Journal of Computational Physics 231 (2012) 3365–3388

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

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

Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions

S. Mishra [a,*], Ch. Schwab [b], J. Šukys [c]

[a] Seminar for Applied Mathematics, ETH, HG G. 57.2, Rämistrasse 101, Zürich, Switzerland
[b] Seminar for Applied Mathematics, ETH, HG G. 57.1, Rämistrasse 101, Zürich, Switzerland
[c] Seminar for Applied Mathematics, ETH, HG G. 62.1, Rämistrasse 101, Zürich, Switzerland

A R T I C L E   I N F O

Article history:
Received 30 January 2011
Received in revised form 6 January 2012
Accepted 9 January 2012
Available online 20 January 2012

Keywords:
Conservation laws
Euler
MHD
Uncertainty quantification
Multi-level Monte Carlo
Parallelization

A B S T R A C T

We extend the multi-level Monte Carlo (MLMC) in order to quantify uncertainty in the solutions of multi-dimensional hyperbolic systems of conservation laws with uncertain initial data. The algorithm is presented and several issues arising in the massively parallel numerical implementation are addressed. In particular, we present a novel load balancing procedure that ensures scalability of the MLMC algorithm on massively parallel hardware. A new code is described and applied to simulate uncertain solutions of the Euler equations and ideal magnetohydrodynamics (MHD) equations. Numerical experiments showing the robustness, efficiency and scalability of the proposed algorithm are presented.

1. Introduction

A number of problems in physics and engineering are modeled in terms of systems of conservation laws:

$$\begin{cases} \mathbf{U}_t + \operatorname{div}(\mathbf{F}(\mathbf{U})) = 0, & \forall(\mathbf{x},t) \in \mathbf{D} \times \mathbb{R}_+, \\ \mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x}), \end{cases} \tag{1.1}$$

Here, $\mathbf{U} : \mathbf{D} \subset \mathbb{R}^d \mapsto \mathbb{R}^m$ denotes the vector of conserved variables and $\mathbf{F} : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}^{m\times d}$ is the collection of directional flux vectors. The partial differential equation is augmented with initial data $\mathbf{U}_0$.

Examples for conservation laws include the shallow water equations of oceanography, the Euler equations of gas dynamics, the magnetohydrodynamics (MHD) equations of plasma physics and the equations of non-linear elasticity.

It is well known that solutions of (1.1) in general develop discontinuities or shock waves in finite time even for smooth initial data. Hence, solutions of (1.1) are sought in the sense of distributions, see [10]. Furthermore, additional admissibility criteria or *entropy conditions* are imposed to ensure uniqueness.

As the equations are non-linear, analytical solution formulas are only available in very special situations. Consequently, numerical schemes are the main tools for the study of systems of conservation laws. Many efficient numerical schemes for approximating systems of conservation laws are currently available. They include the finite volume, conservative finite difference and discontinuous Galerkin methods, see [21,14]. Alternatives include the front tracking method, see [17].

* Corresponding author.
E-mail addresses: smishra@sam.math.ethz.ch (S. Mishra), christoph.schwab@sam.math.ethz.ch (Ch. Schwab), jonas.sukys@sam.math.ethz.ch (J. Šukys).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2012.01.011

Existing numerical methods for approximating (1.1) require the initial data $\mathbf{U}_0$ as the input. However, in most practical situations, it is not possible to measure this input precisely. The measurement of other inputs like sources, boundary data and coefficients is also prone to uncertainty. This uncertainty in the inputs for (1.1) results in the propagation of uncertainty in the solution. The modeling and approximation of the propagation of uncertainty in the solution due to uncertainty in inputs constitutes the theme of uncertainty quantification (UQ).

Uncertainty in inputs and solutions of PDEs is frequently modeled in a probabilistic manner. The inputs are random fields with prescribed probability laws. The solution is also realized as a random field and the statistical moments of the solutions like the expectation and variance are the quantities of interest.

It is highly non-trivial to develop efficient algorithms for quantifying uncertainty in conservation laws. The biggest challenge lies in the fact that the discontinuities in physical space may lead to the propagation of discontinuities in the probability space. A robust numerical method should be able to deal with these discontinuities. Another challenge lies in dealing with the fact that the number of random sources driving the uncertainty may be very large (possibly infinite).

The design of efficient numerical schemes for quantifying uncertainty in solutions of conservation laws has seen a lot of activity in recent years. The most popular methods are the stochastic Galerkin methods based on generalized polynomial chaos (gPC for short). An incomplete list of references on gPC methods for uncertainty quantification in hyperbolic conservation laws includes [3,7,22,32,27,33] and other references therein. Although these *deterministic* methods show some promise, they suffer from the disadvantage that they are highly *intrusive*. Existing codes for computing deterministic solutions of conservation laws need to be completely reconfigured for implementation of the gPC based stochastic Galerkin methods. Furthermore, some of the intrusive schemes appear rather difficult to parallelize. An alternative class of methods for quantifying uncertainty in PDEs are the stochastic collocation methods, see [36] for a general review and [25,35] for modifications of these methods near discontinuities. Stochastic collocation methods are non-intrusive and easier to parallelize than the gPC based stochastic Galerkin methods.

Another class of methods for computational uncertainty quantification in numerical solutions of PDEs are statistical sampling methods, most notably Monte Carlo (MC) sampling. In a MC method, the probability space is *sampled* and the underlying deterministic PDE is solved for each sample. The MC samples of numerical solutions of the PDE are combined into statistical estimates of expectation and other statistical moments of the random solution which are necessary to quantify uncertainty. In uncertainty quantification for hyperbolic scalar conservation laws with random initial data, MC type methods together with finite volume (FV) spatio-temporal discretizations of the PDE were proposed in a recent paper [26]. The MC-FVM methods were analyzed in the context of a scalar conservation law with random initial data and corresponding estimates of the combined discretization and statistical sampling errors were obtained. MC methods are non-intrusive and therefore are very easy to code. Existing deterministic PDE solvers are reused in MC codes. As it was shown in [26], MC methods converge at rate 1/2 as the number M of MC samples increases. The asymptotic convergence rate $M^{-1/2}$ is non-improvable by the central limit theorem. Therefore, MC methods require a large number of "samples" (with each "sample" involving the numerical solution of (1.1) with a given draw of initial data $\mathbf{U}_0$) in order to ensure low statistical errors. This slow convergence entails high computational costs for MC type methods. In particular, quantifying uncertainty with MC methods for systems of conservation laws in several space dimensions with moderately high number of sources of uncertainty becomes very costly.

In order to address this drawback of the MC methods, we proposed a novel *multi-level Monte Carlo* (MLMC) algorithm for *scalar* conservation laws in [26]. Multi-level MC methods were introduced by Heinrich for numerical quadrature [16] and developed by Giles to enhance the efficiency of path simulations for Itô stochastic ordinary differential equations in [11,12]. More recently, MLMC finite element methods for elliptic problems with stochastic coefficients were introduced by Barth et al. in [4]. MLMC methods for SPDEs with applications have been recently proposed in [5,8,13].

In [26], we presented and analyzed MLMC-FVM for scalar conservation laws with *random initial data*. Based on our asymptotic error analysis, we derived in [26] an optimized combination of sampling sizes on different levels of spatial and temporal resolution to achieve maximum accuracy in the statistical estimates of first and higher order moments of the random solution. We proved that for first order FV solvers of the SCL, the MLMC-FVM obtained in this way allows the computation of approximate statistical moments with the *same accuracy versus cost ratio as a single deterministic solve on the same mesh*. While offering dramatically improved efficiency over standard MC methods, the MLMC-FV methods developed in [26] are still totally non-intrusive and are as easy to code and parallelize as traditional, single level MC FV methods. Numerical examples in one space dimension showed that MLMC-FVM was orders of magnitude faster than standard MC-FVM.

Our main aim in this paper is to extend the MLMC-FVM algorithm to systems of conservation laws in several space dimensions. As a convergence theory for numerical schemes approximating such systems of conservation laws is currently out of reach, even for the deterministic problem, we use the error estimates derived in [26] as a guide to choose the number of samples per mesh level in the MLMC algorithm. The absence of rigorous error estimates makes it imperative to test the MLMC-FVM for systems of conservation laws for a large number of benchmark problems and demonstrate the efficiency of this algorithms for systems in several space dimensions. We do so in this paper.

As compared to [26], in the present paper several interesting implementation issues are addressed: due to the massive computational effort entailed by the accurate numerical solution of multi dimensional systems of conservation laws, numerical solves of (1.1) for single samples of random initial data $\mathbf{U}_0$ can not be performed on a single processor. Therefore, in the present paper the MLMC-FVM algorithm is extended towards several forms of parallelism: parallel computation of large numbers of samples on a coarse grid, and parallelism by mesh partitioning of the (few) samples on the finest grid. To this end, we describe a novel (static) load balancing paradigm for a *scalable* version of the MLMC-FVM on a certain class of massively parallel hardware. Although the theoretical MLMC-FVM convergence results of [26] no longer hold for hyperbolic systems (there are no convergence results for numerical schemes even for systems with deterministic data), the MLMC-FVM algorithm developed in [26] is tested on a series of hyperbolic benchmark problems and is shown to be robust: the basic conclusions drawn based on the numerical analysis of MLMC-FVM for SCL in [26] appear to be applicable also to algorithm design for a wide range of hyperbolic conservation laws. In particular, we consider the Euler equations of gas dynamics and the system of compressible, ideal MHD equations in two space dimensions and show that the MLMC-FVM methods can quantify uncertainty in very complex realistic situations; for instance, the MLMC-FVM is able to handle a large number of sources of uncertainty (reminiscent to large number of stochastic dimensions in a gPC method) that appear to be beyond the reach of other existing UQ methods for systems of hyperbolic conservation laws.

The rest of the paper is organized as follows: the mathematical setup is described in Section 2. We note that conservation laws with uncertain sources and boundary data can be handled similarly. We present the MC and algorithms from [26] in Sections 4 and 5, respectively. The serial implementation of the FV solver is briefly recapitulated in Section 6. In Section 7, we describe the parallelization algorithm and numerical experiments in several space dimensions are presented in Section 8.

2. Mathematical preliminaries

Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a complete probability space, then:

Definition 2.1 (*Random field*). A random field is a measurable mapping $\mathbf{U}:\Omega \ni \omega \mapsto \mathbf{U}(\mathbf{x},t,\omega)$ from $(\Omega,\mathcal{F})$ to $((C_b([0,\ T],L^1(\mathbb{R}^d)))^m,\ \mathcal{B}((C_b([0,\ T],L^1(\mathbb{R}^d)))^m)$.

Here $\mathcal{B}(X)$ denotes the Borel sigma algebra corresponding to a set X.

The conservation law (1.1) with random initial data is

$$\begin{cases} \frac{\partial \mathbf{U}(\mathbf{x},t,\omega)}{\partial t} + \operatorname{div}(\mathbf{F}(\mathbf{U}(\mathbf{x},t,\omega))) = 0, \\ \mathbf{U}(\mathbf{x},0,\omega) = \mathbf{U}_0(\mathbf{x},\omega). \end{cases} \mathbf{x} \in \mathbb{R}^d, \quad t > 0, \quad \forall \omega \in \Omega. \tag{2.1}$$

Here, the initial data $\mathbf{U}_0$ is a random field i.e, $\mathbf{U}_0:\Omega \ni \omega \mapsto \mathbf{U}_0(\mathbf{x},\omega)$ that is measurable from $(\Omega,\mathcal{F})$ to $((L^1(\mathbb{R}^d))^m, \mathcal{B}((L^1(\mathbb{R}^d))^m))$.

Assuming that the underlying deterministic conservation law (1.1) has an entropy formulation [10,14], we define the following notion of solutions of (2.1):

Definition 2.2 (*Random entropy solution*). A random field $\mathbf{U}:\Omega \ni \omega \mapsto \mathbf{U}(\mathbf{x},t,\omega)$ is an entropy solution to stochastic conservation law (2.1) if it is a weak solution and if it satisfies an entropy condition (corresponding to the entropy formulation for (1.1)) for $\mathbb{P}$-a.e. $\omega \in \Omega$.

In methods random entropy solutions are sought path-wise. It is not possible to prove that the random entropy solutions defined above exist for systems of conservation laws. This is not surprising as there are no global well-posedness results for systems of conservation laws, even in one space dimension.

In [26], the authors analyzed the special case of scalar conservation laws with uncertain initial data,

$$\begin{cases} \frac{\partial u(\mathbf{x},t,\omega)}{\partial t} + \operatorname{div}(f(u(\mathbf{x},t,\omega))) = 0, \\ u(\mathbf{x},0,\omega) = u_0(\mathbf{x},\omega). \end{cases} \mathbf{x} \in \mathbb{R}^d, \quad t > 0, \quad \forall \omega \in \Omega, \tag{2.2}$$

and showed that the random entropy solutions exist and satisfy certain stability estimates, see Theorem 3.3 of [26]. Furthermore, under suitable regularity assumptions on the initial data, the authors showed that statistical moments of the random entropy solution also exist and are bounded, see Theorem 3.4 of [26].

For the remainder of this paper, we will assume that unique random entropy solutions for the system (2.1) exist. Furthermore, we assume that for $L^1(\mathbb{R}^d)^m$-valued random initial data with finite variances, this entropy solution has finite second moments as $L^1(\mathbb{R}^d)^m$-valued random field. Based on this, we generalize the design principles for MLMC-FVM from [26] to hyperbolic systems (2.1).

3. Finite volume methods

We aim to design an efficient MC type scheme for approximating the stochastic conservation law (2.1). This entails discretizing space, time as well as the probability space. For spatio-temporal discretization, we will employ finite volume methods.

