On the wavelet-based SWIFT method for backward stochastic differential equations

Chau, Ki Wai; Oosterlee, Cornelis W.

DOI
10.1093/imanum/drx022

Publication date
2018

Document Version
Accepted author manuscript

Published in
IMA Journal of Numerical Analysis

On the wavelets-based SWIFT method for backward stochastic differential equations

Ki Wai Chau and Cornelis W. Oosterlee

October 12, 2018

Abstract

We propose a numerical algorithm for backward stochastic differential equations based on time discretization and trigonometric wavelets. This method combines the effectiveness of Fourier-based methods and the simplicity of a wavelet-based formula, resulting in an algorithm that is both accurate and easy to implement. Furthermore, we mitigate the problem of errors near the computation boundaries by means of an antireflective boundary technique, giving an improved approximation. We test our algorithm with different numerical experiments.

1 Introduction

Ever since the general notion of backward stochastic differential equations (BSDEs) was introduced in [15], it has been a popular research subject. Especially in mathematical finance and insurance, BSDEs are powerful tools for the valuation of a contingent claim, both under usual complete market setting or with the incorporation of market imperfections and collateral requirements.

Finding an analytic solution for such equations is often difficult or even impossible, either by directly solving or by transforming the problems into partial differential equations (see [16]). Therefore, numerical methods are in great demand. While the majority of so-called probabilistic methods for solving BSDEs relies on time discretization of the stochastic process, they differ by the methods for calculating the appearing conditional expectations. Techniques used include least-squares Monte Carlo regression in [12], chaos decomposition formulas in [5], cubature methods in [6], among others. In particular, we are interested in methods based on Fourier series. These methods link the computation of the expectations to the characteristic function of the transitional probability density function, which is either given or easy to approximate. One particular method is the BCOS method proposed in [18], which was derived from the COS method for option pricing in [8].

There have been new developments in option pricing methods based on Fourier series. The Shannon Wavelets Inverse Fourier Technique (SWIFT method) was proposed in [14] for pricing European options and a so-called *quick SWIFT variant* was developed in [13] for pricing American and barrier options. The quick SWIFT method, while also based on Shannon wavelets, has the additional benefit of simplifying the algorithm and the error formula. Moreover, it is much easier to adjust individual approximation values because wavelets form a localized basis. We propose a new approach to solving BSDEs by combining a general θ-method for time-integration, as used in [11] and [18], with the SWIFT method. We also improve on previous work on SWIFT by providing an alternative derivation that takes into account the computational range.

In Section 2, the class of BSDEs under our consideration along with some notations and standing assumptions will be introduced. Section 3 contains the derivation of the SWIFT formula and our numerical algorithm for the BSDEs, while Section 4 is related to the error and computational complexity of our algorithm. We further improve our algorithm along the computation boundary in Section 5. Various numerical experiments are performed in Section 6 and concluding remarks are given in Section 7.

2 Backward stochastic differential equations

2.1 Setting

Given a filtered complete probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ with $\mathbb{F} := (\mathcal{F}_t)_{0\le t\le T}$ a filtration satisfying the usual conditions for a fixed terminal time $T > 0$. The process $\omega := (\omega_t)_{0\le t\le T}$ is a Brownian motion adapted to the filtration $\mathbb{F}$ and we are interested in solving the following one-dimension decoupled forward-backward stochastic differential equations, known as FBSDEs, numerically.

$$\begin{cases} dX_t = \mu(t, X_t)dt + \sigma(t, X_t)d\omega_t; \\ dY_t = -f(t, X_t, Y_t, Z_t)dt + Z_t d\omega_t, \end{cases} \tag{2.1}$$

where $0 \le t \le T$. The functions $\mu : \Omega \times [0, T] \times \mathbb{R} \to \mathbb{R}$ and $\sigma : \Omega \times [0, T] \times \mathbb{R} \to \mathbb{R}$ refer to the drift and the diffusion coefficients of the forward stochastic process, X, and $x_0 \in \mathcal{F}_0$ is the initial condition for X. The function $f : \Omega \times [0, T] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ is called the driver function of the backward process and the terminal condition Y_T is given by $g(X_T)$ for a function $g : \Omega \times \mathbb{R} \to \mathbb{R}$. All stochastic integrals with ω are of the Itô type.

It is assumed that both $\mu(t, x)$ and $\sigma(t, x)$ are measurable functions that are uniformly Lipschitz in x and satisfy a linear growth condition in x. Therefore, there exists a unique strong solution for the forward stochastic differential equation,

$$X_t = x_0 + \int_0^t \mu(\tau, X_\tau)d\tau + \int_0^t \sigma(\tau, X_\tau)d\omega_\tau.$$

This process also satisfies the Markov property, namely $\mathbb{E}[X_\tau|\mathcal{F}_t] = \mathbb{E}[X_\tau|X_t]$ for $\tau \ge t$, where $\mathbb{E}[\cdot]$ denotes expectation with respect to probability measure $\mathbb{P}$.

A pair of adapted processes (Y, Z) is said to be the solution of the FBSDE if Y is a continuous real-value adapted process, Z is a real-value predictable process such that $\int_0^T |Z_t|^2 dt < \infty$ almost surely in $\mathbb{P}$ and the pair satisfies Equation (2.1). We wish to find (Y_0, Z_0) by solving the problem backwards in time. We do this by first discretizing Equation (2.1) along the time direction $\Lambda : 0 = t_0 < t_1 < t_2 < \ldots < t_P = T$. In this article, we assume that we have a fixed uniform time-step $\Delta t = t_{p+1} - t_p$, $\forall p$ and define $\Delta\omega_{p+1} := \omega_{t_{p+1}} - \omega_{t_p} \sim \mathcal{N}(0, \Delta t)$, a normally distributed process. The discretized forward process X^Δ is defined by

$$X^\Delta_{t_0} := x_0,\ X^\Delta_{t_{p+1}} := X^\Delta_{t_p} + \mu(t_p, X^\Delta_{t_p})\Delta t + \sigma(t_p, X^\Delta_{t_p})\Delta\omega_{p+1}, \quad p = 0, \ldots, P-1,$$

which is derived from the classical Euler discretization. Note that we only defined the discretized process at the discrete time points here. While it is possible to extend the definition to $[0, T]$, it is not necessary for our presentation.

Adopting the notation $\mathbf{X} = (X, Y, Z)$, we can observe from the backward equation,

$$Y_{t_p} = Y_{t_{p+1}} + \int_{t_p}^{t_{p+1}} f(\tau, \mathbf{X}_\tau) d\tau - \int_{t_p}^{t_{p+1}} Z_\tau d\omega_\tau, \tag{2.2}$$

that a simple discretization is not sufficient to produce an approximation. It is because we would require the value of $Y_{t_{p+1}}$ to approximate Y_{t_p}, but $Y_{t_{p+1}}$ is not $\mathcal{F}_{t_p}$ adapted. To tackle this problem, we follow the standard methods in literature, for example, in [4]. By taking conditional expectations on both sides of Equation (2.2) and approximating the time integral by a θ-time discretization, as in [18], we get

$$\begin{aligned} Y_{t_p} &= \mathbb{E}_p[Y_{t_{p+1}}] + \int_{t_p}^{t_{p+1}} \mathbb{E}_p[f(\tau, \mathbf{X}_\tau)] d\tau \\ &\approx \mathbb{E}_p[Y_{t_{p+1}}] + \Delta t \theta_1 f(t_p, \mathbf{X}_{t_p}) + \Delta t (1-\theta_1) \mathbb{E}_p[f(t_{p+1}, \mathbf{X}_{t_{p+1}})],\ \theta_1 \in [0,1]. \end{aligned}$$

The notation $\mathbb{E}_p$ and $\mathbb{E}_p^x$ are defined as

$$\mathbb{E}_p[\cdot] := \mathbb{E}[\cdot | X_{t_p}] = \mathbb{E}[\cdot | \mathcal{F}_{t_p}],\ \text{and}\ \mathbb{E}_p^x[\cdot] = \mathbb{E}[\cdot | X_{t_p} = x].$$

For the process Z, we derive a recursive approximation formula by multiplying $\Delta\omega_{p+1}$ to both sides of Equation (2.2) and taking conditional expectations,

$$\begin{aligned} 0 =& \mathbb{E}_p[Y_{t_{p+1}} \Delta\omega_{p+1}] + \int_{t_p}^{t_{p+1}} \mathbb{E}_p[f(\tau, \mathbf{X}_\tau) \Delta\omega_{p+1}] d\tau - \int_{t_p}^{t_{p+1}} \mathbb{E}_p[Z_\tau] d\tau \\ \approx& \mathbb{E}_p[Y_{t_{p+1}} \Delta\omega_{p+1}] + \Delta t (1-\theta_2) \mathbb{E}_p[f(t_{p+1}, \mathbf{X}_{t_{p+1}}) \Delta\omega_{p+1}] \\ & - \Delta t \theta_2 Z_{t_p} - \Delta t (1-\theta_2) \mathbb{E}_p[Z_{t_{p+1}}],\ \theta_2 \in (0,1]. \end{aligned}$$

Again, we applied the θ-method to the time integral. However, the two parameters for the θ-method, θ_1 and θ_2, need not necessarily be the same. We define a discrete-time approximation (Y^Δ, Z^Δ) for (Y, Z):

$$Y_{t_P}^\Delta := g(X_{t_P}^\Delta), \quad Z_{t_P}^\Delta = \sigma(t_P, X_{t_P}^\Delta) D_x g(X_{t_P}^\Delta), \tag{2.3a}$$

for $p = P-1, \ldots, 0$,

$$Z_{t_p}^\Delta := -\frac{1-\theta_2}{\theta_2} \mathbb{E}_p[Z_{t_{p+1}}^\Delta] + \frac{1}{\theta_2 \Delta t} \mathbb{E}_p[Y_{t_{p+1}}^\Delta \Delta\omega_{p+1}] + \frac{1-\theta_2}{\theta_2} \mathbb{E}_p[f(t_{p+1}, \mathbf{X}_{t_{p+1}}^\Delta) \Delta\omega_{p+1}], \tag{2.3b}$$

$$Y_{t_p}^\Delta := \mathbb{E}_p[Y_{t_{p+1}}^\Delta] + \Delta t \theta_1 f(t_p, \mathbf{X}_{t_p}^\Delta) + \Delta t (1-\theta_1) \mathbb{E}_p[f(t_{p+1}, \mathbf{X}_{t_{p+1}}^\Delta)], \tag{2.3c}$$

Again, we used the simplifying notation $\mathbf{X}^\Delta = (X^\Delta, Y^\Delta, Z^\Delta)$. Note that various combinations of θ_1 and θ_2 give different approximation schemes. We have an explicit scheme for Y^Δ if $\theta_1 = 0$, and an implicit scheme otherwise. The variable $Z_{t_p}^\Delta$ depends on $\mathbb{E}_p[Z_{t_{p+1}}^\Delta]$ only if $\theta_2 \neq 1$. Also, since the terminal processes $Y_{t_P}^\Delta$ and $Z_{t_P}^\Delta$ are deterministic with respect to $X_{t_P}^\Delta$ and X^Δ is a Markov process, one can show by induction that $Y_{t_p}^\Delta = y_p^\Delta(X_{t_p}^\Delta)$, $Z_{t_p}^\Delta = z_p^\Delta(X_{t_p}^\Delta)$, where z_p^Δ and y_p^Δ are deterministic functions related to the discretization scheme. We shall use the notation $(y_p^\Delta(x), z_p^\Delta(x))$ when we wish to emphasize the dependence of our approximation.

