A semi-intrusive Code Framework for Uncertainty Quantification

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. Intrusive methods require the implementation of a new code and yield

Keywords: conservation laws, hyperbolic, intrusive, UQ, IPM, SG, Collocation

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 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.

A general hyperbolic set of equations with random initial data can be written as

$$\partial_t \boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi}) + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}(t,\boldsymbol{x},\boldsymbol{\xi})) = \boldsymbol{0}, \tag{1a}$$

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

where the solution $u \in \mathbb{R}^p$ depends on time $t \in \mathbb{R}^+$, spatial position $\boldsymbol{x} \in D \subset \mathbb{R}^d$ as well as a vector of random variables $\boldsymbol{\xi} \in \Theta \subset \mathbb{R}^s$ with given probability density functions $f_{\Xi,i}(\xi_i)$ for $i = 1, \cdots, s$. The physical flux is given by $\boldsymbol{F} : \mathbb{R}^p \to \mathbb{R}^p$. To simplify notation, we assume that 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 of the solution, 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 $\langle \cdot \rangle := \int_\Theta \cdot \prod_{i=1}^s f_{\Xi,i}(\xi_i) d\xi_1 \cdots d\xi_s$. More generally, one is interested in determining the moments of the solution for a given set of orthonormal basis functions $\varphi_i : \Theta \to \mathbb{R}$ such that for the multi-index $i = (i_1, \cdots, i_s)$ we have $|i| \leq N$. The moments are then given by $\hat{\boldsymbol{u}}_{\boldsymbol{i}} := \langle \boldsymbol{u} \varphi_i \rangle$.

Numerical methods for approximating the moment $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ can be divided into intrusive and non-intrusive methods. A popular non-intrusive method is the stochastic-Collocation (SC) method, which computes the moments with the help of a numerical quadrature rule: For a given set of N_q quadrature weights w_k and

quadrature points $\boldsymbol{\xi}_k$, the moments are approximated by

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

Since the solution at a fixed quadrature point can be computed by a standard deterministic solver, the SC method does not require a big implementation effort. Furthermore, it is embarrassingly parallel, since the required computations can be carried out in parallel on different cores. A downside of collocation methods are aliasing effects, which stem from the inexact approximation of the integral. Furthermore, collocation methods require more runs of the deterministic solver than intrusive methods [1, 2]. Despite their easy implementation, collocation methods can become cumbersome for unsteady problems, since in this case, the solution needs to written out at all quadrature points in every time step to compute the time evolution of the moments. This problem does not occur in intrusive methods, since the computation is carried out on the moments, i.e. the time evolution can be recorded during the computation. Also for steady problems, the fact that the computation is carried out on the moments provided the ability to compute the stochastic residual, which indicates when to stop the iteration towards the steady state solution. However intrusive methods are in general more difficult to implement and come with higher numerical costs. The main idea of these methods is to derive a system of equations for the moments and implementing a numerical solver for this system: Testing the initial problem (1) with φ_i for $|i| \leq M$ yields

$$\partial_t \hat{\boldsymbol{u}}_i(t, \boldsymbol{x}) + \nabla \cdot \langle \boldsymbol{F}(\boldsymbol{u}(t, \boldsymbol{x}, \cdot))\varphi_i \rangle = \boldsymbol{0}, \tag{2a}$$

$$\hat{\boldsymbol{u}}_{\boldsymbol{i}}(t = 0, \boldsymbol{x}) = \langle \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x}, \cdot)\varphi_i \rangle. \tag{2b}$$

To obtain a closed set of equations, one needs to derive a closure $\mathcal{U}$ such that

$$\boldsymbol{u}(t, \boldsymbol{x}, \boldsymbol{\xi}) \approx \mathcal{U}(\hat{\boldsymbol{u}}_0, \cdots, \hat{\boldsymbol{u}}_N; \xi).$$

A commonly used closure is the stochastic-Galerkin, which represents the solution by a polynomial:

$$\mathcal{U}_{\mathrm{SG}}(\hat{\boldsymbol{u}}_0, \cdots, \hat{\boldsymbol{u}}_N; \xi) := \sum_{i=0}^{N} \hat{\boldsymbol{u}}_i \varphi_i(\boldsymbol{\xi}).$$

