Universiteit Antwerpen

Masterproef scriptie

Recursive Monte Carlo for linear ODEs

Auteur: *Isidoor Pinillo Esquivel*

Promotor: *Wim Vanroose*

Academiejaar 2023-2024

Contents

Abstract

Unbiased algorithms for solving linear initial value problems have received limited attention. This is addressed here by proposing unbiased Recursive Monte Carlo methods for solving linear initial value problems and linear Fredholm integral equations of the second kind. Motivated by understanding randomized parallel complexity and downstream applications to partial differential equations, these methods are developed.

Previously only biased methods were known to achieve optimal convergence rates for nonlinear initial value problems which has primarily theoretical significance. The proposed RRMC method, an unbiased algorithm for linear initial value problems, together with control variates is conjectured to be order optimal with high probability for corresponding smoothness classes.

Parallel to this, the main Poisson algorithm is proposed, generalizing the algorithm in [AR16], which is simple and forward implementable. The main Poisson algorithm is applied to the semi-discretized heat equation demonstrating a path resampling technique, that provides a novel perspective on the Walk on Spheres method, with potential for alternative generalizations.

The proposed methods broadens the understanding of what is possible with unbiased algorithms for solving linear initial value problems, paving the way for further research in this area.

1 Introduction

1.1 Related Work

The primary motivating paper for this work is the work by Sawhney et al. (2022) [Saw+22], which introduces the Walk on Spheres (WoS) method for solving second-order elliptic PDEs with varying coefficients and Dirichlet boundary conditions. Their techniques have shown high accuracy even in the presence of geometrically complex boundary conditions. We were inspired to apply the underlying mechanics of these Monte Carlo (MC) techniques to Inital Value Problems (IVPs) to explore parallel in time and the possibility of extending their techniques to other types of PDEs.

Existing randomized algorithms for solving IVPs generally rely on MC integration or randomizing parts of deterministic algorithms. These are typically motivated by lower smoothness conditions where randomization significantly enhances convergence speed, or solving ODEs where randomization aids in parallelization. The most relevant literature in this context includes papers by Daun and Ermakov and Smilovitskiy.

Daun 2011 [Dau11] proposes a biased algorithm for nonlinear IVPs that has optimal Information-Based Complexity (IBC) for some smoothness classes. Similar to 3.2.3, their approach employs control variates.

Ermakov and Smilovitskiy 2021 [ES21] proposes an unbiased method for functionals of a Cauchy problem for large systems of linear ODEs. Their method, similar to all our IVPs solvers, is based on Volterra integral equations.

2 Background

2.1 Monte Carlo Integration

In this subsection, we review basic MC theory.

Notation 2.1.1 (Random Variables)
Random variables (RVs) will be denoted with capital letters, e.g., X, Y or Z.

MC integration is any method that involves random sampling to estimate an integral.

Definition 2.1.2 (Uniform Monte Carlo Integration)
We define uniform MC integration of $f : \mathbb{R} \to \mathbb{R}$ over $[0, 1]$ as an estimation of the expected value of $f(U)$, with $U \sim \text{Uniform}(0, 1)$. A simple MC Integration scheme, can be summarized in the following formula:

$$\int_0^1 f(s)ds \approx \frac{1}{n} \sum_{j=1}^{n} f(U_j), \tag{1}$$

```
uniform_MC_int(f, n) = sum(f(rand()) for _ in 1:n) / n
```

where n is the amount of samples used and U_j i.i.d. Uniform$(0, 1)$.

Because estimators are random variables (RVs), the cost and error are also random variables. In most cases, obtaining these RVs is difficult to impossible. Directly comparing estimators based on these RVs can be challenging; there is no Pareto front. Instead, comparisons can be made using statistics.

Accuracy comparisons between estimators are typically conducted with (root-)mean-square error (RMSE).

Definition 2.1.3 (Root-Mean-Square Error)
We define the Root-Mean-Square Error (RMSE) of an estimator $\tilde{\theta}$ for θ as follows:

$$\text{RMSE}(\tilde{\theta}) = \sqrt{E[||\tilde{\theta} - \theta||_2^2]}. \tag{2}$$

Comparisons based on RMSE can be counterintuitive; consider Stein's paradox, for example. We limit ourselves to simple cases such as 1-dimensional unbiased estimators, making MSE equivalent to variance. Estimating variance is in most cases possible and can be used to calculate confidence intervals using Chebyshev's inequality.

Average floating point operations or time per simulation are common cost statistics. It may also be useful to consider 'at risk' (analogous to 'value at risk') in terms of memory or wall time.
If we limit ourselves to (1) with a big sample size and finite variance assumption on error and simulation time, simulations can be computed in parallel, making them well-suited for a GPU implementation. These assumptions are useful for establishing a baseline and when they are close to being optimal, they become highly practical. Given these assumption, there is a linear trade-off between average simulation time

and variance. This relationship can be quantified using the following definition of MC efficiency.

Definition 2.1.4 (Monte Carlo Efficiency)
Define MC efficiency of an estimator F as follows:

$$\epsilon[F] = \frac{1}{\mathrm{Var}(F)T(F)}, \tag{3}$$

with T the average simulation time.

Related Work 2.1.5 (Monte Carlo Efficiency)
For further reference see [Vea97] page 45.

For smooth 1-dimensional integration, the linear trade-off between variance and average simulation time is not close to optimal see Theorem 2.4.5.
When it comes to comparing better trade-offs, Information-Based Complexity (IBC) is often employed. IBC primarily serves as a qualitative measure and does not necessarily imply the practicality of an algorithm. We will not delve into a rigorous definition of IBC.

Definition 2.1.6 (Information-Based Complexity)
IBC is a way to describe asymptotically (for increasing accuracy/function calls) the trade-off between the average amount of function calls (information) needed and accuracy.

Note that we consider the average amount of function calls as a measure of cost, rather than the deterministic amount of function calls used. We implicitly assume that the variance of the amount of function calls exists, which provides confidence intervals for the function calls through Chebyshev's inequality.

Example 2.1.7 (IBC of (1))
In (1) the function calls trades of linearly with variance. For n function calls, the RMSE $= O\left(\frac{1}{\sqrt{n}}\right)$ or equivalently, if we want a RMSE of ε we would need $O\left(\frac{1}{\varepsilon^2}\right)$ function calls.

2.2 Recursive Monte Carlo

In this subsection, we introduce Recursive Monte Carlo (RMC) with an example of an Initial Value Problem (IVP) and Walk on Spheres.

Notation 2.2.1 (U, U_j)
We will frequently use the uniform distribution, so we will abbreviate it

$$U_j \text{ i.i.d Uniform}(0,1). \tag{4}$$

The subscripts are used to clarify independence between uniforms. However, when there is no risk of confusion, we simply use U.

Example 2.2.2 $(y' = y)$
Consider following IVP:

$$y_t = y, \quad y(0) = 1. \tag{5}$$

By integrating both sides of (5), we obtain:

$$y(t) = 1 + \int_0^t y(s)ds. \tag{6}$$

(6) represents a recursive integral equation, specifically, a linear Volterra integral equation of the second type.
By estimating the recursive integral in (6) using MC, we derive the following estimator:

$$Y(t) = 1 + ty(Ut). \tag{7}$$

If y is well-behaved, then $E[Y(t)] = y(t)$. However, we cannot directly simulate $Y(t)$ without access to $y(s)$ for $s < t$. Nevertheless, we can replace y with an unbiased estimator without affecting $E[Y(t)] = y(t)$, by the law of total expectation ($E[X] = E[E[X|Z]]$). By replacing y with Y itself, we obtain a recursive expression for Y:

$$y(t) \approx Y(t) = 1 + tY(Ut). \tag{8}$$

```
Y(t, eps) = t > eps ? 1 + t * Y(rand() * t, eps) : 1
```

(8) is a Recursive Random Variable Equation (RRVE).
Simulation of Y with (8), would recurse indefinitely (every Y needs to sample another Y). To stop the recursion, approximate $Y(t) \approx 1$ near $t = 0$ introducing minimal bias. Later, we will discuss Russian roulette; see Definition 2.3.2, which can be used as an unbiased stopping mechanism.

Definition 2.2.3 (Recursive Random Variable Equation (RRVE))
A Recursive Random Variable Equation (RRVE) is an equation that defines a family of random variables in terms of itself.

Example 2.2.4 (Walk on Spheres)
Classical Walk on Spheres estimates the solution of the Laplace equation:

$$\nabla^2 u = 0, \quad u|_{\partial\Omega} = g, \tag{9}$$

in a point by applying RMC to the mean value property of the Laplace equation:

$$u(x) = \frac{1}{\mathrm{Vol}(\partial B(x))} \int_{\partial B(x)} u(y)dy, \tag{10}$$

with $B(x) \subseteq \Omega$ a ball around x chosen sufficiently big. Obtaining following estimator:

$$u(x) \approx Y(x) = Y(U(\partial B(x))). \tag{11}$$

Recursion is terminated by approximating $Y(x)$ with the boundary condition g near the boundary.

Related Work 2.2.5 (Walk on Spheres)
For further reference see [SC].

2.3 Modifying Monte Carlo

In this subsection, we discuss techniques for modifying RRVEs in a way that preserves the expected value of the estimator while acquiring more desirable properties. These techniques are only effective when applied by using prior information about the problem or computational costs.

We will frequently interchange RVs with the same expected values. Therefore introducing following notation:

Notation 2.3.1 ($\cong$)

$$X \cong Y \iff E[X] = E[Y].$$

Russian roulette is a MC technique commonly employed in rendering algorithms. The concept behind Russian roulette is to replace an RV with a less computationally expensive approximation sometimes.

Definition 2.3.2 (Russian roulette)

We define Russian roulette on X with free parameters $Y_1 \cong Y_2$, $p \in [0,1]$ and U independent of Y_1, Y_2, X as follows:

$$X \cong \begin{cases} \frac{1}{p}(X - (1-p)Y_1) & \text{if } U < p \\ Y_2 & \text{else} \end{cases}. \tag{12}$$

Notation 2.3.3 ($B(p)$)

Often Russian roulette will be used with $Y_1 = Y_2 = 0$. In that case, we use Bernoulli variables to shorten notation.

$$B(p) \sim \text{Bernoulli}(p) = \begin{cases} 1 & \text{if } U < p \\ 0 & \text{else} \end{cases}. \tag{13}$$

Example 2.3.4 (Russian roulette)

Consider the estimation of $E[Z]$, with Z:

$$Z = U + \frac{f(U)}{1000}. \tag{14}$$

Here, let $f : \mathbb{R} \to [0,1]$ be 100 times more expensive to compute then sampling U. $f(U)$ contributes relatively little to the variance compared to U. To address this, we Russian roulette $f(U)$:

$$Z \cong U + B\left(\frac{1}{100}\right)\frac{f(U)}{10}. \tag{15}$$

This modification requires calling f on average once every 100 samples. This significantly reduces the computational burden while increasing the variance slightly thereby increasing the overall MC efficiency.

Example 2.3.5 (Russian roulette on (8))

To address the issue of indefinite recursion in (8), Russian roulette can be employed by approximating the value of Y near $t = 0$ with 1 sometimes. Specifically, we replace the coefficient t in front of the recursive term with $B(t)$ when $t < 1$. The modified recursive expression for $Y(t)$ for $t < 1$ becomes:

$$y(t) \cong Y(t) = 1 + B(t)Y(Ut) \tag{16}$$

```
Y(t) = rand() < t ? 1 + Y(rand() * t) : 1 # correct for t<1
```

Interestingly, $\forall t \leq 1 : Y(t)$ is the number of recursion calls to sample $Y(t)$ such that the average number of recursion calls to sample $Y(t)$ equals e^t.

Splitting is a technique that increases samples of important terms, similar to how Russian roulette decreases the samples of unimportant terms.