When solving the approximation in Equation (2.3), one needs to calculate multiple conditional expectations at each time-step. In this article, our method of choice is a wavelet-based method, introduced in [14].

2.2 Standing assumptions

Throughout the paper, in addition to the conditions for μ and σ, we assume the following to be true:

(A1) The function $f(t,x,y,z)$ is continuous with respect to (x,y,z) and all one-sided derivatives exist.

(A2) The function $g(x)$ is continuous in x and all left- and right-side derivatives exist.

When dealing with the discretization scheme with $\theta_1 \neq 1$, we add one more assumption:

(A3) The function f is Lipschitz in (y,z), namely,

$$|f(t,x,y_1,z_1) - f(t,x,y_2,z_2)| \leq M(|y_1 - y_2| + |z_1 - z_2|);\ x, y_1, y_2, z_1, z_2 \in \mathbb{R}, t \in [0,T],$$

for some constant M.

Under assumptions (A1)-(A3) ((A1)-(A2) if $\theta_1 = 1$), the numerical algorithm for the FBSDE, which will be given in Section 3, is well-defined. Although, $D_x g$ in Equation (2.3a) may be undefined at countable many distinctive points, it can just be replaced by a one-sided derivative at these points. The conditions above can also ensure satisfactory performance of our algorithms in general, with more details coming in Section 4. However, the above conditions are not sufficient to assure the existence of the pair of adapted processes (Y,Z), which is the foundation of any numerical algorithm. We introduce an extra assumption to ensure the existence and uniqueness of the solution (Y,Z) to Equation (2.1).

(A4) There exists a constant M such that

$$|f(t,x,y,z)| + |g(x)| \leq M(1 + |x|^\nu + |y| + |z|),\ \forall x,y,z \in \mathbb{R}, t \in [0,T], \nu \geq \frac{1}{2}.$$

For further results on the existence and uniqueness of the solution of BSDEs, readers are referred to [16] and further research extending this result. The last point we would like to raise is that the convergent rate of the discretized process to the original process also depends on the functions μ, σ, f and g. We shall discuss these requirements in Section 4.1; these conditions are not included in the standing assumptions.

3 SWIFT method

For the computation of the expectations appearing in the discrete FBSDEs (2.3), we will use the wavelet-based SWIFT method. In this section, we first provide an alternative derivation for the SWIFT formula used in [14] and [13]. Instead of using an approximation space based on Shannon wavelets on the whole real line, we construct a Shannon wavelet scaling function on a *finite domain* and derive our formula with this scaling function. This approach is beneficial since a truncation of the integration range is required when calculating the wavelet coefficients for the SWIFT formula. Incorporating the truncated range in the scaling function simplifies the formula for the approximation error. Next, we apply the SWIFT method to compute the conditional expectations in the discrete-time approximation of the FBSDE in Equation (2.3) and produce an

algorithm for solving FBSDEs recursively, backwards in time. In Sections 3.1 and 3.2, we derive the SWIFT formula with the finite range approach and compute the relevant expectations for the FBSDE algorithm. Section 3.3 and 3.4 discuss the approximations of the functions $z_p^{\Delta}(x)$ and $y_p^{\Delta}(x)$.

3.1 Scaling Functions

We begin our discussion with some preliminary definitions and results. For any fixed real number m and integer $J \neq 0$, we define an inner product and a norm:

$$< v, w >:= \frac{2^m}{J} \int_{-2^{-m}J}^{2^{-m}J} v(x)w(x)dx, \quad ||v||_2 := \sqrt{< v, v >}.$$

A function v is said to be in the $L^2((-2^{-m}J, 2^{-m}J])$ space if $||v||_2$ is a finite number. It can be shown that the set

$$\Gamma_{m,J} := \left\{ \cos\left(\left(\frac{2n-1}{2J}\pi\right)2^m x\right), \sin\left(\left(\frac{2n-1}{2J}\pi\right)2^m x\right) \middle| n = 1, 2, \dots \right\},$$

is orthonormal with respect to this inner product and is dense in $L^2((-2^{-m}J, 2^{-m}J])$.

Equipped with the above definitions, we construct an approximation space together with a localized basis, which are the foundations of the truncated SWIFT approximation method. Consider $2J$ distinctive functions $\varphi_{J,r} : \mathbb{R} \to \mathbb{R}$,

$$\begin{aligned}
\varphi_{J,r}(x) :=& \sum_{k=1}^{J} \left(\cos\left(\left(\frac{2k-1}{2J}\pi\right)2^m x\right) \cos\left(\left(\frac{2k-1}{2J}\pi\right)2^m \left(\frac{r}{2^m}\right)\right) \right. \\
&\left. + \sin\left(\left(\frac{2k-1}{2J}\pi\right)2^m x\right) \sin\left(\left(\frac{2k-1}{2J}\pi\right)2^m \left(\frac{r}{2^m}\right)\right)\right) \\
=& \sum_{k=1}^{J} \cos\left(\frac{2k-1}{2J}\pi\left(2^m x - r\right)\right) \qquad (3.1) \\
=& \begin{cases} J & \text{if } x = \frac{2J}{2^m}l + \frac{r}{2^m} \text{ for } l \text{ an even integer,} \\ -J & \text{if } x = \frac{2J}{2^m}l + \frac{r}{2^m} \text{ for } l \text{ an odd integer,} \\ \frac{\sin(\pi(2^m x - r))}{2\sin\left(\frac{\pi}{2J}(2^m x - r)\right)} & \text{otherwise,} \end{cases}
\end{aligned}$$

where $r = 1 - J, 2 - J, \dots, J$. This definition is a special case of the scaling functions given in Equation (2.13) of [9], in which the authors presented a uniform approach for the construction of wavelets based on orthogonal polynomials. The properties of $\varphi_{J,r}$ have been extensively studied in [9] and those that are relevant to our numerical method are listed in the next theorem along with their proof.

Theorem 3.1. *The scaling function $\varphi_{J,r}$, which is defined in Equation* (3.1)*, satisfies the following properties:*

(a) The inner product between two scaling functions is given by the following equation:

$$< \varphi_{J,r}, \varphi_{J,s} >= \varphi_{J,r}\left(\frac{s}{2^m}\right), \qquad r, s = 1 - J, 2 - J, \dots, J.$$

Thus, $\{\varphi_{J,r} | r = 1 - J, 2 - J, \dots, J\}$ form an orthogonal set.

(b) The scaling function $\varphi_{J,r}$ is localized around $\frac{r}{2^m}$. By this we mean that for the subspace

$$V_J := span\left\{\cos\left(\frac{(2k-1)\pi}{2J}2^m x\right), \sin\left(\frac{(2k-1)\pi}{2J}2^m x\right)\middle| k=1,2,\ldots,J\right\},$$

we have

$$\left|\left|\frac{\varphi_{J,r}}{\varphi_{J,r}(2^{-m}r)}\right|\right|_2 = \min\left\{||\chi||_2 : \chi \in V_J, \chi\left(\frac{r}{2^m}\right)=1\right\}.$$

(c) $\{\varphi_{J,r}|r=1-J,2-J,\ldots,J\}$ is a basis for V_J.

(d) The scaling function $\varphi_{J,r}$ is also a kernel polynomial in the sense that for any function v in V_j, we have

$$< v, \varphi_{J,r} >= v\left(\frac{r}{2^m}\right).$$

Proof. We can demonstrate (a) by direct computation and applying the orthonormality of the set $\Gamma_{m,J}$, such that

$$\begin{aligned}
< \varphi_{J,r}, \varphi_{J,s} >=& \sum_{k=1}^{n}\left(\cos\left(\left(\frac{2k-1}{2J}\pi\right)2^m\left(\frac{r}{2^m}\right)\right)\cos\left(\left(\frac{2k-1}{2J}\pi\right)2^m\left(\frac{s}{2^m}\right)\right)\right.\\
&\left.+\sin\left(\left(\frac{2k-1}{2J}\pi\right)2^m\left(\frac{r}{2^m}\right)\right)\sin\left(\left(\frac{2k-1}{2J}\pi\right)2^m\left(\frac{s}{2^m}\right)\right)\right)\\
=&\varphi_{J,r}\left(\frac{s}{2^m}\right)\\
=&\begin{cases} J, & \text{if } s=r,\\ \frac{\sin((s-r)\pi)}{2\sin\left(\frac{(s-r)\pi}{2J}\right)}=0, & \text{otherwise.}\end{cases}
\end{aligned}$$