Let the time step be $\Delta t > 0$ and a triangulation $\mathcal{T}$ of the spatial domain $\mathbf{D} \subset \mathbb{R}^d$ of interest. Here, a triangulation $\mathcal{T}$ will be understood as a partition of the physical domain into a finite set of disjoint open, convex polyhedra $K \subset \mathbb{R}^d$ with boundary being a finite union of plane faces. Let $\Delta x_K := \operatorname{diam} K$ and by $\Delta x(\mathcal{T}) := \max\{\Delta x_K : K \in \mathcal{T}\}$ denote the *mesh width* of $\mathcal{T}$. For any volume $K \in \mathcal{T}$, we define the *set $\mathcal{N}(K)$ of neighbouring volumes*

$$\mathcal{N}(K) := \{K' \in \mathcal{T} : K' \neq K \wedge \mathrm{meas}_{d-1}(\overline{K} \cap \overline{K'}) > 0\}. \tag{3.1}$$

For every $K \in \mathcal{T}$ and $K' \in \mathcal{N}(K)$ denote $\nu_{K,K'}$ to be the unit normal pointing outward from the volume K at the face $\overline{K} \cap \overline{K'}$. We set:

$$\lambda = \Delta t / \min\{\Delta x_K : \; K \in \mathcal{T}\} \tag{3.2}$$

by assuming a uniform discretization in time with time step $\Delta\, t$. The constant λ is determined by a standard CFL condition (see [14]) based on the maximum wave speed.

Then, an explicit first-order finite volume discretization ([14]) for approximating (1.1) is given by

$$\mathbf{U}_K^{n+1} = \mathbf{U}_K^n - \frac{\Delta t}{\mathrm{meas}(K)} \sum_{K' \in \mathcal{N}(K)} \mathbf{F}(\mathbf{U}_K^n, \mathbf{U}_{K'}^n), \tag{3.3}$$

where

$$\mathbf{U}_K^n \approx \frac{1}{\mathrm{meas}(K)} \int_K \mathbf{U}(\mathbf{x}, t^n) d\mathbf{x}$$

is an approximation to the cell average of the solution and $\mathbf{F}(\cdot,\cdot)$ is a numerical flux that is consistent with $\mathbf{F} \cdot \nu_{K,K'}$. Numerical fluxes are usually derived by (approximately) solving Riemann problems at each cell edge resulting in the Godunov, Roe and HLL fluxes, see [21].

Higher order spatial accuracy is obtained by reconstructing $\mathbf{U}$ from $\mathbf{U}_K^n$ in non-oscillatory piecewise polynomial functions or by the Discontinuous Galerkin method. Higher order temporal accuracy is achieved by employing strong stability preserving Runge–Kutta methods.

4. Monte Carlo finite volume method

4.1. MC-FVM algorithm

The MC-FVM algorithm consists of the following three steps:

1. **Sample:** We draw M independent identically distributed (i.i.d.) initial data samples $\mathbf{U}_0^i$ with $i = 1,2,\ldots,M$ from the random field $\mathbf{U}_0$ and approximate these by piecewise constant cell averages on the FV mesh.
2. **Solve:** For each realization $\mathbf{U}_0^i$, the underlying conservation law (1.1) is solved numerically by the finite volume method (3.3). We denote the FVM solutions by $\mathbf{U}_{\mathcal{T}}^{i,n}$, i.e. by cell averages $\{\mathbf{U}_K^{i,n} : K \in \mathcal{T}\}$ at the time level t^n,

$$\mathbf{U}_{\mathcal{T}}^{i,n}(\mathbf{x}) = \mathbf{U}_K^{i,n}, \quad \forall \mathbf{x} \in K, \; K \in \mathcal{T}.$$

3. **Estimate statistics:** We estimate the expectation of the random solution field with the sample mean (ensemble average) of the approximate solution:

$$E_M[\mathbf{U}_{\mathcal{T}}^n] := \frac{1}{M} \sum_{i=1}^{M} \mathbf{U}_{\mathcal{T}}^{i,n}. \tag{4.1}$$

Higher statistical moments can be approximated analogously.

The above algorithm is quite simple to implement. We remark that step 1 requires a (pseudo) random number generator. In step 2, any standard (high-order) finite volume scheme can be used. Hence, existing code for FVM can be used and there is no need to rewrite FVM code. Furthermore, the only (data) interaction between different samples is in step 3 when ensemble averages are computed. Thus, the MC-FVM is non-intrusive as well as easily parallelizable.

4.2. Convergence analysis

It is not possible to show rigorously that the MC-FVM algorithm converges for systems of conservation laws. In the special case of the stochastic scalar conservation law (2.2), a rigorous error estimate for the MC-FVM scheme was obtained in [26]. We refer the reader to [26], Theorem 4.6 for details on the assumptions and directly state the error estimate:

$$\|\mathbb{E}[u(\cdot,t^n)] - E_M[u_{\mathcal{T}}^n]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leqslant C\left\{M^{-\frac{1}{2}}\|u_0\|_{L^2(\Omega;L^1(\mathbb{R}^d))} + t^n \Delta x^s \|TV(u_0(\cdot,\omega))\|_{L^\infty(\Omega;d\mathbb{P})}\right\}. \tag{4.2}$$

Here, the $L^2(\Omega;L^1(\mathbb{R}^d))$-norm is defined as

$$\|f(\mathbf{x},\omega)\|_{L^2(\Omega;L^1(\mathbb{R}^d))} := \left(\int_\Omega \|f(\mathbf{x},\omega)\|_{L^1(\mathbb{R}^d)}^2 d\mathbb{P}(\omega)\right)^{\frac{1}{2}}, \tag{4.3}$$

and C is a constant independent of M and Δx and s is the convergence rate of the deterministic FVM solver. The results of Kuznetsov [15] show that $s = \frac{1}{2}$, when a first-order FVM is used to approximate scalar conservation laws.

Note that the error estimate for the mean requires that the initial random field has finite second moments. A similar error estimate for the *k*th moment was obtained in [26] provided that the initial data has finite 2*k* moments.

Using standard error estimates for numerical schemes approximating scalar conservation laws (see e.g. [14,26]), it is straightforward to deduce that the (computational) work needed to solve *one* MC sample is, asymptotically,

$$\mathrm{Work}_1(\Delta x) = \mathcal{O}(\Delta x^{-(d+1)}). \tag{4.4}$$

In order to equilibrate the statistical error in (4.2) with the spatio-temporal discretization error, we need to choose

$$M = \mathcal{O}(\Delta x^{-2s}) \tag{4.5}$$

samples. Hence, the amount of computational work needed to solve M samples is

$$\mathrm{Work}_M(\Delta x) = \mathcal{O}(M\Delta x^{-(d+1)}) = \mathcal{O}(\Delta x^{-2s}\Delta x^{-(d+1)}). \tag{4.6}$$

This leads to the asymptotic error vs. work estimate

$$\left\|\mathbb{E}[u(\cdot,t^n)] - E_M\left[u^n_{\mathcal{T}}\right]\right\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \lesssim (\mathrm{Work})^{-s/(d+1+2s)}. \tag{4.7}$$

The above error vs. work estimate should be compared to the deterministic FVM error which scales as $(\mathrm{Work})^{-s/(d+1)}$.

Hence, the MC-FVM is considerably more expensive than the standard FVM for a deterministic conservation law. As an example, a first order scheme ($s = 1/2$) leads to a convergence rate of 1/6 for the MC-FVM as compared to a convergence rate of 1/4 for the standard FVM for a deterministic conservation law. Moreover, even for smooth solutions and high order FV schemes, i.e. as $s \to \infty$, the convergence rate is dominated by the MC rate of 1/2.

Remark 4.1. We are unable to prove a version of the error estimate (4.2) for systems of conservation laws. This is primarily on account of the fact that there are no convergence results for numerical schemes approximating (1.1). However, we will choose the number of MC samples by (4.5) and test numerically whether the above convergence rates and error vs. work estimates hold also for particular hyperbolic systems.

5. Multi-level Monte Carlo finite volume method

Given the slow convergence of MC-FVM, we propose the multi-level Monte Carlo finite volume method (MLMC-FVM). The key idea behind MLMC-FVM is to simultaneously draw MC samples on a hierarchy of nested grids [26].

5.1. MLMC-FVM algorithm

The algorithm consists of the following four steps:

1. **Nested meshes:** Consider *nested* triangulations $\{\mathcal{T}_\ell\}_{\ell=0}^{\infty}$ of the spatial domain $\mathbf{D}$ with corresponding mesh widths Δx_ℓ that satisfy:

$$\Delta x_\ell = \Delta x(\mathcal{T}_\ell) = \sup\{\mathrm{diam}(K) : K \in \mathcal{T}_\ell\} = O(2^{-\ell}\Delta x_0), \quad \ell \in \mathbb{N}_0, \tag{5.1}$$

where Δx_0 is the mesh width for the coarsest resolution and corresponds to the lowest level $\ell = 0$.

2. **Sample:** For each level of resolution $\ell \in \mathbb{N}_0$, we draw M_ℓ independent identically distributed (i.i.d) samples $\mathbf{U}^i_{0,\ell}$ with $i = 1,2,\ldots,M_\ell$ from the initial random field $\mathbf{U}_0$.
3. **Solve:** For each resolution level ℓ and each realization $\mathbf{U}^i_{0,\ell}$, the underlying conservation law (1.1) is solved by the finite volume method (3.3) with mesh width Δx_ℓ. Let the finite volume solutions be denoted by $\mathbf{U}^{i,n}_{\mathcal{T}_\ell}$, i.e. by cell averages $\{\mathbf{U}^{i,n}_K : K \in \mathcal{T}_\ell\}$ at the time level t^n and resolution level ℓ.
4. **Estimate solution statistics:** Fix some positive integer $L < \infty$ corresponding to the highest level. We estimate the expectation of the random solution field with the following estimator:

$$E^L[\mathbf{U}(\cdot,t^n)] := \sum_{\ell=0}^{L} E_{M_\ell}\left[\mathbf{U}^n_{\mathcal{T}_\ell} - \mathbf{U}^n_{\mathcal{T}_{\ell-1}}\right], \tag{5.2}$$

with E_{M_ℓ} being the MC estimator defined in (4.1) for the level ℓ. Higher statistical moments can be approximated analogously (see, e.g., [26]).

A few remarks are in order. First, the estimator (5.2) indicates that *for each draw of the random data, the hyperbolic system must be solved numerically on two consecutive meshes and time steps, for this data sample*. In the present implementation, we did not exploit this fact to accelerate the solves, e.g. by the use of multilevel techniques, but rather invoked two instances of the deterministic solver ALSVID with the same initial data, but different discretization levels.

Second, MLMC-FVM is also non-intrusive as any standard FVM code can be used in step 3. Furthermore, MLMC-FVM is amenable to efficient parallelization as data from different grid resolutions and different samples only interacts in step 4 where the sample statistics are computed.

5.2. Convergence analysis

Again we have rigorous convergence results for the MLMC-FVM algorithm in the scalar case (2.2), see Theorem 4.8 of [26] for the assumptions on the grid, the finite volume scheme and the initial data. The resulting error estimate is

$$\|\mathbb{E}[u(\cdot,t^n)] - E^L[u(\cdot,t^n)]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leqslant C\left\{\bar{t}\Delta x_L^s \|TV(u_0)\|_{L^1(\Omega,d\mathbb{P})} + \Delta x_L^s \|u_0\|_{L^\infty(\Omega;W^{s,1}(\mathbb{R}^d))}\right\} + C\left\{\sum_{\ell=0}^{L} M_\ell^{-\frac{1}{2}}\Delta x_\ell^s\right\}\left\{\|u_0\|_{L^2(\Omega;W^{s,1}(\mathbb{R}^d))} + t^n\|TV(u_0)\|_{L^2(\Omega;d\mathbb{P})}\right\}. \tag{5.3}$$

Here s refers to the convergence rate of the deterministic finite volume scheme. From the error estimate (5.3), we obtain that the number of samples to equilibrate the statistical and spatio-temporal discretization errors in (5.2) is given by:

$$M_\ell = \mathcal{O}(2^{2(L-\ell)s}). \tag{5.4}$$

Notice that (5.4) implies that the largest number of MC samples is required on the coarsest mesh level $\ell = 0$, whereas only a small fixed number of MC samples are needed on the finest discretization levels.

Combining (5.2) with (5.4) we obtain the following computational cost (Work$_L$) estimates for the MLMC-FVM,

$$\mathrm{Work}_L = \sum_{\ell=0}^{L} M_\ell \mathcal{O}(\Delta x_\ell^{-(d+1)}) \leqslant C\sum_{\ell=0}^{L} 2^{2(L-\ell)s+\ell(d+1)} = C2^{2Ls}\sum_{\ell=0}^{L} 2^{(d+1-2s)\ell} = \begin{cases} C2^{2Ls+[(d+1)-2s]L} = C2^{(d+1)L} = \mathcal{O}\left(\Delta x_L^{-(d+1)}\right) & \text{if } s<(d+1)/2,\\ C2^{(d+1)L}(L+1) = \mathcal{O}\left(\Delta x_L^{-(d+1)}\log\left(\Delta x_L^{-1}\right)\right) & \text{if } s=(d+1)/2. \end{cases} \tag{5.5}$$

Note that the above result is a generalization of the work estimates for MLMC-FVM obtained in [26] as we can cover the case of $s = (d + 1)/2$.

From (5.5), we obtain the corresponding error vs. work estimate for MLMC-FVM,

$$\|\mathbb{E}[u(\cdot,t^n)] - E^L[u(\cdot,t^n)]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leqslant C\Delta x_L^s L \leqslant C\Delta x_L^s \log\left(\Delta x_L^{-1}\right) \lesssim \begin{cases} (\mathrm{Work})^{-s/(d+1)}\cdot\log(\mathrm{Work}) & \text{if } s<(d+1)/2,\\ (\mathrm{Work})^{-1/2}\cdot\log(\mathrm{Work})^{3/2} & \text{if } s=(d+1)/2. \end{cases} \tag{5.6}$$

The second bound in (5.6) is obtained as follows. From (5.5) we deduce that

$$\Delta x_L^s \log(\Delta x_L^{-1}) = \mathcal{O}\left(\left(\frac{\mathrm{Work}}{W(\mathrm{Work})}\right)^{-1/2}\cdot\log\left(\frac{\mathrm{Work}}{W(\mathrm{Work})}\right)\right), \tag{5.7}$$

where $W(\cdot)$ denotes the Lambert W-function [9]. For $y \gg 1, y \leqslant \log(y)\exp(\log(y))$ holds, hence also $W(y) \leqslant \log(y)$. Replacing $W(\cdot)$ by $\log(\cdot)$ in the first term of (5.7) and removing the $W(\cdot)$ from the logarithmic term, the bound (5.6) is obtained.