When using the stochastic-Galerkin method to close (2), the resulting moment system is not necessarily hyperbolic, meaning that it cannot be solved with standard methods. An approach to derive a closure, which ensures hyperbolicity is the Intrusive Polynomial Moment (IPM) closure, which has been presented in [3]: The closure is given by a constraint optimization problem. For a given convex entropy $s : \mathbb{R}^p \to \mathbb{R}$ for the original problem (1), this optimization problem is

$$\mathcal{U}_{ME}(\hat{\boldsymbol{u}}) = \arg\min_{\boldsymbol{u}} \langle s(\boldsymbol{u}) \rangle \quad \text{subject to } \hat{\boldsymbol{u}}_i = \langle \boldsymbol{u}\varphi_i \rangle \text{ for } i = 0, \cdots, N. \tag{3}$$

This problem can be rewritten as

$$\hat{\boldsymbol{\lambda}}(\hat{\boldsymbol{u}}) := \underset{\boldsymbol{\lambda} \in \mathbb{R}^{methods s \cdot (N+1)}}{\arg\min} \left\{ \langle s_*(\sum_{i=0}^{N} \boldsymbol{\lambda}_i \varphi_i) \rangle - \sum_{i=0}^{N} \boldsymbol{\lambda}_i^T \hat{\boldsymbol{u}}_i \right\}, \tag{4}$$

where $s_* : \mathbb{R}^p \to \mathbb{R}$ is the Legendre transformation of s, and $\hat{\boldsymbol{\lambda}}$ are called the dual variables. The solution to (3) is then given by

$$\mathcal{U}_{ME}(\hat{\boldsymbol{u}}) = (s_{\hat{\boldsymbol{u}}})^{-1} (\hat{\boldsymbol{\lambda}}(\boldsymbol{u})^T \boldsymbol{\varphi}). \tag{5}$$

Plugging the derived closure into the moment system (2), one obtains the IPM moment system

$$\partial_t \hat{\boldsymbol{u}}_i(t, \boldsymbol{x}) + \nabla \cdot \langle \boldsymbol{F}(\mathcal{U}_{ME}(\hat{\boldsymbol{u}}))\varphi_i \rangle = \boldsymbol{0}, \tag{6a}$$

$$\hat{\boldsymbol{u}}_{\boldsymbol{i}}(t = 0, \boldsymbol{x}) = \langle \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{x}, \cdot)\varphi_i \rangle. \tag{6b}$$

The IPM method has several advantages: Choosing the entropy $s(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$ yields the stochastic-Galerkin method, i.e. IPM generalizes different intrusive method. Furthermore, at least for scalar problems, IPM is significantly less oscillatory compared to SG [4]. Also, as discussed in [3], when choosing $s(\boldsymbol{u})$ to be a physically correct entropy of the deterministic problem, the IPM solution dissipates the entropy

$$S(\hat{\boldsymbol{u}}) := \langle s(\mathcal{U}_{ME}(\hat{\boldsymbol{u}}))\rangle,$$

i.e. the IPM method yields a physically correct entropy solution. This again underlines a weakness of stochastic-Galerkin: If $s(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$ is not a correct entropy of the original problem, the SG method can lead to non-physical solution, which can then cause a failure of the method. The main weakness of the IPM method is its run time, since it requires the repeated evaluation of (5), which involves the computation of the optimization problem (4). Hence, the desirable costs of IPM (and intrusive methods in general) require a significantly increased run time. A further advantages which we will not use in this work is the possibility of using adaptivity [5], i.e. refining the stochastic space at certain spatial position and time steps in which complex structures such as discontinuities occur. The main task here is to find an adequate refinement indicator [5, 6].

In this paper, we present a semi-intrusive code framework, which facilitates the task of implementing general intrusive methods. The framework only requires the numerical flux of the deterministic problem. If IPM is used, the entropy of the deterministic problem needs to be provided. We thereby provide the ability to recycle existing implementations of deterministic solvers. Furthermore, we investigate intrusive methods for steady problems and compare them to collocation methods. The steady setting provides different opportunities to take advantage of features of intrusive methods:

- Apply intrusive methods as a post-processing step for collocation methods: We converge the moments of the solution to a steady state with an inaccurate, but cheap collocation method and then use the resulting collocation moments as starting values for an expensive but accurate intrusive method, which we then again converge to steady state.
- Since the moments during the iteration process are inaccurate, i.e. they are not the steady state solution, we propose to not fully converge the dual iteration, which computes (4).