Next, let

$$\chi(x)=\sum_{k=1}^{J}\left(c_k\cos\left(\frac{(2k-1)\pi}{2J}2^m x\right)+d_k\sin\left(\frac{(2k-1)\pi}{2J}2^m x\right)\right),$$

and $\chi\left(\frac{r}{2^m}\right)=1$ for some constants c_k and d_k. By a simple application of the Cauchy-Schwarz inequality, we get

$$\begin{aligned}
1=\chi^2\left(\frac{r}{2^m}\right)=&\left(\sum_{k=1}^{J}\left(c_k\cos\left(\frac{(2k-1)\pi}{2J}r\right)+d_k\sin\left(\frac{(2k-1)\pi}{2J}r\right)\right)\right)^2\\
\le&\left(\sum_{k=1}^{J}(c_k^2+d_k^2)\right)\left(\sum_{k=1}^{J}\left(\left(\cos\left(\frac{(2k-1)\pi}{2J}r\right)\right)^2+\left(\sin\left(\frac{(2k-1)\pi}{2J}r\right)\right)^2\right)\right)\\
=&J||\chi||_2^2.
\end{aligned}$$

The last equality follows from the orthonormality of set $\Gamma_{m,J}$ and since c_k and d_k are arbitrary, $||\chi||_2^2 \ge \frac{1}{J}$ for any $\chi \in V_J$, such that $\chi\left(\frac{r}{2^m}\right)=1$. On the other hand, as

$$\varphi_{J,r}\left(\frac{r}{2^m}\right)=\sum_{k=1}^{J}\left(\left(\cos\left(\frac{(2k-1)\pi}{2J}2^m\frac{r}{2^m}\right)\right)^2+\left(\sin\left(\frac{(2k-1)\pi}{2J}2^m\frac{r}{2^m}\right)\right)^2\right)=J,$$

We know that

$$\left|\left|\frac{\varphi_{J,r}}{\varphi_{J,r}(2^{-m}r)}\right|\right|_2^2 = \frac{<\varphi_{J,r},\varphi_{J,r}>}{J^2} = \frac{1}{J},$$

which concludes the proof of (b). Statement (c) is true since $\{\varphi_{J,r}|r = 1-J, 2-J, \ldots, J\}$ has the same number of elements as the spanning set of V_J; its elements are orthogonal and therefore independent of each other.

Part (d) follows from parts (a) and (c). For any $v \in V_J$, $v(\cdot) = \sum_{s=1-J}^{J} \mathcal{V}_s \varphi_{J,s}(\cdot)$ by (c), and from part (a), we have

$$< v, \varphi_{J,r} >= \sum_{s=1-J}^{J} \mathcal{V}_s < \varphi_{J,s}, \varphi_{J,r} >= \sum_{s=1-J}^{J} \mathcal{V}_s \varphi_{J,s}\left(\frac{r}{2^m}\right) = v\left(\frac{r}{2^m}\right).$$

□

The space V_J and the scaling functions $\{\varphi_{J,r}\}_{r=1-J,\ldots,J}$ are our approximation space and our basis functions, respectively. As a result, for any function v in the $L^2((-2^{-m}J, 2^{-m}J])$ space, its projection on V_J, denoted as $H_{V_J}v$, can be written in the form $\frac{1}{J}\sum_{r=1-J}^{J} < H_{V_J}v, \varphi_{J,r} > \varphi_{J,r} = \frac{1}{J}\sum_{r=1-J}^{J} < v, \varphi_{J,r} > \varphi_{J,r}$.

3.2 Quick SWIFT formula and coefficients

Assume we wish to approximate a finite integral $\int_{\mathbb{R}} v(\varsigma)q(\varsigma)d\varsigma$, where v is within $L^2((-2^{-m}J, 2^{-m}J])$ and we have $\int_{\mathbb{R}} q(\varsigma)d\varsigma < \infty$. We shall approach this problem by replacing v by $H_{V_J}v$. This gives us the following approximation:

$$\begin{aligned}
\int_{\mathbb{R}} v(\varsigma)q(\varsigma)d\varsigma &\approx \int_{\mathbb{R}} q(\varsigma)\frac{1}{J}\sum_{r=1-J}^{J} < v, \varphi_{J,r} > \varphi_{J,r}(\varsigma)d\varsigma \\
&= \int_{\mathbb{R}} q(\varsigma)\frac{1}{J}\sum_{r=1-J}^{J}\frac{2^m}{J}\int_{-2^{-m}J}^{2^{-m}J} v(\varrho)\sum_{k_1=1}^{J}\cos\left(C_{k_1}(2^m\varrho - r)\right)d\varrho \sum_{k_2=1}^{J}\cos\left(C_{k_2}(2^m\varsigma - r)\right)d\varsigma \\
&= \sum_{r=1-J}^{J}\int_{\mathbb{R}} q(\varsigma)\frac{2^{\frac{m}{2}}}{J}\sum_{k_2=1}^{J}\cos\left(C_{k_2}(2^m\varsigma - r)\right)d\varsigma \int_{-2^{-m}J}^{2^{-m}J} v(\varrho)\frac{2^{\frac{m}{2}}}{J}\sum_{k_!=1}^{J}\cos\left(C_{k_2}(2^m\varrho - r)\right)d\varrho,
\end{aligned}$$

which is the SWIFT formula, proposed in [14], with $C_k := \frac{2k-1}{2J}\pi$. In the above derivation, we only listed the dependency to dummy variable ς of the functions v and q. In practice, v and q will depend on other variables, like for example, time. We will put the additional dependency in our notation without further notice in the remainder of this article whenever it is necessary for the presentation.

Remark 3.2. While the accuracy of the approximation depends on other properties of the functions v and q and shall be studied in the rest of this article, $v \in L^2((-2^{-m}J, 2^{-m}J])$ and q being integrable are the only conditions needed for the above approximation to be well-defined.

Remark 3.3. We only define the approximation on the set $(-2^{-m}J, 2^{-m}J]$ that centers around zero. For any function v such that $\int_a^b (v(\varsigma))^2 d\varsigma < \infty$ for a finite range $(a, b]$, we need to perform a change of variables $\varsigma' = \varsigma - \frac{a+b}{2}$, for $\varsigma' \in (a - \frac{a+b}{2}, b - \frac{a+b}{2}]$, let $v'(\varsigma') = v(\varsigma' + \frac{a+b}{2}) = v(\varsigma)$. Then, we can

pick J and m accordingly and perform the approximation on v'. With a slight abuse of notation, we assume that we perform this tedious step implicitly and apply the SWIFT formula with any finite range $[a, b]$.

In this article, we shall adopt the quick variant of the SWIFT formula proposed in [13]. Instead of replacing v with $H_{V_J} v$, we approximate v by

$$v(x) \approx \frac{1}{J} \sum_{r=1-J}^{J} v\left(\frac{r}{2^m}\right) \varphi_{J,r}(x).$$

The reason behind this is left for the error section, but this gives an approximation for $\mathbb{E}_p^x[v(t_{p+1}, X_{t_{p+1}}^\Delta)]$, which we see in the discrete-time approximation of FBSDE,

$$\mathbb{E}_p^x[v(t_{p+1}, X_{t_{p+1}}^\Delta)] \approx \mathbb{E}_p^x\left[\frac{1}{J} \sum_{r=1-J}^{J} v\left(t_{p+1}, \frac{r}{2^m}\right) \varphi_{J,r}(X_{t_{p+1}}^\Delta)\right] = \frac{1}{J} \sum_{r=1-J}^{J} v\left(t_{p+1}, \frac{r}{2^m}\right) \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)].$$

The expectation $\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]$ can be computed by

$$\begin{aligned}
\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)] &= \mathbb{E}_p^x\left[\sum_{k=1}^{J} \cos\left(2^m C_k X_{t_{p+1}}^\Delta - C_k r\right)\right] \\
&= \Re\left\{\sum_{k=1}^{J} \mathbb{E}_p^x\left[\exp\left(\imath 2^m C_k (x + \mu(t_p, x)\Delta t + \sigma(t_p, x)\Delta\omega_{p+1})\right) \exp\left(-\imath C_k r\right)\right]\right\} \\
&= \Re\left\{\sum_{k=1}^{J} \exp\left(\imath 2^m C_k x\right) \Phi\left(t_p, x, 2^m C_k\right) \exp\left(-\imath C_k r\right)\right\}, \qquad (3.2)
\end{aligned}$$

where the real part of a complex number is denoted by $\Re\{\}$ and Φ is the characteristic function of $X_{t_{p+1}}^\Delta - X_{t_p}^\Delta$, i.e.

$$\Phi(\tau, \varsigma, \varrho) := \exp\left(\imath\varrho\mu(\tau, \varsigma)\Delta t - \frac{1}{2}\varrho^2\sigma^2(\tau, \varsigma)\Delta t\right).$$

The authors of [13] demonstrated how to calculate the vector $(\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)])_{r=(1-J,\ldots,J)}$ with a Fast Fourier Transform (FFT) algorithm. The computations induced by Equation (3.2) in our algorithm have the computation complexity of $\mathrm{O}(J \log(J))$.

Expectations in the form $\mathbb{E}_p^x[v(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]$ also occur in the discrete-time approximation of the FBSDE and they can be computed by:

$$\begin{aligned}
\mathbb{E}_p^x[v(t_{p+1}, X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}] \approx& \mathbb{E}_p^x\left[\frac{1}{J} \sum_{r=1-J}^{J} v\left(t_{p+1}, \frac{r}{2^m}\right) \varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}\right] \\
=& \frac{1}{J} \sum_{r=1-J}^{J} v\left(t_{p+1}, \frac{r}{2^m}\right) \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}],
\end{aligned}$$

with $\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]$ given by the formula:

$$\begin{aligned}
&\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}] = \sigma(t_p,x)\Delta t\mathbb{E}_p^x[D_x\varphi_{J,r}(X_{t_{p+1}}^\Delta)] \\
=&\Re\left\{\imath\sigma(t_p,x)\Delta t\sum_{k=1}^{J}2^mC_k\mathbb{E}_p^x\left[\exp\left(\imath 2^mC_k(x+\mu(t_p,x)\Delta t+\sigma(t_p,x)\Delta\omega_{p+1})\right)\exp\left(-\imath C_kr\right)\right]\right\} \\
=&\Re\left\{\imath\sigma(t_p,x)\Delta t\sum_{k=1}^{J}2^mC_k\exp\left(\imath 2^mC_kx\right)\Phi\left(t_p,x,2^mC_k\right)\exp\left(-\imath C_kr\right)\right\},
\end{aligned} \tag{3.3}$$

where the first equality sign follows from an integration by parts argument and we also note that $D_x\varphi_{J,r}(x) = -\sum_{k=1}^{J}2^mC_k\sin\left(2^mC_kx - C_kr\right)$. Once again, we can use the FFT to compute these expectations.

