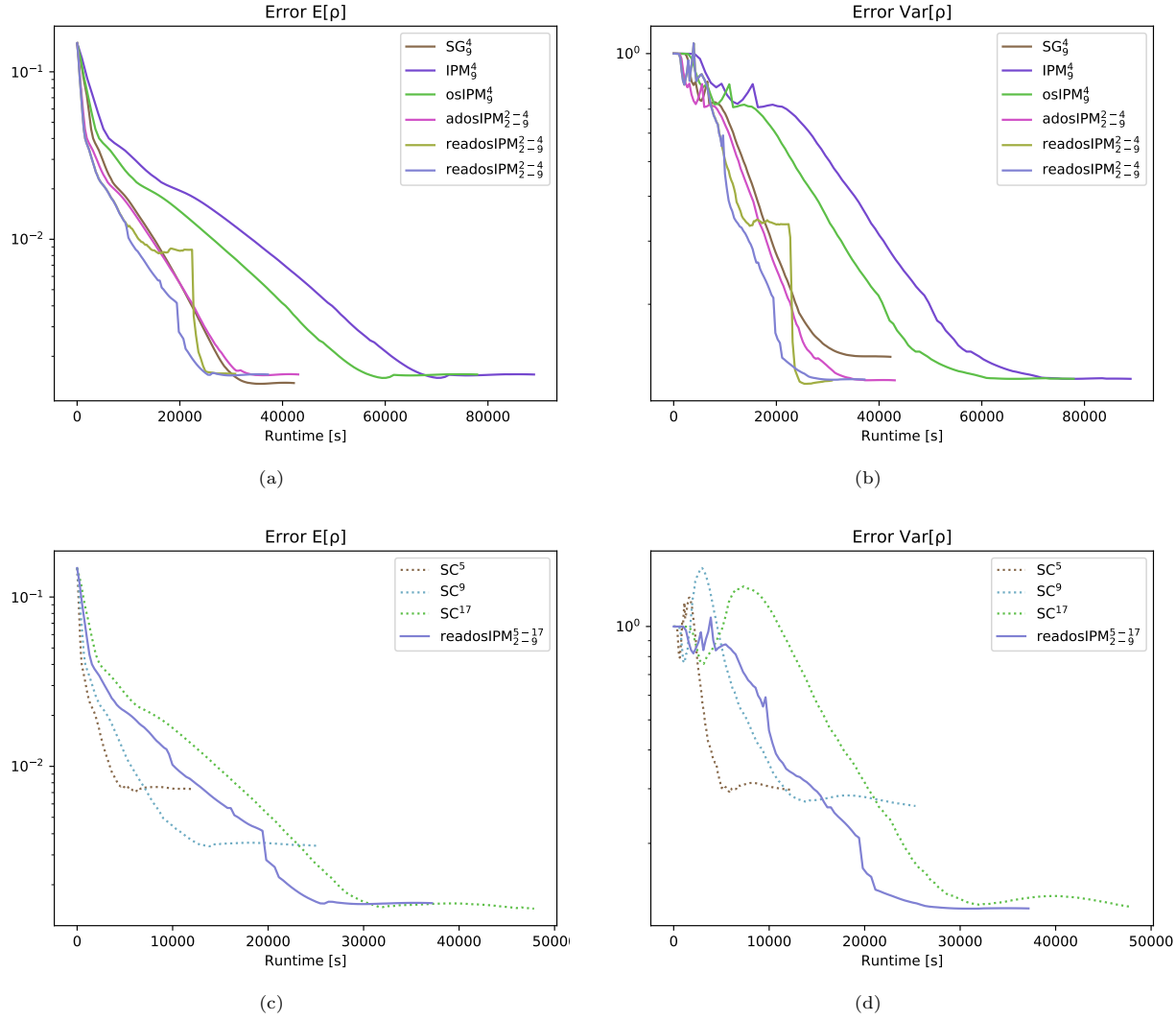


Figure 3: Comparison of the relative L^2-error for the density for IPM related methods (first row) and of the best performing IPM method in comparison with SC (second row). All intrusive methods converge to a residual $\varepsilon = 6 \cdot 10^{-6}$, whereas all non-intrusive methods converge to a residual $\varepsilon = 1 \cdot 10^{-7}$. All computations are performed with with 5 MPI threads.

- If the truncation order is sufficiently small, both quadrature levels yield the same accuracy. We observe this behavior for the expectation value until a truncation order of $N = 5$ and for the variance for $N = 4$. Hence, the variance is more sensitive to aliasing errors.
- Figure 2a reveals that the IPM error of $\mathrm{E}[\rho]$ with 9 quadrature points stagnates at the error level of SC^9, i.e. the aliasing error heavily affects the accuracy. This behavior becomes more dominant when looking at the variance error in Figure 2b. Here, IPM_8^{17} yields a significantly improved result compared to IPM_8^9. Again, the IPM_8^9 result stagnates at the SC^9 accuracy level, which can be reached with IPM when only using four moments (combined with a sufficiently accurate quadrature level). Hence, the number of moments needed for IPM to obtain a certain variance error is significantly smaller than the number of quadrature points needed for SC. This result can be observed throughout the numerical experiments of this work. Especially for high dimensional problems, this potentially decreases the number of unknowns to reach a certain accuracy level significantly.