The paper is structured as follows: After the introduction, we discuss the numerical discretization as well as the implementation and structure of the semi-intrusive framework in section 2. Section 3 discusses the IPM acceleration with a non-intrusive method and in section 4. A comparison of results computed with the presented methods is given in 5, followed by a summary and outlook in section 6.

2. Discretization and code framework

2.1. Discretization

Omitting initial conditions, we can write the IPM moment system (6) as

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

with the flux $\boldsymbol{F} : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$, $\boldsymbol{F}(\boldsymbol{u}) = \langle f(u_{ME}(\hat{\Lambda}(\boldsymbol{u})))\boldsymbol{\varphi}\rangle$ depending on the dual state

$$\hat{\Lambda}(\boldsymbol{u}) = \hat{\boldsymbol{\lambda}}(\boldsymbol{u})^T\boldsymbol{\varphi}.$$

For efficiency of exposition, we sometimes omit the dependence on $\boldsymbol{u}$. The IPM system is hyperbolic, so it is naturally solved by a finite-volume method. First we discretize the spatial domain into cells. The discrete unknowns are chosen to be the spatial averages over each cell at time t_n, given by

$$u_{ij}^n \simeq \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} u_i(t_n, x)dx.$$

If a moment vector in cell j at time t_n is denoted as $\boldsymbol{u}_j^n = (u_{0j}^n, \cdots, u_{Nj}^n)^T \in \mathbb{R}^{N+1}$, the finite-volume scheme can be written in conservative form with the numerical flux $\boldsymbol{G} : \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ as

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{G}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) - \boldsymbol{G}(\boldsymbol{u}_{j-1}^n, \boldsymbol{u}_j^n)\right) \tag{7}$$

for $j = 1, \cdots, N_x$ and $n = 0, \cdots, N_t$, where N_x is the number of spatial cells and N_t is the number of time steps. The numerical flux is assumed to be consistent, i.e., that $\boldsymbol{G}(\boldsymbol{u}, \boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{u})$. To ensure stability, a CFL condition has to be derived by investigating the eigenvalues of $\nabla \boldsymbol{F}$.

When a consistent numerical flux $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $g = g(u_\ell, u_r)$ is available for the deterministic problem (**??**), then for the IPM system we can simply take

$$\boldsymbol{G}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) = \langle g(u_{ME}(\hat{\Lambda}(\boldsymbol{u}_j^n)), u_{ME}(\hat{\Lambda}(\boldsymbol{u}_{j+1}^n)))\boldsymbol{\varphi}\rangle.$$

This choice of the numerical flux is a common choice in kinetic theory and is called kinetic flux. The time update of the moment vector now becomes

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \frac{\Delta t}{\Delta x}\left(\langle g(u_{ME}(\hat{\Lambda}_j^n), u_{ME}(\hat{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle g(u_{ME}(\hat{\Lambda}_{j-1}^n), u_{ME}(\hat{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right), \tag{8}$$

where $\hat{\Lambda}_j^n := \hat{\Lambda}(\boldsymbol{u}_j^n)$ for all j. Note that the computation of $\hat{\Lambda}_j^n$ requires solving the dual problem (4) for the moment vector $\boldsymbol{u}_j^n$.

Unfortunately (8) cannot be implemented because the dual problem cannot be solved exactly.[1] Instead, it must be solved numerically, for example with Newton's method. The stopping criterion for the numerical optimizer ensures that the approximate multiplier vector it returns, which we denote $\overline{\boldsymbol{\lambda}}_j^n \in \mathbb{R}^{N+1}$ for the moment vector $\boldsymbol{u}_j^n$, satisfies the stopping criterion

$$\left\| \boldsymbol{u}_j^n - \left\langle u_{ME}\left(\left(\overline{\boldsymbol{\lambda}}_j^n\right)^T \boldsymbol{\varphi}\right)\boldsymbol{\varphi}\right\rangle \right\| < \tau. \tag{9}$$

This is derived from the first-order necessary conditions for the dual problem. Once the numerical optimizer finds such a $\overline{\boldsymbol{\lambda}}_j^n$, the corresponding dual state $\overline{\Lambda}_j^n := \left(\overline{\boldsymbol{\lambda}}_j^n\right)^T \boldsymbol{\varphi} \in \mathbb{P}(\Theta)$ can be used in (8) for the unknown $\hat{\Lambda}_j^n$. This gives Algorithm 1.

