A sequential Monte Carlo algorithm for solving BSDEs

Emmanuel Gobet *
Laboratoire Jean Kuntzmann, Université de Grenoble and CNRS,
BP 53, 38041 Grenoble Cedex 9, FRANCE

Céline Labart †
INRIA Paris-Rocquencourt, MathFi Project,
Domaine de Voluceau, Rocquencourt,
B.P. 105, 78153 Le Chesnay Cedex, FRANCE

March 3, 2020

Abstract

We present and analyze a numerical algorithm to solve BSDEs based on Picard's iterations and on a sequential control variate method. Its convergence is geometric. Moreover, our algorithm provides a regular solution w.r.t. time and space.

Premia 22

1 Introduction

This note is devoted to the study of a numerical algorithm to solve backward stochastic differential equations (BSDEs hereafter) of the following type, where $X \in \mathbb{R}^d$, $Y \in \mathbb{R}$ and $Z \in \mathbb{R}^q$

$$\begin{cases} -dY_t = f(t, X_t, Y_t, Z_t)dt - Z_t dW_t, \quad Y_T = \Phi(X_T), \\ X_t = x + \int_0^t b(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s. \end{cases} \tag{1.1}$$

Several algorithms to solve BSDEs can be found in the literature. We refer to [9] for an algorithm which solves the associated PDE with a finite difference approximation and to [3], [8], [1] and [4] for algorithms based on the dynamic programming equation. Our algorithm approximates the solution (Y, Z) of (1.1), by combining Picard's iterations — we approximate (Y, Z) by the solutions of a sequence of linear BSDEs converging geometrically fast to (Y, Z) (see [6] for more details)— and an adaptive control variate method. This technique is used to approximate the solutions of linear PDEs, which can be written as expectations of

*Email: emmanuel.gobet@imag.fr
†Email: celine.labart@inria.fr. This work has been partly done while the second author was affiliated to Ecole Polytechnique, CMAP, 91128 Palaiseau, France.

functionals of Markov processes via Feynman-Kac's formula. A control variate, changing at each algorithm iteration, is used to reduce the variance of the simulations. The convergence is also geometric (see [5] for more details). We present the algorithm in the following Section. In Section 3, we state that the approximated solution provided by our algorithm converges geometrically fast to (Y, Z) in a given norm, up to a function approximation error.

2 Description of the algorithm

Before presenting our algorithm, we recall that solving the BSDE (1.1) is equivalent to solving the following semilinear PDE:

$$\begin{cases} \partial_t u(t,x) + \mathcal{L}u(t,x) + f(t,x,u(t,x),(\partial_x u\sigma)(t,x)) = 0, \\ u(T,x) = \Phi(x), \end{cases} \tag{2.1}$$

where $\mathcal{L}$ is defined by

$$\mathcal{L}_{(t,x)}u(t,x) = \frac{1}{2}\sum_{i,j}[\sigma\sigma^*]_{ij}(t,x)\partial^2_{x_i x_j}u(t,x) + \sum_i b_i(t,x)\partial_{x_i}u(t,x).$$

More precisely, we can write

$$(Y_t, Z_t) = (u(t,X_t), \partial_x u(t,X_t)\sigma(t,X_t)), \text{ for all } t \in [0,T].$$

Our algorithm provides an approximated solution of PDE (2.1). Then, by simulating the diffusion X through an Euler scheme, we deduce from the previous equality an approximation of the solution of BSDE (1.1). More precisely, let u_k (resp. (Y^k, Z^k)) denote the approximation of u (resp. (Y, Z)) at step k, and let X^N denote the approximation of X obtained with an N time step Euler scheme. We write

$$(Y^k_t, Z^k_t) = (u_k(t,X^N_t), \partial_x u_k(t,X^N_t)\sigma(t,X^N_t)), \text{ for all } t \in [0,T]. \tag{2.2}$$

The construction of u_k is described below.

Initialization We begin with $u_0 \equiv 0$.