Definition 2.3.6 (Splitting)

Splitting X refers to utilizing multiple $X_j \cong X$ (not necessarily independent) to reduce variance by taking their average:

$$X \cong \frac{1}{n}\sum_{j=1}^{n} X_j. \tag{17}$$

Example 2.3.7 (Splitting on (14))

Consider Example 2.3.4 again. Instead of using Russian roulette on $f(U)$, we split U:

$$Z \cong \frac{\sum_{j=1}^{10} U_j}{10} + f(U_1). \tag{18}$$

This modification requires sampling U, 10 times as often which is insignificant compared to evaluating $f(U)$ once but reduces the variance significantly thereby increasing the overall MC efficiency.

Related Work 2.3.8 (Examples 2.3.4, 2.3.7)

In Example 2.3.4 and Example 2.3.7, it is also possible to estimate the expectations of the 2 terms of Z separately. Given the variances and computational costs of both terms, you can calculate the asymptotically optimal division of samples for each term. However, this is no longer the case with RMC. In [Rat+22], a method is presented to estimate the optimal Russian roulette/splitting factors for rendering.

Splitting the recursive term in an RRVE can result in additive branching recursion, necessitating cautious management of terminating the branches promptly to prevent exponential growth in computational complexity. To accomplish this, termination strategies that have been previously discussed can be employed. Subsequently, we will explore the utilization of coupled recursion as a technique to mitigate additive branching recursion in RRVEs (see Example 3.1.7).

Example 2.3.9 (Splitting on (16))

We can split the recursive term of (16) into two parts as follows:

$$y(t) \cong Y(t) = 1 + \frac{B(t)}{2}(Y_1(U_1 t) + Y_2(U_1 t)) \tag{19}$$

```
function Y(t) # correct for t<1
    u = rand()
    rand() < t ? 1 + (Y(u * t) + Y(u * t)) / 2 : 1
end
```

$$y(t) \cong Y(t) = 1 + \frac{B(t)}{2}(Y_1(U_1 t) + Y_2(U_2 t)) \tag{20}$$

```
Y(t) = rand() < t ? 1 + (Y(rand() * t) + Y(rand() * t)) / 2 : 1
```

where $Y_1(t)$ and $Y_2(t)$ are i.i.d. with $Y(t)$.

Definition 2.3.10 (Control variates)
Define control variating $f(U)$ with $\tilde{f}$ an approximation of f as:

$$\begin{aligned} f(U) &\cong f(U_1) - \tilde{f}(U_1) + E[\tilde{f}(U)] && (21) \\ &= (f - \tilde{f})(U_1) + E[\tilde{f}(U)]. && (22) \end{aligned}$$

Note that control variating requires the evaluation of $E[\tilde{f}(U)]$. When this is estimated instead, we refer to it as 2-level MC and recursively applying 2-level is multilevel MC.

Example 2.3.11 (Control variate on (8))
To create a control variate for (8) that effectively reduces variance, we employ the approximation $y(t) \approx \tilde{y} = 1 + t$ and define the modified recursive term as follows:

$$\begin{aligned} y(t) \cong Y(t) &= 1 + t(Y(U_1 t) - \tilde{y}(U_1 t) + E[\tilde{y}(Ut)]) && (23) \\ &= 1 + t\,(E[1 + Ut]) + t(Y(U_1 t) - 1 - U_1 t) && (24) \\ &= 1 + t + \frac{t^2}{2} + t(Y(U_1 t) - 1 - U_1 t). && (25) \end{aligned}$$

```
function Y(t) # correct for t<1
    u = rand()
    1 + t + t^2 / 2 + (rand() < t ? Y(u * t) - 1 - u * t : 0)
end
```

Note that while we could cancel out the constant term of the control variate, doing so would have a negative impact on the Russian roulette implemented.

Related Work 2.3.12 (MC modification)
For further reference on Russian roulette, splitting and control variates see [Vea97].

2.4 Monte Carlo Trapezoidal Rule

In this subsection, we introduce a MC trapezoidal rule that exhibits similar convergence behavior to the methods discussed later. The MC trapezoidal rule is essentially a regular Monte Carlo method enhanced with control variates based on the trapezoidal rule.

Definition 2.4.1 (MC trapezoidal rule)
We define the MC trapezoidal rule for approximating the integral of function f over the interval $[x, x+\Delta x]$ with a Russian roulette rate l and $\tilde{f}$ represents the linear approximation of f corresponding to the trapezoidal rule as follows:

$$
\begin{aligned}
&\int_x^{x+\Delta x} f(s)ds && (26)\\
&= \int_x^{x+\Delta x} \tilde{f}(s)ds + \int_x^{x+\Delta x} f(s) - \tilde{f}(s)ds && (27)\\
&= \Delta x \frac{f(x)+f(x+\Delta x)}{2} + E\left[f(S) - \tilde{f}(S)\right] && (28)\\
&\cong \Delta x \frac{f(x)+f(x+\Delta x)}{2}\\
&+ \Delta x l B\left(\frac{1}{l}\right)\left(f(S) - f(x) - \frac{S-x}{\Delta x}\left(f(x+\Delta x) - f(x)\right)\right), && (29)
\end{aligned}
$$

where $S \sim \text{Uniform}(x, x+\Delta x)$.

Lemma 2.4.2 (RMSE MC Trapezoidal Rule)
The MC trapezoidal rule for a twice differentiable function has

$$
\text{RMSE} = O\left(\Delta x^3\right). \tag{30}
$$

Proof. Start from (29). The MSE is the variance so we can ignore addition by constants.

$$
\text{MSE} = \text{Var}\left(\Delta x l B\left(\frac{1}{l}\right)\left(f(S) - f(x) - \frac{S-x}{\Delta x}\left(f(x+\Delta x) - f(x)\right)\right)\right) \tag{31}
$$

We substitute $S = \Delta x U + x$ and apply Taylor's theorem finishing the proof:

$$
\begin{aligned}
\text{MSE} &= \text{Var}\left(\Delta x l B\left(\frac{1}{l}\right)\left(f(\Delta x U + x) - f(x) - U\left(f(x+\Delta x) - f(x)\right)\right)\right) && (32)\\
&= \text{Var}\left(\Delta x l B\left(\frac{1}{l}\right)\left(U\Delta x f'(x) + \frac{U^2\Delta x^2}{2} f''(Z_1) - U\left(\Delta x f'(x) + \Delta x^2 f''(z_2)\right)\right)\right) && (33)\\
&= \text{Var}\left(\Delta x l B\left(\frac{1}{l}\right)\left(\frac{U^2\Delta x^2}{2} f''(Z_1) - \frac{U\Delta x^2}{2} f''(z_2)\right)\right) && (34)\\
&= \Delta x^6 \text{Var}\left(l B\left(\frac{1}{l}\right)\left(\frac{U^2}{2} f''(Z_1) - \frac{U}{2} f''(z_2)\right)\right), && (35)
\end{aligned}
$$

for some $Z_1 \in [x, S], z_2 \in [x, x+\Delta x]$. The variance term is bounded because the variance of a bounded RV is bounded. Note that the proof does not rely on Russian roulette ($l = 1$). □

Related Work 2.4.3 (proof of Lemma 2.4.2)
A more generalizable proof for other types of control variates can be constructed by applying the 'separation of the main part' technique, as shown in Lemma 4 of [HM93].

Definition 2.4.4 (Composite (MC) trapezoidal rule)
Define the MC trapezoidal rule for approximating the integral of function f over the interval $[0,1]$ with a uniform grid with n intervals as follows:

$$\int_0^1 f(s)ds \approx \Delta x \sum_{j=0}^{n-1} \frac{f(x_j) + f(x_j + \Delta x)}{2}. \tag{36}$$

```julia
trapezium(f, n) = sum((f(j / n) + f((j + 1) / n)) / 2 for j in 0:n-1) / n
```

Define the corresponding composite MC trapezoidal with a Russian roulette rate l as follows:

$$\int_0^1 f(s)ds \cong \Delta x \sum_{j=0}^{n-1} \frac{f(x_j) + f(x_j + \Delta x)}{2} + lB\left(\frac{1}{l}\right)\left(f(S_j) - f(x_j) - \frac{S_j - x_j}{\Delta x}(f(x_j + \Delta x) - f(x_j))\right), \tag{37}$$

```julia
function MCtrapezium(f, n, l=100)
    sol = 0
    for j in 0:n-1
        if rand() * l < 1
            x, xx = j / n, (j + 1) / n
            S = x + rand() * (xx - x) # \sim Uniform(x, xx)
            sol += l * (f(S) - f(x) - n * (S - x) * (f(xx) - f(x))) / n
        end
    end
    return trapezium(f, n) + sol
end
```

where $S_j \sim \text{Uniform}(x, x + \Delta x)$.

Figure 1 suggests that the order of convergence of RMSE of the composite MC trapezoidal rule is better by 0.5 than the normal composite trapezoidal rule. The MC trapezoidal rule has on average $\frac{1}{l}$ more function calls than the normal trapezoidal rule. For the composite rule with n intervals, there are $\text{Binomial}(n, \frac{1}{l})$ additional function calls (repeated Bernoulli experiments).

Theorem 2.4.5 (RMSE Composite Trapezoidal MC Rule)
The composite trapezoidal MC rule with n intervals for a twice differentiable function has

$$\text{RMSE} = O\left(\frac{1}{n^{2.5}}\right). \tag{38}$$

The proof uses Lemma 2.4.2, and is similar to the proof of the normal trapezoidal rule. The main difference is the accumulation of 'local truncation' errors into 'global truncation' error. Normally there is a loss of one order but the MC trapezoidal loses

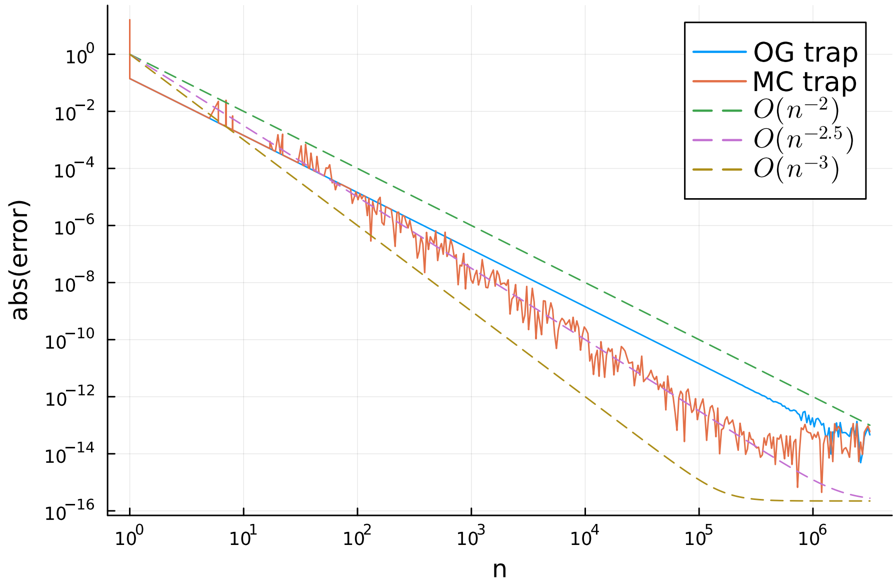

Figure 1: Log-log plot of the error of (37) for $\int_0^1 e^s ds$ with $l = 100$. At floating point accuracy, the convergence ceases.

only a half order because the accumulation happens in variance instead of bias.

$$\sqrt{\operatorname{Var}\left(\sum_{j=1}^{n} \Delta x^3 U_j^2\right)} = \Delta x^3 \sqrt{\sum_{j=1}^{n} \operatorname{Var}(U_j^2)} \tag{39}$$

$$= \Delta x^3 \sqrt{n \operatorname{Var}(U^2)} \tag{40}$$

$$= O(\Delta x^{2.5}). \tag{41}$$

Note that the meaning of a bound on the error, which behaves as $O(\Delta x^2)$, and a bound on the RMSE, also behaving as $O(\Delta x^2)$, is different. A bound on the error implies a bound on the RMSE, but not vice versa.

Related Work 2.4.6 (Monte Carlo Trapezoidal Rule)
Due to an argument like Stein's paradox, it is always possible to bias the composite MC trapezoidal rule to achieve lower RMSE. The optimal IBC for the deterministic, random, and quantum integration are known for some smoothness classes; see [HN01] for details.