In the next two sections, we will combine the quick variant of the SWIFT formula and the discrete-time approximation of the FBSDE in Equation (2.3).

3.3 Quick SWIFT approximation of function $z_p^\Delta(x)$

There are three different expectations, $\mathbb{E}_p^x[Z_{t_{p+1}}^\Delta]$, $\mathbb{E}_p^x[Y_{p+1}^\Delta\Delta\omega_{t_{p+1}}]$, and $\mathbb{E}_p^x[f(t_{p+1},\mathbf{X}_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]$, that need to be approximated in Equation (2.3b). Applying the quick SWIFT formula, we have:

$$\mathbb{E}_p^x[Z_{t_{p+1}}^\Delta] \approx \frac{1}{J}\sum_{r=1-J}^{J}z_{p+1}^\Delta\left(\frac{r}{2^m}\right)\Re\left\{\sum_{k=1}^{J}e^{\imath 2^mC_kx}\Phi\left(t_p,x,2^mC_k\right)e^{-\imath C_kr}\right\};$$

$$\mathbb{E}_p^x[Y_{t_{p+1}}^\Delta\Delta\omega_{p+1}] \approx \frac{1}{J}\sum_{r=1-J}^{J}y_{p+1}^\Delta\left(\frac{r}{2^m}\right)\Re\left\{\imath\sigma(t_p,x)\Delta t\sum_{k=1}^{J}2^mC_ke^{\imath 2^mC_kx}\Phi\left(t_p,x,2^mC_k\right)e^{-\imath C_kr}\right\};$$

$$\begin{aligned}
\mathbb{E}_p^x[f(t_{p+1},\mathbf{X}_{t_{p+1}}^\Delta)\Delta\omega_{p+1}] \approx&\frac{1}{J}\sum_{r=1-J}^{J}f\left(t_{p+1},\frac{r}{2^m},y_{p+1}^\Delta\left(\frac{r}{2^m}\right),z_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right)\cdot \\
&\Re\left\{\imath\sigma(t_p,x)\Delta t\sum_{k=1}^{J}2^mC_ke^{\imath 2^mC_kx}\Phi\left(t_p,x,2^mC_k\right)e^{-\imath C_kr}\right\}.
\end{aligned}$$

We denote the approximation of the FBSDE by a SWIFT-type formula combining with the Euler discretization at point (t_p,x) by $(\hat{y}_p^\Delta(x),\hat{z}_p^\Delta(x))$, then $\hat{z}_p^\Delta$ satisfies the following recursive relation:

$$\begin{aligned}
\hat{z}_p^\Delta(x) =& \Re\left\{\frac{1}{J}\sum_{k=1}^{J}e^{\imath 2^mC_kx}\left[\Phi\left(t_p,x,2^mC_k\right)\sum_{r=1-J}^{J}\left(-\frac{1-\theta_2}{\theta_2}\hat{z}_{p+1}^\Delta\left(\frac{r}{2^m}\right)e^{-\imath C_kr}\right)\right]\right\} \\
&+\Re\left\{\frac{\imath}{J}\sigma(t_p,x)\sum_{k=1}^{J}e^{\imath 2^mC_kx}2^mC_k\Phi\left(t_p,x,2^mC_k\right)\cdot\right. \\
&\left.\left[\sum_{r=1-J}^{J}\left(\frac{1}{\theta_2}\hat{y}_{p+1}^\Delta\left(\frac{r}{2^m}\right)+\frac{(1-\theta_2)\Delta t}{\theta_2}f\left(t_{p+1},\frac{r}{2^m},\hat{y}_{p+1}^\Delta\left(\frac{r}{2^m}\right),\hat{z}_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right)e^{-\imath C_kr}\right)\right]\right\},
\end{aligned} \tag{3.4}$$

for $p = 0,1,\ldots,P-1$.

3.4 Quick SWIFT approximation of function $y_p^\Delta(x)$

Equation (2.3c) for $Y_{t_p}^\Delta$ contains an explicit and an implicit part if $\theta_1 > 0$. The explicit part is denoted by:

$$h(t_p, x) := \mathbb{E}_p^x[Y_{t_{p+1}}^\Delta] + \Delta t(1-\theta_1)\mathbb{E}_p^x[f(t_{p+1}, \mathbf{X}_{t_{p+1}}^\Delta)]. \tag{3.5}$$

The function h is a linear combination of two expectations, $\mathbb{E}_p^x[Y_{t_{p+1}}^\Delta]$ and $\mathbb{E}_p^x[f(t_{p+1}, \mathbf{X}_{t_{p+1}}^\Delta)]$, and they can be approximated by the following quick SWIFT formulas:

$$\mathbb{E}_p^x[Y_{t_{p+1}}^\Delta] \approx \frac{1}{J}\sum_{r=1-J}^{J} y_{p+1}^\Delta\left(\frac{r}{2^m}\right) \Re\left\{\sum_{k=1}^{J} e^{\imath 2^m C_k x}\Phi\left(t_p, x, 2^m C_k\right) e^{-\imath C_k r}\right\}; \tag{3.6a}$$

$$\mathbb{E}_p^x[f(t_{p+1}, \mathbf{X}_{t_{p+1}}^\Delta)] \approx \frac{1}{J}\sum_{r=1-J}^{J} f\left(t_{p+1}, \frac{r}{2^m}, y_{p+1}^\Delta\left(\frac{r}{2^m}\right), z_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right) \Re\left\{\sum_{k=1}^{J} e^{\imath 2^m C_k x}\Phi\left(t_p, x, 2^m C_k\right) e^{-\imath C_k r}\right\}. \tag{3.6b}$$

Therefore, we have an approximation for h:

$$\begin{aligned}\hat{h}(t_p, x) :=& \mathbb{E}_p^x[\hat{y}_{p+1}^\Delta(X_{t_{p+1}}^\Delta)] + \Delta t(1-\theta_1)\mathbb{E}_p^x[f(t_{p+1}, X_{t_{p+1}}^\Delta, \hat{y}_{p+1}^\Delta(X_{t_{p+1}}^\Delta), \hat{z}_{p+1}^\Delta(X_{t_{p+1}}^\Delta))] \\ =& \Re\left\{\frac{1}{J}\sum_{k=1}^{J} e^{\imath 2^m C_k x}\Phi\left(t_p, x, 2^m C_k\right)\cdot \right. \\ & \left. \sum_{r=1-J}^{J}\left[\hat{y}_{p+1}^\Delta\left(\frac{r}{2^m}\right) + \Delta t(1-\theta_1) f\left(t_{p+1}, \frac{r}{2^m}, \hat{y}_{p+1}^\Delta\left(\frac{r}{2^m}\right), \hat{z}_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right)\right] e^{-\imath C_k r}\right\},\end{aligned} \tag{3.7}$$

and the function $\hat{y}_p^\Delta$ is defined implicitly by:

$$\hat{y}_p^\Delta(x) = \Delta t\theta_1 f(t_p, x, \hat{y}_p^\Delta(x), \hat{z}_p^\Delta(x)) + \hat{h}(t_p, x). \tag{3.8}$$

Whenever $\theta_1 \neq 0$, *Picard iterations* are performed I times to recover $\hat{y}_p^\Delta(x)$, which is the same procedure used in both [10] and [18]. The initial guess for the iterations is defined as the approximation of $\mathbb{E}_p^x[Y_{t_{p+1}}^\Delta]$ as in Equation (3.6a). The conditions for convergent iterations and an extra error induced will be discussed in Section 4.3.

The overall algorithm to generate an approximation $(\hat{y}_0^\Delta(x), \hat{z}_0^\Delta(x))$ for (Y_0, Z_0) has been summarized in Algorithm 1.

4 Errors and computational complexity

In this section, we shall discuss the major components of the error when solving an FBSDE with a SWIFT-type method. They are the discretization error of the FBSDE, the error of approximation with the SWIFT formula and the error introduced by the Picard iteration. We will also discuss the computational complexity of the SWIFT method. For notational simplicity, we shall use M to denote a generic constant whose value and dependency may change from line to line.

begin
  for $s = 1 - J$ *to* J **do**
    $\hat{y}_P^{\Delta}(2^{-m}s) = g(2^{-m}s)$, $\hat{z}_P^{\Delta}(2^{-m}s) = \sigma D_x g(2^{-m}s)$ and
    $\hat{f}_P^{\Delta}(2^{-m}s) = f(T, 2^{-m}s, \hat{y}_P^{\Delta}(2^{-m}s), \hat{z}_P^{\Delta}(2^{-m}s))$.
  end
  Compute $(\mathbb{E}_{P-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_P}^{\Delta})])_{r,k=1-J,\dots,J}$ and $(\mathbb{E}_{P-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_P}^{\Delta})\Delta\omega_P])_{r,k=1-J,\dots,J}$ with (3.2) and (3.3).
  for $p = P - 1$ *to 1* **do**
    Compute the function $(\hat{z}_p^{\Delta}(2^{-m}s))_{s=1-J,\dots,J}$ with (3.4).
    Compute the function $(\hat{y}_p^{\Delta}(2^{-m}s))_{s=1-J,\dots,J}$ with (3.8) and Picard iterations if necessary.
    Compute the functions $(f(t_p, 2^{-m}s, \hat{y}_p^{\Delta}(2^{-m}s), \hat{z}_p^{\Delta}(2^{-m}s)))_{s=1-J,\dots,J}$.
    Compute $(\mathbb{E}_{p-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_p}^{\Delta})])_{r,k=1-J,\dots,J}$ and $(\mathbb{E}_{p-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_p}^{\Delta})\Delta\omega_p])_{r,k=1-J,\dots,J}$ with (3.2) and (3.3) if the distribution of $(X_{t_p}^{\Delta} - X_{t_{p-1}}^{\Delta})$ is time-dependent.
  end
  Compute $\hat{z}_0^{\Delta}(x_0)$ and $\hat{y}_0^{\Delta}(x_0)$.
end

Algorithm 1: Quick SWIFT method.