Iteration k, Step 1 Assume that an approximated solution u_{k-1} of class $C^{1,2}$ is available at stage $k-1$, and that we are able to compute $\partial_x u_{k-1}, \partial^2_x u_{k-1}, \partial_t u_{k-1}$ pointwise in $[0,T]\times R^d$. Following the idea of adaptive control variates, we write $u(t,x)$ as $(u-u_{k-1})(t,x) + u_{k-1}(t,x)$. Combining Ito's formula applied to $u(s,X_s)$ and to $u_k(s,X^N_s)$ between t and T and the semilinear PDE (2.1) satisfied by u, we get that the correction term $c_k := u - u_{k-1}$ is $c_k(t,x) = \mathbb{E}[\Phi(X^{t,x}_T) - u_{k-1}(T, X^{N,t,x}_T) + \int_t^T f\left(s, X^{t,x}_s, u(s,X^{t,x}_s), (\partial_x u\sigma)(s,X^{t,x}_s)\right) + (\partial_t + \mathcal{L}^N)u_{k-1}(s,X^{N,t,x}_s)ds|\mathcal{G}_{k-1}]$.

Remark 1. *As we will see a few lines below, u_{k-1} depends on several random variables. $\mathcal{G}_{k-1}$ is the filtration generated by the set of all random variables used to build u_{k-1}. In the above equation, we compute the expectation w.r.t. the law of X and X^N and not to the law of u_{k-1}, which is $\mathcal{G}_{k-1}$ measurable.*

The correction term c_k cannot be used directly: we have to replace u and $\partial_x u$ (unknown terms) appearing in f by u_{k-1} and $\partial_x u_{k-1}$, as suggested by the Picard contraction principle. We should also replace the expectation by a Monte Carlo summation. Let $\bar{c}_k(t,x)$ denotes the following approximation of c_k

$$\frac{1}{M}\sum_{m=1}^{M}[(\Phi - u_{k-1})(T, X_T^{m,N}) + \int_t^T f\left(s, X_s^{m,N}, u_{k-1}(s, X_s^{m,N}), (\partial_x u_{k-1}\sigma)(s, X_s^{m,N})\right) + (\partial_t + \mathcal{L}^N)u_{k-1}(s, X_s^{m,N})ds].$$

Iteration k, Step 2 Since the function $\bar{c}_k$ approximates the error $u - u_{k-1}$, we use it to compute u_k. We compute the value of $\bar{c}_k + u_{k-1}$ on a grid of n random points (t_i, x_i) of $[0,T]\times[-a,a]^d$, and regularize it through a regular operator $\mathcal{P}$:

$$u_k(t,x) = \mathcal{P}(u_{k-1} + \bar{c}_k)(t,x).$$

The operator $\mathcal{P}$ is built with kernel functions. It is regular ($C^{1,2}$) and we can easily differentiate it to get the derivatives of u_k. For more details on $\mathcal{P}$ we refer to [7, Section 11.3].

Thanks to the regularity of the operator $\mathcal{P}$, we have built an approximated solution u_k regular w.r.t. time and space. This is an advantage compared to the algorithms using the dynamic programming equation, which only provide regular solutions w.r.t. the space variable but not w.r.t. the time variable.

3 Convergence

We prove the convergence of our algorithm in the following norm, where V is a predictable process $V : \Omega \times [0,T] \times \mathbb{R}^d$.

$$\|V\|_{\mu,\beta}^2 := \mathbb{E}[\int_0^T \int_{\mathbb{R}^d} e^{\beta s}|V_s(x)|^2 e^{-\mu|x|}dxds].$$

This choice of norm is not harmless. For proving the convergence of the algorithm we combine results on BSDEs stated in a norm leading to the integration w.r.t. $e^{\beta s}ds$ (see [6]), and results on the bounds for solutions of linear PDEs in weighted Sobolev spaces (leading to the integration w.r.t. $e^{-\mu|x|}dx$), coming from [2]. The following Theorem states the convergence of $Y^k - Y$ and $Z^k - Z$ in the $\|\cdot\|_{\mu,\beta}^2$ norm. See [7, Chapter 13] for a proof of it.