Algorithm 1: IPM for Uncertainty Quantification

for $j = 0$ to $N_x + 1$ **do**
  $\boldsymbol{u}_j^0 = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \langle u_0(x, \cdot)\boldsymbol{\varphi}\rangle dx$
end for
for $n = 0$ to N_t **do**
  for $j = 0$ to $N_x + 1$ **do**
    $\overline{\boldsymbol{\lambda}}_j^n \approx \arg\min_{\boldsymbol{\lambda}} \left(\langle s_*(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\rangle - \boldsymbol{\lambda}^T \boldsymbol{u}_j^n\right)$ such that (9) holds
    $\overline{\Lambda}_j^n = \left(\overline{\boldsymbol{\lambda}}_j^n\right)^T \boldsymbol{\varphi}$
  end for
  for $j = 1$ to N_x **do**
    $\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \frac{\Delta t}{\Delta x}\left(\langle g(u_{ME}(\overline{\Lambda}_j^n), u_{ME}(\overline{\Lambda}_{j+1}^n))\boldsymbol{\varphi}\rangle - \langle g(u_{ME}(\overline{\Lambda}_{j-1}^n), u_{ME}(\overline{\Lambda}_j^n))\boldsymbol{\varphi}\rangle\right)$
  end for
end for

[1] Equation (8) also includes integral evaluations which cannot be computed in closed form. Their approximation by numerical quadrature, however, does not play a role in the realizability problems we discuss below.

3. Collocation accelerated IPM for steady state problems

4. One-shot IPM

For classical IPM, the iteration scheme for the moment vectors is

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{c}\left(\boldsymbol{\lambda}(\boldsymbol{u}_{j-1}^n), \boldsymbol{\lambda}(\boldsymbol{u}_j^n), \boldsymbol{\lambda}(\boldsymbol{u}_{j+1}^n)\right), \tag{10}$$

where $\boldsymbol{c} : \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ is given by

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

The map from the moment vector to the dual variables is given by the exact fix point of $\boldsymbol{d} : \mathbb{R}^{N+1} \times \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$,

$$\boldsymbol{d}(\boldsymbol{\lambda}, \boldsymbol{u}) := \boldsymbol{\lambda} - \boldsymbol{B} \cdot \left(\langle u(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\boldsymbol{\varphi}\rangle) - \boldsymbol{u}\right),$$

where $\boldsymbol{B}$ is a preconditioner, which is used in the numerical scheme to ensure convergence of the fix point iteration (which we call dual iteration in the following)

$$\boldsymbol{\lambda}_j^{m+1} = \boldsymbol{d}(\boldsymbol{\lambda}_j^m, \boldsymbol{u}_j^n). \tag{11}$$

Note that here, $\boldsymbol{u}_j^n$ is a fixed parameter. By letting the dual iteration converge, i.e. $m \to \infty$, we obtain $\lim_{m\to\infty} \boldsymbol{d}(\boldsymbol{\lambda}_j^m, \boldsymbol{u}_j^n) =: \boldsymbol{\lambda}_j^n \equiv \boldsymbol{\lambda}(\boldsymbol{u}_j^n)$, which are called the dual variables of $\boldsymbol{u}_j^n$. Hence, the numerical scheme for IPM converges the dual iteration (11) to compute the mapping from the moment vectors to their dual variables and then performs one iteration of (10) to obtain the time update of the moment vector. Note that this can become extremely expensive, since in each iteration of the moment vectors, we fully converge the dual iteration for all spatial cells.

To simplify notation, we leave out the dependency on $\boldsymbol{\lambda}$ when writing the moment scheme in the following: Hence with

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

we can rewrite the moment iteration as

$$\boldsymbol{u}_j^{n+1} = \tilde{\boldsymbol{c}}\left(\boldsymbol{u}_{j-1}^n, \boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n\right). \tag{12}$$

For steady problems, we assume that the IPM scheme converges to a fix point, i.e. we must have that $\rho(\tilde{\boldsymbol{c}}_{\boldsymbol{u}}) < 1$. The main idea of *One-Shot IPM* is to not fully converge the dual iteration, since the moment vectors are not yet converged to the exact steady solution. So if we successively perform one update of the moment iteration and one update of the dual iteration, we obtain

$$\boldsymbol{\lambda}_j^{n+1} = \boldsymbol{d}(\boldsymbol{\lambda}_j^n, \boldsymbol{u}_j^n) \quad \text{for all j} \tag{13a}$$