Estimates in (5.6) show that the MLMC-FVM is superior to the MC-FVM since the asymptotic computational cost for MLMC-FVM scales as Work$^{-s/(d+1)}$ (disregarding the logarithmic term); compare to Work$^{-s/(d+1+2s)}$ for the MC-FVM scheme. Furthermore, if $s < (d + 1)/2$ then this error vs. work estimate is almost (up to logarithmic term) of the same order as the error vs. work of the deterministic finite volume scheme. Hence, the MLMC-FVM is expected to be (asymptotically) considerably faster than the MC-FVM for the same magnitude of error.

Remark 5.1. Although we are unable to prove an error estimate like (5.3) for systems of conservation laws, we choose the number of samples at each resolution level by (5.4) and test numerically whether the MLMC-FVM for systems obeys the same asymptotic estimates. This is done in the next section.

6. Serial implementation of MLMC-FVM

We begin the description of implementation of the MLMC-FVM algorithm by presenting its implementation on a single processor. The implementation on a parallel architecture is more intricate and is postponed to the next section.

As stated in the last section, the MLMC-FVM algorithm has four stages. We discuss implementation issues that arise in each stage below.

6.1. Step 1: Hierarchy of nested grids

We will solve systems of conservation laws (2.1) in one and two space dimensions. In two space dimensions, we choose Cartesian meshes for simplicity. It is relatively straightforward to choose a hierarchy of nested grids in both one and two space dimensions.

6.2. Step 2: Sample

In this step, we have to draw M_ℓ i.i.d. samples for the initial random field $\mathbf{U}_0$ corresponding to the underlying probability distribution. Standard random number generators (RNG) can be readily used to draw such samples. For the serial implementation, any reasonable RNG works well in practice.

6.3. Step 3: Solve

For each realization of the initial random field, we need to solve (2.1) with a finite volume scheme. In this paper, we consider two numerical examples. First, the Euler equations of gas dynamics are considered. In several space dimensions, these equations are

$$\begin{cases} \rho_t + \operatorname{div}(\rho\mathbf{u}) = 0, \\ (\rho\mathbf{u})_t + \operatorname{div}(\rho\mathbf{u}\otimes\mathbf{u} + p\mathbf{ID}) = 0, \\ E_t + \operatorname{div}((E+p)\mathbf{u}) = 0. \end{cases} \tag{6.1}$$

Here, ρ is the density and $\mathbf{u}$ is the velocity field. The pressure p and total energy E are related by the ideal gas equation of state:

$$E := \frac{p}{\gamma-1} + \frac{1}{2}\rho|\mathbf{u}|^2, \tag{6.2}$$

with γ being the ratio of specific heats. It is well known that the Euler equations are strictly hyperbolic and the corresponding eigenvalues and eigenvectors are readily computed, see [21].

The second numerical example that we will consider are the equations of magnetohydrodynamics (MHD) given by

$$\begin{cases} \rho_t + \operatorname{div}(\rho\mathbf{u}) = 0, \\ (\rho\mathbf{u})_t + \operatorname{div}(\rho\mathbf{u}\otimes\mathbf{u} + \left(p+\frac{1}{2}|\mathbf{B}|^2\right)I - \mathbf{B}\otimes\mathbf{B}) = 0, \\ \mathbf{B}_t + \operatorname{div}(\mathbf{u}\otimes\mathbf{B} - \mathbf{B}\otimes\mathbf{u}) = 0, \\ E_t + \operatorname{div}\left(\left(E+p+\frac{1}{2}|\mathbf{B}|^2\right)\mathbf{u} - (\mathbf{u}\cdot\mathbf{B})\mathbf{B}\right) = 0, \\ \operatorname{div}(\mathbf{B}) = 0, \end{cases} \tag{6.3}$$

Here, $\mathbf{B}$ denotes the magnetic field and the total energy is given by the equation of state:

$$E := \frac{p}{\gamma-1} + \frac{1}{2}\rho|\mathbf{u}|^2 + \frac{1}{2}|\mathbf{B}|^2. \tag{6.4}$$

In contrast to the Euler equations, the MHD equations are not strictly hyperbolic. Furthermore, the design of robust numerical schemes for MHD is quite challenging as the Eq. (6.3) involve the divergence constraint. This constraint needs to be handled in a suitable manner in order to avoid spurious oscillations, see [31]. Note that the Euler Eq. (6.1) are recovered from the MHD Eq. (6.3) by setting the magnetic field $\mathbf{B} \equiv 0$.

As the solve step in the MLMC-FVM algorithm will be repeated for a large number of samples on different space-time resolution levels, we need a robust and efficient FVM code for the Euler and MHD equations. We choose the code named `ALSVID` [1] that was designed by researchers at CMA, University of Oslo and SAM, ETH Zürich. As `ALSVID` is extensively used in the examples of this paper, we describe it briefly below.

6.3.1. ALSVID

This finite volume code approximates the Euler equations and MHD equations in one, two and three space dimensions. It is based on the following ingredients:

1. **Approximate Riemann solver:** The numerical fluxes in the finite volume scheme (3.3) used in `ALSVID` are based on approximate Riemann solvers of the HLL type. For the Euler equations (6.1), the code uses the HLL three wave solver proposed by Toro et. al [30]. This approximate Riemann solver resolves contact discontinuities and has the same resolution as the standard Roe solver but computationally is less expensive and is known to preserve positive densities and pressures in the absence of round-off errors, see [19].
   For the MHD Eq. (6.3), `ALSVID` uses the three and five wave HLL type solvers designed in [19]. These solvers aim at approximating the modified Godunov–Powell form of the MHD equations, see [28,19].

2. **Divergence constraint.** The divergence constraint in the MHD Eq. (6.3) is handled in ALSVID by adding the Godunov–Powell source term to the MHD equations. This source term is proportional to the divergence and allows divergence errors to be swept out of the domain. Numerical stability can only be ensured by a careful *upwinding* of the source term, see [19].
3. **Non-oscillatory reconstructions.** ALSVID employs a variety of piecewise polynomial non-oscillatory reconstruction procedures for attaining high order of spatial accuracy. In particular, second order ENO and WENO procedures are employed, see Section 2 of [19]. However, these procedures need to be modified in order to preserve positivity of the density and pressure. Such modifications are described in Section 2 of [19].
4. **Time stepping.** High-order accurate time stepping procedures of the SSP Runge–Kutta [20] are employed in ALSVID.

Fluxes on the boundary of the computational domain are defined using so-called ghost cells, see chapter 10 in [21].

ALSVID uses a modular structure in C++ with a Python front end for pre-and post-processing. One and two dimensional visualizations are performed with MatPlotLib and three dimensional data sets are visualized using MayaVi2. Extensive testing of ALSVID has been performed and reported in [19].

6.4. Computing sample statistics

For both MC-FVM and MLMC-FVM algorithms we need to combine individual realizations to compute ensemble averages. It is straightforward to compute the sample mean for the MC-FVM and the estimator (5.2) for MLMC-FVM. A straightforward algorithm to compute an *unbiased* estimate of the variance for scalar $u = u(x,t)$ with fixed x, t is the following statistical estimator:

$$\mathrm{Var}[u] := \mathbb{E}[u^2] - \mathbb{E}[u]^2 \approx \mathrm{Var}_M[u] := \frac{1}{M-1}\sum_{i=1}^{M}(u^i)^2 - \left(\frac{1}{M-1}\sum_{i=1}^{M}u^i\right)^2, \tag{6.5}$$

where u^i are MC-FVM samples. This way, it suffices to loop over all samples only once; unfortunately, both quantities are almost equal in the regions of vanishing variance which leads to *subtractive cancellation* and *loss of accuracy* in floating point arithmetic. In [34], the authors propose an alternative *stable* "on-line" variance computation algorithm:

Set $\bar{u}^0 = 0$ and $\Phi^0 = 0$; then proceed iteratively:

$$\bar{u}^i = \sum_{j=1}^{i} u^j/i, \tag{6.6}$$

$$\Phi^i := \sum_{j=1}^{i}(u^j - \bar{u}^i)^2 = \Phi^{i-1} + (u^i - \bar{u}^i)(u^i - \bar{u}^{i-1}). \tag{6.7}$$

Then, the unbiased mean and variance estimates are given by:

$$E_M[u] = \bar{u}^M, \quad \mathrm{Var}_M[u] = \Phi^M/(M-1). \tag{6.8}$$

Although identical in exact arithmetic, the above algorithm can deal with small cancellation errors.

We combine a standard RNG in step 2, ASLVID in step 3 and the above estimators in step 4 to obtain an efficient MLMC-FVM for a single processor. The resulting code is also written in C++ with Python front and back ends. Note that the deterministic code ALSVID is reused without any alterations in the solve step as MC-FVM and MLMC-FVM are non-intrusive. As the UQ module is based on ALSVID, we call it as ALSVID-UQ. The parallel version of ALSVID-UQ will be described later.

6.5. Numerical experiments for systems in 1D

We describe numerical experiments for one dimensional systems of conservation laws to test ALSVID-UQ.

6.5.1. Sod shock tube with uncertain shock location

Let $Y \sim 1 + \mathcal{U}(0, \frac{1}{10})$ be a random variable. We consider one dimensional version of the Euler equations (6.1) with $\mathbf{D} = [0,2]$ and with *random* initial shock at *uncertain* location (near $x = 1$):

$$\mathbf{U}_0(x,\omega) = \{\rho_0(x,\omega), u_0(x,\omega), p_0(x,\omega)\} = \begin{cases} \{3.0, 0.0, 3.0\} & \text{if} \quad x < Y(\omega), \\ \{1.0, 0.0, 1.0\} & \text{if} \quad x > Y(\omega). \end{cases} \tag{6.9}$$

The initial data (6.9) and the reference solution at time $t = 0.5$ are depicted in Fig. 1. At every point $x \in [0,2]$ the solid line represents the mean and the dashed lines represent the mean ± standard deviation of the (random) solution. For each *sample* the initial shock splits into three waves: a left going rarefaction wave, a right going contact discontinuity and a right going shock wave. Notice the improvement of the regularity in the stochastic solution: deterministic path-wise solutions for each sample are discontinuous due to formation of the shock and the contact; nevertheless, the mean of the solution is continuous.

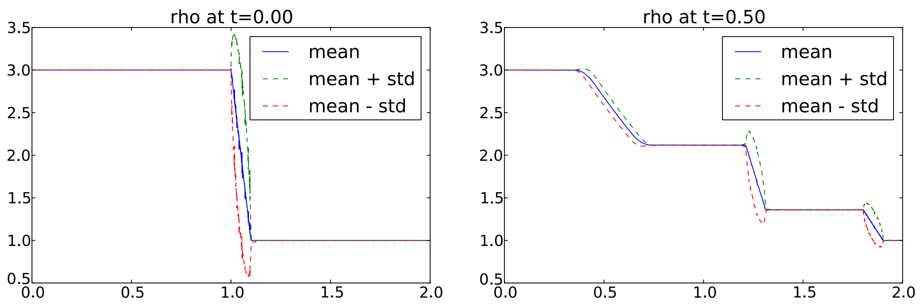


Fig. 1. Reference solution computed using MLMC-FVM. The solution is computed with a second-order WENO scheme using a HLL solver on a grid size of 32768 cells. The number of levels of refinement is 12 and we use 16 samples at the finest level. Note that the initial shock splits into 3 waves all having uncertain location. The ensemble average is *more regular* (continuous) than its paths (with shocks).

Remark 6.1. The "overshoots" in dashed lines representing standard deviation do *not* imply that the random shock amplitude can exceed the mean depicted in solid line; in fact, the shock amplitude is the same for every sample path.

6.5.2. Numerical convergence analysis

Using MLMC-FVM approximation from Fig. 1 as a reference solution, we run MC-FVM and MLMC-FVM methods on the series of mesh resolutions ranging from 32 cells up to 8192 cells and monitor the convergence behavior. The number of levels for the MLMC-FVM method is chosen so that the coarsest level contains 8 cells.

Error estimator. Since the solution is a random field, the discretization error is a random quantity as well. For convergence analysis we therefore compute a statistical estimator by averaging estimated discretization errors from several independent runs. We will compute the error in (4.2) by approximating $L^2(\Omega; L^1(\mathbb{R}^d))$ norm with MC quadrature. Let U_{ref} denote the reference solution and $\{U_k\}_{k=1,\dots,K}$ be a sequence of independent approximate solutions obtained by running MC-FVM or MLMC-FVM solver K times corresponding to K realizations of the stochastic space. Then the $L^2(\Omega; L^1(\mathbb{R}^d))$-based relative error estimator is defined as in [26],

$$\mathcal{RE} = \sqrt{\sum_{k=1}^{K} (\mathcal{RE}_k)^2 / K}, \tag{6.10}$$

where:

$$\mathcal{RE}_k = 100 \times \frac{\|U_{\text{ref}} - U_k\|_{\ell^1(\mathcal{T})}}{\|U_{\text{ref}}\|_{\ell^1(\mathcal{T})}}. \tag{6.11}$$

The extensive analysis for the appropriate choice of K is conducted in [26]; we choose $K = 30$ which was shown to be more than sufficient.

Notation for different combinations of ML(MC) and FVM methods (4.5), (5.4):

MC	Monte Carlo with 1st order FVM scheme	$M = \mathcal{O}(\Delta x^{-1})$,
MC2	Monte Carlo with 2nd order FVM scheme	$M = \mathcal{O}(\Delta x^{-2})$,
MLMC	multilevel MC with 1st order FVM scheme	$M_\ell = M_L\, 2^{(L-\ell)}$,
MLMC2	multilevel MC with 2nd order FVM scheme	$M_\ell = M_L\, 4^{(L-\ell)}$.