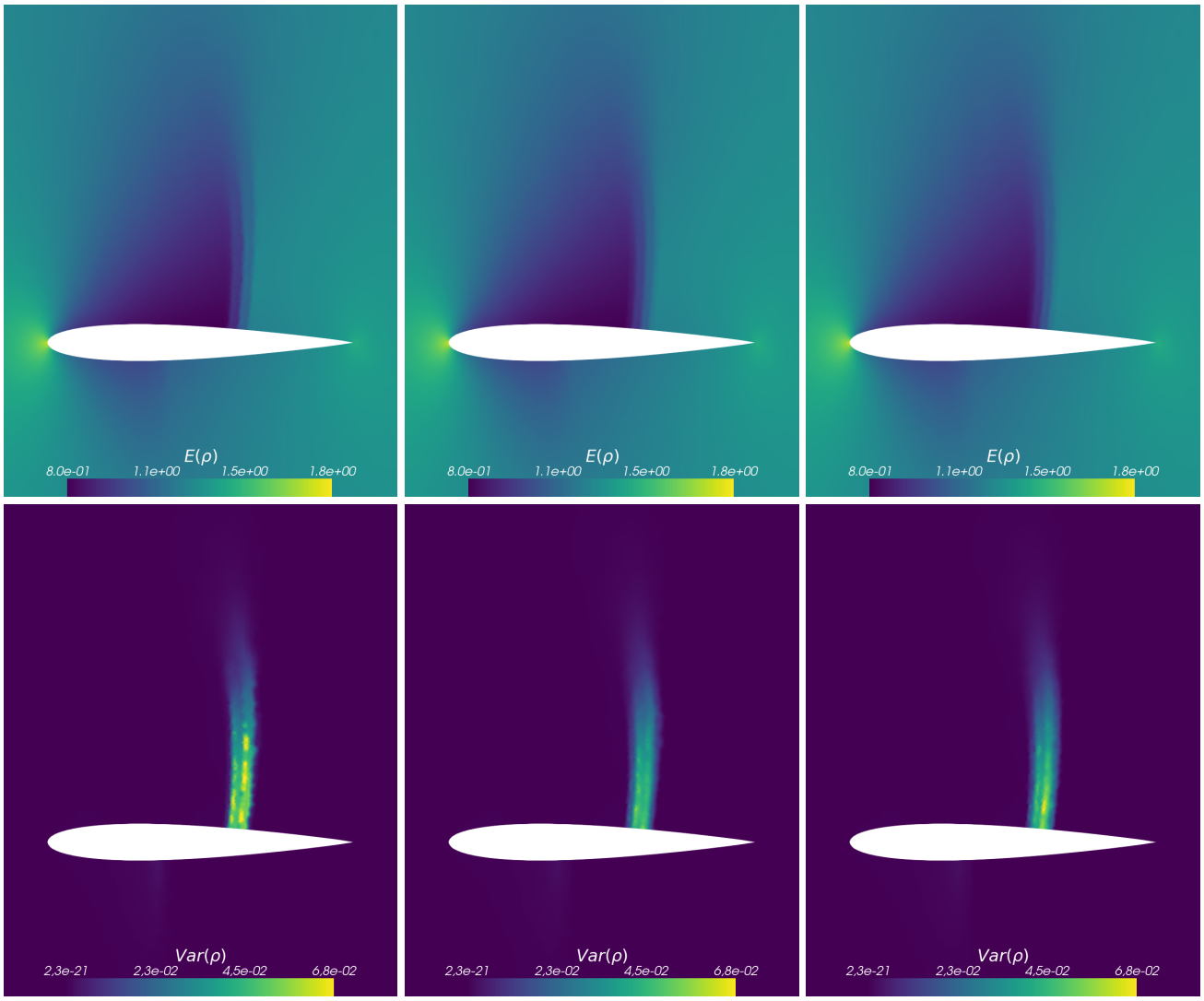


Figure 4: E[ρ] and Var[ρ] computed with SC^5, SG_4, IPM_4 (from left to right). Compare to the reference solution shown in Figure 1.

Let us now compare results obtained with stochastic-Galerkin and IPM as well as its proposed acceleration techniques at a fixed moment order 9. Note, that since IPM generalizes SG, all proposed techniques can be used for stochastic-Galerkin as well. All adaptive methods use gPC polynomials of order 2 to 9 (i.e. 3 to 10 moments). Order 2 uses 5 quadrature points, orders 3 to 6 use 9 quadrature points and orders 8 and 9 use 17 quadrature points. When the smoothness indicator (27) lies below $\delta_- = 2 \cdot 10^{-4}$, the adaptive methods decrease the truncation order, if it lies above $\delta_+ = 2 \cdot 10^{-5}$ the truncation order is increased. The remaining methods have been computed with 17 quadrature points. The iteration in pseudo-time is performed until the expectation value of the density fulfills the stopping criterion (19) with $\varepsilon = 6 \cdot 10^{-6}$, however it can be seen that the error saturates already at a bigger residual.

First, let us mention that the adaptive SG method fails, since it yields negative densities during the iteration. The standard SG method however preserves positivity of mass, energy and pressure. The change of the relative L^2-error during the iteration to the steady state has been recorded in Figure 3a for the expectation

value and in Figure 3b for the variance. When comparing intrusive methods without acceleration techniques as well as SC, the following properties emerge:

- Compared to IPM, stochastic-Galerkin comes at a significantly reduced runtime, meaning that the IPM optimization problem requires a significant computational effort.
- For the expectation value SG_9 shows a smaller error compared to IPM_9, while for the variance, we see the opposite, i.e. IPM yields a better solution approximation than SG.

The proposed acceleration techniques show the following behavior:

- The One-Shot IPM (osIPM) method proposed in Section 4 reduces the runtime while yielding the same error as the classical IPM method.
- When using adaptivity (see Algorithm 3) in combination with the One-Shot idea, the method is denoted by *adaptive One-Shot IPM* (adosIPM). This method reaches the steady state IPM solution with a faster runtime than SG.
- The idea of refinement retardation combined with adosIPM (see Algorithm 4) is denoted by *retardation adosIPM* (readosIPM), which further decreases runtime. Here, we use two different strategies to increase the accuracy: First, we steadily increase the maximal truncation order when the residual approaches zero. To determine residual values for a given set of truncation orders $2, 4, 5$ and 8, we study at which residual level the IPM method reaches a saturated error level for each truncation order. The residual values are then determined to be $6 \cdot 10^{-5}, 3 \cdot 10^{-5}, 2.2 \cdot 10^{-5}$ and $2 \cdot 10^{-5}$. The second, straight forward strategy converges the solution on a low truncation order of 2 to a residual of 10^{-5} and then switches to a maximal truncation order of 9. Strategy 1 is depicted in purple, Strategy 2 is depicted in yellow. It can be seen that both approaches reach the IPM_9 error for the same run time. Hence, we deduce that a naive choice of the refinement retardation strategy suffices to yield a satisfactory behavior.

Let us now compare the results from intrusive methods with those of SC. For every quadrature point, SC iterates the solution until the density lies below a threshold of $\varepsilon = 1 \cdot 10^{-7}$. Recording the error of intrusive methods during the iteration is straight forward. To record the error of SC, we couple all quadrature points after each iteration to evaluate the expectation value and variance, which destroys the black-box nature and results in additional costs. The collocation runtimes, that are depicted on the x-axis in Figures 2, 3c, 3d and 6, are however rescaled to the runtimes that we achieve when running the collocation code in its original, black-box framework without recording the error. Our stochastic-Collocation computations use Clenshaw-Curtis quadrature levels $2, 3$ and 4, i.e. $5, 9$ and 17 quadrature points. The comparison of intrusive methods with non-intrusive methods shows the following:

- SC requires a smaller residual to converge to a steady state solution (we converge the results to $\varepsilon = 1 \cdot 10^{-7}$ compared to $\varepsilon = 6 \cdot 10^{-6}$ for intrusive methods).
- Again, intrusive methods yield improved solutions compared to SC with the same number of unknowns. Actually, the error obtained with 17 unknowns when using SC is comparable with the error obtained with 10 unknowns when using intrusive methods.

