NTNU
Norwegian University of
Science and Technology

Quasi Monte Carlo Methods For Hyperbolic Conservation Laws With Uncertain Initial Data

Jon Christian Halvorsen

Master of Science in Physics and Mathematics

Submission date:	June 2016
Supervisor:	Ulrik Skre Fjordholm, MATH
Co-supervisor:	Siddhartha Mishra, ETH Zürich

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

We consider solving stochastic hyperbolic conservation laws with a quasi Monte Carlo method based on Sobol sequences. As far as we are aware, no one has done any research on this specific combination before. We extend an already available tool for uncertainty quantification, ALSVID-UQ, with the ability to perform quasi Monte Carlo using Sobol sequences. We perform numerical experiments with uncertain initial values for Burgers equation and most of the results converge with the rate $O(1/M)$ where M is the amount of samples, compared to $O(1/\sqrt{M})$ for conventional Monte Carlo methods. The convergence is measured in variance and mean. The problems are easily parallelizable, and simulations are done on a supercomputer with up to 1024 CPU cores used simultaneously.

Sammendrag

Vi løser stokastiske hyperbolske bevaringslover ved hjelp av en kvasi-Monte Carlo metode basert på Sobolsekvenser. Såvidt vi vet har denne kombinasjonen aldri vært forsket på tidligere. Vi utvider usikkerhetskvantifiseringsverktøyet ALSVID-UQ til å kunne bruke Sobolsekvenser, og dermed kunne utføre kvasi-Monte Carlo simuleringer. Vi utfører numeriske forsøk med usikre initialdata for Burgers ligning, og de fleste resultatene konvergerer med rate $O(1/M)$ der M er antall simuleringer, sammenlignet med $O(1/\sqrt{M})$ for konvensjonelle Monte Carlo metoder. Konvergensen er målt i varians og gjennomsnitt. Problemene er parallellskalerbare, og simuleringer er kjørt ved opptil 1024 kjerner samtidig.

Preface

This thesis completes my Master of Science degree within Applied Physics and Mathematics with specialization in Industrial Mathematics. The work for this thesis has been carried out in the spring semester of 2016, with the opportunity of being a visiting researcher at ETH Zurich in Switzerland. The rest of the work has been done at the Norwegian University of Science and Technology (NTNU).

I would like to take this opportunity to thank my supervisor, Ulrik Fjordholm, for all the great support he has given my throughout these months. I would also like to thank Prof. Dr. Siddhartha Mishra who acted as a co-supervisor and host for my stay at ETH. Without access to the resources provided by the Seminary for Applied Mathematics, ETH, this thesis would probably look very different. My stay at ETH couldn't have been any better.

I would also thank Kjetil Lye Olsen, fellow Norwegian and current maintainer of the software package used extensively in this thesis, ALSVID-UQ [1]. He has had to endure a lot of detailed questions, only surpassed by his ability to answer them. Thanks for really thorough answers.

Jon Christian Halvorsen
June 2016, Trondheim

Contents

Chapter 1

Introduction

In recent years, the development of efficient simulation methods for hyperbolic conservation laws with uncertain initial values has been studied extensively. Examples of conservation laws include the Euler equations in fluid dynamics, conservation of energy, traffic flow and shallow water waves. Conservation laws are also used extensively in medium-range weather forecasting (around 10 days into the future). The European Centre for Medium-range Weather Forecasts (ECMWF) experimented with a Monte Carlo method already in 1979 [2], but the results were inconclusive. The current model, called Ensemble prediction [3], can be viewed as a specialized Monte Carlo procedure, generating results from an ensemble consisting of up to 51 perturbations of the full operational model. The results from this model enables the weather prediction to be more correct, but it also enables an estimate about how certain the model is. This is the reason the medium-range forecasts can be labeled with a high, medium or low degree of probability of being correct.

Work has recently been done in speeding up Monte Carlo simulations, for example by utilizing a technique called multi-level Monte Carlo (MLMC) ([4], [5]). The MLMC method makes the Monte Carlo method more effective by doing less work for the same accuracy. The quasi Monte Carlo method's premise is a possibly better convergence rate than conventional Monte Carlo. As far as we are aware, nobody has researched this specific combination before; conservation laws and quasi Monte Carlo. We are interested in whether the quasi Monte Carlo method can be used as a drop-in replacement for the Monte Carlo method in the specific use case that is conservation laws with uncertain initial values.

We begin by presenting theory for conservation laws in Chapter 2, while theory for Monte Carlo and quasi Monte Carlo methods are presented in Chapters 3 and 4. How to generate a Sobol sequence, our quasi random number generator of choice, is presented in Chapter 5. Combining the previous topics, and giving an example, is considered in Chapter 6 while the implementation work is described in Chapter 7. Finally, numerical experiments and conclusion comes in Chapters 8 and 9.

Chapter 2

Conservation Laws

In this section conservation laws and numerical methods for them will be presented. For more in-depth knowledge and understanding of the subject we refer to Holden and Risebro [6]. For other details the lecture notes by S. Mishra [7] are a good resource. The following chapter also builds on previous work done by the author [8].

2.1 Introduction to Conservation Laws

Consider a probability space $\Omega \subset \mathbb{R}^n$ and a quantity of interest u that is defined in all points $x \in \Omega$. In conservation laws, this quantity typically is a temperature, a pressure or a density. For a fixed sub-domain $\omega \subset \Omega$ of any size, the rate of change of u over time is equal to the total amount of u produced (or destroyed) inside ω and the flux across the boundaries. An integral representation of this would be

$$\frac{d}{dt}\int_\omega u dx = -\underbrace{\int_{\delta\omega} F \cdot \nu d\sigma(x)}_{\text{flux}} + \underbrace{\int_\omega S dx}_{\text{source or sink}} \quad ,$$

with ν being the outward normal, $d\sigma(x)$ being the surface measure, F and S being the flux and source. Simplifying this integral by using the divergence theorem we obtain

$$\frac{d}{dt}\int_\omega u dx + \int_\omega \operatorname{div}(F) dx = \int_\omega S dx.$$

Since this holds for all sub-domains $\omega \subset \Omega$, using an infinitesimal ω obtains the following differential equation, known as a *balance law*:

$$\begin{aligned} \partial_t u + \operatorname{div}(F) &= S & \forall(x,t) \in (\Omega, \mathbb{R}_+) \\ u(x,0) &= u_0(x). \end{aligned}$$

If the source (or sink) is zero, we classify this as a conservation law (since u only changes by flux entering or leaving the domain) and it takes the form

$$\begin{aligned} \partial_t u + \operatorname{div}(F) &= 0 & \forall(x,t) \in (\Omega, \mathbb{R}_+) \\ u(x,0) &= u_0(x). \end{aligned} \tag{2.1}$$

2.2 Examples of Conservation Laws

The *scalar transport equation*

$$\partial_t u + \operatorname{div}(a(x,t)u) = 0 \tag{2.2}$$

governs flow over a velocity field $a(x,t)$. If we let u be a concentration of a pollutant in the air, wind will generate the velocity field $a(x,t)$ and the pollutant will be transported with the wind. In a simple case where we consider one dimension and the velocity field is constant, (2.2) reduces to

$$\partial_t u + a\partial_x u = 0,$$

which is often called the transport or *advection equation.*

Given a metal rod that is heated in the center, you would assume that the heat (u) spreads out and in time you get a uniform heat distribution on the rod. For a general medium this diffusion is described by Fick's law:

$$F(u) = -k\nabla u,$$

where k is the conductivity tensor for the medium. Since heat flows from hotter to colder zones we get a minus sign. Plugging this into the general conservation law (2.1) we get

$$\partial_t u - \operatorname{div}(k\nabla u) = 0,$$

which is called the *heat equation.* This equation governs the transportation of heat over time. The heat equation can also model particle diffusion, Brownian motion and a lot of other equations can be reduced to it. The heat equation is a second order partial differential equation. From here and out we only look at first order scalar partial differential equations.

2.3 Properties of Conservation Laws

We look at the first order scalar conservation law

$$\begin{aligned} \partial_t u + \partial_x f(u) &= 0 \\ u(x,0) &= u_0(x). \end{aligned} \tag{2.3}$$

Since the solution (or even the initial data) do not need to be smooth, we multiply with any test function $\phi \in C_c^1(\mathbb{R} \times \mathbb{R}^+)$ and integrate by parts to get

$$\int_{\mathbb{R}\times\mathbb{R}^+} u\partial_t\phi + f(u)\partial_x\phi dxdt + \int_{\mathbb{R}} u_0(x)\phi(x,0)dx = 0. \tag{2.4}$$

Definition 2.1. *Any function $u \in L^1(\mathbb{R} \times \mathbb{R}^+)$ that satisfies* (2.4) *is called a weak solution to* (2.3)*.*

Weak solutions are not necessarily unique. Enforcing an *entropy condition*,

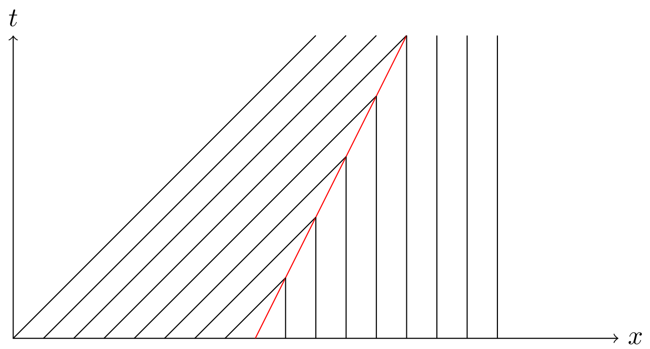


Figure 2.1: Characteristics of a shock solution. We see that as time passes, the location of the shock will move to the right.

such as the *Lax entropy condition*

$$f'(u^-(t)) > s(t) > f'(u^+(t)), \tag{2.5}$$

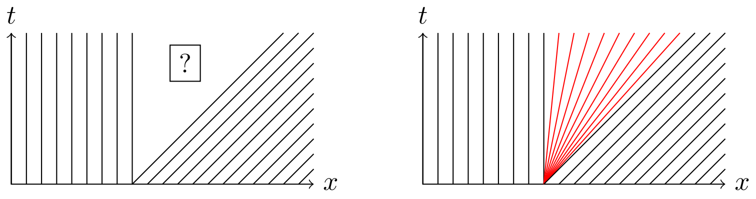
we get an *entropy solution.* Here $s(t)$ is the shock speed while $u^\pm$ is the states on either side of the shock. More general entropy conditions also exist.

Definition 2.2. *If a weak solution also satisfy an entropy condition, such as the Lax entropy condition, this is an entropy solution.*

Solutions of conservation laws can be constructed using the method of characteristics. For decreasing initial data with a convex flux, a *shock* will arise. This is visible as a discontinuity in the solution and the fact that the characteristic lines cross each other. See Figure 2.1 for an example characteristic plot where a shock exists. The traveling speed of the shock is determined by the *Rankine-Hugoniot condition*

$$s(t) = \frac{f(u^+(t)) - f(u^-(t))}{u^+(t) - u^-(t)}. \tag{2.6}$$

If, on the other hand, the characteristics do not collide, but rather spreads out from an area, a *rarefaction wave* is created, see Figure 2.2a for an example.