The parameter M_L corresponds to the number of samples in the finest level and can be freely chosen. Analysis in [26] suggests that $M_L = 16$ is a reasonable choice; however, $M_L = 4$ yielded better results for the problems with particularly large number of levels. In the convergence analysis we choose $M_L = 16$ unless indicated otherwise. Having notations and definitions in place, we proceed to the convergence plots of mean and variance.

Dashed lines in Fig. 2 (and all subsequent figures) indicate *expected* convergence rate slopes obtained by theory for *scalar* case (see (4.2) and (5.3)). We expect them to coincide with the observed convergence rates for *systems* of conservation laws and in this particular case they are actually very similar. Findings coincide with the results published in [26] confirming the robustness of the implementation.

Remark 6.2. Due to additional log(Work) term in (5.6) for the case $s = (d + 1)/2$, the error vs. runtime convergence rate for MLMC2 is achieved *only asymptotically*.

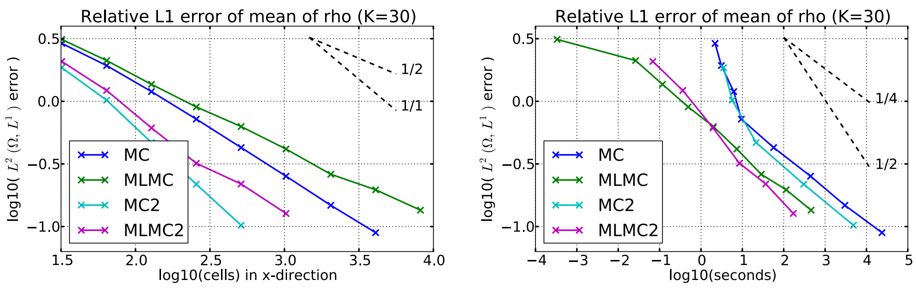


Fig. 2. Convergence of mean of Sod shock tube problem for Euler Eq. (6.1) with random initial data (6.9). Both MLMC (MLMC2) and MC (MC2) give similar errors for the same spatial resolution. However, there is a significant difference in runtime: MLMC methods are more than an order of magnitude faster that pure MC.

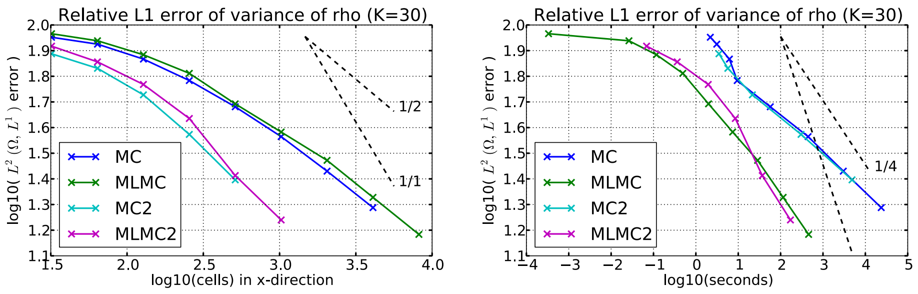


Fig. 3. Convergence of variance of Sod shock tube problem for Euler Eq. (6.1) with random initial data (6.9). MLMC methods are more than 2 orders of magnitude faster that pure MC.

In Fig. 3, we show convergence plots for variance. The observed convergence rate for MLMC2 is again slightly smaller than expected. This could be attributed to two reasons. First, as indicated before, the optimal convergence rate is only expected asymptotically when $s = (d + 1)/2$ which is the case for formally second order schemes in one space dimension. We assume that second order schemes converge with rate one in the presence of shocks. The second reason could be the amplitude of the fourth moment as the error estimate of the variance relies on the fourth moment. We compute the 4th centered moment in Fig. 4 and find that it is of relatively small amplitude.

The performance of the MC and MLMC methods depends on how the number of samples (at each level of resolution) are chosen. Our choices are governed by *rigorously* obtained error estimates (4.2) and (5.3), respectively, for scalar conservation laws. The only parameter in these estimate is the rate of convergence s of the underlying finite volume scheme. Here, we again rely on the rigorously proved convergence rate $s = 1/2$ for a first-order scheme by the theorem of Kuznetsov [15]. It is well known that this rate might not be optimal in some cases and we can obtain better rates of convergence (≈ 1) numerically. However, there is no proof that this is indeed the case. In the absence of a result to the contrary, we rely on the error estimate of Kuznetsov. To test whether this choice is a good one, we repeat the above experiment with a different choice of $s = 1$ for a first-order finite volume scheme. Consequently, the number of MC samples given by (4.5) is $M = \mathcal{O}(\Delta x^{-2})$ and the number of MLMC samples given by (5.4) is given by $M_\ell = M_L 4^{(L-\ell)}$ with M_ℓ being the number of samples at the finest level. The numerical convergence results for this experiment are shown in Fig. 5. In this figure, we show the error vs. number of cells (left) and error vs. runtime (right) obtained with a *first-order* finite volume scheme with assumed rates of convergence $s = 1/2$ and $s = 1$. The results show that both choices led to very similar error for any fixed resolution. However, there is a significant difference in computational efficiency. For both MC as well MLMC methods with $s = 1/2$ require considerably less time to compute the same level of error than their $s = 1$ counterparts. This shows that the choice $s = 1/2$ is not only based on the rigorous error estimate of Kuznetsov, but is also computationally efficient in both the MC as well as MLMC contexts. Furthermore, we see that the MLMC method with either $s = 1/2$ or $s = 1$ is at least two orders of magnitude (asymptotically) faster than its MC counterpart with the same choice of s implying that the MLMC method is fairly robust with respect to the assumed rate of convergence.

7. Parallel implementation of MLMC-FVM

We recall that consists of four steps. In the first step, we select a nested hierarchy of Cartesian grids. This step is straightforward for any parallel architecture. In step 2, we draw samples for the initial random field with a given probability distribution. Here, we need a robust random number generator (RNG) described below.

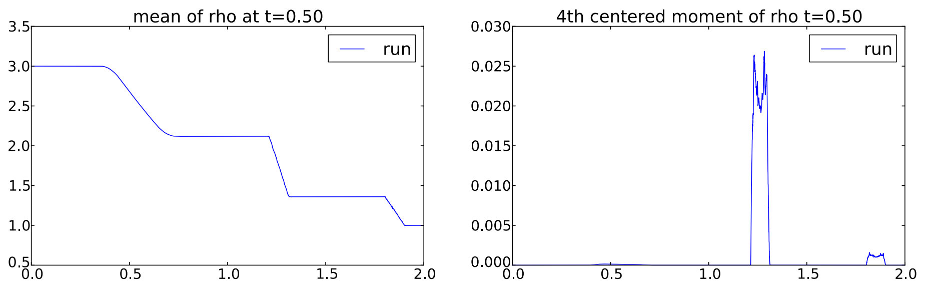


Fig. 4. The fourth centered moment of Sod shock tube problem for Euler Eq. (6.1) with random initial data (6.9). The MLMC-FVM solution is computed with a second-order WENO scheme using a HLL solver on a uniform grid in **D** = [0,2] with 4096 cells so that Δx = 1/2048. The number of levels of refinement is 9 and we use 16 samples at the finest level.

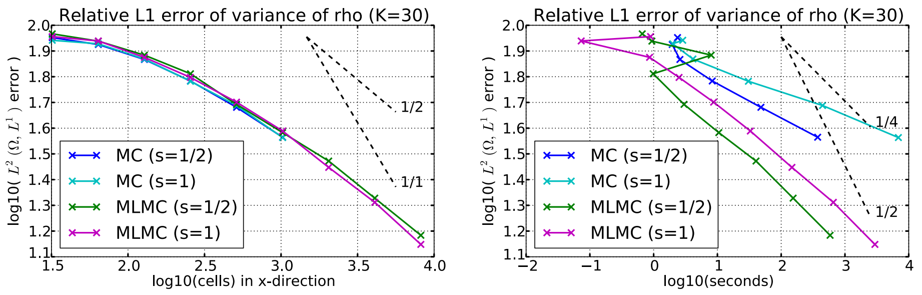
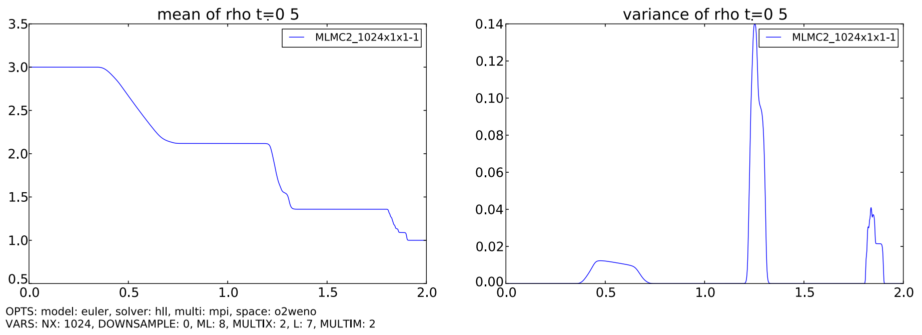


Fig. 5. MLMC and MC methods comparison with *assumed* convergence rates $s = 1/2$ and $s = 1$. We observe $s = 1$ leads to much more work for an equivalent error level, hence $s = 1/2$ is a much better choice for this particular problem and both methods.



OPTS: model: euler, solver: hll, multi: mpi, space: o2weno
VARS: NX: 1024, DOWNSAMPLE: 0, ML: 8, MULTIX: 2, L: 7, MULTIM: 2

Fig. 6. A parallel MLMC2 run with faulty seeding and short RNG period-solution becomes biased and variance is no longer symmetric around shocks. The run is performed for the 1-D Euler equations on a mesh of 1024 cells with 7 levels of refinement and 8 samples at the finest level.

7.1. Robust pseudo random number generation

Random number generation becomes a very sensitive part of Monte Carlo type algorithms on massively parallel architectures. Inconsistent seeding and insufficient period length of the RNG might cause correlations in presumably i.i.d. draws which might potentially lead to biased solutions such as in Fig. 6.

Such spurious correlations are due to two factors: firstly, for a large number of MC samples, we need a longer period and hence a larger buffer of the RNG. Secondly, the seeding of the buffer for each core must be done very carefully to preserve statistical independence.

For the numerical simulations reported below, we used the `WELL`-series of pseudo random number generators from [24,23]. These generators have been designed with particular attention towards large periods and good equidistribution. To deal with the seeding issues, we *injectively* (i.e. one-to-one) map the *unique* rank of each core to some corresponding element in the hardcoded array of prime numbers (henceforth, the array of seeds). In this way the independence is preserved. *Different* random solutions needed for error estimates (6.10) in convergence analysis can be obtained by introducing a deterministic shift on the hardcoded array of seeds; this shift must be sufficiently large to guarantee non-overlapping sets of seeds.

For all numerical experiments reported in this paper the RNG `WELL512a` was used. We found `WELL512a` to have a sufficiently large period $2^{512}-1$ and to be reasonably efficient (33 CPU sec for 10^9 draws). We emphasize that there are plenty of alternatives to `WELL512a` with even longer periods (which, however, use more memory than `WELL512a`). To name a few: `WELL1024a` with period $2^{1024}-1$, takes 34 s and `WELLRNG44497` with period $2^{44497}-1$ which takes 41 s to generate 10^9 draws.

In step 3 of the MLMC-FVM algorithm in Subsection 5.1, we solve the conservation law (2.1) for each draw of the initial data. This is performed with `ALSVID`. A massively parallel version of `ALSVID` has already been developed for deterministic problems; refer to [1] for further details. The parallelization paradigm for `ALSVID` is based on domain decomposition using message passing interface (`MPI`) standard and its particular implementation `OpenMPI`. Refer to [39,40] for detailed descriptions of `MPI` and `OpenMPI`, respectively. For the sake of consistency, we briefly recapitulate frequently used terms:

Core	An independent unit running the program in parallel with other such units. We assume one `MPI` processes per every core.
Node	Multiple cores sharing the same physical memory.
Sample	Initial data computed for some particular random draw of ω.

The key issue in the parallel implementation of the solve steps is to distribute computational work evenly among the cores. In what follows, we assume *a homogeneous computing environment* meaning that all cores are assumed to have identical CPUs and RAM per node, and equal bandwidth and latency to all other cores. Next, we describe our load balancing strategy.