Let us finally take a look at the expectation value and variance computed with different methods. All results are depicted for a zoomed view around the airfoil. Figure 4 shows the expectation value (first row) and variance (second row) computed with 5 quadrature points for SC and 5 moments for SG and IPM. One can observe the following

- All methods yield non-physical step-like profiles of the expectation value and variance along the airfoil. This effect can be observed in various settings [10, 11, 12, 13, 14] and stems from Gibb's phenomena in the polynomial description of the uncertainty, either in the gPC polynomials of intrusive methods or the Lagrange polynomials used in collocation techniques.
- The jump position of the intrusive solution profiles capture the exact behavior more accurately.

The readosIPM results as well as the corresponding refinement levels are depicted in Figure 5. One observes that the solution no longer shows the previously observed discontinuous profile and yield a satisfactory agreement with the reference solution depicted in Figure 1. The refinement level shows that away from the airfoil, a refinement level of 0 (i.e. a truncation order of 2) suffices to yield the IPM_9 solution. The region with a high variance requires a refinement level of 7 (truncation order 9).

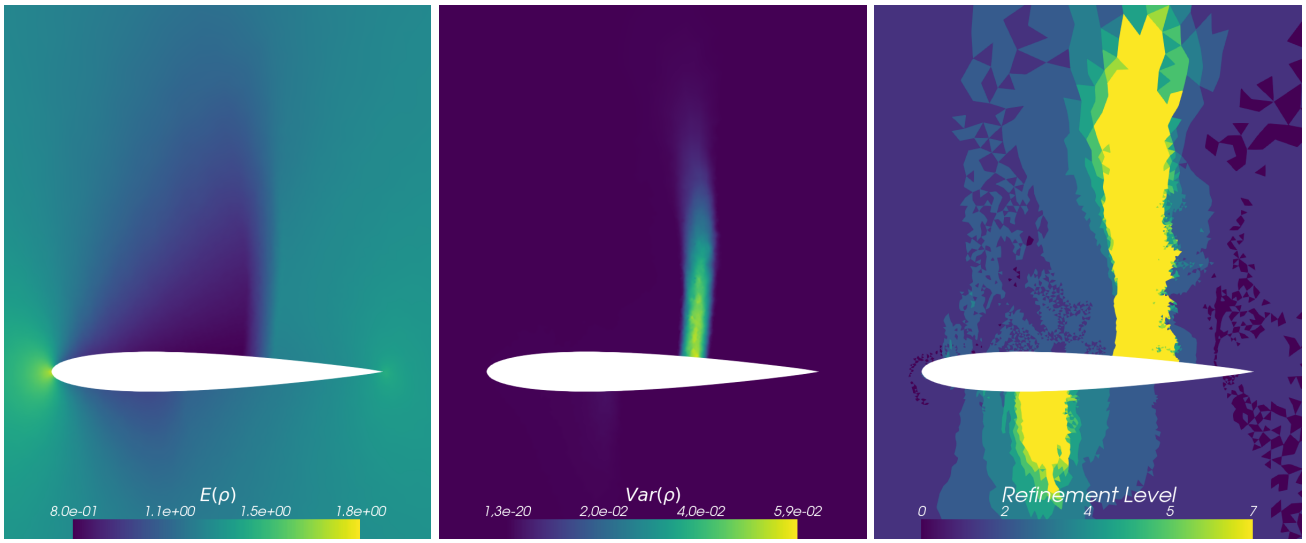


Figure 5: E[ρ], Var[ρ] and refinement level for readosIPM_{2-9}

7.2. 2D Euler equations with a two dimensional uncertainty

In the following, we assume a fixed angle of attack with $\phi = 1.25$ degrees and study the effect of two sources of uncertainties, namely the farfield pressure and Mach number. The farfield pressure is $p \sim U(100\,325, 102\,325)$ Pa and the Mach number is $Ma \sim U(0.775, 0.825)$. Since this problem only has a two-dimensional uncertainty, we use a tensorized quadrature set, which in our experiments proved to be more efficient than a sparse grid quadrature. For SC, we use quadrature sets with 5^2, 9^2 and 17^2 quadrature nodes and compare against SG with moments up to total degree 9 as well as adaptive IPM with refinement retardation (with and without the One-Shot strategy). The IPM method uses moment orders ranging from 1 to 9 adaptively with refinement barriers $\delta_- = 1 \cdot 10^{-4}$ and $\delta_+ = 1 \cdot 10^{-5}$. The refinement retardation allows the truncation order to have a maximal total degree of 1 until a residual of $\varepsilon = 1.5 \cdot 10^{-5}$ and then increases the truncation order by one when the residual is reduced by an amount of $5 \cdot 10^{-6}$. Hence, the maximal truncation order of 9 is reached when the residual is below $\varepsilon = 7 \cdot 10^{-6}$. Again, the refinement strategy let to negative densities for SG resulting in a failure of the method. The error during the iteration has been recorded in Figure 6. To determine the error, we used stochastic-Collocation with a 50 by 50 Gauss-Legendre quadrature rule. As in the one-dimensional NACA testcase, the acceleration techniques lead to a heavily reduced runtime of the IPM method. Furthermore, the error obtained with the intrusive methods requires a total degree of $M = 9$, i.e. $N = 55$ moments to reach the error level of SC with 17 quadrature points per dimension i.e. $17^2 = 289$ collocation points. Note that this effect has also been observed in the one-dimensional case, however now for multi-dimensional problems, the reduced number of unknowns required for intrusive methods to obtain a certain error becomes more apparent. This shows one promising characteristic of intrusive techniques and points to their applicability for higher dimensional problems. In contrast to before, the SG error level is smaller than the IPM error for both, the expectation value and variance. Furthermore, when comparing IPM with and without One-Shot, one can observe that the effect of this acceleration strategy weighs in more heavily than it did for the one-dimensional case. This behavior is expected, since every iterate of the optimization problem becomes more expensive when the dimension is increased. This means, using a method

