Universiteit
Antwerpen

Masterproef scriptie

Unbiased Monte Carlo for Recursive Integrals

Auteur: *Isidoor Pinillo Esquivel*

Promotor: *Wim Vanroose*

Academiejaar 2022-2023

Contents

Abstract

We will write this at the end.

1 Introduction

1.1 Introductory Example

To get familiar with Monte Carlo for estimating recursive integrals we demonstrate it on following problem:

$$y' = y, y(0) = 1. \tag{1}$$

By integrating both sides of (1) following integral equation can be derived:

$$y(t) = 1 + \int_0^t y(s)ds. \tag{2}$$

Equation (2) is a recursive integral equation or to be more specific a linear Volterra integral equation of the second type. By naively using Monte Carlo on the recursive integral of equation (2) one derives following estimator:

$$Y(t) = 1 + ty(Ut).$$

where $U = \text{Uniform}(0, 1)$. If y is well behaved then $E[Y(t)] = y(t)$ but we can't calculate $Y(t)$ without accesses to $y(s), s < t$. Notice that we can replace y by a unbiased estimator of it without changing $E[Y(t)] = y(t)$ by the law of total expectance ($E[X] = E[E[X|Z]]$). By replacing y by Y itself we obtain a recursive expression for Y:

$$Y(t) = 1 + tY(Ut). \tag{3}$$

Equation (3) is a recursive random variable equation (RRVE). If you would implement equation (3) with recursion it will run indefinitely. A biased way of around this is by approximating $Y(t) \approx 1$ near $t = 0$. Later we discuss Russian roulette (2.1.1) which can be used as an unbiased stopping mechanism.

Python Code 1.1.1 (implementation of (3))

```python
from random import random as U
def Y(t, eps): return 1 + t*Y(U()*t, eps) if t > eps else 1
def y(t, eps, nsim):
    return sum(Y(t, eps) for _ in range(nsim))/nsim
print(f"y(1) approx {y(1,0.01,10**3)}")
# y(1) approx 2.710602603240193
```

An issue with (1.1.1) is that the variance increases rapidly when t increases. Which we later solve in the section on ODEs. Note that (1.1.1) keeps desirable properties from unbiased Monte Carlo methods such as: being embarrassingly parallel, robustness and having simple error estimates.

1.2 Contributions

We write this at the end.

1.3 Related Work

work on

- alternative methods for recursive integrals
- MC work on ODEs
- MC work on PDEs
- WoS

This is just to give a general overview we probably reference specific ideas when we first introduce them.

2 Background

2.1 Modifying Monte Carlo

Once we have a RRVE it is possible to modify it to have more desirable properties. Our favorite work that discusses these techniques is [Vea].

Definition 2.1.1 (Russian roulette)
Define Russian roulette on X with free parameters $Y_1, Y_2 : E[Y_1] = E[Y_2]$, $p \in [0,1]$ and U independent of Y_1, Y_2, X the following way:

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

The idea behind Russian roulette is replacing X with a cheaper approximation Y_2 sometimes.

Example 2.1.2
Say that we are interested in estimating $E[Z]$ with Z defined in the following way:

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

where $f : \mathbb{R} \to [0,1]$ expensive to compute. Estimating Z directly would require calling f each simulation. We can modify Z to

$$\tilde{Z} = U + B\left(\frac{1}{100}\right)\frac{f(U)}{10}.$$

where $B(\frac{1}{100}) \sim \text{Bernouli}(\frac{1}{100})$. Now $\tilde{Z}$ just requires calling f on average once every 100 simulations with the variance only increasing slightly compared to Z.

Maybe this wasn't the best example because you could also estimate the expectance of the 2 terms of Z separately.

Example 2.1.3 (Russian roulette on (3))

Russian roulette can fix the indefinite recursion issue of equation (3) by approximating Y near $t = 0$ with 1. Concretely we replace the t in front of the recursive term with $B(t) = \text{Bernouli(t)}$ when $t < 1$.

Python Code 2.1.4 (Russian roulette on (3))

```python
from random import random as U
def Y(t):
    if t>1: return 1 + t*Y(U()*t)
    return 1 + Y(U()*t) if U() < t else 1
def y(t, nsim):
    return sum(Y(t) for _ in range(nsim))/nsim
print(f"y(1) approx {y(1,10**3)}")
# y(1) approx 2.698
```

Definition 2.1.5 (splitting)

Splitting X means using multiple $X_j \sim X$ not independent per se to lower variance by averaging them:

$$\bar{X} = \frac{1}{N}\sum_{j=1}^{N} X_j.$$

Splitting the recursive term in a RRVE can lead to (additive) branching recursion. Extra care should be taken that all branches get terminated with probability 1. This can be achieved by termination strategies already discussed and later we discuss coupled recursion for alleviating additive branching recursion in RRVEs.