(a) A rarefaction wave with undecided characteristics in the center.

(b) A rarefaction wave satisfying the Lax entropy condition (2.5).

Figure 2.2: Characteristics of a rarefaction solution. We see that as time passes, the left side of the rarefaction will stay at the same point, while the right side travels further right and extends the area the rarefaction covers.

This type of wave will emerge for example if you have increasing initial data with a convex flux. Many valid weak solutions exists, but we need a unique entropy solution. Utilizing Rankine-Hugoniot (2.6) and the Lax entropy condition (2.5) the entropy solution is to "fill out" the empty area with a fan shape starting in the origin of the rarefaction. See Figure 2.2b for an illustration of the entropy solution.

2.4 Finite volume schemes

In this section we will develop a finite volume scheme for the one dimensional scalar conservation law(2.3). Since equations of this type does not necessarily have continuous solutions we search for weak solutions and define control cells instead of the usual grid points. This is done because an average cell value better maps discontinuous data. For $x \in [x_L, x_R]$ we define the discrete points as

$$x_j = x_L + \left(j + \frac{1}{2}\right)\Delta x, \qquad j = 0, \dots, N, \qquad \Delta x = \frac{x_R - x_L}{N+1},$$

and the control volumes as

$$C_j = \big[x_{j-1/2}, x_{j+1/2}\big)$$

where $x_{j\pm 1/2}$ are the midpoints between two discrete neighboring grid points. We also use a uniform discretization in time,

$$t^n = n\Delta t.$$

Since the solutions to (2.3) might be discontinuous, instead of looking at point values we try to update the control volumes:

$$U_j^n \approx \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t^n)dx$$

for each time step t^n. Integrating (2.3) over $x \in [x_{j-1/2}, x_{j+1/2}]$ in space and $t \in [t^n, t^{n+1}]$ in time yields

$$\int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} \partial_t u dx dt + \int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} \partial_x f(u) dx dt = 0.$$

Defining the control volume flux

$$F_{j+1/2}^n = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{j+1/2}),t)dt \tag{2.7}$$

we get that

$$\begin{aligned}
\int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} \partial_t u dx dt &= -\int_{t^n}^{t^{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} \partial_x f(u) dx dt \\
\int_{t^n}^{t^{n+1}} \partial_t \Delta x U_j^n dt &= \int_{x_{j-1/2}}^{x_{j+1/2}} \partial_x \Delta t F_j^n dx \\
\Delta x (U_j^{n+1} - U_j^n) &= \Delta t (F_{j+1/2}^n - F_{j-1/2}^n) \\
U_j^{n+1} &= U_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^n - F_{j-1/2}^n\right). \qquad (2.8)
\end{aligned}$$

In (2.7) we define flux passing in and out from a control volume from time t^n to t^{n+1}. The change in U_j^n in the same timespan is the flux passing through the endpoints of the interval. This means that (2.8) is a conservation statement. Since the fluxes are unknown they need to be approximated. This approximation depends on what kind of solver one would want to use.

To approximate (2.8) Gudonov [9] assumed that the control volume averages are constant in each volume C_j and therefore the interfaces between them define a *Riemann problem*:

$$\begin{cases} \partial_t u + \partial_x f(u) & = 0 \qquad\qquad (x \in \mathbb{R}, \quad t > t^n) \\ u(x, t^n) & = \begin{cases} U_j^n & \text{if } x < x_{j+1/2} \\ U_{j+1}^n & \text{if } x > x_{j+1/2}. \end{cases} \end{cases} \qquad (2.9)$$

Our solution is thus a superposition of Riemann problems (2.9). As long as (x, t) is reasonably close to $(x_{j+1/2}, t^n)$ the Riemann problem closely approximates a conservation law made up of several control volumes. The solution of the Riemann problems forms shock waves, rarefactions and compound waves. To prevent the waves from affecting each other we impose the Courant–Friedrichs–Lewy condition, also called the *CFL-condition*:

$$\max_j |f'(u)| \frac{\Delta t}{\Delta x} \leq \frac{1}{2}.$$

This can be inferred from the fact that the solution waves have a finite speed that is bounded by $\max_j |f'(u_j^n)|$. Solving each of the Riemann problems with the CFL-condition active guarantees that waves from adjacent cells do not interact.

2.5 Approximate Riemann solver

We approximate a solution to the Riemann problem by using two waves: one traveling right with speed $s_{j+1/2}^l$ and one traveling left with speed $s_{j+1/2}^r$. This is also called a *two-wave solution.* The approximate solution to the Riemann problem (2.9) is:

$$u_0(x) = \begin{cases} u_j^n & \text{if} \quad x < s_{j+1/2}^l t \\ u_{j+1/2}^* & \text{if} \quad s_{j+1/2}^l t < x < s_{j+1/2}^r t \\ u_{j+1}^n & \text{if} \quad x > s_{j+1/2}^r t. \end{cases}$$

We determine the exact solution to the middle state using the Rankine-Hugoniot condition (2.6):

$$\begin{aligned} f(u_{j+1}^n) - f_{j+1/2}^* &= s_{j+1/2}^r \left(u_{j+1}^n - U_{j+1/2}^* \right) \\ f(u_j^n) - f_{j+1/2}^* &= s_{j+1/2}^l \left(u_j^n - U_{j+1/2}^* \right), \end{aligned}$$

with $f_{j+1/2}^*$ being the middle flux. These equations represents a system which can be solved for $f_{j+1/2}^*$, and if we take the wave speeds to be equal in magnitude but with opposite sign ($s^r = -s^l = s$) we get the simple expression

$$f_{j+1/2}^* = \frac{f(u_j^n) + f(u_{j+1}^n}{2} - \frac{s_{j+1/2}}{2} \left(u_{j+1}^n - u_j^n \right).$$

This is one way to approximate the flux function for a control volume (2.7):

$$F_{j+1/2}^n \approx f_{j+1/2}^*.$$

2.6 Lax-Friedrichs flux

The maximum allowed wave speeds are chosen to ensure neighboring Riemann problems do not interact, and the maximum allowed speeds are

$$s^l_{j+1/2} = -\frac{\Delta x}{\Delta t}, \qquad s^r_{j+1/2} = \frac{\Delta x}{\Delta t}. \tag{2.10}$$

The wave speeds (2.10) lead to the flux function for Lax-Friedrichs scheme:

$$F_{j+1/2} = \frac{f(U^n_j) + f(U^n_{j+1})}{2} - \frac{\Delta x}{2\Delta t}\left(U^n_{j+1} - U^n_j\right), \tag{2.11}$$

and so the full Lax-Friedrichs scheme is given by (2.8) and (2.11).

2.7 Rusanov flux

The Lax-Friedrichs scheme uses the same wave speeds over the whole domain, which is not strictly necessary. Another scheme, the Rusanovs scheme (also called local Lax-Friedrichs) chooses the speeds based on a subset of the grid:

$$s_{j+1/2} = \max\left(|f'(U^n_j)|, |f'(U^n_{j+1})|\right),$$

now assuming that $f(\cdot)$ is convex (which is not strictly needed, but makes the flux much easier to evaluate). The Rusanov method selects the wave speeds based on local data, not on the whole domain like Lax-Friedrich does. The Rusanov flux is thus given as:

$$F^{\text{Rus}}_{j+1/2} = \frac{f(U^n_j) + f(U^n_{j+1})}{2} - \frac{\max\left(|f'(U^n_j)|, |f'(U^n_{j+1})|\right)}{2}\left(U^n_{j+1} - U^n_j\right), \tag{2.12}$$

and the full Rusanov scheme is given by (2.8) and (2.12).

2.8 HLL flux

Harten, Lax and van Leer flux [10] is given by

$$F^{\mathrm{HLL}}_{j+1/2} = \frac{s^r_{j+1/2} f(U^n_j) - s^l_{j+1/2} f(U^n_{j+1}) - s^r_{j+1/2} s^l_{j+1/2} (U^n_{j+1} - U^n_j))}{s^r_{j+1/2} - s^l_{j+1/2}},$$

where $s^r_{j+1/2}$ and $s^l_{j+1/2}$ still are the wave speeds. Note that when solving Burgers' equation we have $f(U_n) = U_n$, and if the wave speeds are of equal magnitude but opposing sign $s = s^r_{j+1/2} = -s^l_{j+1/2}$, we have that

$$\begin{aligned} F^{\mathrm{HLL}}_{j+1/2} &= \frac{sf(U^n_j) + sf(U^n_{j+1})}{2s} - \frac{s^2(U^n_{j+1} - U^n_j)}{2s} \\ &= \frac{f(U^n_j) + f(U^n_{j+1})}{2} - \frac{s(U^n_{j+1} - U^n_j)}{2} \\ &= F^{\mathrm{Rus}}_{j+1/2}. \end{aligned}$$

For these conditions the HLL flux is equal to the Rusanov Flux (2.12).

Chapter 3

Monte Carlo methods

Monte Carlo methods are used extensively in physics, mathematics, chemistry, biology, geology and finance. This section will present the main results from the method. For a deeper understanding of the subject, Caflisch's [11] review article is excellent.

The integral of a one dimensional Lebesgue integrable function $f(\omega) : \mathbb{R} \to \mathbb{R}$ with respect to a probability density $p(\omega)$ can be expressed as

$$I[f] = \int_{\mathbb{R}} f(\omega)p(\omega)d\omega,$$

with $\omega \in \mathbb{R}$. The *cumulative distribution function*, or CDF, is defined as

$$P(x) = \int_{-\infty}^{x} p(\omega)d\omega. \tag{3.1}$$

The expectation of a function can be formulated as an integral, so

$$E[f] = I[f].$$

We further define the Monte Carlo integral as

$$I_M[f] = \frac{1}{M} \sum_{m=1}^{M} f(x_m(\omega)), \tag{3.2}$$

where $\{x_m(\omega)\}_{m=1}^{M}$ is M independently sampled numbers from $p(\omega)$

Theorem 3.1. *With $I[f]$ being the integral of a Lebesgue integrable function and $I_M[f]$ the Monte Carlo approximation to the same integral we have that*

$$\lim_{M \to \infty} I_M[f] = I[f]$$

almost surely.

The above result comes directly from the Strong Law of Large Numbers [11].

Definition 3.2. *The integration error for the Monte Carlo method is*

$$\epsilon_M = |I[f] - I_M[f]| \, .$$

Using the Central Limit Theorem we can bound the Monte Carlo error from Definition 3.2:

Theorem 3.3 (Caflisch [11])**.** *The Monte Carlo error ϵ_M is*

$$e_M \approx \sigma M^{-1/2} \nu, \tag{3.3}$$

where ν is a standard normal random variable and $\sigma = \sigma(f)$ is the square root of the variance (or standard deviation) of f:

$$\sigma(f) = \left(\int_\Omega (f(\omega) - I[f])^2 \, d\omega \right)^{1/2} .$$

Letting $\boldsymbol{\omega} \in \mathbb{R}^n$, all the above still holds in multiple dimensions and we get an error term that scales as $\mathrm{O}(M^{-1/2})$, independent of the spatial dimension. Being independent of the spatial dimension makes the Monte Carlo method easy to work with.