2.5 Unbiased Non-Linearity

In this subsection, we present techniques for handling polynomial non-linearity. The main idea behind this is using independent samples $y^2 \cong Y_1 Y_2$ with Y_1 independent of Y_2 and $Y_1 \cong Y_2 \cong y$.

Example 2.5.1 ($y_t = y^2$)
Consider the following ODE:

$$y_t = y^2, \quad y(1) = -1. \tag{42}$$

The solution to this equation is given by $y(t) = -\frac{1}{t}$. By integrating both sides of (42), we obtain the following integral equation:

$$y(t) = -1 + \int_1^t y(s)y(s)ds. \tag{43}$$

To estimate the recursive integral in (42), we use Y_1, Y_2 i.i.d. Y in following RRVE:

$$y(t) \cong Y(t) = -1 + (t-1)Y_1(S_1)Y_2(S_1), \tag{44}$$

```
function Y(t) # Y(u)^2 != Y(u) * Y(u) !!!
    t > 2 && error("doesn't support t > 2")
    S1 = rand() * (t - 1) + 1
    rand() < t - 1 ? -1 + Y(S1) * Y(S1) : -1
end # Y(t) = 0 or Y(t) = -1
```

where $S_1 \sim \text{Uniform}(1, t)$.

Example 2.5.2 ($e^{E[X]}$)

$e^{\int x(s)ds}$ is a common expression encountered when studying ODEs. In this example, we demonstrate how you can generate unbiased estimates of $e^{E[X]}$ with simulations of X. The Taylor series of e^x is:

$$e^{E[X]} = \sum_{n=0}^{\infty} \frac{E^n[X]}{n!} \tag{45}$$

$$= 1 + \frac{1}{1}E[X]\left(1 + \frac{1}{2}E[X]\left(1 + \frac{1}{3}E[X]\left(1 + ...\right)\right)\right). \tag{46}$$

Change the fractions of (46) to Bernoulli processes and replace all X with independent $X_j \cong X$.

$$e^{E[X]} = E\left[1 + B\left(\frac{1}{1}\right)E[X_1]\left(1 + B\left(\frac{1}{2}\right)E[X_2]\left(1 + B\left(\frac{1}{3}\right)E[X_3]\left(1 + ...\right)\right)\right)\right] \tag{47}$$

$$\cong 1 + B\left(\frac{1}{1}\right)X_1\left(1 + B\left(\frac{1}{2}\right)X_2\left(1 + B\left(\frac{1}{3}\right)X_3\left(1 + ...\right)\right)\right). \tag{48}$$

```
num_B(i) = rand() * i < 1 ? num_B(i + 1) : i - 1
res(X, n) = (n != 0) ? 1 + X() * res(X, n - 1) : 1
expE(X) = res(X, num_B(0))
```

Sampling (48) requires a finite amount of samples from X_j's with probability 1.

Related Work 2.5.3 (Example 2.5.2)

NVIDIA has a great paper on optimizing Example 2.5.2 [Ket+21].

2.6 Recursion

In this subsection, we discuss recursion-related techniques.

Technique 2.6.1 (Coupled recursion)

The idea behind coupled recursion is sharing recursion calls of multiple RRVEs for simulation. This does make them dependent.

Example 2.6.2 (Coupled recursion)

Consider calculating the sensitivity of following ODE to a parameter a:

$$y_t = ay, y(0) = 1 \Rightarrow \tag{49}$$

$$\partial_a y_t = y + a\partial_a y \tag{50}$$

Turn (49) and (50) into RRVEs. To emphasize that they are coupled and should recurse together we write them in a matrix equation:

$$\begin{bmatrix} y(t) \\ \partial_a y(t) \end{bmatrix} \cong \begin{bmatrix} Y(t) \\ \partial_a Y(t) \end{bmatrix} = X(t) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} a & 0 \\ 1 & a \end{bmatrix} X(Ut). \tag{51}$$

```
(q = [1, 0]; A(a) = [a 0; 1 a])
X(t, a) = (rand() < t) ? q + A(a) * X(rand() * t, a) : q
```

Observe how this eliminates the additive branching recursion present in (50).

Technique 2.6.3 (Recursion in recursion)

Later we will use recursion in recursion. Recursion in recursion is like proving an induction step of an induction proof with induction.

Related Work 2.6.4 (Recursion in recursion)

A beautiful example of recursion in recursion is the next flight variant of Walk on Spheres in [Saw+22].

Most programming languages do support recursion, but it often comes with certain limitations such as maximum recursion depth and potential performance issues. Non-branching recursion can be avoided by directly traversing the computational path in the reverse order. Poisson time processes can be sampled in the reverse order because of equal forward paths and backward paths distributions.

Definition 2.6.5 (Main Poisson)

Consider the following linear IVP with $\forall s > 0 \in \mathbb{R} : f(s) \in \mathbb{R}^n, A(s) \in \mathbb{R}^{n \times n}$:

$$y_t = A(t)y + f(t) \Leftrightarrow \tag{52}$$

$$y_t + \sigma y = (A(t) + \sigma I)y + f(t) \Leftrightarrow \tag{53}$$

$$e^{-\sigma t}(e^{\sigma t} y)_t = (A(t) + \sigma I)y + f(t) \Leftrightarrow \tag{54}$$

$$y(t) = e^{-\sigma t} y(0) + \int_0^t e^{(s-t)\sigma} \left((A(s) + \sigma I)y(s) + f(s) \right) ds. \tag{55}$$

With $\sigma > 0 \in \mathbb{R}$. Do following substitution: $e^{(s-t)\sigma} = \tau$ equivalent to importance sampling the exponential term:

$$y(t) = \int_0^{e^{-\sigma t}} y(0) d\tau + \int_{e^{-\sigma t}}^1 \left(\frac{A(s)}{\sigma} + I \right) y(s) + \frac{f(s)}{\sigma} d\tau. \tag{56}$$

Sampling τ uniformly is equivalent to sampling the next event in a Poisson process, which produces an unbiased estimator of the solution to the linear IVP.

Julia Code 2.6.6 (Implementation of 2.6.5)
(56) can be turned into a RRVE by sampling $\tau \sim U$, no Russian roulette is needed for termination because sampling the first integral ends recursion. No recursion is used for the implementation as we can sample the Poisson process in the reverse order.

```
function Y(t, sig, A::Function, f::Function, y0)
    (s = -log(rand()) / sig; sol = y0)
    while s < t
        sol += (A(s) * sol .+ f(s)) ./ sig
        s -= log(rand()) / sig
    end
    sol
end
```

Figure 2 and Figure 3 demonstrate the convergence behavior of the estimator. The estimated convergence of RMSE is $O(\sigma^{-0.5} nsim^{-0.5})$, the amount of time steps is simply $\sim$ Poisson(σt). Note that unlike MC integration the cost of a simulation is not a bounded, resulting in increased wall clock time at risk as $nsim$ grows for parallel simulations.

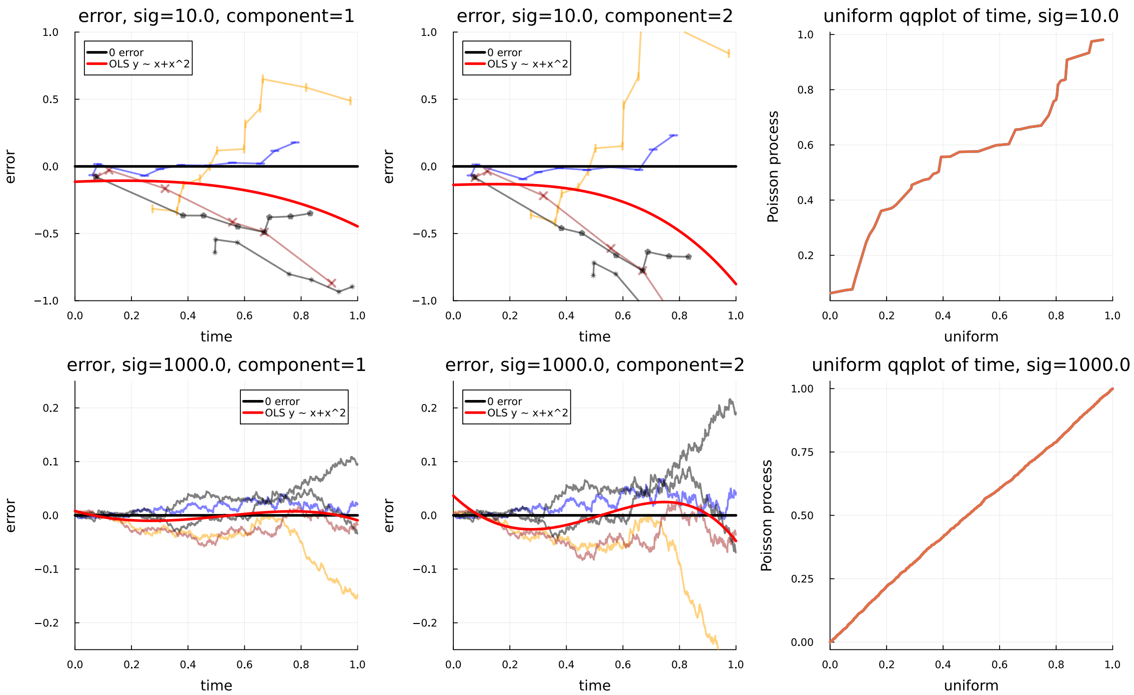

Figure 2: Intermediate points of 5 realizations of the error $(Y(t) - y(t))$ of 2.6.6 for (49) and (50) with $a = 1, \sigma = \{10, 1000\}$. Regression over the realizations and time is done with ordinary least square onto a second degree polynomial. The uniform qqplot of the time realizations indicate that the time realizations are uniformly distributed.

Related Work 2.6.7 (Main Poisson)
[AR16] derives a similar estimator for solving 1-dimensional telegraph equations.

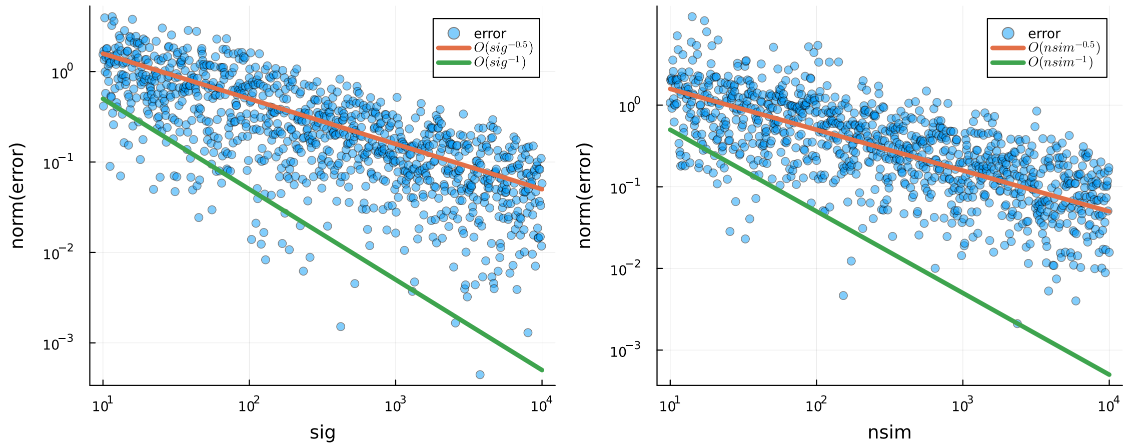

Figure 3: Realizations of norm of the error at $t = 1$ of 2.6.6 for (49) and (50) with $a = 1$, $nsim = 1$ and σ variable or $nsim$ variable and $\sigma = 1$. $nsim$ denoting the number of simulations when splitting the whole estimator.

Tail recursion is commonly used in rendering to implement non-branching recursion.

Technique 2.6.8 (Non-branching tail recursion)

Tail recursion involves reordering all operations so that almost no operation needs to happen after the recursion call. This allows us to return the answer without retracing all steps when we reach the last recursion call.