7.2. Static load balancing

There are 3 levels of parallelization: over mesh resolution levels, over MC samples and *inside* the deterministic solver using domain decomposition. Domain decomposition parallelization is used only in a few largest levels with the finest mesh resolution. On these levels, the number of MC samples is small. However, these levels require most of the computational effort (unless $s = (d + 1)/2$ in (5.5) holds). For the finest mesh level $\ell = L$ we *fix* the number of cores:

$$\underbrace{cores_L}_{\#\text{ of cores}} = \underbrace{subdomains_L}_{\#\text{ of subdomains}} \times \underbrace{samplers_L}_{\#\text{ of groups for MC samples}} \tag{7.1}$$

Then, the number of cores for all the remaining levels ($cores_{L-1},\dots,cores_0$) are computed using the *a-priori* work estimates from (4.4) combined with (5.4):

$$\frac{\mathrm{Work}_{M_\ell}(\Delta x_\ell)}{\mathrm{Work}_{M_{\ell-1}}(\Delta x_{\ell-1})} \sim \frac{M_L 2^{2(L-\ell)s}\Delta x_\ell^{-(d+1)}}{M_L 2^{2(L-(\ell-1))s}\Delta x_{\ell-1}^{-(d+1)}} = \frac{2^{-2\ell s}}{2^{-2(\ell+1)s}2^{-(d+1)}} = 2^{d+1-2s} \tag{7.2}$$

In this way the positive integer parameters $subdomains_L$ and $samplers_L \leqslant M_L$ recursively determine the number of cores needed for each level $\ell < L$:

$$cores_\ell = \left\lceil \frac{cores_{\ell+1}}{2^{d+1-2s}} \right\rceil, \quad \forall \ell < L. \tag{7.3}$$

Notice, that the denominator 2^{d+1-2s} in (7.3) is a *positive integer* (a power of 2) provided $s \in \mathbb{N}/2$ and $s \leqslant (d+1)/2$ (the latter is *not* an additional constraint as it is also present in (5.6)). However, when $s < (d+1)/2$, we have:

$$2^{d+1-2s} \geqslant 2 \tag{7.4}$$

which (when L is large) leads to inefficient load distribution for levels $\ell \leqslant \ell^*$, where:

$$\ell^* := \min\{0 \leqslant \ell \leqslant L : cores_{\ell+1} < 2^{d+1-2s}\} \tag{7.5}$$

We investigate the amount of *total* work required for "inefficient" levels $\ell \in \{0,\dots,\ell^*\}$:

$$\mathrm{Work}_{\{0,\dots,\ell^*\}} := \sum_{\ell=0}^{\ell^*}\mathrm{Work}_\ell \overset{7.2}{=} \sum_{\ell=0}^{\ell^*}\frac{\mathrm{Work}_{\ell^*}}{2^{(d+1-2s)(\ell^*-\ell)}} \leqslant \sum_{\ell=-\infty}^{\ell^*}\frac{\mathrm{Work}_{\ell^*}}{2^{(d+1-2s)(\ell^*-\ell)}} = \frac{\mathrm{Work}_{\ell^*}}{1-(2^{d+1-2s})^{-1}} \overset{7.4}{\leqslant} \frac{\mathrm{Work}_{\ell^*}}{1-\frac{1}{2}} = 2\cdot\mathrm{Work}_{\ell^*} \tag{7.6}$$

For the sake of simplicity, assume that $samplers_L$ and $subdomains_L$ are nonnegative integer powers of 2. Under this assumption, definition (7.5) of ℓ^* together with recurrence relation (7.3) *without* rounding up ($\lceil\cdot\rceil$) implies that $cores_{\ell^*} \leqslant 1/2$. Hence, total work estimate (7.6) for *all* levels $\ell \in \{0,\dots,\ell^*\}$ translates into an estimate for sufficient number of cores, which, instead of ℓ^*+1, turns out to be *only 1*:

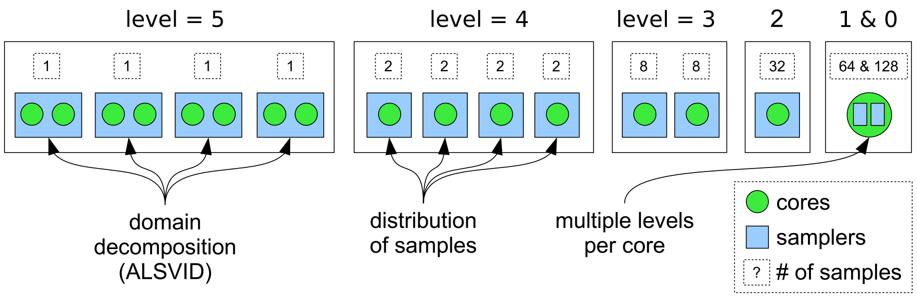


Fig. 7. Static load distribution structure.

$$\text{Work}_{\{0,\ldots,\ell^*\}} \leqslant 2 \cdot \text{Work}_{\ell^*} \rightarrow cores_{\{0,\ldots,\ell^*\}} \leqslant 2 \cdot \frac{1}{2} = 1 \tag{7.7}$$

The implementation of (7.7) (i.e. multiple levels per 1 core) is *essential* to obtain highly scalable and efficient parallelization of MLMC-FVM schemes with $2s < d + 1$.

The example of static load distribution for MLMC-FVM algorithm using all three parallelization levels is given in Fig. 7, where the parameters are set to:

$$L = 5, \quad M_L = 4, \quad d = 1, \quad s = \frac{1}{2}, \quad subdomains_L = 2, \quad samplers_L = 4.$$

This concludes the discussion of static load balancing and of step 3. In step 4 of MLMC-FVM, we combine the results to compute sample mean and variance. Special attention needs to be paid to the computation of variance.

7.3. Variance computation for parallel runs

Assume we have 2 cores A and B each computing M_A and M_B ($M = M_A + M_B$) number of samples respectively. Then keeping the notation and definitions as in the previous section, the *unbiased* estimate for mean and variance can be obtained by: (see [6])

$$\begin{aligned} E_M[u] &= \frac{M_A E_{M_A}[u] + M_B E_{M_B}[u]}{M}, \\ \delta &= E_{M_B}[u] - E_{M_A}[u], \\ \Phi^M &= \Phi^{M_A} + \Phi^{M_B} + \delta^2 \cdot \frac{M_A \cdot M_B}{M}, \\ \text{Var}_M[u] &= \frac{\Phi^M}{M-1}. \end{aligned} \tag{7.8}$$

This algorithm (7.8) is then recursively extended to arbitrary finite number of cores by combining any two of them until only one is left.

7.4. Parallelization results

To verify the reliability of the parallelization procedure, we repeat the convergence analysis simulations that were conducted with the serial version of the algorithm in the last section.

The error vs. resolution plot should be identical and the error vs. runtime plot should have longer runtimes due to parallelization overhead. We observe this behavior in Figs. 8 and 9. The runtime of the *parallel* algorithm is obtained by measuring the *wall clock time*, i.e. the total time passed during the simulation. The wall clock time is accessible as `MPI_Wtime()` routine, see [39]. It is always larger than CPU clock time since additionally it takes into account the time consumed by the operating system, networking, waiting or other auxiliary processes present on the core. In the convergence plots we use *cumulative wall clock time* (obtained by adding wall clock times from each core); this way the dependence on the used number of cores is reduced allowing for straightforward comparison of the results.

7.5. Efficiency and scaling

In Fig. 10, we investigate the parallelization efficiency for the convergence analysis conducted in the previous subsection. Parallel efficiency is defined as a fraction of *simulation time* (which excludes time spent for `MPI` communications and idling) over *wall clock time*,

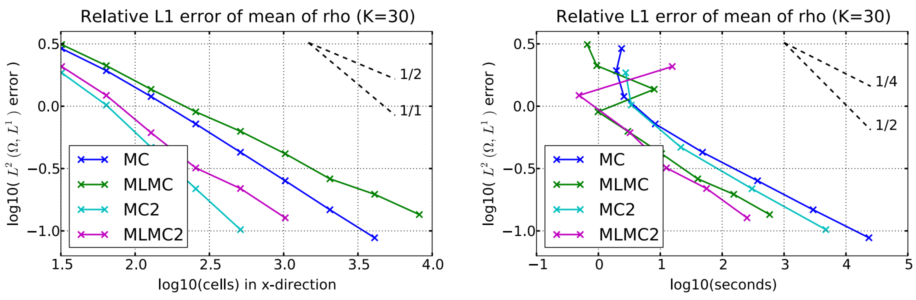


Fig. 8. Convergence of mean is consistent with convergence of mean for the *serial* version of the algorithm in Fig. 2.

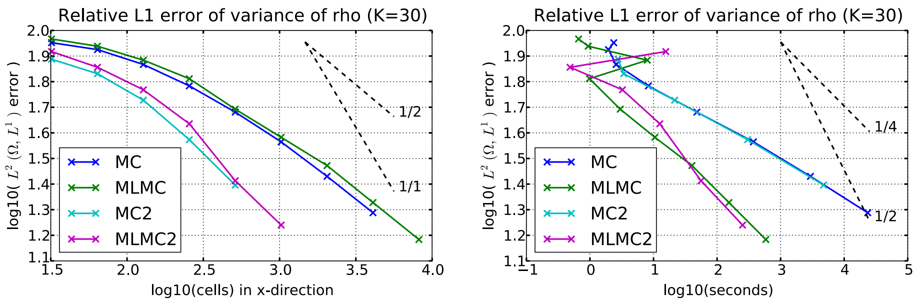


Fig. 9. Convergence of variance is consistent with convergence of variance for the *serial* version of the algorithm in Fig. 3.

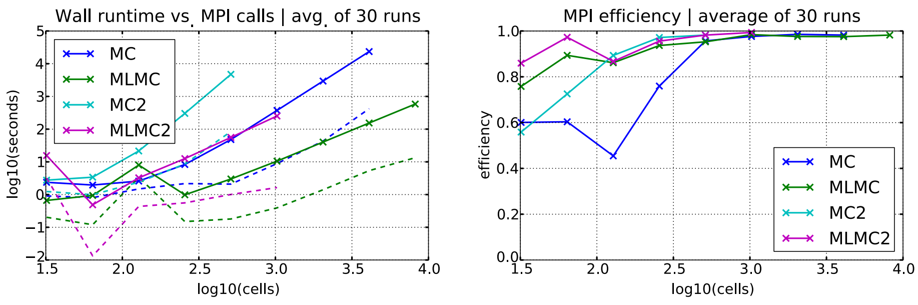


Fig. 10. MPI efficiency of the convergence analysis simulations in Figs. 8 and 9. For large problems (more than 256 cells) the parallelization is very efficient-only negligible amount of runtime is spent on auxiliary networking and waiting.

$$\text{efficiency} := 1 - \frac{(\text{total clock time of all MPI routines})}{(\#\text{cores}) \times (\text{wall clock time})}. \tag{7.9}$$

It separates the amount of time spent in computing from the amount of time spent in communicating or idling (which are indicated with *dashed* lines in runtime plots). We see that high efficiency (~98%) is achieved for large problems.

Furthermore, in Fig. 11 we verify *strong scaling* (fixed discretization and sampling parameters while increasing #cores) of the parallel algorithm and in Fig. 12 we verify *weak scaling* (problem size is equivalent to #cores) of our implementation. We observe that our implementation scales linearly (strongly) up to 4000 cores. Similarly, weak scaling is also realized up to 4000 cores. We believe that our parallelization algorithm will scale linearly for a much larger number of cores. Numerical investigation of scaling for massively parallel versions of ALSVID-UQ will be reported in forthcoming work.

8. Numerical experiments for nonlinear hyperbolic systems in 2d

The efficient parallelization of ALSVID in the previous section provides us with a robust and efficient code (ALSVID-UQ) for uncertainty quantification in hyperbolic systems of conservation laws. Consequently, we are able to conduct large scale numerical experiments in $d = 2$ (and possibly $d = 3$) space dimensions. Note that quantification of uncertainty poses a considerably larger challenge than that of a single run of the deterministic solver since a large number of samples (or stochastic dimensions in gPC based methods) are needed.

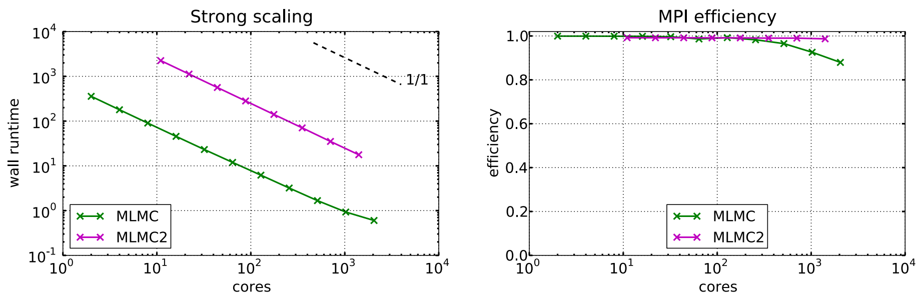


Fig. 11. Strong scaling of MLMC-FVM for Sod shock tube problem (Euler Eq. (6.1) with random initial data (6.9)). The inferior scalability of the domain decomposition method (DDM) due to additional networking between sub-domain boundaries results in lower, but still very high (~90%) efficiency when number of cores is bigger than 1000. 2nd order accurate method MLMC2 appears *not* to suffer from DDM at all.

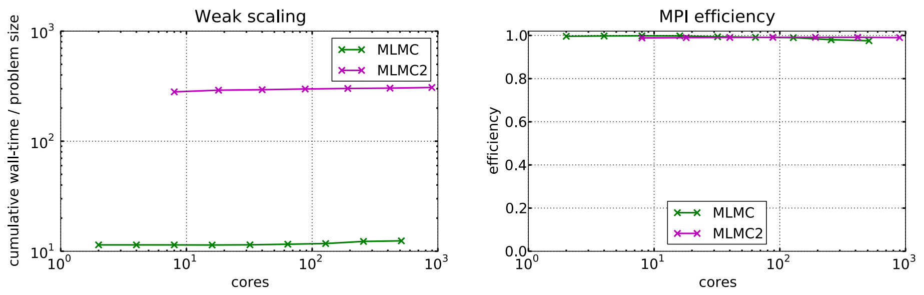


Fig. 12. Weak scaling of MLMC-FVM for Sod shock tube problem (Euler Eq. (6.1) with random initial data (6.9)). The inferior scalability of DDM has no significant influence on the overall weak scaling.

In the following, we present several numerical simulations. For each simulation, we have the following parameters:

Parameter	**Description**
L	Number of hierarchical mesh levels
M_L	Number of samples at the finest mesh level
Grid size	Number of cells in X and in Y directions
CFL	CFL number based on the fastest wave
Cores	Total number of cores used in the simulation
Runtime	Clock-time (serial runs) or wall-time (parallel runs); hrs:min:sec
Efficiency	MPI efficiency, as defined in (7.9)

Each of the above parameters is specified for every simulation in the form of a table below the corresponding figure.

8.1. Euler equations of gas dynamics

We consider Euler equation of gas dynamics (6.1) in two space dimensions. We conduct two numerical experiments for the Euler equations and report the results below.