Chapter 4

Quasi Monte Carlo

Even though the Monte Carlo method is easy to work with it has a somewhat slow convergence. The error bound on the Monte Carlo method (3.3) is a probabilistic error bound. If a random sequence of numbers can give an average error scaling with $O(M^{1/2})$, surely there must exist some sequence that makes the method converge faster. The quasi Monte Carlo method tries to take advantage of this fact, and in this section we hunt for number sequences that can satisfy this requirement.

To perform the Quasi Monte Carlo method we substitute the pseudo random numbers from the Monte Carlo method with numbers from a *low discrepancy sequence.* We begin by normalizing the integration domain to be the unit cube, so the integral we want to approximate is

$$I(f) = \int_{[0,1]^s} f(\boldsymbol{x}) d\boldsymbol{x}.$$

We further define the Quasi Monte Carlo integral just as we defined the ordinary Monte Carlo integral (3.2)

$$I_M^Q(f) = \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{x}_m), \tag{4.1}$$

only now the integral takes a predetermined sequence as input and not pseudo random numbers in the evaluation of $f(\cdot)$. The sequence $\{\boldsymbol{x}_m\}_{m=1}^{M}$, where each $\boldsymbol{x}_m$ is a point in $[0,1]^s$, must be uniformly distributed. Sobol ([12] and [13]) presents three main requirements such a sequence must satisfy to be suitable for quasi Monte Carlo:

- Best uniformity distribution as $M \to \infty$.
- Good distribution for fairly small M.
- A very fast computation algorithm.

To measure uniformity of a sequence, or how well the stochastic room is filled, we define the *discrepancy*:

Definition 4.1. *The discrepancy of any sequence* $\{\boldsymbol{x}_m\}_{m=1}^{M}, \boldsymbol{x}_m \in [0,1]^s$ *is*

$$R_M(J) = \frac{1}{M}\left(\#\{\boldsymbol{x}_m \in J\} - v(J)\right),$$

where J *is any subset of* $[0,1]^s$, $\#(\cdot)$ *is the counting function and* $v(J)$ *is the volume of* J.

In other words, $R_M(J)$ is the error in measuring the volume of J using a Monte Carlo method.

We restrict J to be a rectangular set with sides parallel to the coordinate axes. Let the set of all such subsets be denoted by E. A rectangular set is defined by the points of two opposing corners (x, y), so we can write $J = J(\boldsymbol{x}, \boldsymbol{y})$. Define two different measures of discrepancies, the L^∞ norm

$$D_M = \sup_{J\in E} |R_M(J)|,$$

and the star discrepancy

$$D_M^* = \sup_{J \in E^*} |R_M(J)|,$$

where E^* is all the sets with one vertex at zero, $E^* = \{J(0, \boldsymbol{y})\}$. Since $E^* \subset E$ we have that $D_M^* \leq D_M$.

The discrepancy can be seen as the counterpart to $M^{-1/2v}$ part of the Monte Carlo error (3.3). There is also a direct link between the discrepancy and uniformity of a sequence:

Definition 4.2. *A sequence $\{\boldsymbol{x}_m\}_{m=1}^M$ is uniformly distributed if*

$$\lim_{N \to \infty} D_M = 0.$$

The variation of a function measures how much the function varies on an interval. It can seen as the counterpart of the variation (or standard deviation) part of the Monte Carlo error (3.3).

Definition 4.3. *The Hardy-Krause variation of f, in one dimension, is*

$$V[f] = \int_0^1 \left| \frac{df}{dt} \right| dt,$$

and in s dimensions

$$V[f] = \int_{I^s} \left| \frac{d^s f}{dt_1 \cdots dt_s} \right| dt_1 \cdots dt_s + \sum_{i=1}^{s} V[f_1^{(i)}],$$

where $f_1^{(i)}$ is f restricted to $x_i = 1$.

Definition 4.4. *The integration error for the Quasi Monte Carlo integration be*

$$\epsilon_M^Q(f) = |I(f) - I_M^Q(f)|.$$

Theorem 4.5 (The Koksma-Hlawka Theorem)**.** *For any sequence $\{\boldsymbol{x}_m\}_{m=1}^M$*

and any function f with bounded variation, the integration error is bounded by

$$\epsilon(f) \leq V[f]D_M^* \tag{4.2}$$

.

Notice that we use the definition with the star discrepancy, which is a tighter bound than if we used the normal discrepancy D_M. The Koksma-Hlawka Theorem is valid for any sequence, but for a Quasi Random sequence the bound is

$$\epsilon_M^Q(f) \leq V[f]D_M^*.$$

Comparing the Koksma-Hlawka inequality for error in the Quasi Monte Carlo integration (4.2) and the Monte Carlo error (3.3), there are some similarities and some important differences. First, the Koksma-Hlawka is an inequality and therefore a strict upper bound, while the Monte Carlo error is a probabilistic error. Both the bounds consist of one factor depending on the sequence ($M^{-\frac{1}{2}}$ for Monte Carlo, and D_M^* for Koksma-Hlawka) and one factor depending on f ($\sigma(f)$ and $V[f]$ respectively).

Definition 4.6. *A sequence is quasi random if*

$$D_M \leq c(\log M)^k M^{-1},$$

where c and k are independent from M, but may depend on s.

Often we only call the sequences quasi random if $k = s$. By Theorem 4.5 these quasi random sequences will have an error

$$\epsilon_M^Q \leq cV[f](\log M)^s M^{-1},$$

but the practical error is typically much lower. What we have experienced in practice, and aim to fulfill here is an error scaling as $O(M^{-1})$.

The first candidate for such a quasi random sequence was presented by Halton [14] in 1960. Sobol [15] sequences were presented in 1967, and Faure sequences [16] in 1982. The implementation and efficiency of the Faure and Halton sequences are discussed by Fox [17], while the Sobol sequence is briefly mentioned as another good candidate. Newer additions include lattice methods and higher order digital nets for sampling, but from here we focus on the Sobol sequence.

Chapter 5

Sobol Sequence Generator

We want to generate a quasi random sequence in the unit cube for some target dimension s, each with M realizations. This could seem like a trivial task, but may need more work than we realize. To make a low-discrepancy sequence in one dimension we could make a regular grid $x_j = j/N$ where N is the number of grid points. This would however violate Sobol's second requirement if we suddenly only wanted $M/2$ points and are stuck with points only in the left half of our domain. Another strategy could be halving our domain and picking a midpoint each time. In fact, the first Sobol dimension does something like that, see Table 5.1 for a list of the first 10 Sobol numbers in dimension one. We can not, however, do this for all the dimensions our sequence. An example of doing that with only 2 dimensions can be shown in Figure 5.1. *Sobol's property A* can guarantee some uniformity.

Definition 5.1 (Sobol's property A [18])**.** *Divide the unit cube* $[0,1]^s$ *into* 2^s *multi dimensional sub-cubes. Let* $P_0, P_1 \cdots$ *be a sequence of points and let* A^s *be a binary segment 5.2 of this sequence with length* 2^s*. The sequence* $P_0, P_1 \cdots$ *possesses Sobol's Property A if all points in* A^s *lies in different sub-cubes.*

Table 5.1: The 10 first realizations of the Sobol sequence in dimension 1.

i	x_i
1	0.5
2	0.75
3	0.25
4	0.375
5	0.875
6	0.625
7	0.125
8	0.1875
9	0.6875
10	0.9375

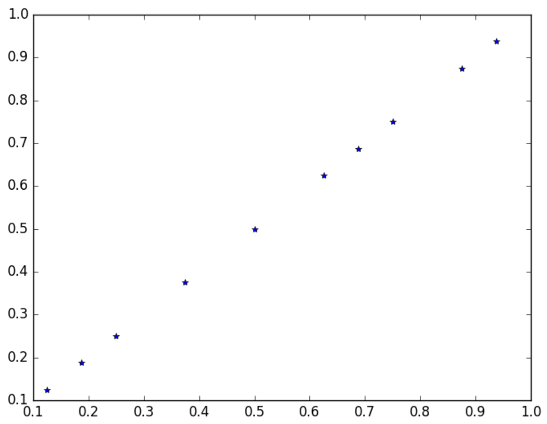


Figure 5.1: Plotting the first dimension on both axis does not yield a good uniformity, and also contradicts property A.

Definition 5.2. *A binary segment of length 2^s is a set of points P_i where the subscripts satisfy*

$$l2^s \leq i(l+1)l^s \qquad l = 0, 1, \cdots$$

Theorem 5.3. *A sequence $P_0, P_1 \cdots$ possesses the property A if and only if*

$$det \begin{vmatrix} v_{111} & v_{121} & \dots & v_{1s1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{s11} & v_{s21} & \dots & v_{ss1} \end{vmatrix} = 1, \tag{5.1}$$

where v_{ijk} is the kth bit after the decimal, in the jth direction number in the ith component of a point sequence in s dimensions. This is proved by Sobol [18].

5.1 Generating a Sobol sequence

We generate a *Sobol sequence* [15] in s dimensions. Fox [19] describes a method to generate such sequences. For each dimension we need a set of *direction numbers* $v_1, v_2, \cdots$. Each direction number v_i is a binary fraction:

$$v_i = 0.v_{i1}v_{i2}v_{i3} \cdots .$$

This binary fraction can also be represented by a single odd integer m_i:

$$v_i = \frac{m_i}{2^i}.$$

Each part of the binary fraction is made out of one bit, so the v_{ij}s are either zero or one. Note that $j > i$ implicitly defines $v_{ij} = 0$ as $0.10 = 0.1$. This is also the reason m_i has to be odd, since the rightmost bit of v_i is one.

To generate these direction numbers we choose a polynomial with coefficients either zero or one, which is a *primitive polynomial* in the field $\mathbb{Z}_2$. We choose

$$P(x) = x^d + a_1 x^{d-1} + \cdots + a_{d-1}x + 1,$$

so P is a polynomial of degree d in $\mathbb{Z}_2$.

Using the chosen arbitrary polynomial we define a recurrence for v_i:

$$v_i = a_1 v_{i-1} \oplus a_2 v_{i-2} \oplus \cdots \oplus a_{d-1} v_{i-d+1} \oplus v_{i-d} \oplus \lfloor \frac{v_{i-d}}{2^d} \rfloor, i > d,$$

or equivalently, for m_i:

$$m_i = 2a_1 m_{i-1} \oplus 2^2 a_2 m_{i-2} \oplus \cdots \oplus 2^{d-1} a_{d-1} m_{i-d+1} \oplus 2^d m_{i-d} \oplus 2^d m_{i-d}.$$

Here $\oplus$ denotes the *exclusive or*, bit by bit operation, meaning that you sum each bit and take the modulo 2 of it. One can also note that dividing $\lfloor \frac{v_{i-d}}{2^d} \rfloor$ is the same operation as moving (or bit-shifting) all the bits to the right d places. For computations the m_i recurrence can be calculated with integer arithmetic using a simple scaling. This means that all the calculations can be done in logical operations, which should be very effective. The values of the first d m_i can chosen freely, but must be odd and $m_i < 2^i$.