4.1 Discretization error of the FBSDE

The error due to the discrete-time approximation of the stochastic process depends on the parameters θ_1 and θ_2, the drift μ, the volatility σ, the driver function f and the terminal condition g. It is difficult to provide a universal result for all FBSDEs for which our method can be applied. However, under some specific assumptions, we can derive an error bound for the global error due to time discretization. Adopting the following error notation:

$$\varepsilon_p^y(X_p) := y_p(X_{t_p}) - y_p^{\Delta}(X_{t_p}^{\Delta}),\ \varepsilon_p^z := z_p(X_{t_p}) - z_p^{\Delta}(X_{t_p}^{\Delta}),\ \varepsilon_p^f(X_{t_p}) := f(t_p, \mathbf{X}_{t_p}) - f(t_p, \mathbf{X}_{t_p}^{\Delta}),$$

one of the results for the discretization error is in the following theorem.

Theorem 4.1 ([18],Theorem 1.)**.** *Assume that the forward process has constant coefficients μ and σ and the discretization scheme with $\theta_1 = \theta_2 = \frac{1}{2}$. If*

$$\mathbb{E}_{P-1}^x[|\varepsilon_P^z|] \sim \mathcal{O}((\Delta t)^3),\ \mathbb{E}_{P-1}^x[|\varepsilon_P^y|] \sim \mathcal{O}((\Delta t)^3),$$

then

$$\mathbb{E}_0^x\left[|\varepsilon_p^y| + \sqrt{\Delta t}|\varepsilon_p^z|\right] \leq M(\Delta t)^2,\ \textit{for } 1 \leq p \leq P,$$

where the constant M depends on numbers T, μ and σ and functions g and f.

For general drift and diffusion coefficients, we may only have *first-order weak convergence* for the Euler discretization of forward process X and it may become the dominating error of our algorithm. For the proof of Theorem 4.1 and other convergence results, readers are referred to [18] and the references therein.

4.2 Error in SWIFT formulas

In this subsection, we shall discuss the error of the numerical approximation by the SWIFT-type formulas. In order for our formula to be applicable for both the one-step approximation and the recursive case, we adopt the following setting.

Consider an expectation $\mathbb{E}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]$ for a function v defined on $\mathbb{R}$ and assume that v is continuous with all its left- and right-side derivatives well-defined, we define our expectation approximation as

$$\hat{\mathbb{E}}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] := \frac{1}{J}\sum_{r=1-J}^{J} \rho_v\left(\frac{r}{2^m}\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x], \tag{4.1}$$

where $\{\rho_v(2^{-m}r)\}_{r=1-J,\ldots,J}$ is an approximation vector related to function v, to be defined. For any function $v : \mathbb{R} \to \mathbb{R}$ and given range $(-2^{-m}J, 2^{-m}J]$, we associate v with *an alternating extension* defined below.

Definition 4.2. An *alternating extension of a function* v, denoted by $\tilde{v}$, is a function defined on $\mathbb{R}$ such that it satisfies:

(a) $\tilde{v}(x) = v(x) \quad \forall x \in (-2^{-m}J, 2^{-m}J]$;

(b) $\tilde{v}(x + 2^{1-m}J) = -\tilde{v}(x) \quad \forall x \in \mathbb{R}$.

The difference between the approximation value and the true value is given by

$$\begin{aligned}
&\mathbb{E}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] - \hat{\mathbb{E}}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]\\
=&\mathbb{E}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] - \mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] + \mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] - \mathbb{E}[H_{V_J}v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]\\
+&\mathbb{E}\left[\frac{1}{J}\sum_{r=1-J}^{J} < H_{V_J}v, \varphi_{J,r} > \varphi_{J,r}(X^{\Delta}_{t+\Delta t})\middle| X^{\Delta}_t = x\right] - \frac{1}{J}\sum_{r=1-J}^{J}\rho_v\left(\frac{r}{2^m}\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]\\
=&\mathbb{E}[v(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t = x] - \mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t = x]\\
+&\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t}) - H_{V_J}v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] + \frac{1}{J}\sum_{r=1-J}^{J}\left(H_{V_J}v\left(\frac{r}{2^m}\right) - \rho_v\left(\frac{r}{2^m}\right)\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]
\end{aligned} \tag{4.2}$$

$$\begin{aligned}
=&\mathbb{E}[v(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t = x] - \mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t = x]\\
+&\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t}) - H_{V_J}v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] + \frac{1}{J}\sum_{r=1-J}^{J}\left(H_{V_J}v\left(\frac{r}{2^m}\right) - v\left(\frac{r}{2^m}\right)\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]\\
+&\frac{1}{J}\sum_{r=1-J}^{J}\left(v\left(\frac{r}{2^m}\right) - \rho_v\left(\frac{r}{2^m}\right)\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x].
\end{aligned} \tag{4.3}$$

We derive the above formula by telescoping and using the properties of the scaling function. By taking absolute values on both sides of Equation (4.3), we get a simple bound for the approximation

error:

$$
\begin{aligned}
&\left|\mathbb{E}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]-\hat{\mathbb{E}}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]\right|\\
&\leq|\mathbb{E}[v(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t=x]|+\mathbb{E}[|\tilde{v}(X^{\Delta}_{t+\Delta t})|\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t=x]\\
&+\mathbb{E}[|\tilde{v}(X^{\Delta}_{t+\Delta t})-H_{V_J}v(X^{\Delta}_{t+\Delta t})|\,|X^{\Delta}_t=x]+\frac{1}{J}\sum_{r=1-J}^{J}\left|H_{V_J}v\left(\frac{r}{2^m}\right)-\tilde{v}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]|\\
&+\frac{1}{J}\sum_{r=1-J}^{J}\left|v\left(\frac{r}{2^m}\right)-\rho_v\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]|.
\end{aligned}
$$

Errors of a SWIFT-type method can be separated into four (in Equation (4.2)) or five (in Equation (4.3)) parts and they will be discussed one by one. Note that the first three terms in Equation (4.2) and (4.3) are the same.

The first error term is related to the tail behaviour of the probability measure and function v. It is finite, as otherwise the original expectation would be infinite. Also, its value should decrease (heuristically speaking, not in strict mathematical sense) when a wider computational domain is used. Assuming v to be uniformly bounded by a number M, this term is bounded by $M\cdot\mathbb{P}(X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]|X^{\Delta}_t=x)$. Similarly, the second term is related to the tail probability. Because of the continuity of v and the periodicity of $\tilde{v}$, $\tilde{v}$ is uniformly bounded by some number M' and the second error term is bounded by $M'\cdot\mathbb{P}(X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]|X^{\Delta}_t=x)$.

The third part is related to the projection error on V_J. From the assumption that v is continuous with all left- and right-side derivatives of v existing, $\tilde{v}=\lim_{J\to\infty}H_{V_J}v$, a.e.. This can be shown by adopting the classical Dirichlet's kernel argument. Applying the dominated convergence theorem,

$$
\begin{aligned}
\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})-H_{V_J}v(X^{\Delta}_{t+\Delta t})\,|X^{\Delta}_t=x]=&\sum_{k=J+1}^{\infty}<v,\cos(2^mC_j\cdot)>\mathbb{E}[\cos(2^mC_jX^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]\\
&+\sum_{k=J+1}^{\infty}<v,\sin(2^mC_j\cdot)>\mathbb{E}[\sin(2^mC_jX^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x].
\end{aligned}
$$

Note that in this part, we only require $\tilde{v}=\lim_{J\to\infty}H_{V_J}v$, a.e., so that we can relax the requirement on v from being continuous to being piecewise continuous. Using an integration by parts argument, if the forward process has a smooth density, this error converges exponentially with respect to J but increases with the computational range.

Remark 4.3. In fact, the projection error in the SWIFT formula can be controlled under alternative conditions. Assume that the probability density function q of $X^{\Delta}_{t+\Delta t}|X^{\Delta}_t=x$, is in $L^2(\mathbb{R})$, then,

$$
\begin{aligned}
&|\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})-H_{V_J}v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t=x]-\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t=x]|\\
=&|\mathbb{E}[\tilde{v}(X^{\Delta}_{t+\Delta t})-H_{V_J}v(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\in(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t=x]\\
&-\mathbb{E}[H_{V_J}v(X^{\Delta}_{t+\Delta t})\mathbf{1}_{\{X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]\}}|X^{\Delta}_t=x]|\\
\leq&||\tilde{v}-H_{V_J}v||_2\left(\int_{\mathbb{R}}q^2(\varsigma|x)d\varsigma\right)^{\frac{1}{2}}+M_3\mathbb{P}(X^{\Delta}_{t+\Delta t}\notin(-2^{-m}J,2^{-m}J]|X^{\Delta}_t=x),
\end{aligned}
$$

with M_3 depending on the function $H_{V_J}v$. While we do not use this alternative proof in this article, it implies that the SWIFT formula can be used in a more general setting and is suitable for other applications as well.

In the usual variant of the SWIFT formula, we set $\rho_v(2^{-m}r) =< v, \varphi_{J,r} >$, so the fourth term of Equation (4.2) is by definition zero and the error of applying the SWIFT formula will only consist of the first three terms. However, the calculation of $< v, \varphi_{J,r} >$ is difficult and time-consuming, if not impossible in practice, especially in the recursive case. Therefore, we propose picking $\rho_v(2^{-m}r) = \hat{v}(2^{-m}r)$, *an approximation of the original function* v. While it will introduce an extra error, we shall demonstrate that this error can be controlled and that the computation is much simpler.