The non-branching recursion presented in the RRVEs can be implemented using tail recursion due to the associativity of all operations ($(xy)z = x(yz)$) involved.

Julia Code 2.6.9 (Tail recursion for Example 2.6.5)

We collect the sum in the variable sol and the matrix multiplications in W. Notice that the computation is going backwards in time.

```julia
using LinearAlgebra
function Y(t, sig, A::Function, f::Function, y0)
    (sol = zero(y0); W = I; s = t + log(rand()) ./ sig)
    while 0 < s
        sol += W * f(s) ./ sig
        W += W * A(s) ./ sig
        s += log(rand()) ./ sig
    end
    sol + W * y0
end
```

Tail recursion loses the intermediate values of the recursive calls and may increase computational costs because of the reordering of operations. The computational cost of matrix multiplications are sensitive to reordering operations.

Related Work 2.6.10 (Non-branching tail recursion)

[VSJ21] proposes an efficient unbiased backpropagation algorithm for rendering exploiting properties of non-branching tail recursion by inverting local Jacobian matrices which is only feasible for small matrices. To access intermediate values in tail recursion they use path replay.

The expensive matrix multiplications disappear when solving for $v^T y(t)$.

Julia Code 2.6.11 (Adjoint tail recursion for Example 2.6.5)

We collect the sum in the variable sol and instead of matrix multiplications in W matrix-vector multiplications in v.

```
function Yadj(t, sig, A::Function, f::Function, y0, v) # unbiased v'*y
    (sol = 0; s = t + log(rand()) ./ sig)
    while 0 < s
        sol += dot(v, f(s)) ./ sig
        v += v * A(s) ./ sig
        s += log(rand()) ./ sig
    end
    sol + dot(v, y0)
end
```

We are interested in methods that can keep v sparse. When $(v)_j = \delta_{jn}$ we obtain an estimator for $(y(t))_n$.

Related Work 2.6.12 (Adjoint tail recursion)

[ES21] also proposes an unbiased method for functionals of linear IVPs.

3 Ordinary Differential Equations

3.1 Green's Functions

In this subsection, we discuss how to transform ODEs into integral equations using Green's functions and subsequently solve them.

A Green's function is a type of kernel function used for solving linear problems with linear conditions. In this context, Green's functions are analogous to homogeneous and particular solutions to a specific problem, which are combined through integration to solve more general problems.

Related Work 3.1.1 (Green's function)

Our notion of the Green's function is similar to that presented in [HGM01].

Notation 3.1.2 (H)

We denote the Heaviside step function with:

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{else} \end{cases}. \tag{57}$$

Notation 3.1.3 (δ)

We denote the Dirac delta function as $\delta(x)$.

To clarify Green's functions, let us look at the following example.

Example 3.1.4 ($y_t = y$ average condition)

Consider equation:

$$y_t = y, \tag{58}$$

with following condition making the solution unique:

$$\int_0^1 y(s)ds = e - 1. \tag{59}$$

The solution to this equation remains the same: $y(t) = e^t$. We define the corresponding source Green's function $G(t, x)$ for y_t and this type of condition as follows:

$$G_t = \delta(x - t), \quad \int_0^1 G(s, x)ds = 0. \tag{60}$$

Solving this equation yields:

$$G(t, x) = H(t - x) + x - 1. \tag{61}$$

We define the corresponding boundary Green's function $G(t, x)$ for y_t and this type of condition as follows:

$$P_t = 0, \quad \int_0^1 P(s)ds = e - 1. \tag{62}$$

Solving this equation yields:

$$P(t) = e - 1. \tag{63}$$

The Green's functions are constructed to form the following integral equation for (58):

$$y(t) = P(t) + \int_0^1 G(t,s)y(s)ds. \tag{64}$$

Converting (64) into an RRVE using RMC, we obtain:

$$y(t) \cong Y(t) = e - 1 + 2B\left(\frac{1}{2}\right) Y(S)(H(t-S) + S - 1), \tag{65}$$

where $S \sim U$. We plot realizations of (65) in Figure 4.

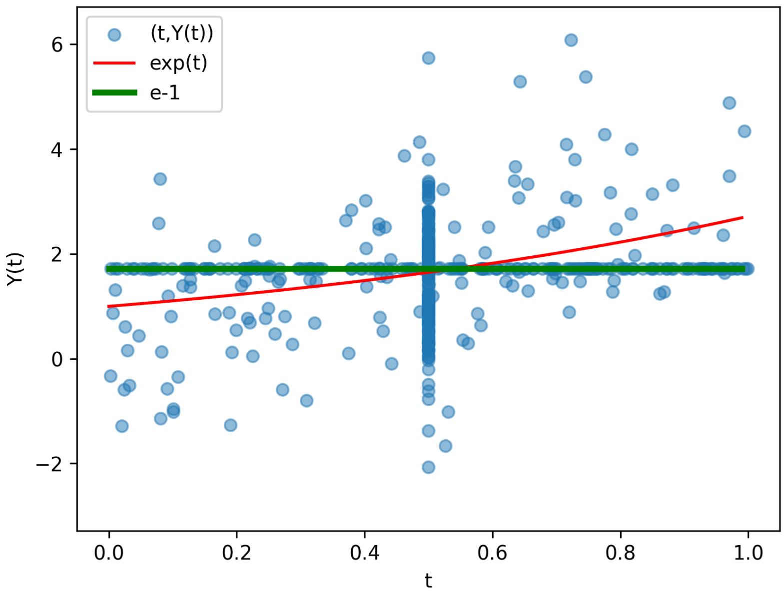

Figure 4: Recursive calls $(t, Y(t))$ of (65) when calling $Y(0.5)$ 300 times. Points accumulate on the Green's line due to the Russian roulette, and at $t = 0.5$ because of the first recursion call.

(64) is a Fredholm integral equation of the second kind.

Definition 3.1.5 (Fredholm equation of the second kind)
A Fredholm equation of the second kind for φ is of the following form:

$$\varphi(t) = f(t) + \lambda \int_a^b K(t,s)\varphi(s)ds. \tag{66}$$

Here, $K(t,s)$ represents a kernel, and $f(t)$ is a given function.

If both K and f satisfy certain regularity conditions, then for sufficiently small λ, it is relatively straightforward to establish the existence and uniqueness of solutions using a fixed-point argument.

Example 3.1.6 (Dirichlet $y_{tt} = y$)
Consider following BVP:

$$y_{tt} = y. \tag{67}$$

With boundary conditions $y(b_0), y(b_1)$. The Green's functions corresponding to y_{tt} and Dirichlet conditions are:

$$P(t|x) = \begin{cases} \frac{b_1-t}{b_1-b_0} & \text{if } x = b_0 \\ \frac{t-b_0}{b_1-b_0} & \text{if } x = b_1 \end{cases}, \tag{68}$$

$$G(t,s) = \begin{cases} -\frac{(b_1-t)(s-b_0)}{b_1-b_0} & \text{if } s < t \\ -\frac{(b_1-s)(t-b_0)}{b_1-b_0} & \text{if } t < s \end{cases}. \tag{69}$$

Directly from these Green's functions, we obtain the following integral equation and RRVE:

$$y(t) = P(t|b_0)y(b_0) + P(t|b_1)y(b_1) + \int_{b_0}^{b_1} G(t,s)y(s)ds, \tag{70}$$

$$Y(t) = P(t|b_0)y(b_0) + P(t|b_1)y(b_1) + lB\left(\frac{1}{l}\right)(b_1 - b_0)G(t,S)y(S), \tag{71}$$

with the Russian roulette rate $l \in \mathbb{R}$ and $S \sim \text{Uniform}(b_0, b_1)$. In Figure 5 and 6 we test convergence of (71).

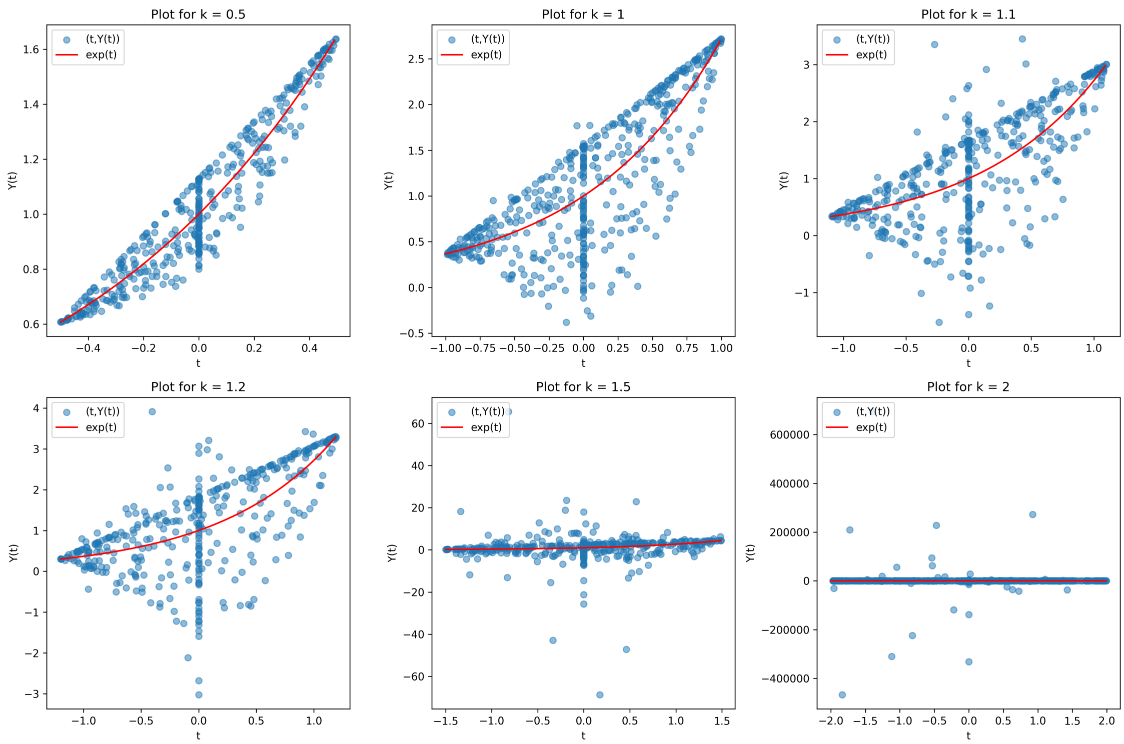

Figure 5: Recursive calls $(t, Y(t))$ of (71) when calling $Y(0)$ 75 times, with $l = 1.2$ and boundary conditions $y(-k) = e^{-k}$ and $y(k) = e^k$.

We tested coupled splitting on Example 3.1.6. Coupled splitting removes the additive branching of splitting by coupling (reusing) samples.

Example 3.1.7 (Coupled splitting on Example 3.1.6)

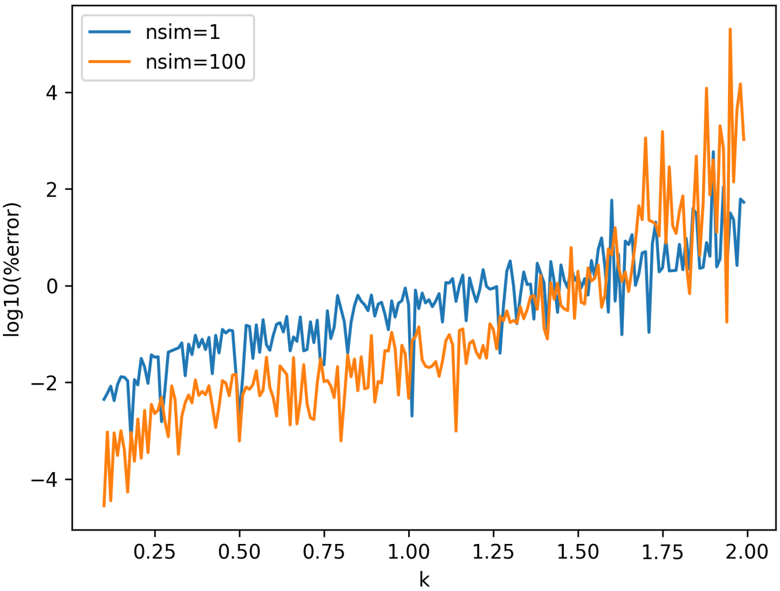

Figure 6: The logarithmic percentage error of $Y(0)$ for (71), with $l = 1.2$ and boundary conditions $y(-k) = e^{-k}$ and $y(k) = e^{k}$, displays an exponential increase until approximately $k = 1.5$, beyond which additional simulations fail to reduce the error, indicating that the variance does not exist.