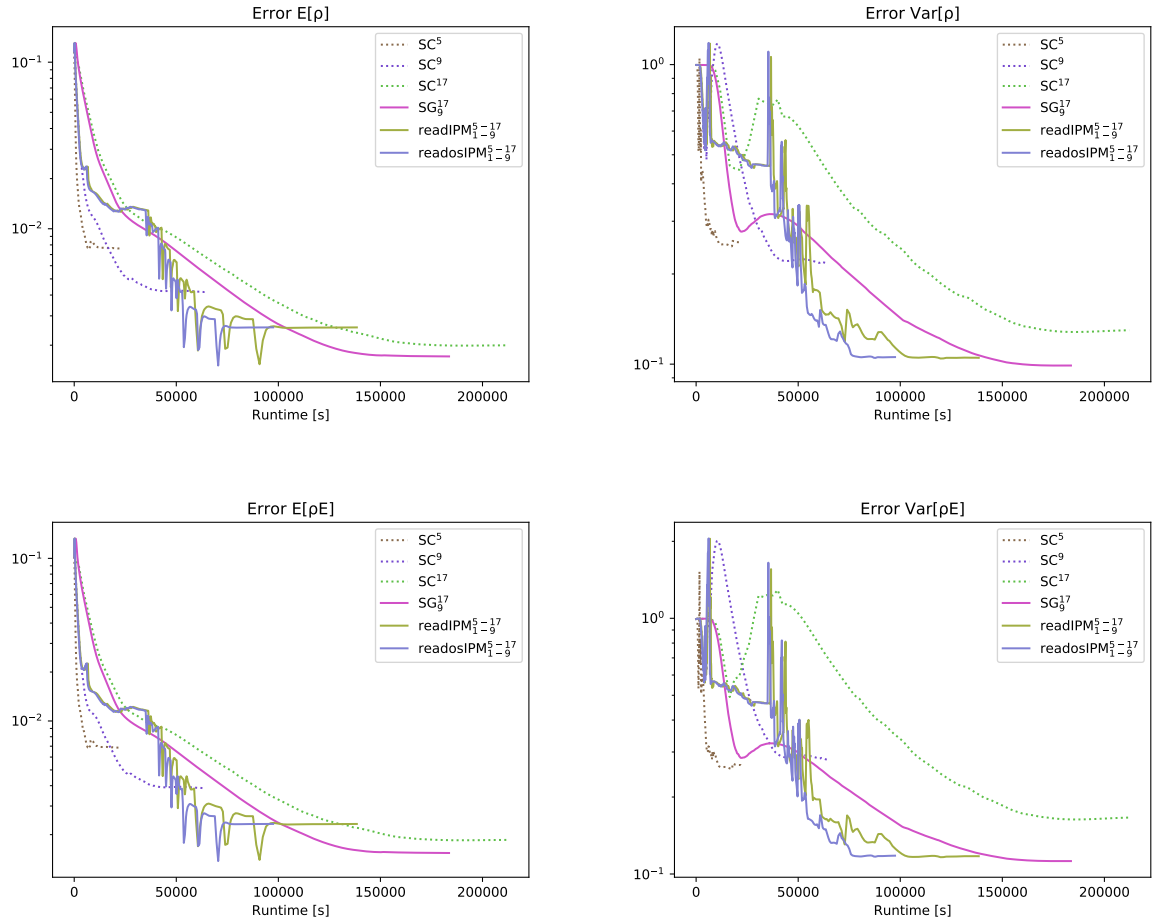


Figure 6: Relative L^2-error with 16 MPI threads for density and energy.

with less iterations for every computation of the dual variables heavily reduces computational costs, which is why we consider the One-Shot strategy to be an effective approach, especially for problems with high uncertain dimension. When looking at the computed IPM expectation value and variance (see Figure 7d and 7e) and comparing the results against the reference solution in Figure 7a and 7b, one can observe that since the uncertainties have a bigger effect on the result than in the one-dimensional case, the IPM solution still shows a step-like profile. However, we are able to capture the main features of the reference solution. The increased effect of the uncertainty can again be seen in Figure 7f, where a refinement level of 8, i.e. a truncation order of 9 is chosen on a larger portion than for the one-dimensional case. Visualizing the variance on a logarithmic scale in Figure 7c further shows that the applied refinement indicators work well and enforce the usage of higher order moments in areas of high variance.

7.3. 2D Euler equations with a three dimensional uncertainty

For the last numerical study we again use the Euler equations, but this time not in the context of the NACA airfoil, but rather with a bend Sod shock tube experiment. The prescribed initial condition for this testcase describes a gas at rest, but with a discontinuity, in density as well as energy in the upper part of the tube. The density $\rho_u = p/(R \cdot T)$ in the upper part is set to $\rho_u = 1.289$ with $p = 101\,325$ Pa, a temperature of 273.15 K and the specific gas constant for dry air $R = 287.87$. The energy ρe_u is set to exactly 1.0. For