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{c}\left(\boldsymbol{\lambda}_{j-1}^{n+1}, \boldsymbol{\lambda}_j^{n+1}, \boldsymbol{\lambda}_{j+1}^{n+1}\right). \tag{13b}$$

In the following, we study convergence of this scheme. For this we take an approach commonly chosen to prove the local convergence properties of Newton's method: In Theorem 1, we show that the iteration function is contractive at its fix point and conclude in Theorem 2 that this yields local convergence.

Theorem 1. *Assume that the classical IPM iteration* (12) *is contractive at its fix point* $\boldsymbol{u}^*$. *Then the Jacobi matrix* $\boldsymbol{J}$ *of the One-Shot IPM iteration* (13) *has a spectral radius* $\rho(\boldsymbol{J}) < 1$ *at the fix point* $(\boldsymbol{\lambda}^*, \boldsymbol{u}^*)$ *if the preconditioner* $\boldsymbol{B} = \langle u'(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-1}$, *i.e. the Hessian of the dual problem is used.*

Proof. We first point out that since we assume that the classical IPM scheme is contractive at its fix point, we have $\rho(\tilde{\boldsymbol{c}}_{\boldsymbol{u}}(\boldsymbol{u}^*)) < 1$ with $\tilde{\boldsymbol{c}}_{\boldsymbol{u}} \in \mathbb{R}^{(N+1)\cdot N_x \times (N+1)\cdot N_x}$ defined by

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

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

$$\partial_{\boldsymbol{u}_\ell}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{\lambda}_\ell}\frac{\partial \boldsymbol{\lambda}(\boldsymbol{u}^*_{j-1})}{\partial \boldsymbol{u}},\ \partial_{\boldsymbol{u}_c}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{\lambda}_c}\frac{\partial \boldsymbol{\lambda}(\boldsymbol{u}^*_j)}{\partial \boldsymbol{u}},\ \partial_{\boldsymbol{u}_r}\tilde{\boldsymbol{c}}_j = \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{\lambda}_r}\frac{\partial \boldsymbol{\lambda}(\boldsymbol{u}^*_{j+1})}{\partial \boldsymbol{u}}. \tag{14}$$

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

$$\boldsymbol{u} = \langle u(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\boldsymbol{\varphi}\rangle =: \boldsymbol{h}(\boldsymbol{\lambda}), \tag{15}$$

which is why we have the mapping $\boldsymbol{u} : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$, $\boldsymbol{u}(\boldsymbol{\lambda}) = \boldsymbol{h}(\boldsymbol{\lambda})$. 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

$$\boldsymbol{\lambda} = \boldsymbol{h}^{-1}(\boldsymbol{u}(\boldsymbol{\lambda})) \tag{16}$$

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

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

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

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

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 $\boldsymbol{h}^{-1}(\boldsymbol{u}(\boldsymbol{\lambda}))$ as $\boldsymbol{\lambda}(\boldsymbol{u})$ and simply computing the term $\frac{\partial \boldsymbol{u}(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}$ by differentiating (15) w.r.t. $\boldsymbol{\lambda}$, one obtains

$$\partial_{\boldsymbol{u}}\boldsymbol{\lambda}(\boldsymbol{u}) = \langle u'(\boldsymbol{\lambda}^T\boldsymbol{\varphi})\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-1}. \tag{17}$$

Now we begin to derive the spectrum of the *One-Shot IPM* iteration (13). Note that in its current form this iteration is not really a fix point iteration, since it uses the time updated dual variables in (13b). To obtain a fix point iteration, we plug the dual iteration step (13a) into the moment iteration (13b) to obtain

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

The Jacobian $\boldsymbol{J} \in \mathbb{R}^{2(N+1)\cdot N_x \times 2(N+1)\cdot N_x}$ has the form

$$\boldsymbol{J} = \begin{pmatrix} \partial_{\boldsymbol{\lambda}}\boldsymbol{d} & \partial_{\boldsymbol{u}}\boldsymbol{d} \\ \partial_{\boldsymbol{\lambda}}\boldsymbol{c} & \partial_{\boldsymbol{u}}\boldsymbol{c} \end{pmatrix}, \tag{18}$$