For the sum in the fourth term of Equation (4.3), we need to consider two cases. When $r \neq J$, the pointwise convergence of $H_{V_J}v$ guarantees that $\left|H_{V_J}v\left(\frac{r}{2^m}\right) - \tilde{v}\left(\frac{r}{2^m}\right)\right| \to 0$. Therefore, these terms are bounded when J is large enough. When $r = J$, it is likely that $\tilde{v}$ is discontinuous at $2^{-m}J$ and the above argument does not hold. However, we note that this error term is also a weighted sum of $H_{V_J}v(2^{-m}r) - v(2^{-m}r)$, with the weight given by $\frac{1}{J}\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]$. Assume that $\mathbb{P}(X^{\Delta}_{t+\Delta t} \notin (\lambda - b, \lambda + b)) < \epsilon_1$ and $\frac{1}{J}\varphi_{J,J}(x) < \epsilon_2$, when $x \in (\lambda - b, \lambda + b)$, for some positive numbers b, ϵ_1 and ϵ_2 and some number λ, then

$$\frac{1}{J}\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x] <\mathbb{P}(X^{\Delta}_{t+\Delta t} \notin (\lambda - b, \lambda + b)) + \epsilon_2\mathbb{P}(X^{\Delta}_{t+\Delta t} \in (\lambda - b, \lambda + b)) < \epsilon_1 + \epsilon_2.$$

These assumptions are satisfied when the distribution of X^{Δ} is centered around a point λ, which is true with a diffusion process whose diffusion coefficient is small, the computational range is sufficiently large so that λ is far away from the boundary, and the wavelet order is sufficiently high. If these conditions are met, the weight for the term $\left|H_{V_J}v\left(\frac{J}{2^m}\right) - \tilde{v}\left(\frac{J}{2^m}\right)\right|$ is small and the weighted term can be bounded. By combining the two arguments above, we can bound this error term when the computational range is sufficiently large and the wavelet order is sufficiently high.

In the one-step case, we pick $\rho_v = v$. Equation (4.2) along with the above analysis covers all approximation errors.

In the backward recursive case, we can use Equation (4.3) to study the error propagation at each time step. For example, in our BSDE algorithm, we let $v = y^{\Delta}_{p+1}$ or z^{Δ}_{p+1} and $\rho_v = \hat{y}^{\Delta}_{p+1}$ or $\hat{z}^{\Delta}_{p+1}$. In these cases, the fifth term of Equation (4.3) is comparing our approximation with the true value at the next time step. The error accumulates in a recursive way by applying this error analysis from time step t_0 to t_{p-1}. Further discussion of the error propagation will be given in Section 4.4.

The error for approximating $\mathbb{E}[v(X^{\Delta}_{t+\Delta t})\Delta\omega_{p+1}|X^{\Delta}_t = x]$ with

$$\hat{\mathbb{E}}[v(X^{\Delta}_{t+\Delta t})(\omega_{t+\Delta_t} - \omega_t)|X^{\Delta}_t = x] := \frac{1}{J}\sum_{r=1-J}^{J}\rho_v\left(\frac{r}{2^m}\right)\mathbb{E}[\varphi_{J,r}(X^{\Delta}_{t+\Delta t})(\omega_{t+\Delta} - \omega_t)|X^{\Delta}_t = x], \quad (4.4)$$

can be studied in a similar way.

Remark 4.4. It is clear from the derivation that the assumption of the function v being continuous with left- and right-side derivatives is crucial in applying the quick version of the SWIFT formula. In our FBSDE algorithm, the functions y and z at intermediate time points, $p = 1, \ldots, P-1$, satisfy the above conditions. This can be observed from Equations (3.4) and (3.8). However, we may still face an issue at the terminal time, as $D_x g$ may contain discontinuities. *We propose a mixed*

algorithm to deal with this situation. At the terminal time, the usual SWIFT formulas are used and then the algorithm switches to the quick version in all subsequent time steps, see Algorithm 2.

begin
  for $s = 1 - J$ *to* J **do**
    $\hat{y}_P^{\Delta}(2^{-m}s) =< g, \varphi_{J,s} >$, $\hat{z}_P^{\Delta}(2^{-m}s) =< \sigma D_x g, \varphi_{J,s} >$ and
    $\hat{f}_P^{\Delta}(2^{-m}s) =< f(T, \cdot, g(\cdot), \sigma D_x(\cdot)), \varphi_{J,r} >$.
  end
  Compute $(\mathbb{E}_{P-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_P}^{\Delta})])_{r,k=1-J,\dots,J}$ and $(\mathbb{E}_{P-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_P}^{\Delta})\Delta\omega_P])_{r,k=1-J,\dots,J}$ with (3.2) and (3.3).
  for $p = P - 1$ *to* 1 **do**
    Compute the function $(\hat{z}_p^{\Delta}(2^{-m}s))_{s=1-J,\dots,J}$ with (3.4).
    Compute the function $(\hat{y}_p^{\Delta}(2^{-m}s))_{s=1-J,\dots,J}$ with (3.8) and Picard iterations if necessary.
    Compute the functions $(f(t_p, 2^{-m}s, \hat{y}_p^{\Delta}(2^{-m}s), \hat{z}_p^{\Delta}(2^{-m}s)))_{s=1-J,\dots,J}$.
    Compute $(\mathbb{E}_{p-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_p}^{\Delta})])_{r,k=1-J,\dots,J}$ and $(\mathbb{E}_{p-1}^{2^{-m}r}[\varphi_{J,k}(X_{t_p}^{\Delta})\Delta\omega_p])_{r,k=1-J,\dots,J}$ with (3.2) and (3.3) if the distribution of $(X_{t_p}^{\Delta} - X_{t_{p-1}}^{\Delta})$ is time-dependent.
  end
  Compute $\hat{z}_0^{\Delta}(x_0)$ and $\hat{y}_0^{\Delta}(x_0)$.
end

Algorithm 2: Mixed quick SWIFT method.

4.3 Picard iteration error

When $\theta_1 \neq 0$, a Picard iteration must be performed with the equation:

$$y = \Delta t\theta_1 f(t_p, x, y, \hat{z}_p^{\Delta}(x)) + \hat{h}(t_p, x),$$

to find the fixed point y. It is well-known that this iterative algorithm will converge to the unique fixed point if the function $\Delta t\theta_1 f$ is a contraction map of y, namely,

$$|\Delta t\theta_1 f(t_p, x, y_1, \hat{z}_p^{\Delta}(x)) - \Delta t\theta_1 f(t_p, x, y_2, \hat{z}_p^{\Delta}(x))| \leq \xi |y_1 - y_2|,$$

with $\xi \in [0, 1)$ for all $x \in (-2^{-m}J, 2^{-m}J]$. This condition is satisfied when the driver function is Lipschitz in y and Δt is small enough.

We adopt the following notation:

$$\begin{cases} \hat{y}_P^{\Delta,I}(x) := g(x); \\ \hat{y}_p^{\Delta,0}(x) := \dfrac{1}{J} \displaystyle\sum_{r=1-J}^{J} \hat{y}_{p+1}^{\Delta,I}\left(\frac{r}{2^m}\right) \Re\left\{ \sum_{k=1}^{J} e^{\imath 2^m C_k x} \Phi(t_p, x, 2^m C_k) e^{-\imath C_k r} \right\}; \\ \hat{y}_p^{\Delta,i+1}(x) := \Delta t\theta_1 f(t_p, x, \hat{y}_p^{\Delta,i}(x), \hat{z}_p^{\Delta}(x)) + \hat{h}(t_p, x), \end{cases}$$

for $p = 0, \dots, P-1$ and $i = 0, \dots, I-1$. It is clear that $\hat{y}_p^{\Delta,I}(x) = \hat{h}(t_p, x)$ when $\theta_1 = 0$ and $I \geq 1$, which is the explicit scheme. The above notations are consistent with the notations in Section 3.4

except we should replace the $\hat{y}^{\Delta}_{p+1}$ with $\hat{y}^{\Delta,I}_{p+1}$ in equation (3.5). Furthermore, for any given x, we know by definition that $y^{\Delta}_p(x)$ is the unique fixed point that satisfies

$$y = \Delta t\theta_1 f(t_p, x, y, z^{\Delta}_p(x)) + h(t_p, x).$$

Note that the notation $\hat{y}^{\Delta}_p$ was defined by Equation (3.8).

With the above notations and given the extra information that f is Lipschitz with respect to z, we can derive the one-step approximation error for function y^{Δ}:

$$\begin{aligned}
&|\hat{y}^{\Delta,I}_p(x) - y^{\Delta}_p(x)| \le |\hat{y}^{\Delta,I}_p(x) - \hat{y}^{\Delta}_p(x)| + |\hat{y}^{\Delta}_p(x) - y^{\Delta}_p(x)| \\
\le&\varepsilon^{Picard}_p + \Delta t\theta_1 |f(t_p, x, \hat{y}^{\Delta}_p(x), \hat{z}^{\Delta}_p(x)) - f(t_p, x, y^{\Delta}_p(x), z^{\Delta}_p(x))| + |\hat{h}(t_p, x) - h(t_p, x)| \\
\le&\varepsilon^{Picard}_p + \xi|\hat{y}^{\Delta}_p(x) - y^{\Delta}_p(x)| + \xi|\hat{z}^{\Delta}_p(x) - z^{\Delta}_p(x)| + |\hat{h}(t_p, x) - h(t_p, x)| \\
\le&(1+\xi)\varepsilon^{Picard}_p + \xi|\hat{y}^{\Delta,I}_p(x) - y^{\Delta}_p(x)| + \xi|\hat{z}^{\Delta}_p(x) - z^{\Delta}_p(x)| + |\hat{h}(t_p, x) - h(t_p, x)|.
\end{aligned}$$

The term $\varepsilon^{Picard}_p := |\hat{y}^{\Delta,I}_p(x) - \hat{y}^{\Delta}_p(x)|$ is the error of applying Picard iterations, which depends on Δt and the Lipschitz coefficient of f with respect to y, as stated before in this section. The constant M is related to the Lipschitz coefficient of f from standing assumption (A3) and $\xi := M\Delta t\theta_1 \le 1$. The last inequality is due to a telescoping argument of the term $|\hat{y}^{\Delta}_p(x) - y^{\Delta}_p(x)|$. Rearranging the terms gives us the following error bound:

$$|\hat{y}^{\Delta,I}_p(x) - y^{\Delta}_p(x)| \le \frac{1+\xi}{1-\xi}\varepsilon^{Picard}_p + \frac{1}{1-\xi}(\xi|\hat{z}^{\Delta}_p(x) - z^{\Delta}_p(x)| + |\hat{h}(t_p, x) - h(t_p, x)|). \tag{4.5}$$