Theorem 2. *Assume that σ is uniformly elliptic, $b \in C_b^{1,2}$ (bounded with bounded derivatives) and $\sigma \in C_b^{1,3}$, $\partial_t\sigma \in C_b^1$ in space. We also assume that f is a bounded Lipschitz function (with Lipschitz parameter L_f) and $\Phi \in C_b^{2+\alpha}$. Then, there exists a constant $K(T)$ such that*

$$\|Y - Y^k\|_{\mu,\beta}^2 + \|Z - Z^k\|_{\mu,\beta}^2 \leq S_k + O(\frac{1}{N}), \text{ where } S_k \leq \eta_1 S_{k-1} + \epsilon_1 \text{ and}$$

$$\eta_1 = \frac{4(1+T)L_f^2}{\beta} + K(T)\left(h_x^2 + h_t^2 + \frac{T(2a)^d}{nh_t h_x^d}(1 + M^{-1}h_x^{-2})\right),$$

$$\epsilon_1 = \frac{K(T)}{N^2 h_x^2} + K(T)\left(h_x^2 + \frac{T(2a)^d}{nh_t h_x^d}h_x^{-2} + e^{-\mu a}a^{d-1}h_x^{-1} + h_x^{-1}e^{-\frac{\mu}{\sqrt{d}}a}\right),$$

where h_t (resp. h_x) denotes the bandwidth associated to the time (resp. space) kernel function. Moreover, if $(\beta, h_x, h_t, n, M, N, a)$ are chosen such that η_1 is smaller than 1, it holds

$$\limsup_k \left(\|Y^k - Y\|^2_{\mu,\beta} + \|Z^k - Z\|^2_{\mu,\beta} \right) \leq \frac{K(T)}{1-\eta_1}\epsilon_1 + O(\frac{1}{N}).$$

Remark 3. *Looking at η_1 and ϵ_1, we can distinguish five different error terms : the discretisation error $\frac{1}{N^2 h_x^2}$ due to the Euler scheme, the contraction term $\frac{4(1+T)L_f^2}{\beta}$ corresponding to the geometric convergence of Picard's iterations, the Monte Carlo error $\frac{1}{M}$, the boundary discontinuity error $e^{-\mu a} a^{d-1} h_x^{-1} + h_x^{-1} e^{-\frac{\mu}{\sqrt{d}}a}$ and the error due to the estimator $\mathcal{P}$, i.e. $h_x^2 + h_t^2 + \frac{T(2a)^d}{n h_t h_x^{d+2}}$, which corresponds to the squared bias and variance ensuing from the error $\|u - \mathcal{P}u\|^2_{L^2} + \|\partial_x u - \partial_x(\mathcal{P}u)\|^2_{L^2}$.*

Remark 4. *Using a sequential Monte Carlo method leads to an error of $\frac{1}{M}$ only appearing in the contraction term η_1. Had we implemented a non adaptive method (i.e. using only Picard's iterations), M^{-1} would have appeared in ϵ_1. This would have led us to choose a much larger M, while practically $M = 10$ does the trick.*

References

[1] V. Bally, and G. Pagès, Stochastic Proc. and their Appl., **106(1)**, 1-40 (2003). 1

[2] A. Bensoussan, and J.L. Lions, Application of variational inequalities in stochastic control, Dunod, (1978). 3

[3] B. Bouchard, and N. Touzi, Stochastic Proc. and their Appl.,**111**, 175-206 (2004). 1

[4] F. Delarue and S. Menozzi, Annals of Applied Probab., **16(1)**, 140-184 (2006). 1

[5] E. Gobet, and S. Maire, SIAM Journal on Num. Anal., **43(3)**, 1256-1275 (2005). 2

[6] N. El Karoui, S. Peng, and M.C. Quenez, Mathematical Finance, **7(1)**, 1-71 (1997). 1, 3

[7] C. Labart. Phd thesis (2007). 3

[8] E. Gobet, J.P. Lemor, and X. Warin, Annals of Applied Probab., **15(3)**, 2172-2202 (2005). 1

[9] J. Ma, P. Protter, and J. Yong, Probab. Theory. Rel. Fields, **98(1)**, 339-359 (1994). 1