In addition to normal splitting (see definition 2.3.6), we can also split the domain in (70) as follows:

$$y(t_1) = P(t_1|b_0)y(b_0) + P(t_1|b_1)y(b_1) \tag{72}$$
$$+ \int_{b_0}^{\frac{b_0+b_1}{2}} G(t_1, s_1)y(s_1)ds_1 + \int_{\frac{b_0+b_1}{2}}^{b_1} G(t_1, s_2)y(s_2)ds_2 \tag{73}$$
$$y(t_2) = P(t_2|b_0)y(b_0) + P(t_2|b_1)y(b_1) \tag{74}$$
$$+ \int_{b_0}^{\frac{b_0+b_1}{2}} G(t_2, s_1)y(s_1)ds_1 + \int_{\frac{b_0+b_1}{2}}^{b_1} G(t_2, s_2)y(s_2)ds_2. \tag{75}$$

By coupling, we can eliminate the additive branching recursion in the RRVEs corresponding to (72) and (74). In other words, to estimate $y(t_1)$ and $y(t_2)$ we need to estimate $y(s_1)$ and $y(s_2)$ instead of doing it twice we reuse the same estimate.
This results in the following RRVE:

$$X(t_1, t_2) = \begin{bmatrix} P(t_1|b_0) & P(t_1|b_1) \\ P(t_2|b_0) & P(t_2|b_1) \end{bmatrix} \begin{bmatrix} y(b_0) \\ y(b_1) \end{bmatrix} + \frac{b_1 - b_0}{2} \begin{bmatrix} G(t_1, S_1) & G(t_1, S_2) \\ G(t_2, S_1) & G(t_2, S_2) \end{bmatrix} X(S_1, S_2), \tag{76}$$

```
Pb0(t, b0, b1) = (b1 - t) / (b1 - b0)
Pb1(t, b0, b1) = (t - b0) / (b1 - b0)
G(t, s, b0, b1) = (t < s) ?
                  -(b1 - s) * (t - b0) / (b1 - b0) :
```

```
                        -(b1 - t) * (s - b0) / (b1 - b0)
function X(T::Array, y0, y1, b0, b1, l)
    PP = hcat([Pb0(t, b0, b1) for t in T], [Pb1(t, b0, b1) for t in T])
    sol = PP * [y0, y1]
    if rand() * l < 1
        (u = rand(); n = length(T))
        SS = [b0 + (u + j) * (b1 - b0) / n for j in 0:n-1]
        GG = reshape([G(t, S, b0, b1) for t in T, S in SS], n, n)
        sol += l * GG * X(SS, y0, y1, b0, b1) .* (b1 - b0) ./ n
    end
    sol
end
```

with $S_1 = b_0 + \frac{U}{2}(b_1 - b_0)$ and $S_2 = b_0 + \frac{U+1}{2}(b_1 - b_0)$. (76) is unbiased in the following way: $X(t_1, t_2) \cong [y(t_1) \quad y(t_2)]^T$.

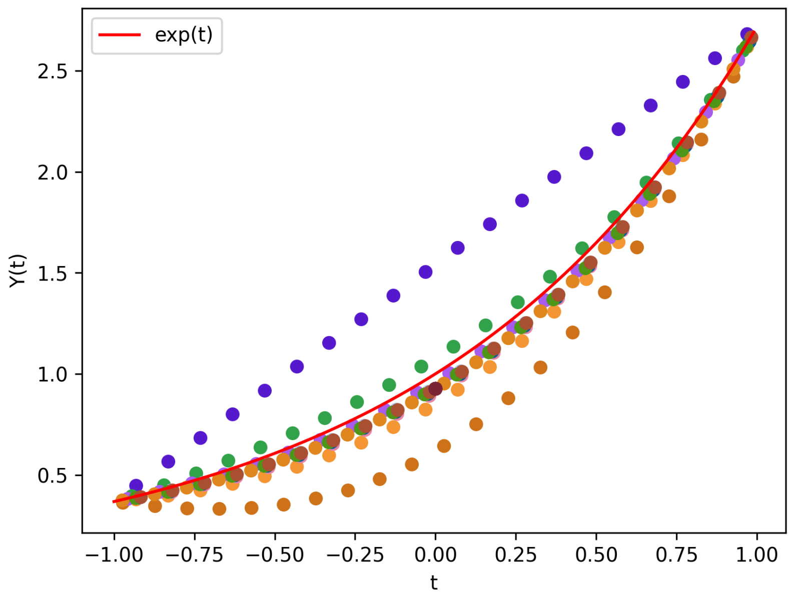

Figure 7: Recursive calls of (76) when calling $X(0)$ once. We chose X to have 20 points and colored each call. The S_j are coupled such that they are equally spaced. The initial conditions for this call are $y(-1) = e^{-1}$ and $y(1) = e^1$, with Russian roulette rate $l = 1.2$.

Related Work 3.1.8 (Coupled splitting)
Coupled splitting is partly inspired by the approach of [SM09], which reduces variance by using larger submatrices in unbiased sparsification of matrices. Reusing samples for Walk on Spheres is discussed in both [Mil+23] and [BP23].

Figure 7 resembles fixed-point iterations, leading us to hypothesize that the convergence speed is very similar to fix-points methods until the accuracy of the stochastic approximation of the operator is reached (the approximate operator bottleneck). The approximation of the operator can be improved by increasing the coupled splitting amount when approaching the bottleneck. Alternatively when reaching the bottleneck it is possible to rely on splitting to converge.

Related Work 3.1.9 (Convergence coupled splitting)
See [GH21] for a discussion on the convergence of recursive stochastic algorithms.

Related Work 3.1.10 (IBC integral equations)
Optimal IBC is known for integral equations see [Hei98] for the solution at 1 point and the global solution.

3.2 Initial Value Problems

Classic IVP solvers rely on shrinking the time steps for convergence. In this subsection, we introduce Recursion in Recursion MC (RRMC) for IVPs which tries to emulate this behavior.

Example 3.2.1 (RRMC $y_t = y$)
We demonstrate RRMC for IVPs with

$$y_t = y, \quad y(0) = 1. \tag{77}$$

Imagine we have a time-stepping scheme $(t_n), \forall n : t_{n-1} < t_n$ then the following integral equations hold:

$$y(t) = y(t_n) + \int_{t_n}^{t} y(s)ds, \quad t > t_n. \tag{78}$$

Turn these in the following class of RRVEs:

$$y(t) \cong Y_j(t) = y(t_j) + (t - t_j)Y_j((t - t_j)U + t_j), \quad t > t_j. \tag{79}$$

The problem with these RRVEs is that we do not know $y(t_j)$. Instead, we can replace it with an unbiased estimate y_j which we keep fixed in the inner recursion:

$$y(t) \cong Y_j(t) = y_j + (t - t_j)Y_j((t - t_j)U + t_j), \quad t > t_j \tag{80}$$

$$y(t_j) \cong y_j = \begin{cases} Y_{j-1}(t_j) & \text{if } j \neq 0 \\ y(t_0) & \text{if } j = 0 \end{cases}. \tag{81}$$

```
function Y_in(t, tn, yn, h)
    S = tn + rand() * (t - tn)  # \sim Uniform(T,t)
    return rand() < (t - tn) / h ? yn + h * Y_in(S, tn, yn, h) : yn
end
function Y_out(tn, h) # h is out step size
    TT = tn > h ? tn - h : 0
    return tn > 0 ? Y_in(tn, TT, Y_out(TT, h), h) : 1
end
```

We refer to (80) as the inner recursion and (81) as the outer recursion of the recursion in recursion. The measured RMSE for estimating $y(t)$ is of the order $O\left(h^{1.5}\right)$, where h represents the step size. For the implementation we used a scaled version of the time process from Example 2.3.5 for the inner recursion, such that the average number of total inner recursion calls is en, where n represents the total number of outer recursion calls.

Conjecture 3.2.2 (local RMSE RMC)
Consider a general linear IVP:

$$y_t(t) = A(t)y(t) + g(t), y(t_0), \tag{82}$$

with A a matrix and g a vector function, both once Lipschitz differentiable with the corresponding RRVE:

$$y(t) \cong Y(t) = y(t_0) + hB\left(\frac{t-t_0}{h}\right) A(S)Y(S) + g(S), \tag{83}$$

where $S \sim \text{Uniform}(t_0, t)$. Then, the following relation holds:

$$E[||y(t_1) - Y(t_1)||^2] = O(h^2), \tag{84}$$

with $t_1 = t_0 + h$.

To achieve a higher order of convergence RRMC can be combined with control variates.

Example 3.2.3 (CV RRMC $y_t = y$)
Let us control variate Example 3.2.1.

$$y(t) = y(t_n) + \int_{t_n}^{t} y(s)ds, \quad t > t_n. \tag{85}$$

We build a control variate with a lower-order approximation of the integrand:

$$\begin{align}
y(s) &= y(t_n) + (s-t_n)y_t(t_n) + O((s-t_n)^2) \tag{86}\\
&\approx y(t_n) + (s-t_n)f(y(t_n), t_n) \tag{87}\\
&\approx y(t_n) + (s-t_n)\left(\frac{y(t_n) - y(t_{n-1})}{t_n - t_{n-1}}\right) \tag{88}\\
&\approx y(t_n)(1 + s - t_n). \tag{89}
\end{align}$$

Using (89) as a control variate for the integral:

$$\begin{align}
y(t) &= y(t_n) + \int_{t_n}^{t} y(s)ds \tag{90}\\
&= y(t_n) + \int_{t_n}^{t} y(s) - y(t_n)(1+s-t_n) + y(t_n)(1+s-t_n)ds \tag{91}\\
&= y(t_n)\left(1 + (1-t_n)(t-t_n) + \frac{t^2 - t_n^2}{2}\right) + \int_{t_n}^{t} y(s) - y(t_n)(1+s-t_n)ds. \tag{92}
\end{align}$$

Figure 8 displays the error of realizations of an RRVE constructed from (92) for various step sizes.

Similar to explicit solvers, RRMC performs poorly on stiff problems. We attempted to make RRMC more like implicit solvers but this proved difficult. Instead, we believe that an approach more like exponential integrators is more viable.

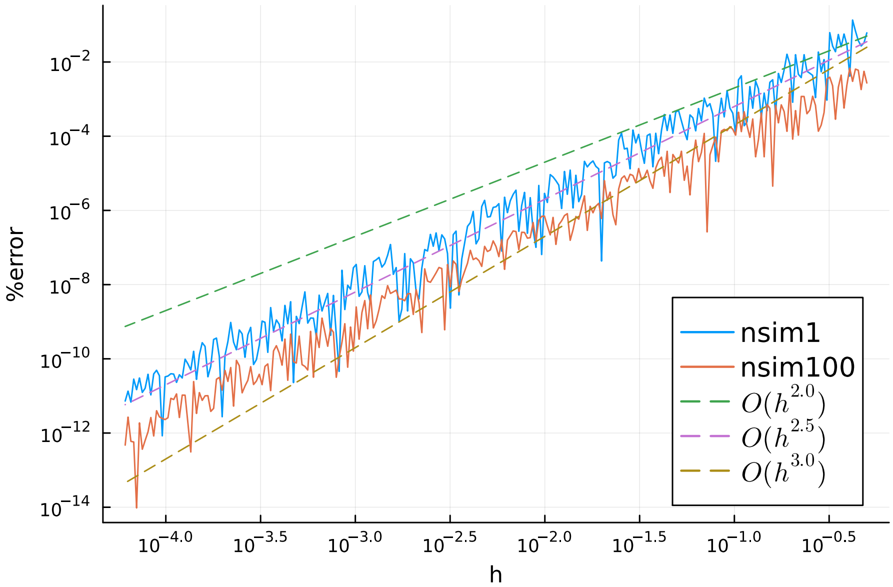

Figure 8: Log-log plot of the error for Example 3.2.3 at $Y(10)$.