As an example we choose the polynomial

$$x^3 + x + 1,$$

which gives us

$$a_1 = 0$$
$$a_2 = 1,$$

and the recurrence is

$$m_i = 4m_{i-2} \oplus 8m_{i-3} \oplus m_{i-3}.$$

. Further we choose $m_1 = 1$, $m_2 = 3$, $m_3 = 7$ and calculate the next two m_is

Table 5.2: The first direction numbers for one Sobol sequence.

i	m_i	v_i
1	1	0.1
2	3	0.11
3	7	0.111
4	5	0.0101
5	7	0.00111

(denote $(\cdot)_2$ as the binary representation of a number):

$$\begin{aligned} m_4 &= 12 \oplus 8 \oplus 1 \\ &= (1100)_2 \oplus (1000)_2 \oplus (0001)_2 \\ &= (0101)_2 = 5 \\ m_5 &= 28 \cdot 24 \cdot 3 \\ &= (11100)_2 \cdot (11000)_2 \cdot (00011)_2 \\ &= (00111)_2 = 7, \end{aligned}$$

and the resulting direction numbers are tabulated in Table 5.2.

To generate the sequence $\boldsymbol{x}_M = \{x_i\}_{i=0}^{M}, x_i \in [0,1]$ we use

$$x_i = b_1 v_1 \oplus v_2 b_2 \oplus \cdots \oplus b_i v_i,$$

where $\cdots b_3 b_2 b_1$ is the binary representation of m. To produce a sequence $\boldsymbol{q}_M^s$ in s dimensions we choose s different primitive polynomials to calculate direction numbers:

$$\boldsymbol{q}_M^s = (\boldsymbol{x}_M^1, \boldsymbol{x}_M^2, \cdots, \boldsymbol{x}_M^s) \in [0,1]^{M \times s}$$

To guarantee some uniformity these direction numbers should be chosen so the resulting sequence satisfies property A (5.1).

Antonov and Salew [20] proves that a reordering of the bits in m by using the *Gray code*[21] representation of m gives the same asymptotic discrepancy.

Table 5.3: The Gray code of some numbers.

i	gray(i)
$0 = (000)_2$	$(000)_2 = 0$
$1 = (001)_2$	$(001)_2 = 1$
$2 = (010)_2$	$(011)_2 = 3$
$3 = (011)_2$	$(010)_2 = 2$
$4 = (100)_2$	$(110)_2 = 6$
$5 = (101)_2$	$(111)_2 = 7$
$6 = (110)_2$	$(101)_2 = 5$
$7 = (111)_2$	$(100)_2 = 4$

The reordering is defined as

$$x_m = g_1 v_1 \oplus g_2 v_2 \oplus \cdots ,$$

where $\cdots g_3 g_2 g_1$ is the Gray code binary representation of n. The Gray code of a number is defined by:

$$\begin{aligned} \text{gray}(i) &= i \oplus \lfloor \frac{i}{2} \rfloor \\ &= (\cdots i_3 i_2 i_1) \oplus (\cdots i_4 i_3 i_2) \end{aligned}$$

and some examples can be seen in table 5.3.

The Gray code representation has the interesting property that gray(i) and gray($i-1$) only differ in one bit (the index of the rightmost zero-bit in the binary representation of $i-1$). This means that we can setup a recursive definition of x_{n+1}:

$$x_{n+1} = x_n \oplus v_c,$$

where c is the rightmost zero-bit in the binary representation of n. This implies that a computer implementation can be made extremely efficient.

5.2 Implementation of the Sobol Sequence

The original implementation by Bratley and Fox [19] gave the direction numbers for generating a sequence with up to 40 dimensions. Since then there has been a lot of work in generating more direction numbers and in making them more robust. Joe and Kuo [22] extends the original Sobol generator with direction numbers up to 1111 dimensions. Later [23] they extended this to 21201 dimensions, and also optimized the generator for getting good two-dimensional projections. For most of the numerical computations we use an implementation by Gruenschloss[24]. In the above implementations all directions numbers satisfy Sobol's Property A.

5.3 Uniformity and normality

In this section we experiment with the Sobol generator to generate uniform and normal distributed numbers. For the uniformity we plot a histogram of $N = 10000$ realizations using the first Sobol dimension, where the first numbers are presented in Table 5.1. The results are presented in Figure 5.2a.

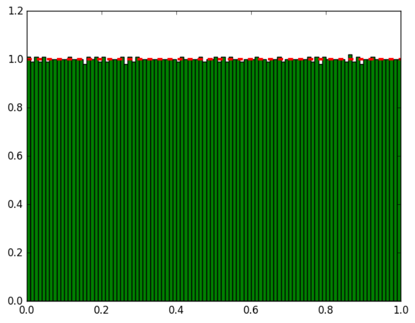
We further transform these numbers using the Box-Muller transform.

Theorem 5.4 (Box-Muller transform [25]). *For independent variables $U_1 \sim \mathcal{U}[0,1]$ and $U_2 \sim \mathcal{U}[0,1]$ the numbers X_1 and X_2*

$$X_1 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$
$$X_2 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

are standard normal distributed $\sim \mathcal{N}(0,1)$ and independent.

Since the numbers must be independent we use numbers from two different Sobol dimensions to generate standard normal numbers. The result of taking the Box-Muller transform is presented in Figure 5.2b. For reference a plot using the numpy random module is also presented in Figure 5.3. We conclude that the modified Box-Muller procedure works correctly.


(a) Histogram of the uniform sequence.

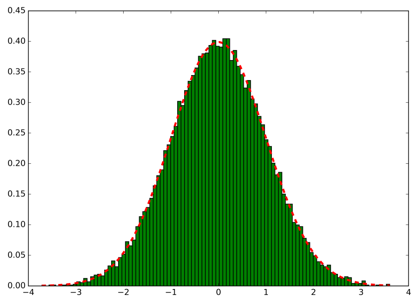
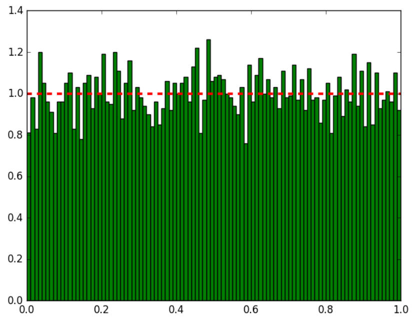

(b) Histogram of Box-Muller transformed uniform sequence.

Figure 5.2: Uniformity and normality of the Sobol sequence.


(a) Histogram of the uniform numbers.

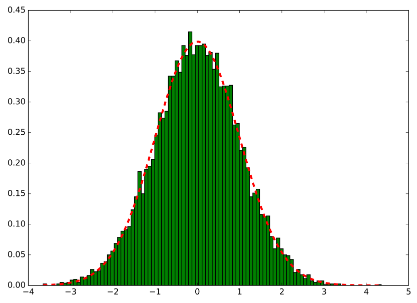

(b) Histogram of Box-Muller transformed uniform sequence.

Figure 5.3: Uniformity and normality of the numpy random sequence.

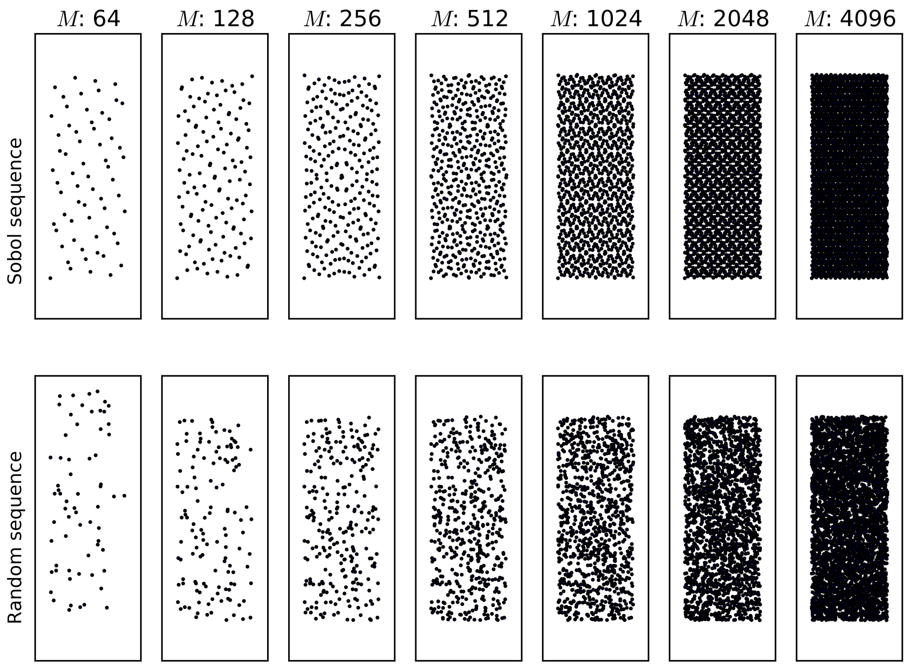


Figure 5.4: The spacing of a two-dimensional projection of the Sobol sequence (top) and Pseudo-random sequence (bottom) for different length of sequences, M.

5.4 2D projections

As already seen, a possible problem with the Sobol generator is to what degree the numbers are spaced out evenly. An example of how the Sobol sequence fills out the space (a 2D-projection) can be seen in Figure 5.4. A comparison between the Pseudo-random sequence and the Sobol sequence for all 2D projections for the first five dimensions can be viewed in Figure 5.5 and 5.6. The projections seems to indicate that the Sobol sequence fills out the space faster than the pseudo-random sequence, but not all of the two dimensional projections are equally good. They are better than the pseudo random ones though, so Sobol's

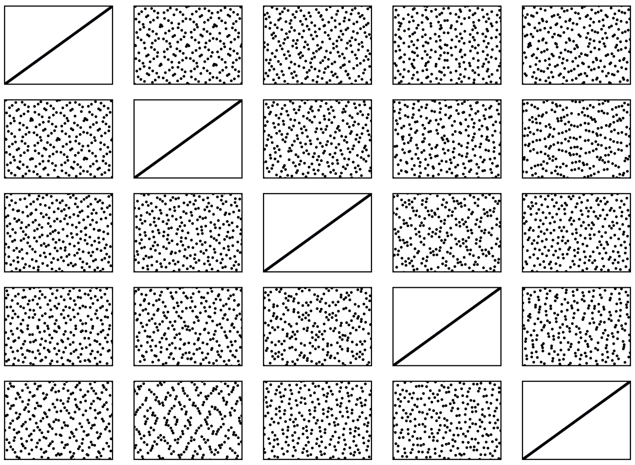


Figure 5.5: Two-dimensional projection for the first five dimensions of the Sobol Sequence.

second requirement is fulfilled.