8.1.1. Shock-vortex interaction

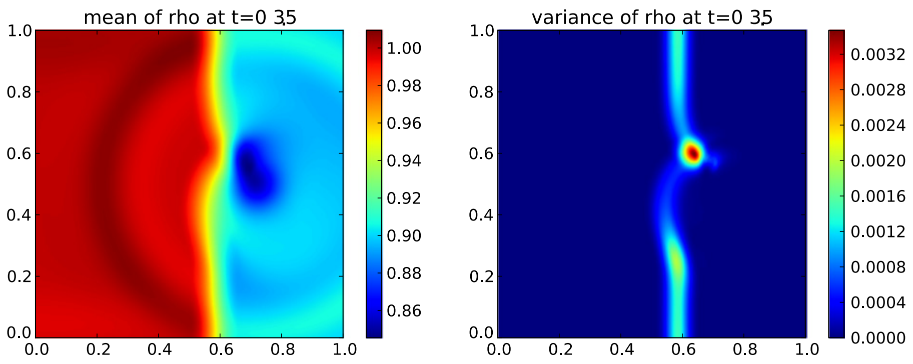
In this standard test case for deterministic solvers (see [29]), the computational domain is taken to be $\mathbf{D} = [0,1] \times [0,1]$. Let $Y \sim \frac{1}{2} + \mathcal{U}(0, \frac{1}{10})$. The initial *random* stationary Mach 1.1 shock is normal to x axis and has *uncertain location* with mean at $\mathbf{x}_1 = \frac{1}{2} + \frac{1}{20}$. The resulting initial data are:

$$\{\rho_0(\mathbf{x},\omega), \mathbf{u}_0(\mathbf{x},\omega), p_0(\mathbf{x},\omega)\} = \begin{cases} \{1, (\sqrt{\gamma}, 0)^\top, 1\} & \text{if } \mathbf{x}_1 < Y(\omega), \\ \{\frac{1}{1.1}, (1.1\sqrt{\gamma}, 0)^\top, 1 - \frac{\gamma}{10}\} & \text{if } \mathbf{x}_1 > Y(\omega). \end{cases} \tag{8.1}$$

This base flow is superposed with the a vortex centered at (x_c, y_c):

$$\bar{\mathbf{u}}_0(\mathbf{x},\omega) = (\epsilon\tau e^{\alpha(1-\tau^2)} \sin\theta, -\epsilon\tau e^{\alpha(1-\tau^2)} \cos\theta)^\top, \tag{8.2}$$

$$\bar{p}_0(\mathbf{x},\omega) = -(\gamma - 1)\frac{\epsilon^2 e^{2\alpha(1-\tau^2)}}{4\alpha\gamma\rho}, \tag{8.3}$$



L	M_L	grid size	CFL	cores	runtime	efficiency
8	16	2048x2048	0.475	128	5:47:15	91.6%

Fig. 13. Shock-vortex interaction solution at time $t = 0.35$ using MLMC.

where:

$$r = \sqrt{(\mathbf{x}_1 - x_c)^2 + (\mathbf{x}_2 - y_c)^2}, \quad \tau = \frac{r}{r_c}, \quad \sin\theta = \frac{\mathbf{x}_2 - y_c}{r}, \quad \cos\theta = \frac{\mathbf{x}_1 - x_c}{r}. \tag{8.4}$$

and we choose the parameters to be:

$$\epsilon = 0.3, \quad r_c = 0.05, \quad \alpha = 0.204, \quad x_c = 0.25, \quad y_c = 0.5. \tag{8.5}$$

The location of the initial stationary supersonic shock is uncertain in this experiment. For each realization of the initial data, the path-wise solution consists of the vortex moving to the right, interacting with the shock and emerging out of the shock. Since the shock location is uncertain, we expect the mean to be *smoother* than the path-wise solutions. A priori, it is unclear how the variance is going to be redistributed.

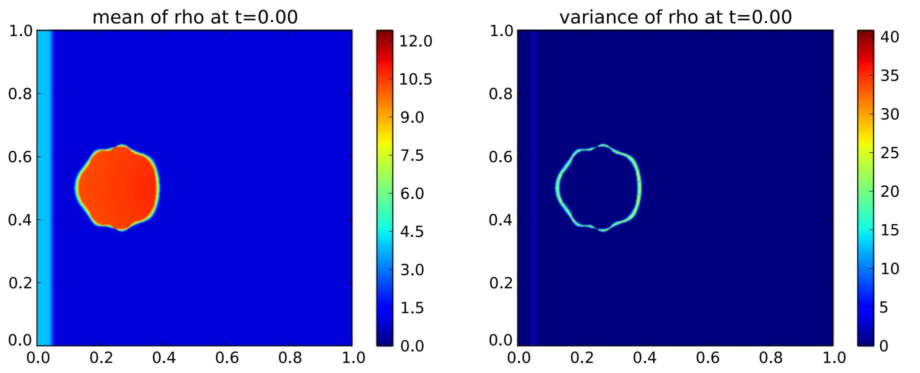
The results of uncertain shock-vortex interaction simulation at time $t = 0.35$ are given in Fig. 13. At this time instant, the vortex has just emerged out of the shock. We present results for a run with a second-order WENO discretization in physical space. The HLLC solver is used for computing the numerical fluxes. The results are computed on 9 nested levels of resolution ($L = 8$) with the finest resolution being on a 2048×2048 mesh and with timesteps reduced accordingly in order to maintain the same CFL constant over all discretization levels. The simulation is run on 128 cores and 16 samples are taken for the finest mesh resolution.

The results in Fig. 13 show that the MLMC-WENO scheme is quite robust. The mean of the pressure shows that the stationary shock (with uncertain location) is smoothed out. Similarly, the vortex (in mean) has emerged from the shock at this time instant. The vortex is resolved quite sharply. The plot for the variance of the pressure is more interesting. Although the initial variance was concentrated on the shock, we see that the variance is redistributed by the flow. A large proportion of the variance is still concentrated on the shock. However, the profile is deviated near the point where the vortex was incident at the shock. Furthermore, there is a clear signature of the vortex in the variance although the vortex (in mean) has already crossed the shock. The above experiment reveals that the variance (and possibly higher moments) can have a much more complex behavior than the mean flow field. It requires a very efficient numerical method to be able to resolve such complex features.

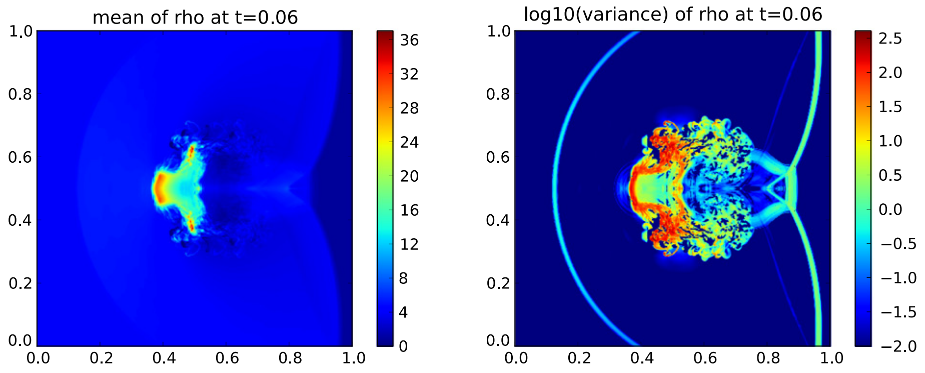
8.1.2. Cloud shock

So far, we have presented numerical experiments for the Euler equations with only one source of uncertainty-uniformly random initial shock location. In engineering applications, the number of uncertainty sources can be quite large. Stochastic Galerkin methods based on gPC expansions will be deficient at resolving such problems as the computational complexity grows exponentially with the number of uncertainty sources (number of stochastic dimensions or terms in the gPC expansion). On the other hand, methods that are based on MC-sampling scale favourably with respect to high dimension of the parameter space. As an example, consider uncertain initial data with a large number of sources for uncertainty. There is a very negligible increase of computational cost over the case of a single source of uncertainty as a random vector (with the number of components corresponding to the number of uncertainty sources) is drawn instead of a random number at each mesh point and only at the initial time step. If the boundary conditions or sources are uncertain, then the computational cost is a constant cost with a very small linear increase with respect to the number of parameters.

We put the above hypothesis to test on a problem with a high number of uncertainty sources. We consider the so-called *cloud-shock* interaction problem from [18]. The computational domain is taken to be $\mathbf{D} = [0,1] \times [0,1]$. Let $Y \sim \frac{1}{25} + \mathcal{U}(0, \frac{1}{50})$ and let $Y_1, \ldots, Y_7 \sim \mathcal{U}(0,1)$ be i.i.d. random variables independent from Y.



(a) initial data



(b) solution at $t = 0.06$

L	M_L	grid size	CFL	cores	runtime	efficiency
9	8	4096x4096	0.4	1023	5:38:17	96.9%

Fig. 14. Cloud shock at $t = 0$ and $t = 0.06$ using MLMC-FVM.

The initial data consists of an initial shock with the uncertain amplitude and uncertain location and is given by:

$$\{\rho_0(\mathbf{x},\omega),\mathbf{u}_0(\mathbf{x},\omega),p_0(\mathbf{x},\omega)\} = \begin{cases} \{3.86859 + \frac{1}{10}Y_6(\omega),(11.2536,0)^\top,167.345 + Y_7(\omega)\} & \text{if } \mathbf{x}_1 < Y(\omega), \\ \{1,(0,0)^\top,1\} & \text{if } \mathbf{x}_1 > Y(\omega). \end{cases} \tag{8.6}$$

Furthermore, a high density cloud or bubble with uncertain amplitude and uncertain shape of the form

$$\begin{aligned} &\rho_{\mathbf{0}}(\mathbf{x},\omega) = 10 + \frac{1}{2}Y_1(\omega) + Y_2(\omega)\sin\left(4\left(\mathbf{x}_1 - \frac{1}{4}\right)\right) + \frac{1}{2}Y_3(\omega)\cos\left(8\left(\mathbf{x}_2 - \frac{1}{2}\right)\right) \\ &\text{if } r \leqslant 0.13 + \frac{1}{50}Y_4(\omega)\sin\theta + \frac{1}{100}Y_5(\omega)\sin(10\theta), \end{aligned} \tag{8.7}$$

where

$$r = \sqrt{(\mathbf{x}_1 - 0.25)^2 + (\mathbf{x}_2 - 0.5)^2}, \quad \theta = \frac{\mathbf{x}_1 - 0.25}{r}, \tag{8.8}$$

lies to the right of the shock. The mean and the variance of the initial data is depicted in Fig. 14(a). Note that there are 8 sources of uncertainty in the above problem. A parametric representation of the initial data results in a 11 dimensional problem consisting of two space, one time and eight stochastic dimensions. To the best of our knowledge, such high dimensional problems have not been considered in the literature.

The mean and variance of the solution at time $t = 0.06$ is shown in Fig. 14(b). The results are from a MLMC-WENO run with 10 nested levels of resolution ($L = 9$) and the finest resolution is set to 4096×4096 mesh. The number M_L of MC samples at the finest resolution is 8 and number of cores for this run is 1023. Note that time taken for this simulation is comparable to that for the shock-vortex interaction where we had only one source of uncertainty. This justifies our claim that MLMC methods are robust with respect to a high number of sources of uncertainty. Since this two-dimensional problem on a

4096×4096 mesh is eight times more computation-intensive than the shock-vortex interaction on a 2048×2048 mesh, the simulation was executed on approximately 8 times as many cores.

The physics of the flow in this case consists of the supersonic initial shock moving to the right, interacting with the high density bubble and leading to a complex flow pattern that consists of a leading bow shock, trailing tail shocks and a very complex region (near the center) possessing sharp gradients as well as turbulent like smooth features. The mean flow (for the density) consists of the bow shock, tail shocks and a complex region with sharp gradients as well as smooth regions. The variance is concentrated in the smooth region at the center; it is significantly smaller at the tail shocks and almost vanishing at the bow shock. The initial uncertainty in the shape of the bubble seems to lead to a more complex distribution of the variance.

8.2. MHD equations of plasma physics

Next, we describe several numerical experiments for MHD equations of plasma physics.

8.2.1. Orszag–Tang vortex

The deterministic version of the Orszag–Tang vortex is the standard benchmark for MHD codes. We adapt it to include stochastic initial data. The computational domain is taken to be $\mathbf{D} = [0,2] \times [0,2]$. The standard initial data (from [31]) for the Orszag–Tang vortex is *randomly* perturbed in two different ways:

(1) **2 sources of uncertainty.** Let $Y_1, Y_2 \sim \mathcal{U}(0,1)$. The phases of the velocities are uncertain and depend on the scaled random variables Y_1, Y_2:

$$\begin{aligned} &\{\rho_0(\mathbf{x},\omega), p_0(\mathbf{x},\omega)\} = \{\gamma^2, \gamma\}, \\ &\mathbf{u}_0(\mathbf{x},\omega) = \left(-\sin\left(\pi \mathbf{x}_2 + \frac{1}{20}Y_1(\omega)\right), \sin\left(\pi \mathbf{x}_1 + \frac{1}{10}Y_2(\omega)\right)\right)^\top, \\ &\mathbf{B}_0(\mathbf{x},\omega) = (-\sin(\pi \mathbf{x}_2), \sin(2\pi \mathbf{x}_1))^\top. \end{aligned} \tag{8.9}$$

(2) **8 sources of uncertainty.** Let $Y_i \sim \mathcal{U}(-1,1),\ i = 1,\ldots,8$. The amplitudes of the initial density and pressure are uncertain

$$\begin{aligned} \rho_0(\mathbf{x},\omega) &= \gamma^2\left(1 + \frac{1}{20}Y_1(\omega)\right), \\ p_0(\mathbf{x},\omega) &= \gamma\left(1 + \frac{1}{20}Y_4(\omega)\right), \end{aligned} \tag{8.10}$$

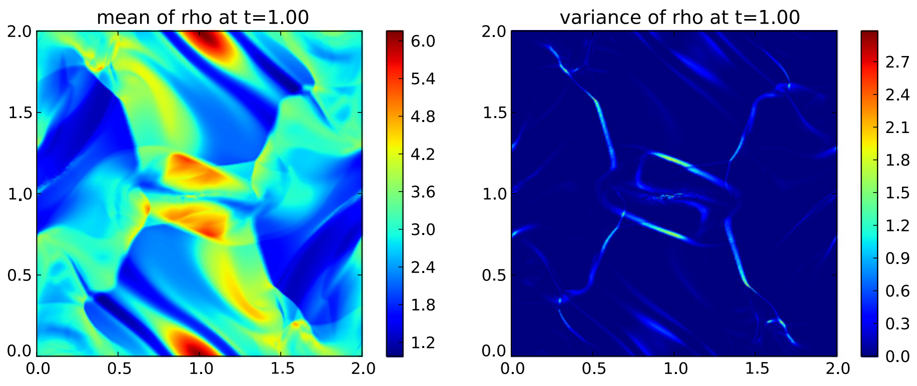
and, additionally, the phases of the initial velocities and the phases with the amplitudes of the initial magnetic fields are also uncertain,

$$\begin{aligned} \mathbf{u}_0(\mathbf{x},\omega) &= \left(-\sin\left(\pi \mathbf{x}_2 + \frac{1}{20}Y_2(\omega)\right), \sin\left(\pi \mathbf{x}_1 + \frac{1}{10}Y_3(\omega)\right)\right)^\top, \\ \mathbf{B}_1(\mathbf{x},\omega) &= -\left(1 + \frac{1}{20}Y_6(\omega)\right)\sin\left(\pi \mathbf{x}_2 + \frac{1}{25}Y_5(\omega)\right), \\ \mathbf{B}_2(\mathbf{x},\omega) &= \left(1 + \frac{1}{20}Y_8(\omega)\right)\sin\left(2\pi \mathbf{x}_1 + \frac{1}{20}Y_7(\omega)\right). \end{aligned} \tag{8.11}$$

Here, as in the setup for cloud shock problem in Fig. 14(a), a parametric representation of the initial data results in a 11 dimensional problem consisting of two space, one time and eight stochastic dimensions.