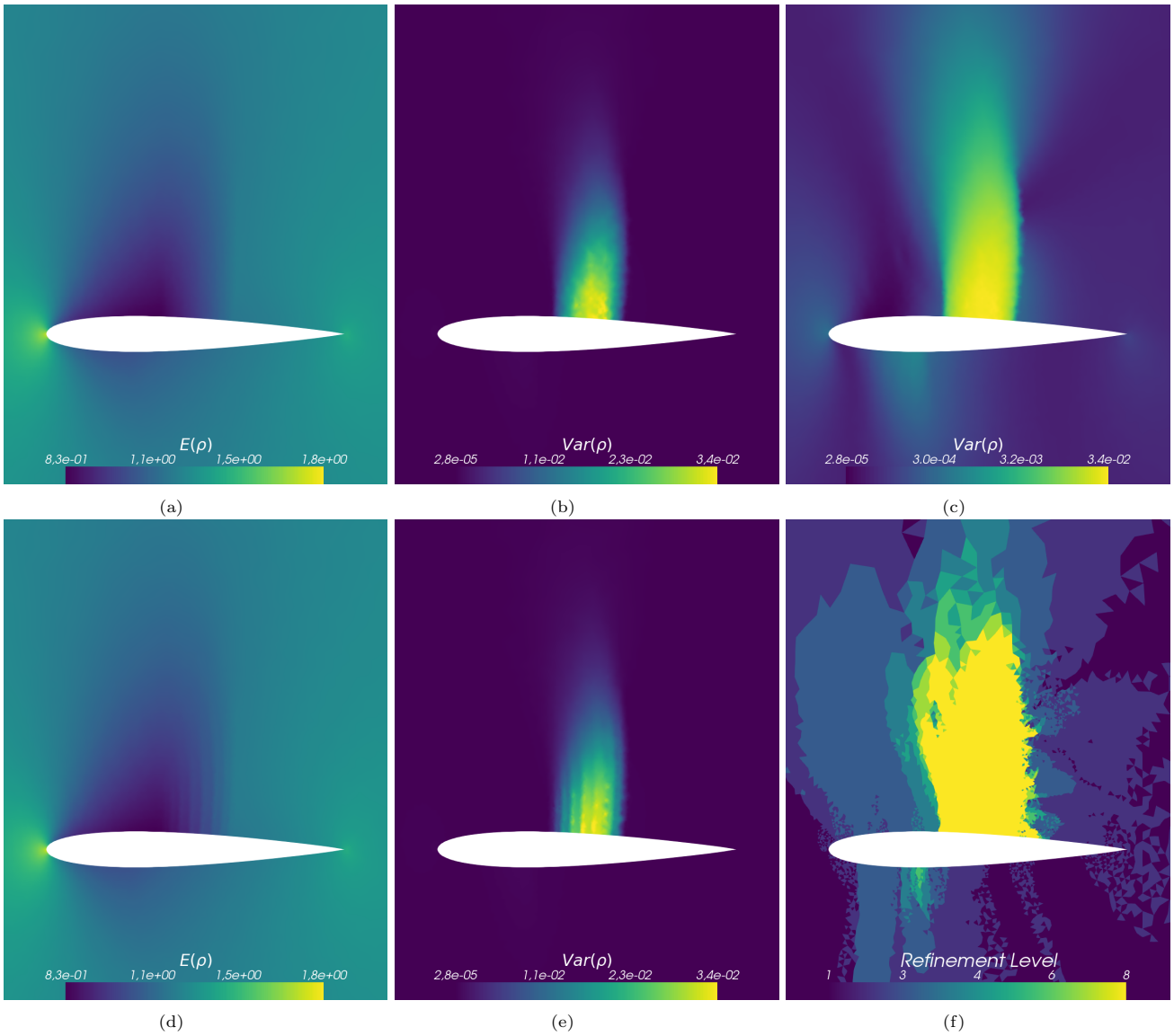


Figure 7: Reference solution (a) E[ρ] and (b) Var[ρ] and with logarithmic scaled (c) Var[ρ]. readosIPM$_{1-9}$ solution for (d) E[ρ] and (e) Var[ρ], with the resulting (f) refinement levels.

the initial conditions in the lower part we set $\rho_l = 0.5 \cdot \rho_u$ and $\rho e_l = 0.3$. The heat capacity ratio γ is set to 1.4 as in the previous studies. We again augment the deterministic case by inflicting uncertainties and again increase the number uncertainties to three. The applied boundary conditions are Euler slip conditions $\boldsymbol{v}^T \boldsymbol{n} = 0$ for the walls and Dirichlet type boundary conditions at the ends of the tube which set to the deterministic initial conditions. The first two uncertainties are the density and energy of the lower part of the shock tube. Here we set $\rho_l \sim U(1.189, 1389)$ and $\rho e_l \sim U(0.2, 0.4)$. The third uncertainty enters through the position of the shock itself and thus $y_{\text{shock}} \sim U(1.0, 1.2)$. Note, that in contrast to the upper testcases, here we are not interested in the steady state of the system, but rather the of time evolution of the expectancy and variance. The computational mesh (see Figure 8c) is an unstructured mesh of 25 458 triangular cells, where the cells are all similar in size, i.e. there are not refined regions as in the NACA testcase. The simulations were run until a time of 2.0s. For the refinement barries values of $\delta_- = 2 \cdot 10^{-2}$ and $\delta_+ = 4 \cdot 10^{-3}$ were set for all adaptive simulations. The corresponding results can be seen in Figure

Sparse grids				
Moment order	1	2	-	-
Number of quadrature points	25	441	-	-
Tensorized grids				
Moment order	1	2	3	4
Number of quadrature points	27	125	125	729

Table 1: Order of moments and corresponding number of quadrature nodes used.

8. The reference solution in Figure 8a and 8b was computed using SC with 50 quadrature points in each stochastic dimension, yielding a total of $50^3 = 125\,000$ quadrature points.
For this testcase we want to compare the results for sparse grids with respect to tensorized grids for higher stochastical dimensions, where both quadrature sets are based on Clenshaw-Curtis nodes. As we made used of adaptivity, the corresponding moment orders and used quadrature points can be seen in Table 1. The following properties emerge:

- It was not possible to obtain a result for sparse grids with moments of order 3 or higher. Any level up to level 11 (72 705 quadrature nodes) of the used Clenshaw-Curtis sparse grids, resulted in a failure of the method during the computation of the mapping from the dual variables to the moment vector.
- Sparse grids required a significantly higher number of quadrature points for the same order of moments in comparison to tensorized grids.
- When using sparse grids with adaptivity the computational cost per quadrature node was observed to be smaller, showing that the nestedness property can potentially increase the methods performance.

Even though the solutions in Figure 8 show the same characteristics and sparse grids are generally well suited and often used in combination with the SC method, our tests reveal that, at least for the Clenshaw-Curtis based sparse grids, these quadrature rules are not well suited to be used in combination with IPM as they require significantly more quadrature points and are only able to generate results for low orders of moments. Tensorized quadrature on the other hand performs similarly well as in the previous testcases and manages to yield a closer solution for the expectation as well as the variance due to the usage of higher moments. Tensorized quadrature rules should thus be preferred for lower stochastical dimensions.