5.5 Random number generator runtime

Another potential problem for the Sobol generator might be that they are slower to generate, and therefore reduces the computational efficiency of the method. As can be seen in Figure 5.7 we see that this is not the case. We are quite surprised to get about twice the speed out of the Sobol generator than the Mersenne Twister generator. The Gray code algorithm can, on the other hand, be implemented extremely efficient, si it makes sense after all.

This marks Sobol's final requirement as fulfilled, and we state that the Sobol sequence is a suitable quasi random number generator.

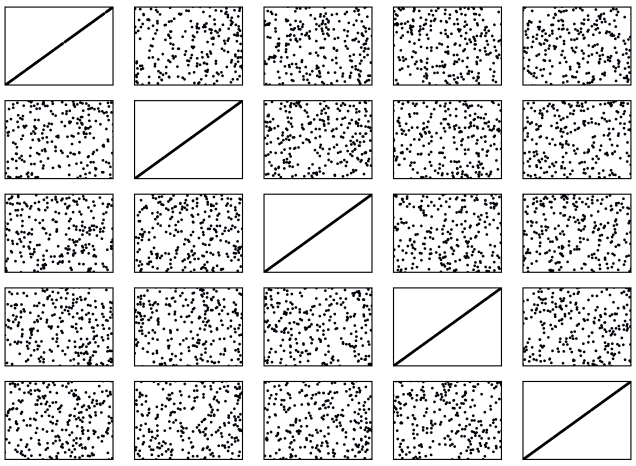


Figure 5.6: Two-dimensional projection for the first five dimensions of the pseudo random Sequence.

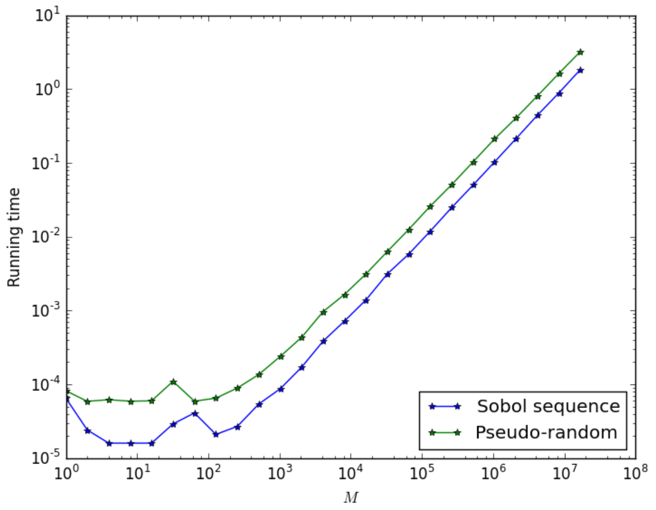


Figure 5.7: Runtime for generating a sequence of M random numbers using either the Sobol sequence or the Mersenne Twister pseudo-random generator.

Chapter 6

Conservation Laws with uncertain initial data

6.1 (Quasi) Monte Carlo Solver

In this section we describe a procedure for generating a numerical solution to the conservation law with uncertain initial data. We are interested in taking a look at what happens when the initial data is non-deterministic. To do this we sample the random parameter, generate initial data and send this initial data to a deterministic solver. We use this solver as a black box. We repeat this process M times. The mean value and standard deviation are calculated from the results. For the mean value we use the definition of the Monte Carlo integral (3.2) (or equivalently the QMC integral (4.1)), rewritten to fit this data as

$$E_M[u](x,t) = \frac{1}{M}\sum_{i=1}^{M} u_i(x,t,w),$$

where each u_i is a different realization of the problem with uncertain initial data. We also utilize the discrete version of the standard deviation:

$$\mathrm{SD}_M(u) = \left(\frac{1}{M} \sum_{i=1}^{M} \left(u_i(x,t,\omega) - E_M[u](x,t) \right)^2 \right)^{1/2}$$

6.2 Uncertain shock location

We follow Schwab and Tokareva [26] and consider the following stochastic Riemann problem on the scalar conservation law (2.3)

$$u_0(x,w) = \begin{cases} u^- & \text{if } x < x_0 + Y(\omega) \\ u^+ & \text{if } x > x_0 + Y(\omega), \end{cases} \tag{6.1}$$

with ω a random variable with distribution $Y(\omega)$. We introduce the stochastic variable $y = x_0 + Y(\omega) \in \mathbb{R}$. Using the method of characteristics with an entropy condition like the Lax Entropy Condition (2.5) this can be shown to be solved by

$$u(x,t,\omega) = u^- + (u^+ - u^-)H(x - st - y),$$

where $H(\cdot)$ is the Heaviside function and $s = s(t)$ is the shock speed determined by the Rankine-Hugeniot condition (2.6). Given that Y has the density $p(y)$, the expectation can be written as

$$\begin{aligned} E(x,t) &= \int_{-\infty}^{\infty} u(x,t,y)p(y)dy \\ &= u^- + (u^+ - u^-) \int_{-\infty}^{\infty} H(x - st - y)p(y)dy \\ &= u^- + (u^+ - u^-) \int_{-\infty}^{\infty} \left(1 - H(y - (x - st)\right) p(y)dy, \end{aligned}$$

where the last equality comes from a property of the Heaviside function. In the sense of distributions we have that $H'(y - y_0) = \delta(y - y_0)$ where $\delta(\cdot)$ is the delta function:

$$\int_{-\infty}^{\infty} \phi(y)\delta(y-y_0) = \phi(y_0)$$

for any test function $\phi(y) \in C_c^\infty(\mathbb{R})$. Putting all this together we have that the expected solution to (2.1) with initial condition (6.1) is

$$\begin{aligned} E[u](x,t) &= u^- + (u^+ - u^-)\left(1 - H(y-(x-st))\right) P(y)|_{-\infty}^{\infty} \\ &\quad + (u^+ - u^-) \int_{y=-\infty}^{\infty} \delta(y-(x-st))P(y)dy \\ &= u^- + (u^+ - u^-)P(x-st) \end{aligned} \tag{6.2}$$

with $P(\cdot)$ being the cumulative distribution function (3.1). Wee see that the solution varies only with the distribution of the shock location.

The variance is given as

$$\begin{aligned} \mathrm{Var}[u](x,t) &= \int_{-\infty}^{\infty} (u - E[u])^2 p(y)dy \\ &= \int_{-\infty}^{\infty} \left[(u^+ - u^-)H(x-st-y) - (u^+ - u^-)P(x-st)\right]^2 p(y)dy \\ &= (u^+ - u^-)^2 \int_{-\infty}^{\infty} H(x-st-y)^2 p(y)dy \\ &\quad + (u^+ - u^-)^2 \int_{-\infty}^{\infty} -2H(x-st-y)P(x-st)p(y)dy \\ &\quad + (u^+ - u^-)^2 \int_{-\infty}^{\infty} P(x-st)^2 p(y)dy \\ &= (u^+ - u^-)^2 \left(P(x-st) - 2P(x-st)^2 + P(x-st)^2\right) \\ &= (u^+ - u^-)^2 \left(P(x-st) - P(x-st)^2\right) \end{aligned}$$

6.3 Example on uncertain shock location

We let the uncertain shock position be distributed with a uniform variable $c(2\mathcal{U}(\omega)-1)$ where $\mathcal{U}(\omega) \sim \mathcal{U}(0,1)$ and $c \in \mathbb{R}$. This implies that the stochastic variable $y = x_0 + c(2\mathcal{U}(\omega)-1) \sim \mathcal{U}(x_0 - c, x_0 + c)$. The density and distribution function of y is now

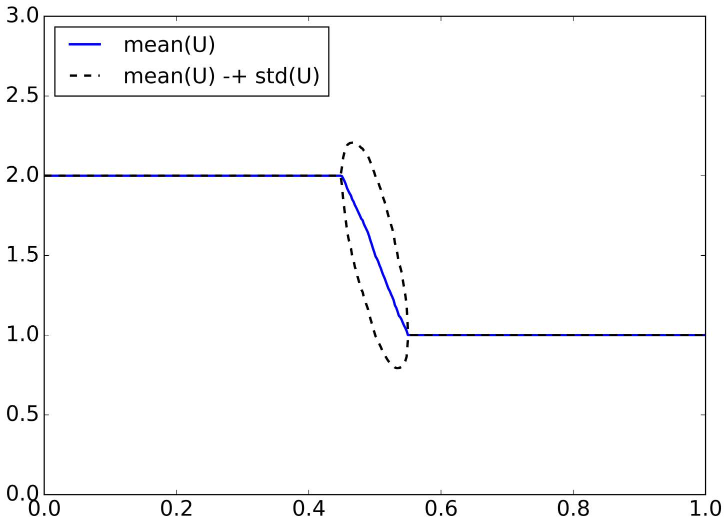
$$p(y) = \begin{cases} \frac{1}{2c} & \text{if } y \in [x_0 - c, x_0 + c] \\ 0 & \text{otherwise.} \end{cases}$$

$$P(y) = \begin{cases} 0 & \text{if } y < x_0 - c \\ \frac{1}{2c}(y - x_0 + c) & \text{if } y \in [x_0 - c, x_0 + c] \\ 1 & \text{if } y > x_0 + c. \end{cases}$$

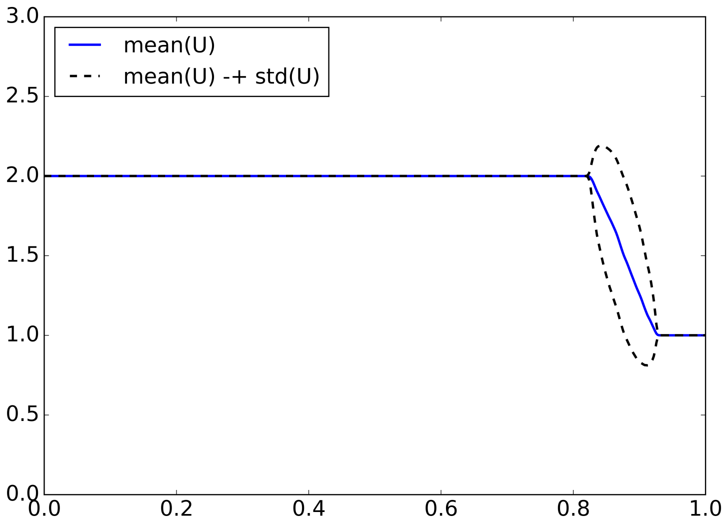
Using (6.2) to solve this initial value problem we get the solution

$$\begin{aligned} u(x,t) &= u^- + (u^+ - u^-)P(x - st) \\ &= u^- + (u^+ - u^-)\begin{cases} 0 & \text{if } x - st < x_0 - c \\ \frac{1}{2c}(x - st - x_0 + c) & \text{if } x - st \in [x_0 - c, x_0 + c] \\ 1 & \text{if } x - st > x_0 + c. \end{cases} \end{aligned}$$

This example is solved numerically for Burgers' equation in Figure 6.1 for $x_0 = 0.5$, $c = 0.1$ and $t = 0.5$.



(a) Initial condition for the Riemann problem.