4 Partial Differential Equations

In this section we discuss how to solve ODEs derived from the semi-discretized heat equation with Dirichlet boundary conditions in 1 point and acceleration with recursive first-passage sampling combined with presampled path stitching.

4.1 Heat Equation

Definition 4.1.1 (1D heat equation Dirichlet)
We define the 1D heat equation for $u(x,t)$ with $(x,t) \in \Omega = [0,1]^2$ with Dirichlet boundary conditions the following way:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \Leftrightarrow \tag{93}$$

$$u_t(x,t) = u_{xx}(x,t), \quad \forall (x,t) \in \Omega = [0,1]^2. \tag{94}$$

Given $\{u(x,0), u(1,t), u(0,t) | \forall x, t \in [0,1]\}$.

Definition 4.1.2 (Semi-discretization 1D heat equation Dirichlet)
We define the semi-discretization 1D heat equation for $\tilde{u}$ as space discretized heat equation using the central difference scheme:

$$\tilde{u}_t(j\Delta x, t) = \frac{\tilde{u}((j+1)\Delta x, t) - 2\tilde{u}(j\Delta x, t) + \tilde{u}((j-1)\Delta x, t)}{\Delta x^2} \tag{95}$$

$\forall (j,t) \in \{0,1,\dots,N\} \times [0,1]$.
Given $\{\tilde{u}(j\Delta x, 0), \tilde{u}(1,t), \tilde{u}(0,t) | \forall j \in \{0,1,\dots,N\}, \forall t \in [0,1]\}$.
It is well-known that the solution to the semi-discretization 1D heat equation converges to the solution of the 1D heat equation with corresponding boundary conditions as $\Delta x \to 0$.

Example 4.1.3 (Point estimator of 4.1.2)
We want a point estimator for the solution to the semi-discretization 1D heat equation. Apply (56) with $\sigma = \frac{2}{\Delta x^2}$ at interior point j this is:

$$\tilde{u}(j\Delta x, t) = \int_0^{e^{-\frac{2t}{\Delta x^2}}} \tilde{u}(j\Delta x, 0) d\tau + \int_{e^{-\frac{2t}{\Delta x^2}}}^1 \frac{\tilde{u}((j+1)\Delta x, s) + \tilde{u}((j-1)\Delta x, s)}{2} d\tau. \tag{96}$$

This can be turned into an estimator $Y(j\Delta x, t, \Delta x) \cong \tilde{u}(j\Delta x, t)$ by sampling $\tau \sim U$, sampling either $\tilde{u}((j+1)\Delta x, s)$ or $\tilde{u}((j-1)\Delta x, s)$ to avoid branching recursion and terminating recursion when either hitting the boundary condition or sampling the first integral. Either sampling $\tilde{u}((j+1)\Delta x, s)$ or $\tilde{u}((j-1)\Delta x, s)$ loses the convergence in σ demonstrated in Figure 3.

Julia Code 4.1.4 (Implementation of estimator of (96))

```
function Ytail(x, t, dx, u_bound::Function)
    while true
        xold = x
        t += dx^2 * log(rand()) / 2
        x += rand(Bool) ? dx : -dx
```

```
        return t - eps() <= 0 ? u_bound(xold, 0) :
               x - eps() <= 0 ? u_bound(0, t) :
               x + eps() >= 1 ? u_bound(1, t) : continue
    end
end
```

$Y(j\Delta x, t, \Delta x)$ is just the boundary condition for the first passage point from the time space domain for a Poisson time process and a random walk in space. Therefore if the boundary conditions are bounded, $Y(j\Delta x, t, \Delta x)$ is bounded and has bounded variance. In this implementation as $\Delta x \to 0$ the cost of calculating the first passage time grows $\approx O(\Delta x^{-2})$ and the process converges to Brownian motion. Note that the estimator needs only 1 function call from the boundary conditions.

Example 4.1.5 (4.1.3 + source + coefficients)
Consider the equations in 4.1.2 with a source term and 0-order coefficients, the semi-discretization of:

$$u_t = u_{xx} + a(x,t)u + f(x,t). \tag{97}$$

Again by applying (56) with $\sigma = \frac{2}{\Delta x^2} + a_0$ we obtain at an interior point:

$$\tilde{u} = \int_0^{e^{-t\sigma}} \tilde{u}_0 d\tau + \int_{e^{-t\sigma}}^1 \sigma^{-1} \left(\frac{\tilde{u}_+ + \tilde{u}_-}{\Delta x^2} + (a(x,s) + a_0)\tilde{u} + f \right) d\tau. \tag{98}$$

This can be turned into a estimator $Y(j\Delta x, t, \Delta x) \cong \tilde{u}(j\Delta x, t)$ by sampling $\tau \sim U$. To avoid branching recursion we sample based on the magnitude of coefficients of the recursive terms, let a_m be approximately the magnitude of $(a(x,s) + a_0)$. We chose to sample the $(a(x,s) + a_0)\tilde{u}$ term and f with probability $p_{\text{source}} = \frac{a_m}{a_m + \frac{2}{\Delta x^2}}$. As $\Delta x \to 0$ the $\frac{\tilde{u}_+ + \tilde{u}_-}{\Delta x^2}$ term is the main contribution to the second integral therefore being sampled almost always. In total for 1 fully recursed sample, the $f(x,s)$ and the $(a(x,s) + a_0)\tilde{u}$ term only is sampled a few times and this does not scale with Δx. We sampled f together with $(a(x,s) + a_0)\tilde{u}$ for simplicity.

Julia Code 4.1.6 (Implementation of estimator of (98))
Because of the high chance of sampling the $\frac{\tilde{u}_+ + \tilde{u}_-}{\Delta x^2}$ it is efficient to sample the inter-arrival until not sampling it which is geometrically distributed.

```
using Distributions
function Yvar(x, t, dx, a0, am, u_bound::Function, f::Function, a::Function)
    siginv = 1 / (2 / dx^2 + a0)
    (p_source = am / (am + 2 / dx^2); p_avg = 1 - p_source)
    (geom = Geometric(p_source); expon = Exponential(siginv))
    (sol = 0; w = 1)
    sterm_counter = rand(geom)
    while true
        t -= rand(expon)
        t <= eps() && return sol + w * u_bound(x, 0)
        if sterm_counter != 0
            x += rand(Bool) ? dx : -dx
            w *= 2 * siginv / (dx^2 * p_avg)
            sterm_counter -= 1
            return x - eps() <= 0 ? w * u_bound(0, t) + sol :
                   x + eps() >= 1 ? w * u_bound(1, t) + sol : continue
        else # this is only done O(am tavg) times in an estimator
```

```
18            sterm_counter = rand(geom)
19            sol += w * f(x, t) * siginv / p_source
20            w *= (a(x, t) + a0) * siginv / p_source
21        end
22    end
23 end
```

Computing the estimator requires knowing more then the first passage point unlike Example 4.1.3 but this is limited to a few points of the time space process and this amount does not scale with Δx.

Technique 4.1.7 (Presampling the process of 4.1.6)
As $\Delta x \to 0$ the bottleneck is sampling the process that moves through space and time. This process is independent of the boundary conditions, $a(x,t)$ and $f(x,t)$. So it is possible to precompute samples of the process and reuse them. Reusing paths makes the solutions correlated. To compute the estimator we need to know where we sampled the source term to update *sol* and w, how many steps we took in between to update w for the multiplications in not source points (line 13 of 4.1.6) and where termination happens.

```
using Distributions
function genPath(x, t, dx, a0, am)
    (siginv = 1 / (2 / dx^2 + a0); p_source = am / (am + 2 / dx^2))
    (geom = Geometric(p_source); expon = Exponential(siginv))
    sterm_counter = rand(geom)
    spoints = [(x, t, sterm_counter)] #maybe other data struct
    while true
        t -= rand(expon)
        t <= eps() && return (spoints, (x, t, sterm_counter))
        if sterm_counter != 0
            x += rand(Bool) ? dx : -dx
            sterm_counter -= 1
            return x - eps() <= 0 ? (spoints, (x, t, sterm_counter)) :
                   x + eps() >= 1 ? (spoints, (x, t, sterm_counter)) :
                   continue
        else # this is only done O(am tavg) times in an estimator
            sterm_counter = rand(geom)
            push!(spoints, (x, t, sterm_counter))
        end
    end
end
```

Paths for smaller $\tilde{a}_m$'s can be obtained by thinning the Bernoulli process, by rejecting source points with probability $\frac{\tilde{a}_m}{a_m}$. As $\Delta x \to 0$ *genPath* converges, the dependence on a_0 also disappears. For the heat equation *genPath* presampled paths only depend on geometry and the starting point.

Julia Code 4.1.8 (4.1.6 using precomputed paths)
We implement 4.1.6 with a precomputed path from 4.1.7 using tail recursion. Note that (x,t) is never explicitly used, only for figuring out how we exited the domain. The whole calculation is linear in a, f evaluated in the source points and u_{bound} evaluated in the exit points so there are various ways to optimize it for example using unbiased estimates instead of exact evaluations.

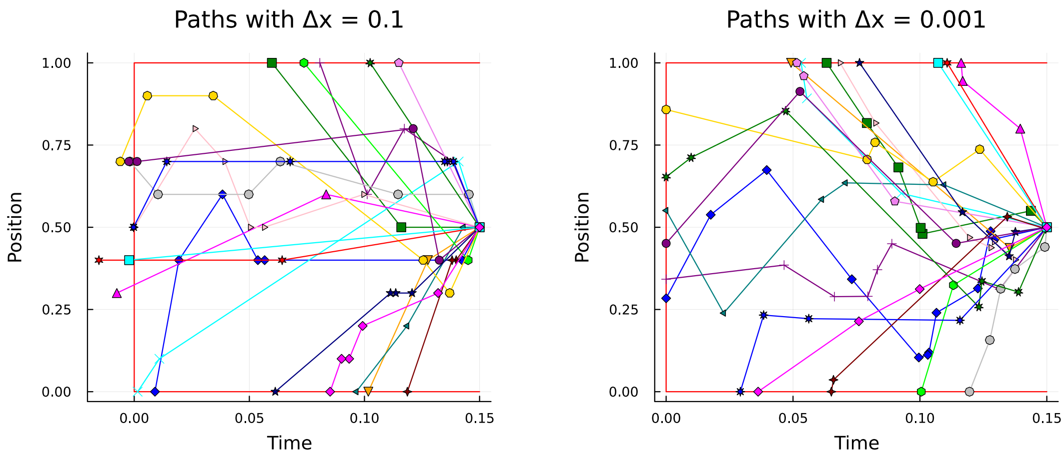

Figure 9: 20 samples of genPath($x = 0.5, t = 0.15, dx = \{0.1, 0.001\}, a_0 = 0, a_m = 30$). The source term is sampled at the intermediate points. Note that the amount of intermediate points does not differ much between $\Delta x = 0.1$ and $\Delta x = 0.001$. Points with $t < 0$ do not cause any bias, the method is unbiased for the semi-discretized heat equation.

```
function YvarPath(path, dx, a0, am, u_bound, f, a)
    spoints, exit = path
    (siginv = 1 / (2 / dx^2 + a0); p_source = am / (am + 2 / dx^2))
    w_multiplier = 2 * siginv / (dx^2 * (1 - p_source))
    (sol = 0; w = w_multiplier^spoints[1][3])
    for (x, t, sterm_counter) in spoints[2:end]
        sol += w * f(x, t) * siginv / p_source
        w *= (a(x, t) + a0) * siginv / p_source
        w *= w_multiplier^sterm_counter
    end
    (x, t, sterm_counter) = exit # small w can produce NaNs
    w /= sterm_counter > 0 && abs(w) > eps() ? w_multiplier^sterm_counter : 1
    t >= eps() ? sol + w * u_bound(x, t) : sol + w * u_bound(x, 0)
end
```

4.2 First Passage Sampling

In this subsection we discuss recursive first passage sampling to efficiently first passage sample complicated geometries.