4.4 The errors of the FBSDE recursive scheme

Given an FBSDE system, a time partition and a discretization scheme, we may apply the result in Section 4.2 to derive a local approximation error for the related expectations. When we are approximating expectations with the functions y^{Δ}_{p+1} and z^{Δ}_{p+1} in the BSDE scheme, the approximation vector is given by

$$\begin{cases} \rho_{y^{\Delta}_{p+1}} = \{\hat{y}^{\Delta,I}_{p+1}(2^{-m}r)\}_{r=1-J,\dots,J}; \\ \rho_{z^{\Delta}_{p+1}} = \{\hat{z}^{\Delta}_{p+1}(2^{-m}r)\}_{r=1-J,\dots,J}, \end{cases}$$

for $p = 0, \dots, P-2$. At the terminal time $t_P = T$, they are defined as

$$\begin{cases} \rho_{y^{\Delta}_P} = \{Y^{\Delta}_T | X^{\Delta}_T = 2^{-m}r\}_{r=1-J,\dots,J}; \\ \rho_{z^{\Delta}_P} = \{Z^{\Delta}_T | X^{\Delta}_T = 2^{-m}r\}_{r=1-J,\dots,J}, \end{cases}$$

or

$$\begin{cases} \rho_{y^{\Delta}_P} = \{< Y^{\Delta}_T, \varphi_{J,r} >\}_{r=1-J,\dots,J}; \\ \rho_{z^{\Delta}_P} = \{< Z^{\Delta}_T, \varphi_{J,r} >\}_{r=1-J,\dots,J}, \end{cases}$$

depending on the scheme we used. When approximating expectations involving $f(t_{p+1}, x, y^{\Delta}_{p+1}(x), z^{\Delta}_{p+1}(x))$, the approximation vector $\rho_{f_{p+1}} = \{f(t_{p+1}, 2^{-m}r, \rho_{y^{\Delta}_{p+1}}(2^{-m}r), \rho_{z^{\Delta}_{p+1}}(2^{-m}r)\}_{r=1-J,\dots,J}$, for $p = 0, \dots P-1$.

From Equation (4.3), we know that the approximation error for the SWIFT formula consists of four parts. Therefore, the local approximation errors at point (t,x) by the SWIFT formula for any function v, denoted as $\zeta_v^{m,J}(t_p,x)$ or $\zeta_v^{m,J,\omega}(t_p,x)$ for the two types of expectation, are given by

$$\begin{aligned}
\zeta_v^{m,J}(t_p,x) :=& \mathbb{E}_p^x[v(X_{t_{p+1}}^\Delta)\mathbf{1}_{\{X_{t_{p+1}}^\Delta \notin (-2^{-m}J, 2^{-m}J]\}}] - \mathbb{E}_p^x[\tilde{v}(X_{t_{p+1}}^\Delta)\mathbf{1}_{\{X_{t_{p+1}}^\Delta \notin (-2^{-m}J, 2^{-m}J]\}}] \\
&+\mathbb{E}_p^x[\tilde{v}(X_{t_{p+1}}^\Delta) - H_{V_J}v(X_{t_{p+1}}^\Delta)] + \frac{1}{J}\sum_{r=1-J}^{J}\left(H_{V_J}v\left(\frac{r}{2^m}\right) - v\left(\frac{r}{2^m}\right)\right)\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]; \\
\zeta_v^{m,J,\omega}(t_p,x) :=& \mathbb{E}_p^x[v(X_{t_{p+1}}^\Delta)\mathbf{1}_{\{X_{t_{p+1}}^\Delta \notin (-2^{-m}J, 2^{-m}J]\}}\Delta\omega_{p+1}] - \mathbb{E}_p^x[\tilde{v}(X_{t_{p+1}}^\Delta)\mathbf{1}_{\{X_{t_{p+1}}^\Delta \notin (-2^{-m}J, 2^{-m}J]\}}\Delta\omega_{p+1}] \\
&+\mathbb{E}_p^x[(\tilde{v}(X_{t_{p+1}}^\Delta) - H_{V_J}v(X_{t_{p+1}}^\Delta))\Delta\omega_{p+1}] + \frac{1}{J}\sum_{r=1-J}^{J}\left(H_{V_J}v\left(\frac{r}{2^m}\right) - v\left(\frac{r}{2^m}\right)\right)\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}].
\end{aligned}$$

Applying all the results above and the standing assumptions, we can derive the recurring error formulas of our SWIFT BSDE scheme:

$$\begin{aligned}
\frac{1}{M_1}|z_p^\Delta(x) - \hat{z}_p^\Delta(x)| \leq& |\zeta_{z_{p+1}^\Delta}^{m,J}(t_p,x)| + |\zeta_{y_{p+1}^\Delta}^{m,J,\omega}(t_p,x)| + |\zeta_{f_{p+1}}^{m,J,\omega}(t_p,x)| \\
&+\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \rho_{z_{p+1}^\Delta}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]| + \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|) \\
&+\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \rho_{y_{p+1}^\Delta}\left(\frac{r}{2^m}\right)\right||\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|, \qquad (4.6)
\end{aligned}$$

and

$$\begin{aligned}
\frac{1}{M_2}|y_p^\Delta(x) - \hat{y}_p^{\Delta,I}(x)| \leq& M_3 + |\zeta_{z_{p+!}^\Delta}^{m,J}(t_p,x)| + |\zeta_{y_{p+1}^\Delta}^{m,J,\omega}(t_p,x)| + |\zeta_{f_{p+1}}^{m,J,\omega}(t_p,x)| + |\zeta_{y_{p+1}^\Delta}^{m,J}(t_p,x)| + |\zeta_{f_{p+1}}^{m,J}(t_p,x)| \\
&+\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \hat{z}_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]| + \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|) \\
&+\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \hat{y}_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]| + \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|), \\
&\qquad (4.7)
\end{aligned}$$

with constants M_1, M_2 and M_3 depending on the underlying FBSDE, the discretization scheme and the Picard iteration errors, but not depending on m and J. Their proof is left for the appendix.

4.4.1 Error bound and choice of parameters

An error bound at $(0,x_0)$ for applying the SWIFT scheme to a FBSDE system can be found by repeatedly applying Equations (4.6) and (4.7). This bound is given by the weighted sum of the local approximation errors for each point in the grid $\{(0,x_0)\} \cup \{(t_p, 2^{-m}r) | t = 1,\dots,P \text{ and } r = 1-J,\dots,J\}$, with the weight being 1 for $(0,x_0)$ and the weight being

$$\sum_{u\in\upsilon_p}\frac{1}{J^p}\prod_{l=1}^{p}(|\mathbb{E}_l^{u_{l-1}}[\varphi_{J,u_l}(X_{t_l}^\Delta)]| + |\mathbb{E}_l^{u_{l-1}}[\varphi_{J,u_l}(X_{t_l}^\Delta)\Delta\omega_{l+1}]|), \qquad (4.8)$$

for $\{(t_p, 2^{-m}r)|t = 1, \ldots, P \text{ and } r = 1 - J, \ldots, J\}$, where υ_p is the set containing length $p+1$ vectors $u = (u_0, u_1, \ldots, u_p)$, with the first element $u_0 = x_0$ and other elements equal to $2^{-m}r$ for $r = 1 - J, \ldots, J$. A simple example of deriving such an error bound is included in the appendix.

However, since this error bound uses local approximation errors from multiple points in a grid, actually calculating the error bound would be costly. The weight and the local error behave differently at different grid points. Whenever $|r|$ is small, the local error converges exponentially in numerical tests with fixed $2^{-m}J$ and increasing J, but it may fail to converge to zero when $|r|$ is close to J. On the other hand, when $|r|$ is close to J, the weight in Equation (4.8) tends to zero quickly and reduces the error. We do not have a simple formula to describe this balance. Last but not least, parameter P, the number of time points, affects the total number of terms in the error bound, the value of the local error and the value of the weight in Equation (4.8). It is highly non-trivial to quantify the effect of P on the overall error from this error bound.

In practice, we would recommend a three-step approach to select the parameters for the scheme and get a global picture of what the error may be. First of all, pick parameter P based on the discretization error for (X, Y, Z). This can be done either through the error bound in Section 4.1 or other existing error bounds in the literature. The reason is that m and J have no effect on the discretization error while P affects each part of the approximation error. Next, we should choose our parameters J and m according to error formula (4.3). The interest is in the third term in the equation, which increases with the truncation range but decreases with the wavelet order J. Therefore we should first fix the truncation range $2^{-m}J$ to a constant value a in our scheme, the tail probability outside our computational range is below a fixed tolerance level. This can be done heuristically by considering the cumulants of X_T, see [8]. Finally, we pick a J value such that the overall error converges and adjust m accordingly so that the truncation range remains the same. This approach is very useful for applications (compared to the error bound itself).

4.5 Computational Complexity

At each time-step t_p, the following operations have to be performed:

- *Computation* of $\mathbb{E}_p^x[\varphi_{J,k}(X_{t_{p+1}}^{\Delta})]$ and $\mathbb{E}_p^x[\varphi_{J,k}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]$ by the FFT algorithm, in $\mathrm{O}(J\log(J))$ operations. It is calculated only once at the terminal time-step if the characteristic function of X^{Δ} does not depend on the time point;
- *Computation* of $\hat{z}_p^{\Delta}(x)$, $\hat{h}(t_p, x)$ and $\hat{y}_p^{\Delta,0}(x)$ by matrix-vector multiplications, in $\mathcal{O}(J^2)$ operations;
- *Computation* of $\hat{y}^{\Delta,I}$ by I Picard iterations on an x-grid, in $\mathcal{O}(IJ)$ operations;
- *Evaluation* of $f(t_p, x, \hat{y}_p^{\Delta,I}(x), \hat{z}_p^{\Delta}(x))$ in $\mathcal{O}(J)$ operations.