(b) Solution of the Riemann problem at $t = 0.5$.

Figure 6.1: The initial condition and solution for the example Riemann problem.

6.4 Karhunen-Loève expansion

We would also like to experiment on continuous initial value problems, and a *Karhunen-Loève expansion* is a good fit. A derivation of the method is detailed by Alexanderian [27]. We consider a centered Gaussian random field X, characterized by the variance σ^2.

Theorem 6.1 (Karhunen-Loève expansion [27])**.** *Let $X : [t_1, t_2] \times \Omega \to \mathbb{R}$ be a centered mean-square continuous stochastic process with $X \in L^2(\Omega \times [t_1, t_2])$. There exist a basis $\{\epsilon_i\}$ of $L^2([t_1, t2])$ such that for all $t \in [t_1, t_2]$*

$$X(t, \omega) = \sum_{i=1}^{Q} \sqrt{\lambda_i} \chi_i(\omega) \epsilon(t)$$

where χ_i are centered mutually uncorrelated random variables with unit variance. If the stochastic process is Gaussian χ_i are standard normal variables.

The Gaussian process X has a Karhunen-Loève expansion[28] with

$$\epsilon_i = \begin{cases} \frac{\cos(\omega_i(x-0.5))}{\sqrt{0.5+\frac{\sin(\omega_i)}{2\omega_i}}} & \text{if } i \text{ is even,} \\ \frac{\sin(\omega_i(x-0.5))}{\sqrt{0.5-\frac{\sin(\omega_i)}{2\omega_i}}} & \text{if } i \text{ is odd.} \end{cases}$$

$$\lambda_i = \sigma^2 \frac{2b}{(1+\omega_i b)^2},$$

where w_i are the ordered positive roots of the characteristic equation

$$[1 - b\tan(\omega/2)]\,[b\omega + \tan(\omega/2)] = 0.$$

The first eigenvalues and bases $\sqrt{\lambda_i}\epsilon_i(x)$ are plotted in Figure 6.2, and ten realizations of the field X is plotted in figure 6.3 with the expansion truncated at $Q = 10$.

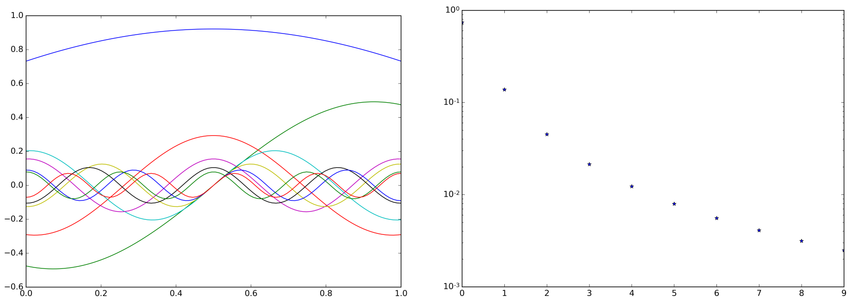


Figure 6.2: The ten first eigenvalues in a logarithmic plot and the ten first bases plotted.

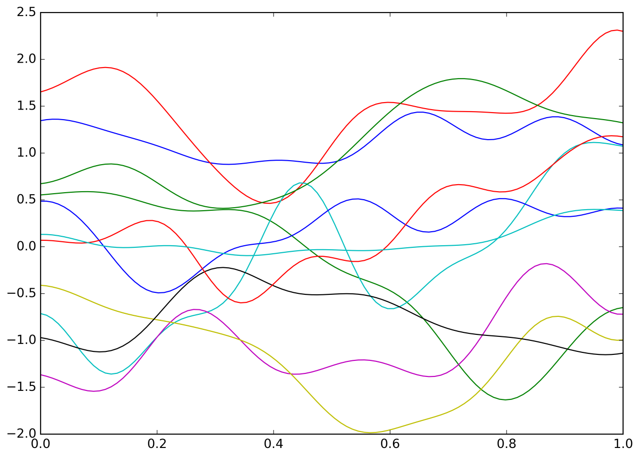


Figure 6.3: 10 realizations of the random field X.

Chapter 7

Implementation

7.1 Implementation from scratch

We have implemented as much as possible from scratch both to get a better understanding of the methods, but also to compare results with the already available solver package ALSVID-UQ [1]. We have written a C++ implementation of the Lax-Friedrichs, Rusanov and Godunov flux solvers for scalar one-dimensional conservation laws, and tested extensively using Burgers' equation. This implementation uses Armadillo [29] as an underlying matrix library, and everything has been wrapped in Python for easy scripting. MPI has been used to parallelize the processing, and the solver has been tested concurrently with up to 1024 cores at the Euler cluster [30].

We have implemented a Sobol generator in both Python and C++, and together with the solver part we are able to solve Burgers' equation with any initial data. When we tested the ALSVID-UQ package we did get a nice speedup above what we could achieve with our own implementation, perhaps because of load balancing of the compute nodes ([31], [32]), more aggressive optimization options when compiling, or some other reasons. A decision was made to extend ALSVID-UQ with the necessary components to be able to perform Quasi Monte Carlo with it.

7.2 Integrating with ALSVID-UQ

ALSVID-UQ was not intended to be utilized as a framework for quasi Monte Carlo, and thus needed some updates. Gruenschloss' [24] Sobol generator was integrated into the existing framework. Care had to be taken as you have to request which Sobol dimension you want data from. We implemented this feature in ALSVID-UQ, along with a special method to generate numbers from $\mathcal{N}(0,1)$ from Sobol numbers. New initial data has been added where needed.

Every new feature added to ALSVID-UQ has been checked to produce the same results as with the implementation written from scratch.

ALSVID-UQ also offers scripting possibilities in Python. A framework for creating and processing data has been written, extending the native capabilities of ALSVID-UQ.

Chapter 8

Numerical Experiments

This section presents results from numerical experiments solved by using the augmented ALSVID-UQ framework to solve initial value problems with mostly Burgers' equation:

$$\begin{aligned} \partial_t u(x,t) + \partial_x \left(\frac{u^2(x,t)}{2} \right) &= 0 \qquad\qquad t > 0 \\ u(x,0) &= u_0(x). \end{aligned}$$

If the initial data is uncertain we also have a dependency on a random variable ω in some probability space Ω:

$$\begin{aligned} \partial_t u(x,t,\omega) + \partial_x \left(\frac{u^2(x,t,\omega)}{2} \right) &= 0 \qquad\qquad t > 0 \qquad \omega \in \Omega \\ u(x,0,\omega) &= u_0(x,\omega). \end{aligned}$$

The following experiments will utilize the HLL-solver built into ALSVID-UQ, but as has been shown in Section 2.8, this is equivalent to using a Rusanov flux (2.12). For all problems we will plot the mean and variance, and also the convergence for both the mean and the variance. The convergence is measured in the L$_1$ norm:

$$||\boldsymbol{u}||_1 = \frac{1}{N}\sum_{i=1}^{N}|u_i|,$$

where N is the length of the vector $\boldsymbol{x}$. We will first present results for a simple discontinuous initial value problem, using both uniform and normal distributed random numbers. This is mostly to check what kind of convergence we can expect for more advanced simulations, but also to check that everything functions as expected. We will then gradually solve more complicated initial value problems.

8.1 Discontinuous initial value with uniform height

A test problem is given by

$$u_0(x,\omega) = \begin{cases} h+\omega & \text{if } x \le x_0 \\ -h-\omega & \text{if } x > x_0 \end{cases},$$

$$\omega \sim \mathcal{U}(0,c), \qquad h > 0, \qquad 0 < c < h, \tag{8.1}$$

where x_0 is the location of a shock. The solution is stationary and given by

$$\begin{aligned} u &= h+\omega+(-h-\omega-(h+\omega))H(x-x_0) \\ &= (h+\omega)(1-2H(x-x_0)). \end{aligned}$$

The mean value is

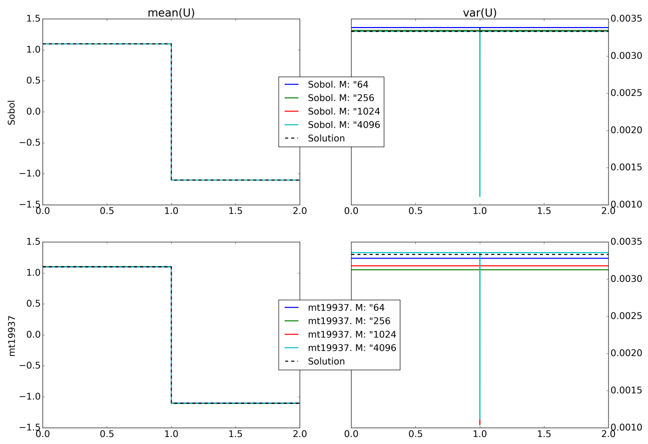


Figure 8.1: Mean and variance for the solutions of the discontinuous initial value problem with uniform height.

$$
\begin{aligned}
E[u](x,t) &= (1-2H(x-x_0))\int_{-\infty}^{\infty}(h+\omega)p(\omega)d\omega \\
&= \frac{1-2H(x-x_0)}{c}\int_0^c (h+\omega)d\omega \\
&= \frac{1-2H(x-x_0)}{c}(hc+\frac{c^2}{2}) \\
&= (1-2H(x-x_0))(h+\frac{c}{2}) \\
&= \begin{cases} h+0.5c & \text{if } x \le x_0 \\ -h-0.5c & \text{if } x > x_0 \end{cases},
\end{aligned}
$$

and the variance is $\frac{c^2}{12}$ for all x. Letting $N = 16384$, $t = 0.3$, $h = 1$, $c = 0.2$ and $x_0 = 1$ and letting M vary we present the results in Figure 8.1 and the convergence in Figure 8.2. Since the underlying finite volume method converges with $O(1/N)$ in space for all nonlinear flux, we use a large N to isolate the stochastic error.

A visual inspection of the mean results in Figure 8.1 both Monte Carlo methods seem to solve this problem efficiently. If we however study the convergence

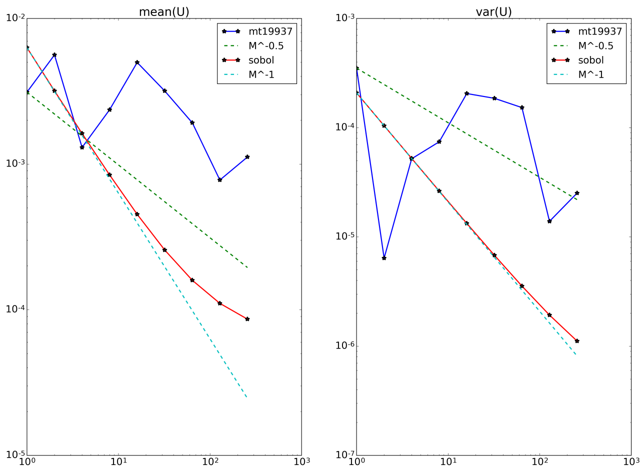


Figure 8.2: Convergence of mean and variance in the discontinuous initial value problem with uniform height.

in Figure 8.2 we notice that the quasi Monte Carlo method seems to converge a lot faster, and with more certainty, than the regular Monte Carlo method using the Mersenne Twister. For this problem the Quasi Monte Carlo method converges with rate $O(1/M)$ in stochastic space. We suspect the slight bump in the end comes because the spatial error in the finite volume method begins to dominate from there and out. The error in the conventional Monte Carlo method seems to fluctuate. We are not yet sure why this occurs, it may be a problem by our implementation, or some other factor may play a role. We know that for all these problems the error for the conventional Monte Carlo method should scale as $O(1/\sqrt{M})$, so from here on we focus on the quasi Monte Carlo error.