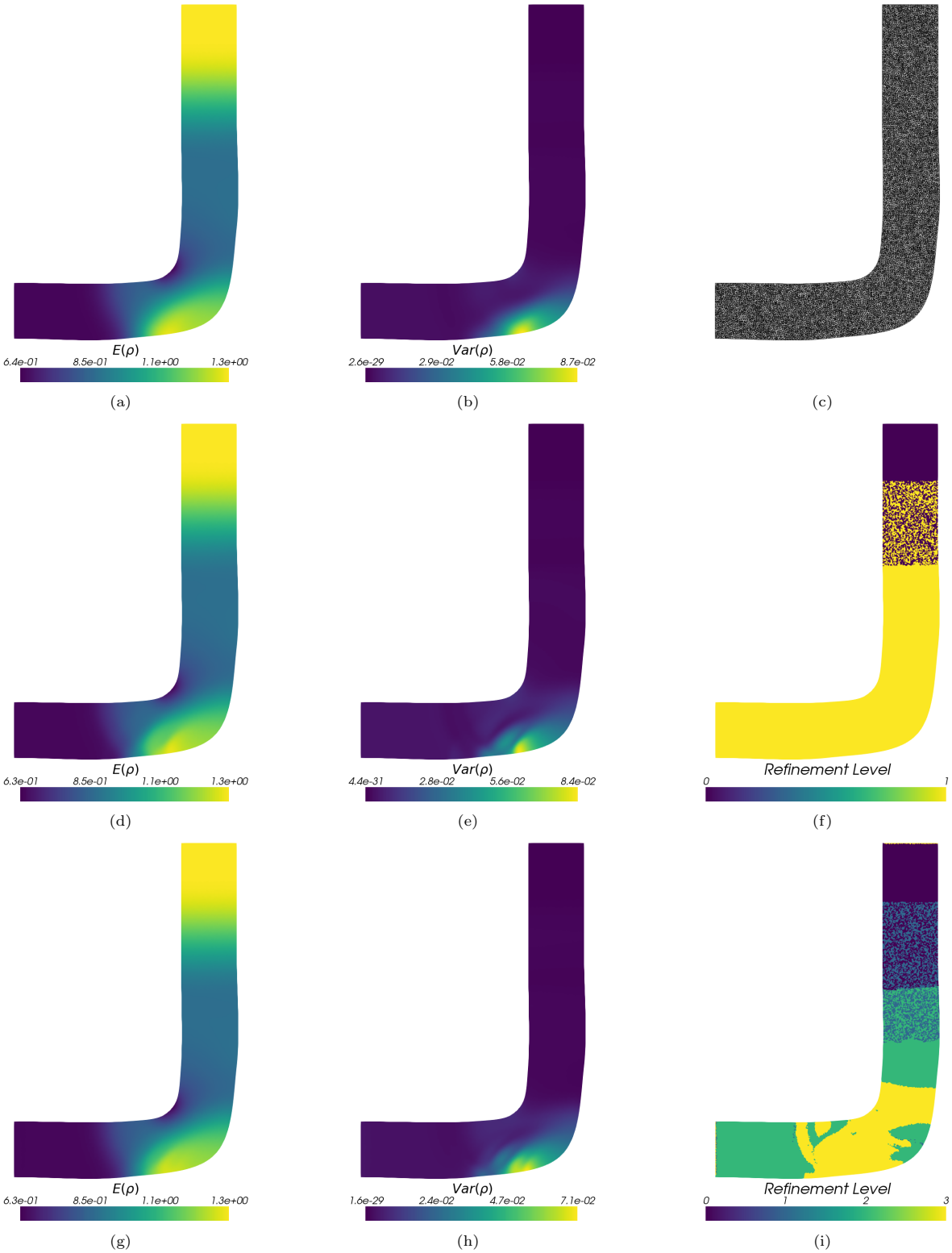


Figure 8: Reference solution E[ρ] and Var[ρ] and used computational mesh (top row) and corresponding adaptive IPM solutions for sparse (middle row) and tensorized grids (bottom row) with the used refinement levels at the last time step

8. Summary and outlook

In this work, we proposed acceleration techniques, which are applicable to the intrusive nature of IPM and SG: We use a One-Shot technique to iterate moments and their corresponding dual variables to their steady state simultaneously. By not fully converging the dual iteration, we can reduce computational costs. Additionally, we make use of adaptivity and propose to keep a low maximal truncation order for most of the iteration to the steady state solution. Since complicated structures in the uncertain dimension only appear on a small portion of the spacial mesh, we are able to heavily reduce computational costs. The effects of the proposed techniques have been demonstrated by comparing results obtained with IPM as well as SG against SC. In our test cases, the intrusive methods yield the same error level as SC for a reduced runtime, especially since intrusive methods require less unknowns to achieve a certain accuracy due to aliasing errors. In higher-dimensional problems, this effect is amplified since the number of unknowns to achieve a certain total degree is asymptotically smaller than the number of quadrature points in a tensorized or sparse grid ((2) instead of $O(M(\log_2(M)^{p-1}))$). Furthermore, we could observe that the required residual at which the solution of intrusive methods reaches a steady state is smaller than for SC. Additionally, the ability to adaptively change the truncation order helps intrusive methods to compete with SC in terms of computational runtime.

In future work, we aim at further accelerating the IPM method by using non-exact Hessian approximations. Similar to the One-Shot idea of not fully converging the dual problem, it seems to be plausible to not spend too much time on computing the Hessian when the moments are not close to a steady state. Hessian approximations that can be interesting are BFGS and sparse BFGS [41, Chapter 6.1], which construct the Hessian from previously computed gradients. Note that this strategy will increase the used memory, since old Hessians or gradients from a certain number of old time steps need to be saved in every spacial cell. Even though the non-intrusive nature of stochastic-Collocation or Monte Carlo methods allows an easy implementation, as shown in this work, it can be helpful to intrusively modify the code in order to fully exploit all acceleration potentials. Synchronizing the time updates of the solution at different quadrature points yields an increased control over the solution during the computation, which can for example be uses to employ adaptive methods. In this case one can switch to a fine quadrature level in a certain spatial cell by for example computing moments with the given coarse set of collocation points. From these moments one can compute an IPM reconstruction, which one can then evaluate at a finer quadrature set. Another example of breaking up the non-intrusive nature of Monte Carlo methods can be found in [42], where the generation of random samples is combined with the sampling after collisions to increase efficiency. Furthermore, we aim at applying the proposed acceleration techniques to SG methods with hyperbolicity limiters [43, 44].

References

[1] Norbert Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[2] Dongbin Xiu and George Em Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[3] Dongbin Xiu and Jan S Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

[4] Ivo Babuška, Fabio Nobile, and Raul Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

[5] GJA Loeven and H Bijl. Probabilistic collocation used in a two-step approach for efficient uncertainty quantification in computational fluid dynamics. *Computer Modeling in Engineering & Sciences*, 36(3):193–212, 2008.

[6] Roger G Ghanem and Pol D Spanos. *Stochastic finite elements: a spectral approach*. Courier Corporation, 2003.

[7] Gaël Poëtte, Bruno Després, and Didier Lucor. Uncertainty quantification for systems of conservation laws. *Journal of Computational Physics*, 228(7):2443–2467, 2009.

[8] Jonas Kusch, Graham W. Alldredge, and Martin Frank. Maximum-principle-satisfying second-order intrusive polynomial moment scheme. *The SMAI journal of computational mathematics*, 5:23–51, 2019.

[9] C. Kristopher Garrett, Cory Hauck, and Judith Hill. Optimization and large scale computation of an entropy-based moment closure. *Journal of Computational Physics*, 302:573–590, 2015.

[10] OP Le Maıtre, OM Knio, HN Najm, and RG Ghanem. Uncertainty propagation using Wiener–Haar expansions. *Journal of computational Physics*, 197(1):28–57, 2004.

[11] Jonas Kusch, Ryan G McClarren, and Martin Frank. Filtered stochastic galerkin methods for hyperbolic equations. *Journal of Computational Physics*, 2019.

[12] Gaël Poëtte. *Contribute to the mathematical and numerical analysis of uncertaint systems of conservation laws and of the linear and nonlinear Boltzmann equation*. PhD thesis, 2019.