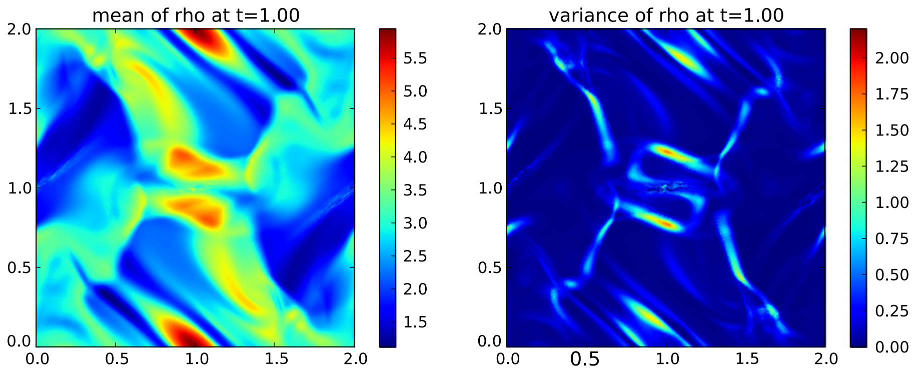
The MLMC-FVM solution is then considered for both versions of the initial data, i.e. with 2 sources (8.9) and with 8 sources (8.11) of uncertainty. The mean field and the variance (for the plasma density) of the solutions are shown in Figs. 15 and 16, respectively.

The computation is performed using the MLMC-FVM scheme with second-order WENO reconstruction, and with the HLL three wave solver of [19]. The code uses an upwind discretization of the Godunov–Powell source term. The results shown in these figures are from a computations with 8 levels of refinement ($L = 7$) and the finest mesh resolutions of 2048×2048 mesh cells and 4096×4096 mesh cells for the problem with two sources of uncertainty (8.9) and with eight sources of uncertainty (8.11), respectively. The number of MC samples at the finest resolution for both problems is 4. The problems have more than 10^9 degrees of freedom per time step and the total number of time steps is about 10^4 amounting to an overall computational cost of this simulation between 10^{12} and 10^{13} FLOPS. These numbers show that the simulations are extremely challenging and requires massively parallel architectures. In fact, the problem with 2 sources of uncertainty took about 6 hours (wall-clock) on 128 cores (simulated on ETH's parallel cluster Brutus [37]) and the problem with 8 sources of uncertainty took about 3.5 h (wall-clock) on 2040 cores (simulated on Palu, CSCS [38]). We also observe that the variance for the problem with eight sources of uncertainty is more diffused than the variance for the problem with two sources of uncertainty.



L	M_L	grid size	CFL	cores	runtime	efficiency
7	4	2048x2048	0.475	128	5:02:14	98.4%

Fig. 15. Uncertain Orszag–Tang vortex solution at $t = 1.0$ using MLMC-FVM (2 sources of uncertainty). Variance is very large near discontinuities of the path-wise solutions.



L	M_L	grid size	CFL	cores	runtime	efficiency
8	4	4096x4096	0.475	2044	3:17:18	97.0%

Fig. 16. Uncertain Orszag–Tang vortex solution at $t = 1.0$ using MLMC-FVM (8 sources of uncertainty). Again, the largest variances appear near discontinuities of the path-wise solutions.

It is well known (see [31,19]) that stable computation of numerical solutions of the Orszag–Tang problem on highly refined meshes (which, by the CFL condition (3.2) entails a correspondingly large number of timesteps) is quite challenging. Since our spatial resolution at mesh level $L = 7$ is very fine, we need an *extremely* robust code like ALSVID for the solve step in MLMC-FVM in order to resolve this problem.

The mean density is quite complicated with shocks along the diagonals of the domain as well a (smooth) current sheet at the center of the domain. The solution consists of discontinuities interspersed within interesting smooth features. Our simulations show that the variance is concentrated at shocks as well as at the current sheets and other interesting smooth regions. From this problem as well as the results of the previous section, we observe that the variance is a very good indicator of where the discontinuities and sharp gradients of the solution are concentrated and would serve as a good *a posteriori* error indicator for adaptive mesh refinement.

8.2.2. Numerical convergence analysis

We analyze these particular two dimensional numerical experiments (Orszag–Tang vortex with 2 and 8 sources of uncertainty) in greater detail. Again, we use the high-resolution MLMC-FVM simulations from Figs. 15 and 16 as the reference solutions, respectively. We investigate convergence of error vs. work in Figs. 17 and 19 for 2 sources of uncertainty and in Figs. 18 and 20 for 8 sources of uncertainty. The error in the mean field converges at expected rates. At comparable numerical resolution and accuracy, the MLMC(2) is about two orders of magnitude faster than the MC(2) method for both problems. We observe a slight deterioration in the estimated convergence rates for the variance. This could well be a pre-asymptotic effect. As seen in Figs. 19 and 20, the curves are steepening which seems to indicate better rates with further refinement.

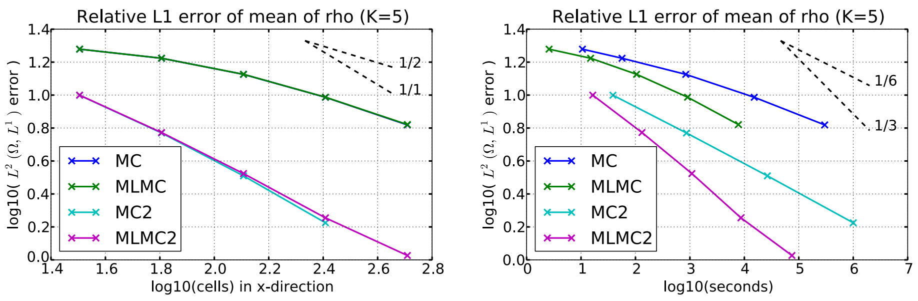


Fig. 17. Convergence of mean in the uncertain Orszag–Tang vortex simulation (2 sources of uncertainty).

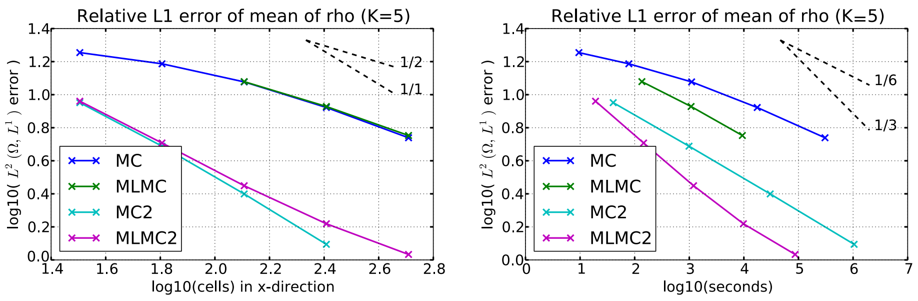


Fig. 18. Convergence of mean in the uncertain Orszag–Tang vortex simulation (8 sources of uncertainty).

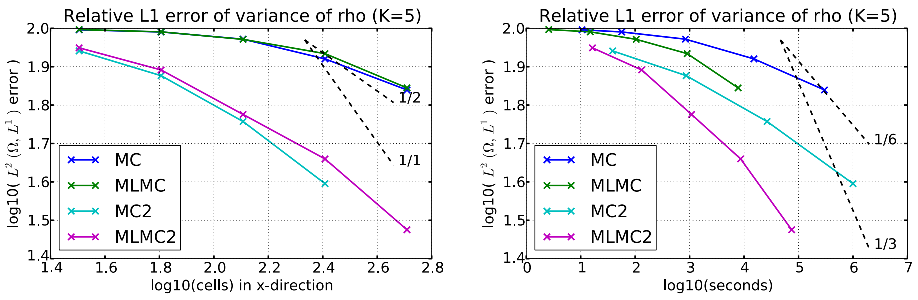


Fig. 19. Convergence of variance in the uncertain Orszag–Tang vortex simulation (2 sources of uncertainty).

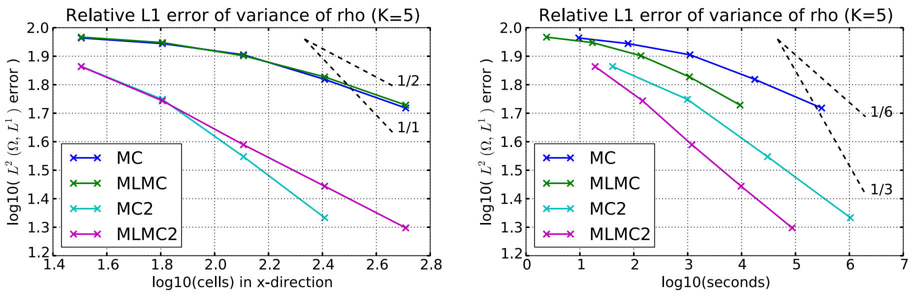


Fig. 20. Convergence of variance in the uncertain Orszag–Tang vortex simulation (8 sources of uncertainty).

Again, the MLMC(2) appears considerably faster than the corresponding MC(2) method in delivering variance estimates of comparable numerical accuracy.

Remark 8.1. Our aim in computing the Orszag–Tang vortex with two and with eight sources of uncertainty in the initial data is to compare the robustness of the MLMC method with respect to an increase in the sources of uncertainty. To this end, we plot the error vs. resolution and the error vs. runtime for the MLMC (2) method with both two and with eight sources of

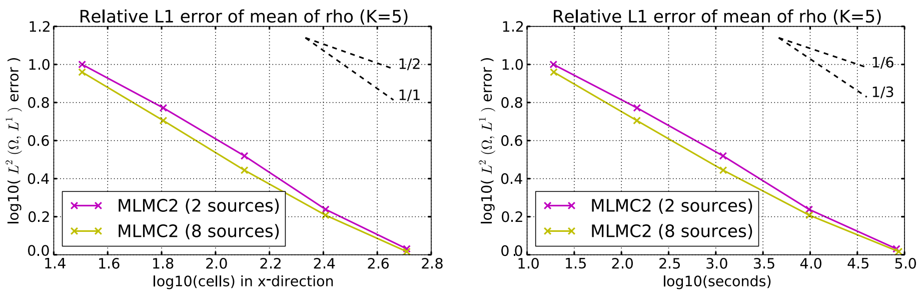


Fig. 21. Error convergence sensitivity to the number of sources of uncertainty. For both 2 and 8 sources of uncertainty, the convergence of error vs. computational work is almost identical.

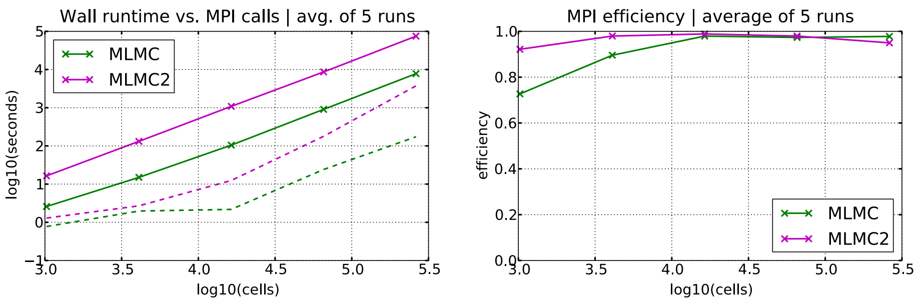


Fig. 22. MPI overhead. For large problems (more than 64 cells in each dimension) efficiency of parallelization is as good as it was for $d = 1$ experiments in Fig. 10.

uncertainty in Fig. 21. The results in this figure show that the runtime for a fixed level of error is nearly identical whether there are two or eight sources of uncertainty in the initial data. This shows that the MLMC method is quite robust with respect to large number of sources of uncertainty in the data: an increase in the number of sources of uncertainty does not lead to a deterioration in the computational efficiency. Thus, the MLMC method can be used for computing uncertainty in problems with very large number of sources of randomness.

8.2.3. Efficiency of parallelization

We test the efficiency of static load balancing for parallelization procedure described in Section 7 in this two-dimensional example. In Fig. 22, we show the parallelization efficiency of the MLMC-FVM and see that the algorithm is quite efficient and most of the time is spent computing rather than communicating or waiting.

The strong scaling for this problem is shown in Fig. 23. We see that the algorithm scales linearly up to around 4000 cores. Similarly, Fig. 24 shows a weak scaling up to a similar number of processors. We have not tested the algorithm for a larger number of processors since we are limited by the size of the machine [38], but we expect it to scale up to a much larger number of cores. The results in both one and two space dimensions show that our static load balancing algorithm is quite efficient.

8.2.4. Isothermal blast wave

In the numerical experiments presented so far, we have considered the initial randomness to be uniformly distributed. The MLMC-FVM algorithm allows, however, initial data with any probability distribution which can be sampled numerically. We demonstrate its versatility with respect to the initial probability distribution in the following experiment where the initial uncertainty is distributed normally.