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

$$\begin{aligned} \partial_{\boldsymbol{\lambda}}\boldsymbol{d}(\boldsymbol{\lambda}^n_j, \boldsymbol{u}^n_j) &= \boldsymbol{I}_d - \boldsymbol{B}\cdot\langle u'(\boldsymbol{\varphi}^T\boldsymbol{\lambda}^n_j)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle - \partial_{\boldsymbol{\lambda}}\boldsymbol{B}\cdot\left(\langle u(\boldsymbol{\varphi}^T\boldsymbol{\lambda}^n_j)\boldsymbol{\varphi}\rangle - \boldsymbol{u}\right) \\ &= -\partial_{\boldsymbol{\lambda}}\boldsymbol{B}\cdot\left(\langle u(\boldsymbol{\varphi}^T\boldsymbol{\lambda}^n_j)\boldsymbol{\varphi}\rangle - \boldsymbol{u}\right), \end{aligned}$$

where we used $\boldsymbol{B} = \langle u'(\boldsymbol{\varphi}^T \boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-1}$. Recall that at the fix point $(\boldsymbol{\lambda}^*, \boldsymbol{u}^*)$, we have $\langle u(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_j^n)\boldsymbol{\varphi}\rangle = \boldsymbol{u}$, hence one obtains $\partial_{\boldsymbol{\lambda}}\boldsymbol{d} = \boldsymbol{0}$. For the block $\partial_{\boldsymbol{u}}\boldsymbol{d}$, we get

$$\partial_{\boldsymbol{u}}\boldsymbol{d}(\boldsymbol{\lambda}_j^n, \boldsymbol{u}_j^n) = \boldsymbol{B},$$

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

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

since we already showed that by the choice of $\boldsymbol{B}$ 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

$$\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\frac{\partial \boldsymbol{d}(\boldsymbol{\lambda}_{j-1}^n, \boldsymbol{u}_{j-1}^n)}{\partial \boldsymbol{u}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\boldsymbol{B} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}_\ell}\langle u'(\boldsymbol{\varphi}^T\boldsymbol{\lambda}_{j-1}^n)\boldsymbol{\varphi}\boldsymbol{\varphi}^T\rangle^{-1} = \partial_{\boldsymbol{u}_\ell}\tilde{\boldsymbol{c}}_j$$

by the choice of $\boldsymbol{B}$ as well as (14) and (17). Hence, we have that $\partial_{\boldsymbol{u}}\boldsymbol{c} = \tilde{\boldsymbol{c}}_{\boldsymbol{u}}$, which only has eigenvalues between -1 and 1 by the assumption that the classical IPM iteration is contractive. Since $\boldsymbol{J}$ is an upper triangluar block matrix, the eigenvalues are given by $\lambda(\partial_{\boldsymbol{\lambda}}\boldsymbol{d}) = 0$ and $\lambda(\partial_{\boldsymbol{u}}\boldsymbol{c}) \in (-1, 1)$, hence the One-Shot IPM is contractive around its fix 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 $(\boldsymbol{\lambda}^0, \boldsymbol{u}^0) \in B_\delta(\boldsymbol{\lambda}^*, \boldsymbol{u}^*)$ we have*

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

Proof. By Theorem 1, the One-Shot scheme is contractive at its fix 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

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

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

Remark 3. *Theorem 2 gives some insights in how to set up the One-Shot IPM iteration: If we look at the Jacobian* (18) *at an arbitrary point, we obtain*

Let us assume that the classical IPM scheme converges globally.

- *Choose a moment vector $\boldsymbol{u}^0$ and compute the corresponding exact dual variables $\boldsymbol{\lambda}^0 := \boldsymbol{\lambda}(\boldsymbol{u}^0)$ as starting vector. In this case, the term $\partial_\lambda \boldsymbol{d}$ is equal to zero for the starting conditions.*

5. Results

6. Summary and outlook

References

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

[2] 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.

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

[4] Jonas Kusch, Graham W Alldredge, and Martin Frank. Maximum-principle-satisfying second-order intrusive polynomial moment scheme. *arXiv preprint arXiv:1712.06966*, 2017.
[5] Ilja Kröker and Christian Rohde. Finite volume schemes for hyperbolic balance laws with multiplicative noise. *Applied Numerical Mathematics*, 62(4):441–456, 2012.
[6] 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.