Definition 4.2.1 (First passage time)
Define the first passage time for a process X_t for a set of valid states V as

$$\mathrm{FPt}(X_t, V) = \inf\{t > 0 | (X_t, t) \notin V\}. \tag{99}$$

Note that the first passage time is a RV itself.

Definition 4.2.2 (First passage)
Define the first passage for a process X_t for a set of valid states V as

$$\mathrm{FP}(X_t, V) = (X_\tau, \tau), \tau = \mathrm{FPt}(X_t, V). \tag{100}$$

Lemma 4.2.3
When a process has more valid states the first passage time gets larger i.e.

$$V_1 \subset V_2 \Rightarrow \mathrm{FPt}(X_t(\omega), V_1) \leq \mathrm{FPt}(X_t(\omega), V_2). \tag{101}$$

The ω is to indicate the same realization of X_t.

Theorem 4.2.4 (Green's functions and first passage distribution)
The density of first passages of Code 4.1.4 is the corresponding Dirichlet boundary Green's function for the semi-discretized heat equation.

Proof. Follows directly from the definition. Let $P(x, t|x_0, t_0)$ denote the boundary Green's function for the semi-discretized heat equation and X_t the corresponding space time process.

$$\begin{aligned} P(x, t|x_0, t_0) &= E[Y(x_0, t_0)] & (102)\\ &= E[\tilde{u}(X_\tau, \tau)|X_{t_0} = x_0] & (103)\\ &= E[\delta_{((X_\tau,\tau)=(x,t))}|X_{t_0} = x_0] & (104)\\ &= P((X_\tau, \tau) = (x, t)|X_{t_0} = x_0). & (105)\end{aligned}$$

□

Another way to obtain an unbiased estimator for the semi-discretized heat equation is using unbiased estimates of the boundary Green's boundary function. Because of:

$$\begin{aligned} \tilde{u}(x_0, t_0) &= \int_{\partial\Omega} \tilde{u}(x, t) P(x, t|x_0, t_0) d(x, t) & (106)\\ &= \int_{\partial\Omega} \tilde{u}(x, t) dP((X_\tau, \tau) = (x, t)|X_{t_0} = x_0). & (107)\end{aligned}$$

Related Work 4.2.5 (Recursively estimating Green's functions)
In [Qi+22] unbiased estimators for the boundary Green's function are constructed by recursively sampling known Green's functions.

Similar to recursively sampling simpler Green's function we recursively sample first passages.

Technique 4.2.6 (Recursive first passage sampling)
Recursive first passage sampling involves sampling an initial, simpler first passage, the base that includes fewer valid states. Using this sampled first passage as a starting point, we then perform the same sampling process until the sampled first passage is almost invalid.

Example 4.2.7 (Euler first passage sampling)
In this example, we approximately sample the first passage of Brownian motion for a parabolic barrier by simulating Brownian motion with the Euler scheme. We plotted this in Figure 10. The accuracy of the sampled first passage are $O(\Delta t)$ and the cost to sample is $O(\Delta t^{-1})$.

```
in_triangle(time, pos) = (1 - time > abs(pos))
function sample_euler_triangle(dt=0.001)
    (time = 0; pos = 0)
    while in_triangle(time, pos)
        time += dt
        pos += sqrt(dt) * randn()
    end
    return (time, pos)
end
```

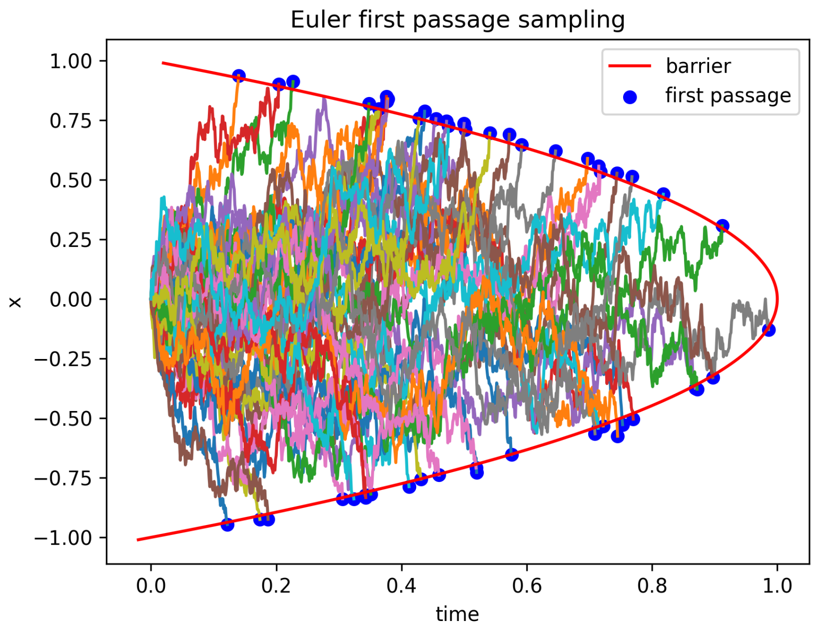

Figure 10: 50 realizations of Euler first passage sampling with step size 0.001.

Example 4.2.8 (Recursive first passage sampling)
In this example, we sample the first passage of Brownian motion from a parabolic barrier with recursive first passage sampling. For the simpler first passage sampler, we exploit the self-affinity of Brownian motion ($\frac{W_{ct}}{\sqrt{c}} \sim W_t$) by scaling and translating samples from a triangular barrier so its valid states are contained in the parabola generated by the Euler scheme. The precomputed samples are created by 4.2.7. Assuming that the error of the base sampler is insignificant. The accuracy of the sampled first passage are $O(\varepsilon)$ and the cost to sample is $O(\ln(\varepsilon^{-1}))$. Note that resampling with replacement produces dependent samples and introduces additional error. For MC applications dependence is not an issue and when the additional error is significant resampling without replacement can be used.

```
function triangle_scale_in_para(time, pos)
    xx = sqrt(1 - time) - abs(pos)
    tt = abs(1 - abs(pos)^2 - time)
    return sqrt(tt) < xx ? sqrt(tt) : xx
end
function sample_recursive_para(accuracy=0.01, scale_mul=0.9)
```

```
    (time = 0.0; pos = 0.0)
    scale = triangle_scale_in_para(time, pos)
    while scale > accuracy
        scale *= scale_mul
        dtime, dpos = triangle_sample[rand(1:length(triangle_sample))]
        dtime, dpos = scale^2 * dtime, scale * dpos
        (time += dtime; pos += dpos)
        scale = triangle_scale_in_para(time, pos)
    end
    return (time, pos)
end
```

When the domain is convex, the maximum scaling of the triangular barrier that fits is easier to obtain. To dampen barrier overstepping of 4.2.7 we use a smaller scaling than the maximum.

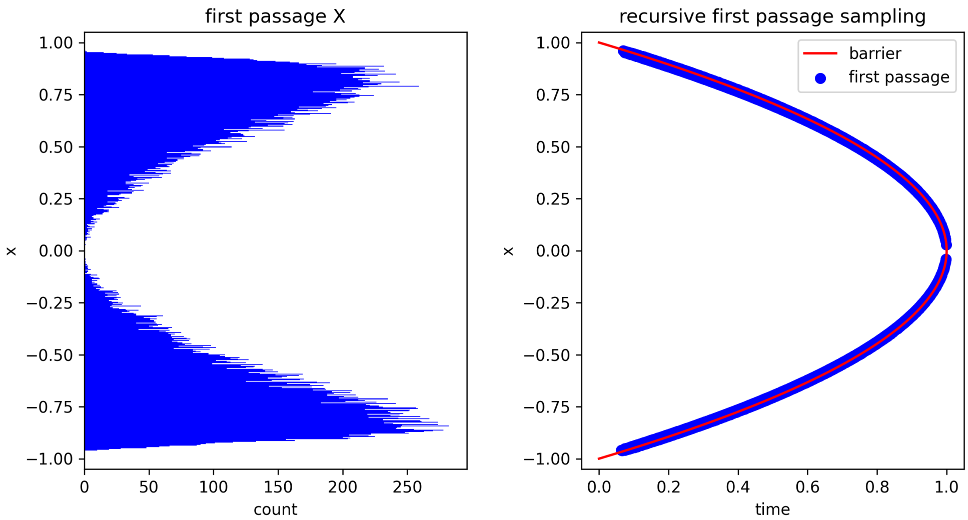

Figure 11: 50000 of realizations of recursive first passage sampling produced by (4.2.8). The precomputed sample of 5000 first passages of a triangular barrier uses the Euler scheme with step size 0.001.

Related Work 4.2.9 (Recursive first passage sampling)
Walk on Spheres see 2.2.4 is a recursive first passage algorithm. Recursive first passage sampling for Brownian motion is discussed in [HT16] and by transformation also first passage problems for the Ornstein-Uhlenbeck process. An alternative to resampling from an Euler scheme is to use tabulated inverse cumulative probability functions, as demonstrated in [HGM01].

Technique 4.2.10 (Brownian motion path stitching)
Sampling approximate Brownian motion paths also can be framed as a first passage problem. Instead of sampling first passage points we sample first passage paths. This requires a simple first passage paths sampler and the paths sampled this way are stitched together.

Example 4.2.11 (Path stitching parabola)
Example 4.2.8 but resampling full Euler scheme generated paths. The advantages

over generating paths directly from the Euler scheme are that computations for all small steps from the base sampler are avoided and the full path can be represented by its subpaths and their scalings requiring less memory than a fully stored path. The only downsides are correlation between paths and inhomogeneous time steps.

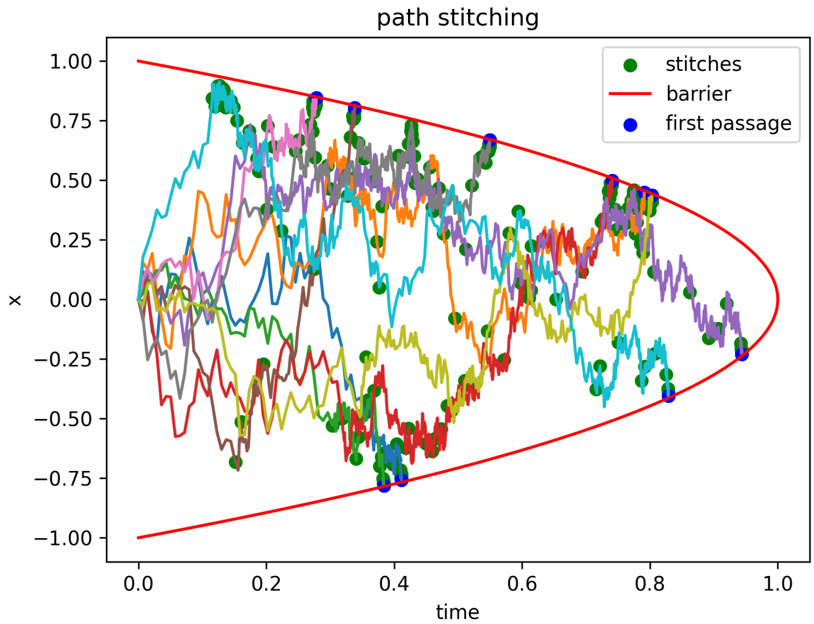

Figure 12: 10 paths build with path stitching build out of resampled Euler scheme generated paths with step size 0.01*.*

Related Work 4.2.12 (Path stitching)
Path stitching appears frequently in rendering and also in [Das+15] and [JL12].

5 Limitations and Future Work

While we discussed many different topics, the scope was limited. There are still several areas that require further exploration, development, and better integration into the existing literature.