8.2 Discontinuous initial data with normal height

As far as we know, there was no clear evidence that the Sobol sequence should not function correctly in a quasi Monte Carlo method using normally distributed

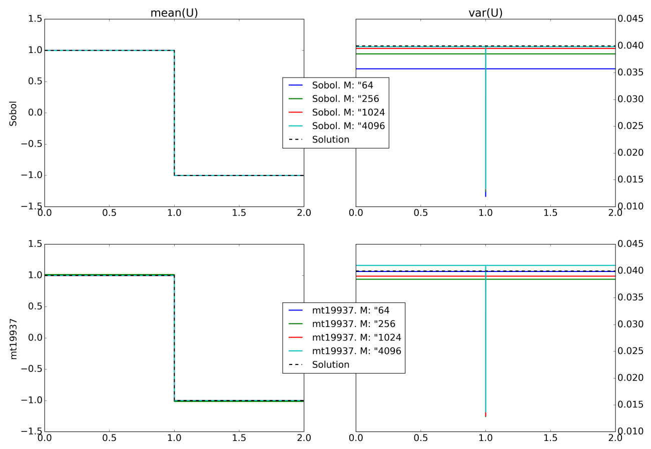


Figure 8.3: Mean and variance for the solutions of the discontinuous initial value problem with normal height.

numbers, but here we check that. For the next problem we use the exact same problem as (8.1), only substituting the uniform variable with a normal distributed variable. The normal distributed variable is generated by the modified Box-Muller algorithm as described in Section 5.3.

$$u_0(x) = \begin{cases} h + \omega & \text{if } x \leq x_0 \\ -h - \omega & \text{if } x > x_0 \end{cases}, \qquad \omega \sim \mathcal{N}(0, c),$$

Following precisely the same procedure as above we get that

$$\begin{aligned} E[u](x,t) &= (1 - 2H(x - x_0))h \\ \text{Var}[U](x,t) &= c^2 h^2. \end{aligned}$$

We use the same constants as before and plot the results in Figures 8.3 and 8.4.

We are not quite as happy with the performance on this experiment. The convergence of the mean in Figure 8.4 is not as good as before, but the convergence of the variance is $O(1/M)$. Since normal distributed variables are not

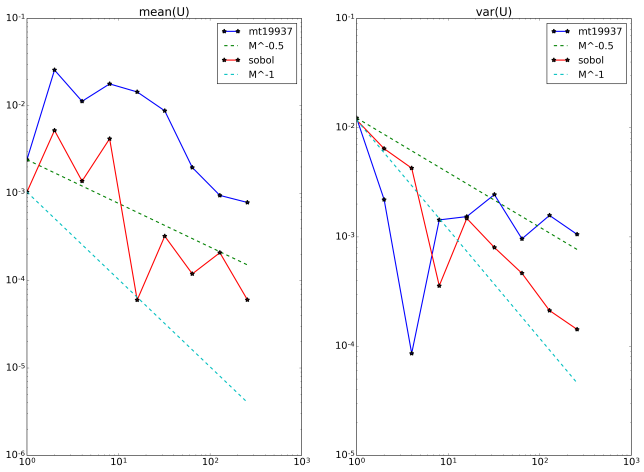


Figure 8.4: Convergence of mean and variance in the discontinuous initial value problem with normal height.

bounded and we in principle could have that $h + c < 0$ this could be a source of error. In that case we would get a rarefaction wave instead of a shock wave, and this could skew the results. The probability of this happening in any single run is only $\mathbb{P}(1 + 0.2c < 0) \approx 2.8 \cdot 10^{-7}$, so we disregard that here.

8.3 5-dimensional problem with uncertain shock placements

Further it would be interesting to see how the quasi Monte Carlo method performs on problems with more dimensions. We craft an initial value problem with different shock placements:

$$u_0(x) = \begin{cases} 1 & \text{if} \quad 0 \le x < 0.1 + Y_1(\omega) \\ 0.8 & \text{if} \quad 0.1 + Y_1(\omega) \le x < 0.3 + Y_2(\omega) \\ 0.6 & \text{if} \quad 0.3 + Y_2(\omega) \le x < 0.5 + Y_3(\omega) \\ 0.4 & \text{if} \quad 0.5 + Y_3(\omega) \le x < 0.7 + Y_4(\omega) \\ 0.2 & \text{if} \quad 0.7 + Y_4(\omega) \le x < 0.9 + Y_5(\omega) \\ 0.0 & \text{if} \quad x \ge 0.9 + Y_5(\omega) \end{cases},$$

$$Y_i(\omega) \sim \mathcal{U}(-c, c), \qquad i \in [1, 2, 3, 4, 5], \tag{8.2}$$

where c yet again is some small positive constant. The following is then the mean solution to each of the stochastic Riemann problems:

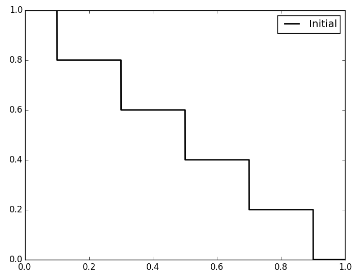
$$E[u](x,t) = u^- + (u^+ - u^-) \begin{cases} 0 & \text{if } x - st < x_0 - c \\ Z & \text{if } x - st \in [x_0 - c, x_0 + c] \\ 1 & \text{if } x - st > x_0 + c, \end{cases}$$

where x_0 is the mean value of the current shock location and the value of the constant is $Z = \frac{1}{2c}(x - st - x_0 + c)$. This can further be patched together to a solution of problem (8.2). The variance can be expressed as

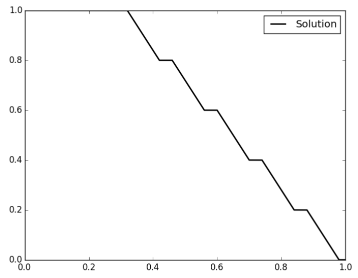
$$\text{Var}[u](x,t) = (u^+ - u^-)^2 \begin{cases} 0 & \text{if } x - st < x_0 - c \\ Z - (Z)^2 & \text{if } x - st \in [x_0 - c, x_0 + c] \\ 0 & \text{if } x - st > x_0 + c. \end{cases}$$

The initial conditions and analytical mean solution can be seen in Figure 8.5. The results for $c = 0.1$, $t = 0.3$ and $N = 16384$ are presented in Figures 8.6 and 8.7.

These results are not conclusive for the convergence of the variance, that prompted us to run again with higher spatial accuracy. In Figure 8.8 the convergence results are presented for $N = 65536$. Increasing N yields a much better convergence. This implies that the finite volume error dominated with lower N, and we conclude that the quasi Monte Carlo method converges with $O(1/M)$



(a) Initial condition for five dimensional problem.



(b) Analytical mean solution for $t = 0.3$ for five dimensional problem.

Figure 8.5: The initial condition and solution for the 5D shock problem.

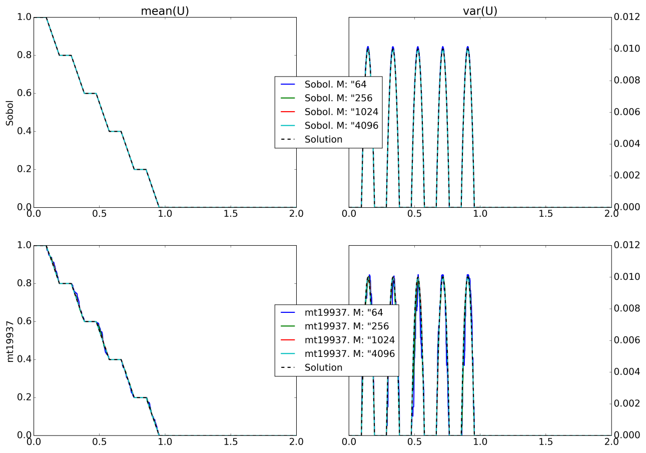


Figure 8.6: Numerical and analytical solutions to the 5D shock problem.

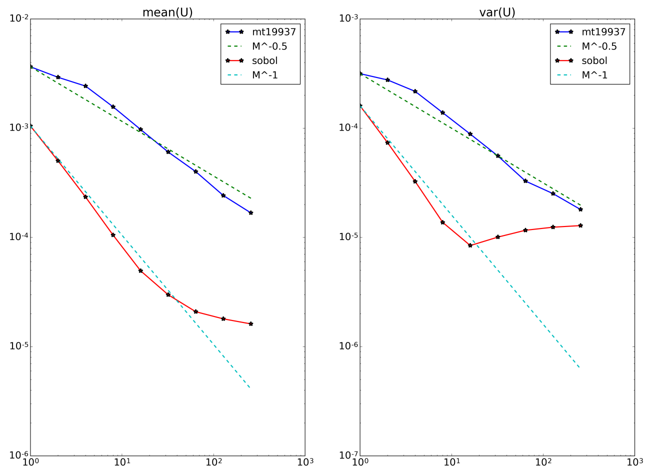


Figure 8.7: Convergence for the 5D shock problem with $N = 16384$.

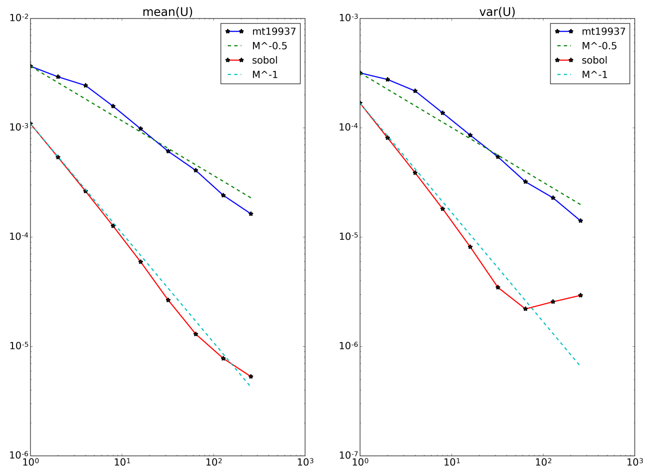


Figure 8.8: Convergence for the 5D shock problem with $N = 65536$.