[13] Timothy Barth. Non-intrusive uncertainty propagation with error bounds for conservation laws containing discontinuities. In *Uncertainty quantification in computational fluid dynamics*, pages 1–57. Springer, 2013.

[14] Richard P Dwight, Jeroen AS Witteveen, and Hester Bijl. Adaptive uncertainty quantification for computational fluid dynamics. In *Uncertainty Quantification in Computational Fluid Dynamics*, pages 151–191. Springer, 2013.

[15] Per Pettersson, Gianluca Iaccarino, and Jan Nordström. Numerical analysis of the Burgers' equation in the presence of uncertainty. *Journal of Computational Physics*, 228(22):8394–8412, 2009.

[16] Philipp Öffner, Jan Glaubitz, and Hendrik Ranocha. Stability of correction procedure via reconstruction with summation-by-parts operators for Burgers' equation using a polynomial chaos approach. *arXiv preprint arXiv:1703.03561*, 2017.

[17] Xiaoliang Wan and George Em Karniadakis. Multi-element generalized polynomial chaos for arbitrary probability measures. *SIAM Journal on Scientific Computing*, 28(3):901–928, 2006.

[18] Jakob Dürrwächter, Thomas Kuhn, Fabian Meyer, Louisa Schlachter, and Florian Schneider. A hyperbolicity-preserving discontinuous stochastic Galerkin scheme for uncertain hyperbolic systems of equations. *arXiv preprint arXiv:1805.10177*, 2018.

[19] Siddhartha Mishra, Ch Schwab, and Jonas Šukys. Multi-level monte carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions. *Journal of Computational Physics*, 231(8):3365–3388, 2012.

[20] Siddhartha Mishra and Ch Schwab. Sparse tensor multi-level monte carlo finite volume methods for hyperbolic conservation laws with random initial data. *Mathematics of computation*, 81(280):1979–2018, 2012.

[21] Siddhartha Mishra, Nils Henrik Risebro, Christoph Schwab, and Svetlana Tokareva. Numerical solution of scalar conservation laws with random flux functions. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):552–591, 2016.

[22] Julie Tryoen, O Le Maıtre, and Alexandre Ern. Adaptive anisotropic spectral stochastic methods for uncertain scalar conservation laws. *SIAM Journal on Scientific Computing*, 34(5):A2459–A2481, 2012.

[23] Ilja Kröker and Christian Rohde. Finite volume schemes for hyperbolic balance laws with multiplicative noise. *Applied Numerical Mathematics*, 62(4):441–456, 2012.

[24] Jan Giesselmann, Fabian Meyer, and Christian Rohde. A posteriori error analysis for random scalar conservation laws using the Stochastic Galerkin method. *arXiv preprint arXiv:1709.04351*, 2017.

[25] Dongbin Xiu. Fast numerical methods for stochastic computations: a review. *Communications in computational physics*, 5(2-4):242–272, 2009.

[26] AK Alekseev, IM Navon, and ME Zelentsov. The estimation of functional uncertainty using polynomial chaos and adjoint equations. *International Journal for numerical methods in fluids*, 67(3):328–341, 2011.

[27] SB Hazra, V Schulz, J Brezillon, and NR Gauger. Aerodynamic shape optimization using simultaneous pseudo-timestepping. *Journal of Computational Physics*, 204(1):46–64, 2005.

[28] Jonas Kusch, Jannick Wolters, and Martin Frank. *UQCreator*, 2019. `https://git.scc.kit.edu/uqcreator`.

[29] Lloyd N Trefethen. Cubature, approximation, and isotropy in the hypercube. *SIAM Review*, 59(3):469–491, 2017.

[30] S Deshpande. Kinetic theory based new upwind methods for inviscid compressible flows. In *24th Aerospace Sciences Meeting*, page 275, 1986.

[31] Amiram Harten, Peter D Lax, and Bram van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1):35–61, 1983.

[32] Benoît Perthame. Boltzmann type schemes for gas dynamics and the entropy property. *SIAM Journal on Numerical Analysis*, 27(6):1405–1421, 1990.

[33] Benoıt Perthame. Second-order boltzmann schemes for compressible euler equations in one and two space dimensions. *SIAM Journal on Numerical Analysis*, 29(1):1–19, 1992.

[34] Bert J Debusschere, Habib N Najm, Philippe P Pébay, Omar M Knio, Roger G Ghanem, and Olivier P Le Maıtre. Numerical challenges in the use of polynomial chaos representations for stochastic processes. *SIAM journal on scientific computing*, 26(2):698–719, 2004.
[35] Jingwei Hu, Shi Jin, and Dongbin Xiu. A stochastic galerkin method for hamilton–jacobi equations with uncertainty. *SIAM Journal on Scientific Computing*, 37(5):A2246–A2269, 2015.
[36] Jingwei Hu and Shi Jin. A stochastic galerkin method for the boltzmann equation with uncertainty. *Journal of Computational Physics*, 315:150–168, 2016.
[37] J Tryoen, O Le Maitre, M Ndjinga, and A Ern. Instrusive projection methods with upwinding for uncertain non-linear hyperbolic systems. *Preprint*, 2010.
[38] Jakob Dürrwächtera, Fabian Meyerb, Thomas Kuhna, Andrea Becka, Claus-Dieter Munza, and Christian Rohdeb. A high-order stochastic galerkin code for the compressible euler and navier-stokes equations.
[39] Roger Ghanem and S Dham. Stochastic finite element analysis for multiphase flow in heterogeneous porous media. *Transport in porous media*, 32(3):239–262, 1998.
[40] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous galerkin methods. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 112, 2006.
[41] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
[42] Gaël Poëtte. A gpc-intrusive monte-carlo scheme for the resolution of the uncertain linear boltzmann equation. *Journal of Computational Physics*, 385:135–162, 2019.
[43] Kailiang Wu, Huazhong Tang, and Dongbin Xiu. A stochastic galerkin method for first-order quasilinear hyperbolic systems with uncertainty. *Journal of Computational Physics*, 345:224–244, 2017.
[44] Louisa Schlachter and Florian Schneider. A hyperbolicity-preserving stochastic Galerkin approximation for uncertain hyperbolic systems of equations. *Journal of Computational Physics*, 375:80–98, 2018.