Initially, our primary reference was the Walk on Spheres literature. The application of its techniques to IVPs guided us to random convergence for integration/IVPs, using a Poisson process to bypass recursion, and the recursive estimator for $e^{E[X]}$. These different discoveries resulted in numerous structural modifications in our thesis, resulting in a disjointed narrative throughout.

One area that definitely deserves further investigation is the main Poisson algorithm, which we discovered after completing our section on RRMC. Our investigation of the algorithm is rushed, and we are confident that there are additional valuable properties that remain to be discovered. We are interested in methods that exhibit behavior similar to RRMC derived from or related to the Main Poisson algorithm. Additionally, we are also interested in methods of these types for stiff problems.

In addition to this, our methods for linear IVPs, which involve transformation to integral equations, could potentially be expanded to linear delay differential equations. Besides the integral equation approach, another direction involves direct estimation of the Magnus expansion. This would require unbiased estimators for expressions of the type $e^{\int_0^{\Delta t} A(s)ds}y(0)$. This is related to the work on evaluating matrix functions using MC methods, as referenced in [GAM23]. An alternative approach could be to combine unbiased piecewise constant estimators of $A(t)$ and $f(t)$ with solvers for linear constant variable IVPs.

In the subsection discussing Green's functions, we formulated global integral equations for a test problem. We interested if the local characteristics of ODEs persist in these global integral equations. Specifically, if this local aspect can be utilized to improve the efficiency of estimators tasked with solving these integral equations.

The practicality of coupled splitting remains uncertain. Despite its favorable convergence behavior and unbiasedness at each iteration, the implementation presents practical difficulties due to the use of recursion and Russian roulette. Additionally, the dense matrix multiplications make it too costly to be efficient. However, these challenges do not rule out the potential for performance optimization of coupled splitting using standard MC techniques.

We used the adjoint main Poisson method as the basis for deriving our point estimators for the heat equation. Another point of view can be obtained by using the Feynman-Kac formula by MC estimating the integral over time and doing a space discretization. However, it's unclear which types of problems can be tackled using the adjoint main Poisson method in a way similar to the heat equation. Ideally, we would like to find something akin to the Feynman-Kac formula for different PDEs that retains as many of the desirable properties as possible, but is more broadly applicable. Furthermore, we have yet to fully investigate how to merge presampling with path stitching and under what circumstances this is feasible. An interesting observation when combining presampling with path stitching is that we can interpret subpaths as iid variables in a U-estimator.

Identifying suitable applications for the proposed algorithms could result in specialized versions tailored specifically for those use cases. Potential applications could be in fields where MC methods are already prevalent. Unbiased algorithms naturally fit scenarios where the primary computational load or the quantity of interest is an average. This is particularly true in situations with minimal structure, where the advantage of IBC over deterministic algorithms are meaningful, or where the linear trade-off between cost and variance is nearly optimal. We believe that unbiased linear IVPs solvers are important to developing/understanding randomized ODE/PDE solvers. The applications beyond general ODEs/PDEs that we have in mind include enhancing the efficiency of existing WoS methods, adapting MC algorithms in reinforcement learning, and creating stochastic gradient descent methods from an ODE perspective that utilize stochastic step sizes to improve stability.

Finally, one of the elements lacking in our findings is rigor. We attempted to derive an expression for the variance by employing the law of total variance, similar to [Rat+22]. However, we were unaware of [ES21] at the time, which presents theorems to show that their estimators have finite variance and provide an expression for it but this aplies only directly on the adjoint Main Poisson algorithm and may require some work to further generalize. We believe that proving the optimality of IBC in Example 3.2.3 is feasible but tedious, by a proof using a lower bound on IBC from integration and proving it is attained.

Abstract

Unbiased algoritmes voor het oplossen van lineaire beginwaardeproblemen zijn nauwelijks bestudeerd. In deze thesis presenteren wij unbiased recursieve Monte Carlo methoden voor het oplossen van lineaire beginwaardeproblemen en lineaire Fredholm-integraalvergelijkingen van de tweede soort. Gemotiveerd door random parallelle complexiteit en downstream-toepassingen voor partiële differentiaalvergelijkingen.

Voorheen waren alleen biased methoden bekend die optimale convergentiesnelheden bereiken voor niet-lineaire beginwaardeproblemen, wat voornamelijk theoretische betekenis heeft. De voorgestelde RRMC methode, een unbiased algoritme voor lineaire beginwaardeproblemen, samen met control variates, wordt vermoed orde-optimaal te zijn met hoge kans voor overeenkomstige gladheidsklassen.

Daarnaast wordt het main Poisson algoritme voorgesteld, dat het algoritme in [AR16] veralgemeend, dat simpel en voorwaarts-implementeerbaar is. Het main Poisson algoritme wordt toegepast op de semi-gediscretiseerde warmtevergelijking om een pad-resampling-techniek te demonstreren, dit biedt een nieuw perspectief van het Walk on Spheres algoritme, met potentieel voor alternatieve veralgemeningen.

De voorgestelde methoden verbreden wat mogelijk lijkt te zijn met unbiased algoritmes voor het oplossen van lineaire beginwaardeproblemen, en wijzen de weg uit voor verder onderzoek in dit gebied.

References

[AR16] Juan A. Acebrón and Marco A. Ribeiro. "A Monte Carlo method for solving the one-dimensional telegraph equations with boundary conditions". en. In: *Journal of Computational Physics* 305 (Jan. 2016), pp. 29–43. ISSN: 00219991. DOI: 10.1016/j.jcp.2015.10.027. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999115006919 (visited on 09/09/2023).

[BP23] Ghada Bakbouk and Pieter Peers. "Mean Value Caching for Walk on Spheres". en. In: (2023). Artwork Size: 10 pages Publisher: The Eurographics Association, 10 pages. DOI: 10.2312/SR.20231120. URL: https://diglib.eg.org/handle/10.2312/sr20231120 (visited on 08/01/2023).

[Das+15] Atish Das Sarma et al. "Fast distributed PageRank computation". en. In: *Theoretical Computer Science* 561 (Jan. 2015), pp. 113–121. ISSN: 03043975. DOI: 10.1016/j.tcs.2014.04.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304397514002709 (visited on 01/05/2023).

[Dau11] Thomas Daun. "On the randomized solution of initial value problems". en. In: *Journal of Complexity* 27.3-4 (June 2011), pp. 300–311. ISSN: 0885064X. DOI: 10.1016/j.jco.2010.07.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0885064X1000066X (visited on 03/06/2023).

[ES21] S. M. Ermakov and M. G. Smilovitskiy. "The Monte Carlo Method for Solving Large Systems of Linear Ordinary Differential Equations". en. In: *Vestnik St. Petersburg University, Mathematics* 54.1 (Jan. 2021), pp. 28–38. ISSN: 1063-4541, 1934-7855. DOI: 10.1134/S1063454121010064. URL: https://link.springer.com/10.1134/S1063454121010064 (visited on 01/20/2023).

[GAM23] Nicolas L. Guidotti, Juan A. Acebrón, and José Monteiro. *A Fast Monte Carlo algorithm for evaluating matrix functions with application in complex networks*. en. arXiv:2308.01037 [cs]. Aug. 2023. URL: http://arxiv.org/abs/2308.01037 (visited on 12/17/2023).

[GH21] Abhishek Gupta and William B. Haskell. *Convergence of Recursive Stochastic Algorithms using Wasserstein Divergence*. en. arXiv:2003.11403 [cs, eess, math, stat]. Jan. 2021. URL: http://arxiv.org/abs/2003.11403 (visited on 01/11/2023).

[Hei98] S. Heinrich. "Monte Carlo Complexity of Global Solution of Integral Equations". In: *Journal of Complexity* 14.2 (1998), pp. 151–175. ISSN: 0885-064X. DOI: https://doi.org/10.1006/jcom.1998.0471. URL: https://www.sciencedirect.com/science/article/pii/S0885064X9890471X.

[HGM01] Chi-Ok Hwang, James A. Given, and Michael Mascagni. "The Simulation–Tabulation Method for Classical Diffusion Monte Carlo". en. In: *Journal of Computational Physics* 174.2 (Dec. 2001), pp. 925–946. ISSN: 00219991. DOI: 10.1006/jcph.2001.6947. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999101969475 (visited on 12/18/2022).

[HM93] Stefan Heinrich and Peter Mathé. "The Monte Carlo complexity of Fredholm integral equations". In: *mathematics of computation* 60.201 (1993). ISBN: 0025-5718, pp. 257–278.

[HN01] S. Heinrich and E. Novak. *Optimal Summation and Integration by Deterministic, Randomized, and Quantum Algorithms*. en. arXiv:quant-ph/0105114. May 2001. URL: http://arxiv.org/abs/quant-ph/0105114 (visited on 03/19/2023).

[HT16] Samuel Herrmann and Etienne Tanré. "The first-passage time of the Brownian motion to a curved boundary: an algorithmic approach". en. In: *SIAM Journal on Scientific Computing* 38.1 (Jan. 2016). arXiv:1501.07060 [math], A196–A215. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/151006172. URL: http://arxiv.org/abs/1501.07060 (visited on 12/20/2022).

[JL12] Hao Ji and Yaohang Li. "Reusing Random Walks in Monte Carlo Methods for Linear Systems". en. In: *Procedia Computer Science* 9 (2012), pp. 383–392. ISSN: 18770509. DOI: 10.1016/j.procs.2012.04.041. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050912001627 (visited on 09/17/2022).

[Ket+21] Markus Kettunen et al. "An unbiased ray-marching transmittance estimator". en. In: *ACM Transactions on Graphics* 40.4 (Aug. 2021). arXiv:2102.10294 [cs], pp. 1–20. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3450626.3459937. URL: http://arxiv.org/abs/2102.10294 (visited on 06/18/2023).

[Mil+23] Bailey Miller et al. *Boundary Value Caching for Walk on Spheres*. en. arXiv:2302.11825 [cs]. Feb. 2023. URL: http://arxiv.org/abs/2302.11825 (visited on 05/02/2023).

[Qi+22] Yang Qi et al. "A bidirectional formulation for Walk on Spheres". en. In: *Computer Graphics Forum* 41.4 (July 2022), pp. 51–62. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.14586. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14586 (visited on 12/22/2022).

[Rat+22] Alexander Rath et al. "EARS: efficiency-aware russian roulette and splitting". en. In: *ACM Transactions on Graphics* 41.4 (July 2022), pp. 1–14. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3528223.3530168. URL: https://dl.acm.org/doi/10.1145/3528223.3530168 (visited on 02/10/2023).

[Saw+22] Rohan Sawhney et al. "Grid-free Monte Carlo for PDEs with spatially varying coefficients". en. In: *ACM Transactions on Graphics* 41.4 (July 2022), pp. 1–17. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3528223.3530134. URL: https://dl.acm.org/doi/10.1145/3528223.3530134 (visited on 09/17/2022).

[SC] Rohan Sawhney and Keenan Crane. "Monte Carlo Geometry Processing:A Grid-Free Approach to PDE-Based Methods on Volumetric Domains". en. In: 38.4 (), p. 18.

[SM09] K. Sabelfeld and N. Mozartova. "Sparsified Randomization Algorithms for large systems of linear equations and a new version of the Random Walk on Boundary method". en. In: *Monte Carlo Methods and Applications* 15.3 (Jan. 2009). ISSN: 0929-9629, 1569-3961. DOI: 10.1515/MCMA.2009.015. URL: https://www.degruyter.com/document/doi/10.1515/MCMA.2009.015/html (visited on 09/17/2022).

[Vea97] Eric Veach. "Robust Monte Carlo Methods for Light Transport Simulation. Ph.D. Dissertation. Stanford University." en. In: *Robust Monte Carlo Methods for Light Transport Simulation.* (1997).

[VSJ21] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. "Path replay backpropagation: differentiating light paths using constant memory and linear time". en. In: *ACM Transactions on Graphics* 40.4 (Aug. 2021), pp. 1–14. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3450626.3459804. URL: https://dl.acm.org/doi/10.1145/3450626.3459804 (visited on 02/17/2023).