The computational domain for this isothermal blast wave experiment is taken to be $\mathbf{D} = [0,1] \times [0,1]$. Let $Y \sim \mathcal{N}(0,1)$. The magnetic field from the initial data for the deterministic isothermal blast wave is perturbed-its magnitude depends on Y that is drawn from a normal distribution:

$$\rho_0(\mathbf{x},\omega) = \begin{cases} 100 & \text{if} \quad \left(\mathbf{x}_1 - \frac{1}{2}\right)^2 + \left(\mathbf{x}_2 - \frac{1}{2}\right)^2 < \left(\frac{1}{20}\right)^2, \\ 0 & \text{otherwise}. \end{cases} \tag{8.12}$$

$$\{\mathbf{u}_0(\mathbf{x},\omega), p_0(\mathbf{x},\omega), \mathbf{B}(\mathbf{x},\omega)\} = \left\{ (0,0)^\top, \rho_0(\mathbf{x},\omega), \left(\frac{5}{\sqrt{\pi}} + \frac{1}{5}Y(\omega), 0\right)^\top \right\}.$$

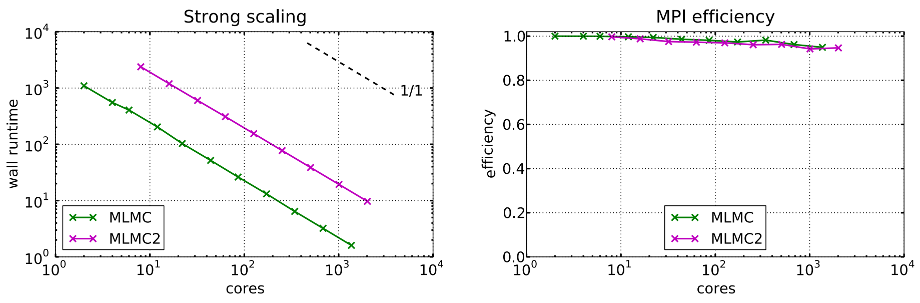


Fig. 23. Strong scaling. The inferior scalability of DDM has no significant influence on the overall strong scaling for $d = 2$.

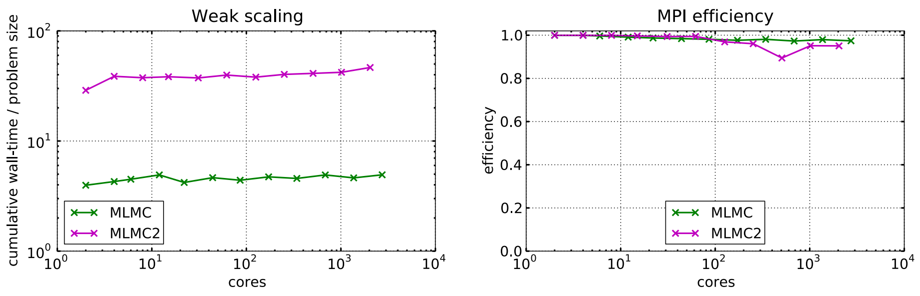
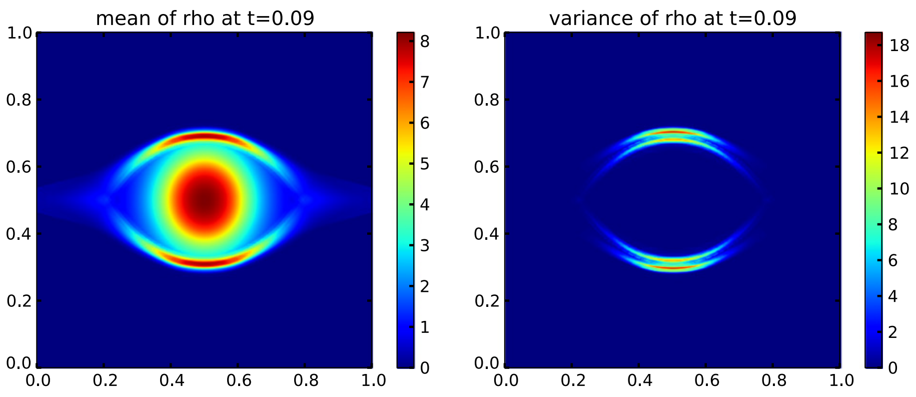


Fig. 24. Weak scaling. The inferior scalability of DDM has no significant influence on the overall weak scaling for $d = 2$.



L	M_L	grid size	CFL	cores	runtime	efficiency
6	4	1024x1024	0.475	508	5:22:50	94.0%

Fig. 25. Isothermal blast wave solution at $t = 0.09$ using MLMC-FVM with *normally* distributed perturbation of initial magnetic field.

The resulting solution is shown in Fig. 25. We present results of a simulation performed with five levels of refinement and with the mesh at the finest level consisting of 1024×1024 cells. The number of samples on the finest level is 4 and the computation is performed on 508 cores.

The results show that the mean plasma density has a rotating profile with the initial blast wave spreading out from the center of the domain. Furthermore, the variance is concentrated at the outer blast wave.

We point out that even a single deterministic run for the isothermal blast wave at mesh level $L = 6$ is known to be challenging, (see, e.g. [19]). Additionally, as the MC and MLMC are based on randomly generated initial scenarios, robustness of the path-wise solver ALSVID is crucial to handle each run. Furthermore, given the time scales of the problem, only a fast method like MLMC-FVM can handle such complex problems.

9. Conclusion

We consider systems of conservation laws in several space dimensions but with uncertain initial data. The problem of quantifying the uncertainty in such solutions is quite challenging. In this paper we extend the MLMC-FVM of [26] and present it in the context of systems of conservation laws in several space dimensions.

The algorithm is described in detail and some underlying theoretical properties (for scalar conservation laws) are discussed. Several key issues that are encountered in the efficient implementation of this algorithm are presented. They include the choice of the underlying finite volume solver. We choose the ALSVID FVM code [1] as our base code as it provides a robust and efficient approximation of the Euler and MHD equations. Another key issue is the implementation of the MLMC algorithm on parallel architectures. We discuss this issue in considerable detail and propose a novel load balancing procedure of MLMC-FVM for parallel architectures.

The MLMC-FVM is tested on a suite of one and two dimensional numerical experiments for both Euler as well as MHD equations. We demonstrate that the MLMC-FVM algorithm is robust, efficient and able to resolve very complex flows governed by systems of nonlinear hyperbolic conservation laws with uncertain initial conditions. In particular, this method can handle problems with a large number of stochastic dimensions or sources of uncertainty. Such problems are beyond the reach of other existing methods. Furthermore, the parallel version of ALSVID-UQ [2] described here is shown to scale weakly and strongly upto 4000 cores. It is expected to scale substantially beyond this range of cores. We describe several complex flows with random initial data, simulated with MLMC-FVM, and emphasize the subtle role played by the non-linearity in the evolution of the variance as well as regularity improvement of the mean due to ensemble averaging.

The main advantages of the MLMC-FVM method are its simplicity and non-intrusiveness. It is easy to code and to parallelize. The efficiency of the parallelization on homogeneous MIMD hardware can be improved by the load balancing procedure proposed in the present paper. Since the MLMC is nonintrusive, existing FVM solvers for hyperbolic systems of conservation laws can be used in their entirety. Furthermore, existing deterministic codes for solving hyperbolic systems of conservation laws can be reused in entirety. Hence, we advocate the use of MLMC-FVM algorithms for numerical quantification of uncertainty in solutions of systems of conservation laws. The source code ALSVID-UQ can be downloaded from [2].

We confine ourselves to the case of uncertain initial data in this paper. Uncertain boundary conditions and source terms can be handled analogously. The development of MLMC-FVM algorithms for this kind of uncertainty is currently in progress.

Acknowledgement

This work is performed as part of ETH interdisciplinary research grant CH1-03 10-1. CS acknowledges partial support by the European Research Council under grant ERC AdG 247277-STAHDPDE. JŠ is grateful to W. Petersen for discussions on parallelization issues. We would also like to thank teams of Brutus (ETH Zürich, [37]) and Swiss National Supercomputing Centre (CSCS, [38]) for their extensive support.

References

[1] ALSVID. Available from <http://folk.uio.no/mcmurry/amhd>.
[2] ALSVID-UQ. Available from <http://mlmc.origo.ethz.ch/>.
[3] R. Abgrall, A simple, flexible and generic deterministic approach to uncertainty quantification in non-linear problems. Rapport de Recherche, INRIA, 2007.
[4] A. Barth, Ch. Schwab, N. Zollinger, Multilevel MC method for elliptic PDEs with stochastic coefficients, Numer. Math. 119 (1) (2011) 123–161.
[5] A. Barth, A. Lang, Ch. Schwab, Multi-level Monte Carlo finite element method for parabolic stochastic partial differential equations. Preprint 2011, SAM report 2011-30, ETH Zürich.
[6] T.F. Chan, G.H. Golub, R.J. LeVeque, Updating formulae and a pairwise algorithm for computing sample variances. STAN-CS-79-773, 1979.
[7] Q.Y. Chen, D. Gottlieb, J.S. Hesthaven, Uncertainty analysis for steady flow in a dual throat nozzle, J. Comput. Phys. 204 (2005) 378–398.
[8] K.A. Cliffe, M.B. Giles, R. Scheichl, A.L. Teckentrup, Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients, Computing and Visualization in Science 14 (1) (2011) 3–15.
[9] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, D. Knuth, On the Lambert *W* Function, Advances in Computational Mathematics, vol. 5, Springer-Verlag, Berlin, New York, 1996, pp. 329–359.
[10] C.M. Dafermos, Hyperbolic Conservation Laws in Continuum Physics, 2nd ed., Springer-Verlag, 2005.
[11] M. Giles, Improved Multilevel Monte Carlo Convergence Using the Milstein Scheme, in: Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, 2008, pp. 343–358.
[12] M. Giles, Multilevel Monte Carlo path simulation, Oper. Res. 56 (2008) 607–617.
[13] M. Giles, Ch. Reisinger, Stochastic finite differences and multilevel Monte Carlo for a class of SPDEs in finance, Preprint July 2011, Oxford-Man Institute of Quantitative Finance and Mathematical Institute, University of Oxford.
[14] E. Godlewski, P.A. Raviart, Hyperbolic Systems of Conservation Laws, in: Mathematiques et Applications, Ellipses Publications, Paris, 1991.
[15] R. Eymard, Th. Gallouët, R. Herbin, Finite Volume Methods Handbook of Numerical Analysis, vol. VII, North-Holland, Amsterdam, 2000.
[16] S. Heinrich, Multilevel Monte Carlo methods, in: Large-Scale Scientific Computing, Third International Conference LSSC 2001, Lecture Notes in Computer Science, 2170, Springer-Verlag, Sozopol, Bulgaria, 2001, pp. 58–67.
[17] H. Holden, N.H. Risebro, Front Tracking for Hyperbolic Conservation Laws, Springer-Verlag, New York, 2002.
[18] U.S. Fjordholm, S. Mishra, E. Tadmor, Arbitrarily high-order essentially non-oscillatory entropy stable schemes for systems of conservation laws SIAM J. Num. Anal., accepted for publication.
[19] F. Fuchs, A.D. McMurry, S. Mishra, N.H. Risebro, K. Waagan, Approximate Riemann solver based high-order finite volume schemes for the Godunov–Powell form of ideal MHD equations in multi-dimensions, Commun. Comput. Phys. 9 (2011) 324–362.
[20] S. Gottlieb, C.W. Shu, E. Tadmor, High order time discretizations with strong stability property, SIAM Rev. 43 (2001) 89–112.
[21] R.A. LeVeque, Numerical Solution of Hyperbolic Conservation Laws, Cambridge University Press, 2005.

[22] G. Lin, C.H. Su, G.E. Karniadakis, The stochastic piston problem, PNAS 101 (2004) 15840–15845.
[23] P. L'Ecuyer, F. Panneton, Fast Random Number Generators Based on Linear Recurrences Modulo 2: Overview and Comparison, in: Proceedings of the 2005 Winter Simulation Conference, IEEE press, 2005, pp. 110–119.
[24] P. L'Ecuyer, F. Panneton, Fast random number generators based on linear recurrences modulo 2, ACM Trans. Math. Softw. 32 (2006) 1–16.
[25] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, J. Comput. Phys. 228 (2009) 3084–3113.
[26] S. Mishra, Ch. Schwab, Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data, Math. Comput. in press. Also available from <http://www.sam.math.ethz.ch/reports/2010/24>.
[27] G. Poette, B . Després, D. Lucor, Uncertainty quantification for systems of conservation laws, J. Comput. Phys. 228 (2009) 2443–2467.
[28] K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, D.L. De Zeeuw, A solution adaptive upwind scheme for ideal MHD, J. Comput. Phys. 154 (2) (1999) 284–309.
[29] C.W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. ICASE Technical report, NASA, 1997.
[30] E.F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the Harten–Lax–van Leer Riemann solver, J. Shock Waves 4 (1994) 25–34.
[31] G. Toth, The DivB = 0 constraint in shock capturing magnetohydrodynamics codes, J. Comput. Phys. 161 (2000) 605–652.
[32] J. Tryoen, O. Le Maitre, M. Ndjinga, A. Ern, Intrusive projection methods with upwinding for uncertain non-linear hyperbolic systems. Preprint, 2010.
[33] X. Wan, G.E. Karniadakis, Long-term behaviour of polynomial chaos in stochastic flow simulations, Comput. Methods Appl. Mech. Eng. 195 (2006) 5582–5596.
[34] B.P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (1962) 419–420.
[35] J.A.S. Witteveen, A. Loeven, H. Bijl, An adaptive stochastic finite element approach based on Newton–Cotes quadrature in simplex elements, Comput. Fluids 38 (2009) 1270–1288.
[36] D. Xiu, J.S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (2005) 1118–1139.
[37] Brutus, ETH Zürich, <de.wikipedia.org/wiki/Brutus_(Cluster)>.
[38] Cray XE6, Swiss National Supercomputing Center (CSCS), Manno, <www.cscs.ch>.
[39] MPI: A Message-Passing Interface Standard. Version 2.2, 2009, available from: <http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf>.
[40] Open MPI: Open Source High Performance Computing. Available from <http://www.open-mpi.org/>.