Appendix A. Costs of evaluating the numerical flux

In the following, we briefly discuss the number of operations needed when precomputing integrals versus the use of a kinetic flux for Burgers' equation. The stochastic Burgers' equation reads

$$\begin{aligned}\partial_t u + \partial_x \frac{u^2}{2} &= 0,\\ u(t=0,x,\xi) &= u_{IC}(x,\xi).\end{aligned}$$

The scalar random variable ξ is uniformly distributed in the interval $[-1,1]$, hence the gPC basis functions $\boldsymbol{\varphi} = (\varphi_0, \cdots, \varphi_M)^T$ are the Legendre polynomials. Choosing the SG ansatz (3) and testing with the gPC basis functions yields the SG moment system

$$\partial_t \hat{u}_i + \partial_x \frac{1}{2} \sum_{n,m=0}^{M} \hat{u}_n \hat{u}_m \langle \varphi_n \varphi_m \varphi_i \rangle = 0.$$

Defining the matrices $\boldsymbol{C}_i := \langle \boldsymbol{\varphi}\boldsymbol{\varphi}^T \varphi_i \rangle \in \mathbb{R}^{N\times N}$ gives

$$\partial_t \hat{\boldsymbol{u}} + \partial_x \boldsymbol{F}(\hat{\boldsymbol{u}}) = 0$$

with $F_i(\hat{\boldsymbol{u}}) = \frac{1}{2}\hat{\boldsymbol{u}}^T \boldsymbol{C}_i \hat{\boldsymbol{u}}$. Note that $\boldsymbol{C}_i$ can be computed analytically, hence choosing a Lax-Friedrichs flux

$$G_i^{(LF)}(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \frac{1}{4}\left(\hat{\boldsymbol{u}}_\ell^T \boldsymbol{C}_i \hat{\boldsymbol{u}}_\ell + \hat{\boldsymbol{u}}_r^T \boldsymbol{C}_i \hat{\boldsymbol{u}}_r\right) - \frac{\Delta x}{2\Delta t}(\hat{\boldsymbol{u}}_r - \hat{\boldsymbol{u}}_\ell)_i \tag{A.1}$$

requires no integral evaluations. Recall, that the numerical flux choice made in this work gives

$$\boldsymbol{G}(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \sum_{k=1}^{Q} w_k g(\mathcal{U}(\hat{\boldsymbol{u}}_\ell;\xi_k), \mathcal{U}(\hat{\boldsymbol{u}}_r;\xi_k))\boldsymbol{\varphi}(\xi_k) f_\Xi(\xi_k), \tag{A.2}$$

where $\mathcal{U}$ is the SG ansatz (3). When the chosen deterministic flux g is Lax-Friedrichs, the order of the polynomials inside the sum is $3M = 3(N-1)$. Choosing a Gauss-Lobatto quadrature rule, $Q = \frac{3}{2}N - 1$ quadrature points suffice for an exact computation of the numerical flux. Indeed, with this choice of

quadrature points, the numerical fluxes (A.1) and (A.2) are equivalent. Counting the number of operations, one observes that our choice of the numerical flux (A.2) uses $O(N^2)$ operations whereas (A.1) requires $O(N^3)$ operations: When computing and storing the values in a matrix $\boldsymbol{A} \in \mathbb{R}^{Q\times N}$ with entries $a_{ki} = \varphi_i(\xi_k)$ before running the program, the numerical flux (A.2) can be split into two parts. First, we determine the SG solution at all quadrature points, i.e. we compute $\boldsymbol{u}^{(\ell)} := \boldsymbol{A}\hat{\boldsymbol{u}}_\ell$ and $\boldsymbol{u}^{(r)} := \boldsymbol{A}\hat{\boldsymbol{u}}_r$ which requires $O(N \cdot Q)$ operations. These solution values are then used to compute the numerical flux

$$G_i(\hat{\boldsymbol{u}}_\ell, \hat{\boldsymbol{u}}_r) = \sum_{k=1}^{Q} w_k g(u_k^{(\ell)}, u_k^{(r)}) a_{ki} f_\Xi(\xi_k),$$

which again requires $O(N\cdot Q)$ operations, i.e. the costs are $O(N^2)$. The evaluation of (A.1) however requires $O(N^3)$ operations.

Appendix B. IPM for the 2D Euler equations

In the following, we provide details on the implementation of IPM for the 2D Euler equations. For ease of presentation, we denote the momentum by $m_1 := \rho v_1$ and $m_2 := \rho v_2$ and the energy by $E := \rho e$. Then, the vector of conserved variables is $\boldsymbol{u} = (\rho, m_1, m_2, E)^T$. The entropy used is

$$s(\boldsymbol{u}) = -\rho \ln\left(\rho^{-\gamma}\left(E - \frac{m_1^2 + m_2^2}{2\rho}\right)\right).$$

Now the gradient of the entropy $\nabla_{\boldsymbol{u}} s$ has the components

$$\begin{aligned}
\frac{\partial s}{\partial \rho} &= -\ln\left(\rho^{-\gamma}\left(E - \frac{m_1^2 + m_2^2}{2\rho}\right)\right) + \frac{m_1^2 + m_2^2}{-2\rho E + m_1^2 + m_2^2} + \gamma,\\
\frac{\partial s}{\partial m_i} &= -\frac{2\rho m_i}{-2\rho E + m_1^2 + m_2^2},\\
\frac{\partial s}{\partial E} &= -\frac{1}{\rho}\left(E - \frac{m_1^2 + m_2^2}{2\rho}\right).
\end{aligned}$$

To compute $\boldsymbol{u}_s(\boldsymbol{\Lambda}) = (\nabla_{\boldsymbol{u}} s)^{-1}(\boldsymbol{\Lambda})$, we set $\boldsymbol{\Lambda} = \nabla_{\boldsymbol{u}} s(\boldsymbol{u})$ and rearrange with respect to $\boldsymbol{u}$. Let us define

$$\alpha(\boldsymbol{\Lambda}) := \exp\left(\frac{\Lambda_2^2 + \Lambda_3^2 - 2\Lambda_1\Lambda_4 - 2\Lambda_4\gamma}{2\Lambda_4(1-\gamma)}\right) \cdot (-\Lambda_4)^{\frac{1}{1-\gamma}}$$

Then the solution ansatz $\boldsymbol{u}_s$ is given by

$$\begin{aligned}
&\rho(\boldsymbol{\Lambda}) = \alpha(\boldsymbol{\Lambda}), \qquad m_1(\boldsymbol{\Lambda}) = -\frac{\Lambda_2\alpha(\boldsymbol{\Lambda})}{\Lambda_4}, \qquad m_2(\boldsymbol{\Lambda}) = -\frac{\Lambda_3\alpha(\boldsymbol{\Lambda})}{\Lambda_4},\\
&E(\boldsymbol{\Lambda}) = -\frac{\alpha(\boldsymbol{\Lambda})(-\Lambda_2^2 - \Lambda_3^2 + 2\Lambda_4)}{2\Lambda_4^2}.
\end{aligned}$$