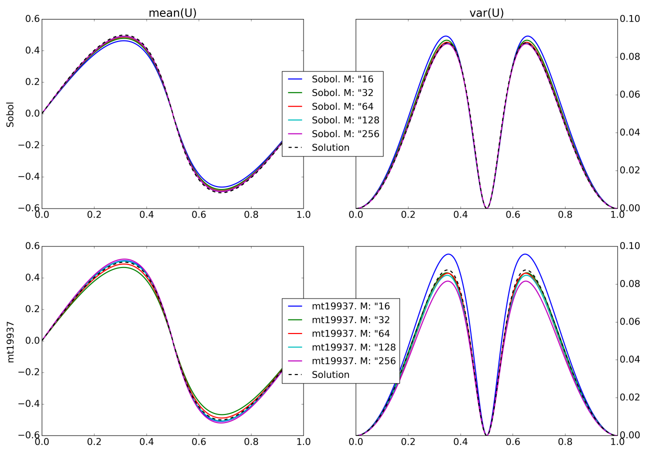


Figure 8.9: Numerical and reference solutions to the sine problem.

also for this problem, even though there is a slip dip in the convergence graph for the variance. We suspect that also this dip would be corrected had we run with even higher spatial accuracy. For the regular Monte Carlo method the convergence rate is as predicted, $O(1/\sqrt{M})$.

8.4 Sine with uncertain amplitude

Experimenting with continuous initial value problems, We want to solve the problem

$$\begin{aligned} u_0(x,\omega) &= \omega\sin(2\pi x) \\ \omega &\sim \mathcal{U}(0,1), \end{aligned}$$

and compare it with a reference solution computed with $t = 0.2$, $N = 4096$ and $M = 2048$. We use $t = 0.2$, $N = 512$ and vary M. The mean and variance results are plotted in Figure 8.9, and for reference the convergence of the quasi Monte Carlo is plotted in Figure 8.10. The results here indicate the same convergence $O(1/M)$ for the quasi Monte Carlo method.

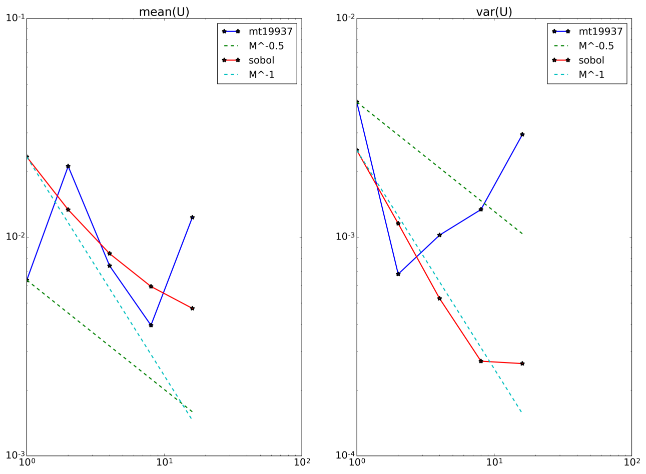


Figure 8.10: Convergence for the sine problem with $N = 512$.

8.5 Karhunen-Loève expansion

Following the procedure described in Section 7, only now we truncate the expansion at $Q = 9$, let $N = 16384$, $t = 0.05$ and vary M. The mean and variance results are plotted in Figure 8.11 and the convergence result is plotted in Figure 8.12. We do not know the analytical solution to this problem, so we compare with a reference solution computed with $N = 16384$ and $M = 16384$. The convergence results here indicate $O(1/M)$ convergence for the quasi Monte Carlo method and $O(1/\sqrt{M})$ for the regular Monte Carlo method. The dip in the convergence plots when M gets large makes us suspect also this solution could be better with better spatial resolution in the finite volume method.

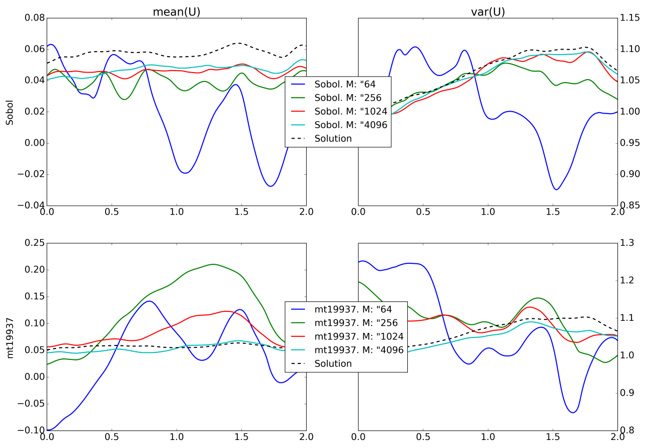


Figure 8.11: Results of the Karhunen-Loève experiment.

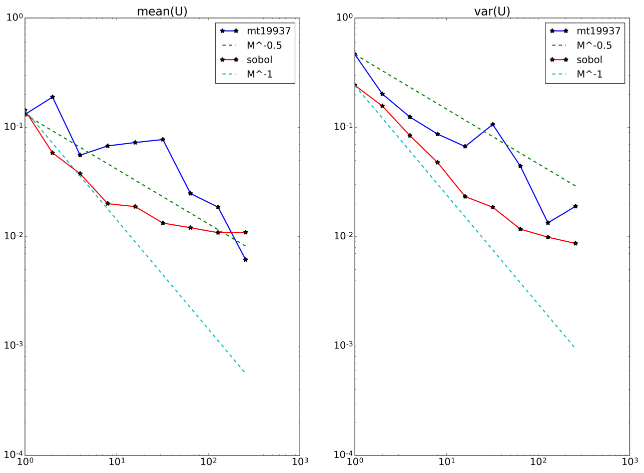


Figure 8.12: Convergence for the Karhunen-Loève experiment.

Chapter 9

Conclusion

We consider solving hyperbolic systems of conservation laws with uncertain initial values, solved using a quasi Monte Carlo method. The quasi Monte Carlo method utilize Sobol sequences, while the underlying solver is a finite volume solver. We implement the Sobol generator and the finite volume solver from scratch in C++. Combining these we have a complete system for solving these kind of problem for scalar conservation laws. For speed and efficiency we also integrate these into ALSVID-UQ, a tool for uncertainty quantification.

We present numerical experiments in one space dimension and up to nine stochastic dimensions. For all experiments needing uniformly sampled initial data the error is demonstrated to scale as $O(1/M)$, both in the mean and variance. Using a modified Box-Muller algorithm we can also generate normally distributed initial data. Here the results are not that robust, but is no worse than conventional Monte Carlo. For the most advanced experiment, using a Karhunen-Loève expansion truncated after 9 terms, we get the error scaling as $O(1/M)$, even though these initial values depend on normally distributed numbers. For some of the numerical examples the conventional Monte Carlo error behave erratically. Due to the complexity of running these high-resolution problems on a cluster we have not yet been able to pinpoint the exact cause of this.

Aside from some small technical difficulties, such as sampling from the correct dimension and using the modified Box-Muller method, the quasi Monte Carlo method using Sobol sequences can substitute conventional Monte Carlo methods with little to no work. The quasi Monte Carlo method offers a faster convergence, and often a lower constant error in all our experiments. Especially in cases where only uniformly distributed numbers are needed this method shines.

Bibliography

[1] ALSVID-UQ. Software project available from `http://www.sam.math.ethz.ch/alsvid-uq`.

[2] A. Hollingsworth. An experiment in Monte Carlo forecasting. In *Workshop on Stochastic Dynamic Forecasting, 17-19 October 1979*, pages 65–85, Shinfield Park, Reading, 1979. ECMWF, ECMWF.

[3] R. Buizza, M. Milleer, and T. N. Palmer. Stochastic representation of model uncertainties in the ECMWF ensemble prediction system. *Quarterly Journal of the Royal Meteorological Society.*

[4] J. Šukys et al. *Robust multi-level Monte Carlo Finite Volume methods for systems of hyperbolic conservation laws with random input data.* PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 21990, 2014, 2014.

[5] S. Mishra and C. Schwab. Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data. *Mathematics of Computation*, 81(280):1979–2018, 2012.

[6] H. Holden and N. H. Risebro. *Front tracking for hyperbolic conservation laws*, volume 152 of *Applied Mathematical Sciences.* Springer, 2011.

[7] S. Mishra. Numerical Methods for conservation laws and related equations. Lecture notes for Numerical Methods for Partial Differential Equations, ETH.

[8] J. C. Halvorsen. Monte Carlo Finite Volume Methods For Hyperbolic Conservation Laws With Uncertain Initial Data. Available at `https://github.com/jchalvorsen/Monte_Carlo_Hyperbolic`.

[9] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959.

[10] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, chapter The HLL and HLLC Riemann Solvers, pages 315–344. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[11] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta numerica*, 7:1–49, 1998.

[12] I. M. Sobol. On quasi-Monte Carlo integrations. *Mathematics and Computers in Simulation*, 47(2):103–112, 1998.

[13] I. M. Sobol. Quasi-Monte Carlo methods. *Progress in Nuclear Energy*, 24(1):55–61, 1990.

[14] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

[15] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[16] H. Faure. Discrépances de suites associées à un système de numération (en dimension un). *Bulletin de la Société Mathématique de France*, 109:143–182, 1981.

[17] B. L. Fox. Algorithm 647: Implementation and relative efficiency of quasirandom sequence generators. *ACM Transactions on Mathematical Software (TOMS)*, 12(4):362–376, 1986.

[18] I. M. Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[19] P. Bratley and B. L. Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

[20] I. A. Antonov and V. M. Saleev. An economic method of computing lp τ-sequences. *USSR Computational Mathematics and Mathematical Physics*, 19(1):252–256, 1979.

[21] F Gray. Pulse code communication. 1953.

[22] S. Joe and F. Y. Kuo. Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):49–57, 2003.

[23] S. Joe and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[24] L. Gruenschloss. A C++ implementation of a simple and fast random-access Sobol'-sequence generator, based on the direction numbers published in S. Joe and F. Y. Kuo: Constructing Sobol sequences with better two-dimensional projections. `http://gruenschloss.org`.

[25] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29(2):610–611, 06 1958.

[26] C. Schwab and S. Tokareva. High order approximation of probabilistic shock profiles in hyperbolic conservation laws with uncertain initial data. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(03):807–835, 2013.

[27] A. Alexanderian. A brief note on the Karhunen-Loève expansion.

[28] O. P. Le Matre and O. M. Knio. Spectral methods for uncertainty quantification. *Scientific Computation. Springer, New York*, 2010.

[29] C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. In *NICTA*, Australia, Oct 2010.

[30] Central hpc clusters euler and brutus. Information can be found at `https://www1.ethz.ch/id/services/list/comp_zentral/cluster/index_EN`.

[31] J. Šukys, S. Mishra, and C. Schwab. *Parallel Processing and Applied Mathematics: 9th International Conference, PPAM 2011, Torun, Poland, September 11-14, 2011. Revised Selected Papers, Part I*, chapter Static Load Balancing for Multi-level Monte Carlo Finite Volume Solvers, pages 245–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[32] J. Šukys. *Parallel Processing and Applied Mathematics: 10th International Conference, PPAM 2013, Warsaw, Poland, September 8-11, 2013, Revised Selected Papers, Part I*, chapter Adaptive Load Balancing for Massively Parallel Multi-Level Monte Carlo Solvers, pages 47–56. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.