Example 2.1.6 (splitting on (3))

We can "split" the recursive term of (3) in 2:

$$Y(t) = 1 + \frac{t}{2}(Y_1(Ut) + Y_2(Ut)).$$

with $Y_1(t), Y_2(t)$ i.i.d. $Y(t)$.

Python Code 2.1.7 (splitting on (3))

```python
from random import random as U
def Y(t):
    u = U()
    if t > 1:
        return 1 + t*(Y(u*t)+Y(u*t))/2
    return 1 + (Y(u*t)+Y(u*t))/2 if U() < t else 1
def y(t, nsim):
    return sum(Y(t) for _ in range(nsim))/nsim
print(f"y(1) approx {y(1,10**3)}")
# y(1) approx 2.73747265625
```

Definition 2.1.8 (2-level MC)

2-level MC on X with parameters $\tilde{X}, Y : E[\tilde{X}] = E[Y]$:

$$X \to X - \tilde{X} + Y.$$

This gets discussed in [Gil13].

Definition 2.1.9 (control variates)
Control variate on $f(X)$ is

$$f(X) \to f(X) - \tilde{f}(X) + E[\tilde{f}(X)].$$

Control variates are a special case of 2-level MC. Usually $\tilde{f}$ is an approximation of f to reduce variance.

Example 2.1.10 (control variate on (3))
To make a control variate for (3) that reduces variance we use following approximation of $y(t) \approx 1 + t$:

$$Y(t) = 1 + t + \frac{t^2}{2} + t(Y(Ut) - 1 - Ut).$$

Notice that we can cancel the constant term of the control variate but that would affect the Russian roulette negatively.

Python Code 2.1.11

```python
from random import random as U
def Y(t):
    u = U()
    if t > 1: return 1+t**2/2 + t*(Y(u*t)-u*t)
    return 1 + t + t**2/2 + (Y(u*t)-1-u*t if U() < t else 0)
def y(t, nsim):
    return sum(Y(t) for _ in range(nsim))/nsim
print(f"y(1) approx {y(1,10**3)}")
# y(1) approx 2.734827303480301
```

2.2 Monte Carlo Trapezoidal Rule

The Monte Carlo trapezoidal rule has some interesting convergence behavior [Wu20]. We aren't yet comfortable with our understanding of the randomized trapezoidal rule.
comparing normal vs Monte Carlo trapezoidal rule and highlighting the "half variance phenomenon". + maybe integrating polynomials for intuition

2.3 Unbiased Non-Linearity

Example 2.3.1 ($y' = y^2$)
see python note book

Example 2.3.2 ($e^{E[X]}$)
see python note book

2.4 Recursion

Example 2.4.1 (coupled recursion)
example with $y' = y$ (I need to redo this example)

Example 2.4.2 (recursion in recursion)
maybe induction in induction proof example

Example 2.4.3 (tail recursion)
discuss problems with implementing recursion and solutions.
inverse problem example

2.5 Green Functions

green function stuff that we will be needing, we aren't sure in how much detail we're going to go.

Example 2.5.1 (numerical green functions)
There will be probably some green functions that we need that don't have an analytic expression yet.

3 1-Dimensional Recursive Integrals

3.1 Linear Recursive Integrals

We have algo in mind for this case based on coupled recursion on disjunct sets.

3.2 IVPs ODEs

An IVP example probably using DRRMC maybe compare it to parareal. Maybe also non-linear algo

3.3 BVPs ODEs

A BVP example using yet another algo that hopefully has the half variance phenomenon.

4 Higher Dimensional Recursive Integrals

4.1 Complicated Geometry

Example 4.1.1 (nasty 2D integral)
2D integral that is difficult because of its geometry

4.2 Recursive Brownian Motion

WoS like way to simulate Brownian motion which is related to the green function of the heat equation

Example 4.2.1 (recursive Brownian motion)
see period5

4.3 Heat Equation

a geometric robust way to solve the heat equation and maybe a higher order method to solve the heat equation

4.4 Wave Equation

probably won't get to it

References

[Gil13] Michael B. Giles. *Multilevel Monte Carlo methods*. en. arXiv:1304.5472 [math]. Apr. 2013. URL: `http://arxiv.org/abs/1304.5472` (visited on 12/04/2022).

[Vea] Eric Veach. "ROBUST MONTE CARLO METHODS FOR LIGHT TRANSPORT SIMULATION". en. In: ().

[Wu20] Yue Wu. *A randomised trapezoidal quadrature*. en. arXiv:2011.15086 [cs, math]. Dec. 2020. URL: `http://arxiv.org/abs/2011.15086` (visited on 03/06/2023).

5 Appendix

Derivation of the green functions and some expressions.