The most time-consuming part in our algorithm is the matrix-vector multiplication. The proposed algorithms have linear computational complexity with respect to the time-steps P and the starting evaluation at terminal time is of order $\mathcal{O}(J)$. In total, the complexity of the SWIFT-type methods is $\mathcal{O}(J + P[J^2 + IJ + J\log(J) + J])$.

Figure 1: The approximation range $(-2^{-m}J, 2^{-m}J]$ and the accuracy range $[\alpha, \beta]$.

5 Antireflective boundary

Recalling Equation (4.3), the approximation of $\mathbb{E}[v(X^{\Delta}_{t+\Delta t})|X^{\Delta}_t = x]$ by the SWIFT formula may have a significant local error when two conditions are satisfied. The first condition being that the alternating extension $\tilde{v}$ diverges from v in the range $[-\eta - 2^{-m}J, -2^{-m}J]$ or $[2^{-m}J, \eta + 2^{-m}J]$, for some number $\eta > 0$ and the second condition being that the probability of $X^{\Delta}_{t+\Delta t}|X^{\Delta}_t = x$ in the aforementioned ranges is large. While the first condition is almost always true, given that X^{Δ} is a diffusion process, the second condition is true only when the starting point x is close to the boundary $-2^{-m}J$ or $2^{-m}J$. Therefore, there may be intervals $(-2^{-m}J, \alpha)$ and $(\beta, 2^{-m}J]$ where the SWIFT formula is inaccurate.

We propose using an antireflective boundary technique to deal with this issue. Antireflective boundary conditions form a popular technique for extrapolation in image deblurring methods. For its applications in image deblurring, the reader may refer to [7] and the references therein.

In practice, assume that we approximate a function $\vartheta(x)$ by $\hat{\vartheta}(x)$ on $(-2^{-m}J, 2^{-m}J]$ and we know that the approximation is accurate for $x \in [\alpha, \beta]$, namely, $|\vartheta(x) - \hat{\vartheta}(x)| < \epsilon$, for some small positive real number ϵ. Given the numbers $\alpha > -2^{-m}J$ and $\beta < 2^{-m}J$, so that there is some inaccuracy near the boundary but $(\alpha, 2\alpha + 2^{-m}J), (2\beta - 2^{-m}J, \beta) \subset [\alpha, \beta]$ (see Figure 1). We would extrapolate an approximation of $\vartheta(x)$ for $x \in (-2^{-m}J, \alpha)$ and $x \in (\beta, 2^{-m}J]$ by applying antireflective conditions with the accurate approximation. For $x \in (-2^{-m}J, 2^{-m}J]$, we define

$$\begin{cases} \hat{\vartheta}^a(x) := 2\hat{\vartheta}(\alpha) - \hat{\vartheta}(2\alpha - x) & \text{for } x \in (-2^{-m}J, \alpha); \\ \hat{\vartheta}^a(x) := \hat{\vartheta}(x) & \text{for } x \in [\alpha, \beta]; \\ \hat{\vartheta}^a(x) := 2\hat{\vartheta}(\beta) - \hat{\vartheta}(2\beta - x) & \text{for } x \in (\beta, 2^{-m}J), \end{cases} \tag{5.1}$$

and use $\hat{\vartheta}^a$ instead of $\hat{\vartheta}$ as our approximation.

If ϑ is two times continuously differentiable on $\mathbb{R}$, then, by a simple application of Taylor's theorem, we have:

$$\begin{aligned} \vartheta(x) =& \vartheta(\alpha) + \frac{d\vartheta}{dx}(\alpha)(x - \alpha) + \frac{1}{2}\frac{d^2\vartheta}{dx^2}(\varsigma_1)(x - \alpha)^2; \\ \vartheta(2\alpha - x) =& \vartheta(\alpha) - \frac{d\vartheta}{dx}(\alpha)(x - \alpha) + \frac{1}{2}\frac{d^2\vartheta}{dx^2}(\varsigma_2)(x - \alpha)^2, \end{aligned}$$

where $x \in (2^{-m}J, \alpha)$, $\varsigma_1 \in (x, \alpha)$ and $\varsigma_2 \in (\alpha, 2\alpha - x)$. The approximation error for $x \in (-2^{-m}J, \alpha)$ is then bounded by

$$\begin{aligned} |\vartheta(x) - \hat{\vartheta}^a(x)| \leq& |\vartheta(x) - 2\vartheta(\alpha) + \vartheta(2\alpha - x)| + 2|\vartheta(\alpha) - \hat{\vartheta}(\alpha)| + |\vartheta(2\alpha - x) - \hat{\vartheta}(2\alpha - x)| \\ \leq& (-2^{-m}J - \alpha)^2 \sup_{\varsigma \in (-2^{-m}J, 2\alpha + 2^{-m}J)} \left|\frac{d^2\vartheta}{dx^2}(\varsigma)\right| + 3\epsilon. \end{aligned}$$

A similar formula can be derived for the set $(2\beta - 2^{-m}J, 2^{-m}J)$. For the recursive scheme, one can just apply Equation (5.1) at each time-step.

Remark 5.1. The range of accurate approximations for the SWIFT formula depends on the distribution of $X^{\Delta}_{t+\Delta t}|_{X^{\Delta}_t=x}$, and, therefore, it is model dependent. The performance of applying the antireflective boundary technique with the SWIFT formula depends on the smoothness of the target function with respect to starting point x, the accuracy of the SWIFT formula in the range $[\alpha, \beta]$ and the length of the range.

6 Numerical experiments

Several numerical studies have been performed in MATLAB 9.0.0. The computer is equipped with Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz and 7.7 GB RAM. Four different discretization schemes have been used to test the effect of various choices of θ on the numerical algorithm. They are:

Scheme A:	$\theta_1 = 0$, $\theta_2 = 1$,	Scheme C:	$\theta_1 = 1$, $\theta_2 = 1$,
Scheme B:	$\theta_1 = 0.5$, $\theta_2 = 1$,	Scheme D:	$\theta_1 = 0.5$, $\theta_2 = 0.5$.

The function z^{Δ}_p is solved explicitly in all schemes while y^{Δ}_p is solved explicitly for scheme A and implicitly with 5 Picard iterations for the other schemes.

For each given example, we associate our numerical algorithm with the computational domain $[\kappa_1 - L\sqrt{\kappa_2}, \kappa_1 + L\sqrt{\kappa_2}]$, where cumulants $\kappa_1 = x_0 + \mu(0, x_0)T$ and $\kappa_2 = \sigma^2(0, x_0)T$ and $L = 10$. It is similar to the setting in [18]. The value $2^{-m}J = L\sqrt{\kappa_2}$ is a constant for each example. The value of J is assumed to be 2^9.

6.1 Example 1

This example has been studied in [18] and is originally from [20]. The considered FBSDE is

$$\begin{cases} dX_t = d\omega_t, \\ dY_t = -(Y_t Z_t - Z_t + 2.5Y_t - \sin(t + X_t)\cos(t + X_t) - 2\sin(t + X_t))dt + Z_t d\omega_t. \end{cases} \tag{6.1}$$

We take the initial and terminal conditions $x_0 = 0$ and $Y_T = \sin(X_T + T)$.

The exact solution is given by

$$(Y_t, Z_t) = (\sin(X_t + t), \cos(X_t + t)). \tag{6.2}$$

The terminal time is set to be $T = 1$ and $(Y_0, Z_0) = (0, 1)$. The driver function f depends on both time t and current state X_t. The results for the quick SWIFT method are presented in Figure 2a while the results for the *mixed SWIFT method* are presented in Figure 2b. We observe that there are no significant differences between the quick SWIFT and mixed SWIFT method. For schemes A, B and C, both approximation errors for $Y_0(x_0)$ and $Z_0(x_0)$ are of $\mathrm{O}(\Delta t)$ order, while the errors converge with $\mathrm{O}((\Delta t)^2)$ for scheme D.

Remark 6.1. The driver functions for some examples in this section are not universally Lipschitz. However, one should notice that for the contraction argument in Section 4.3 to be valid, for any fixed z, the driver function should be Lipschitz with respect to y. All the driver functions in this section satisfy this weaker condition. We aim for clear presentation rather than general applicability when we designed the assumptions and conditions and our method can be applied to a broader class of BSDEs than we described here. For a more in-depth analysis of the application of Picard iterations in the numerical treatment of BSDEs, the reader is referred to [10].

6.2 Example 2: European call option

Next, we calculate the price $v(t, S_T)$ of a call option under the Black-Scholes model by solving an FBSDE. The underlying process satisfies:

$$dS_t = \bar{\mu} S_t dt + \bar{\sigma} S_t d\omega_t. \tag{6.3}$$

Along the line of reasoning in [18], we assume that the financial market is complete, there are no trading restrictions and the call option can be perfectly hedged. Then $(v(t, S_t), \bar{\sigma} S_t D_s v(t, S_t))$ solves the FBSDE,

$$\begin{cases} dS_t = \bar{\mu} S_t dt + \bar{\sigma} S_t d\omega_t, \\ dY_t = -(-rY_t - \frac{\bar{\mu}-r}{\bar{\sigma}} Z_t)dt + Z_t d\omega_t, \end{cases} \tag{6.4}$$

with terminal condition $Y_T = \max(S_T - K, 0)$. The driver function is continuous and linear with respect to y and z. We use the following parameter values in our test:

$$S_0 = 100, K = 100, r = 0.1, \bar{\mu} = 0.2, \bar{\sigma} = 0.25, T = 0.1. \tag{6.5}$$

The exact solutions $Y_0 = 3.65997$ and $Z_0 = 14.14823$ are given by the Black-Scholes formula first shown in [3]. We switch to the log-asset domain $X_t = \log(S_t)$ and solve

$$\begin{cases} dX_t = \left(\bar{\mu} - \frac{1}{2}\bar{\sigma}^2\right) dt + \bar{\sigma} d\omega_t, \\ dY_t = -(-rY_t - \frac{\bar{\mu}-r}{\bar{\sigma}} Z_t)dt + Z_t d\omega_t, \end{cases} \tag{6.6}$$

where $Y_T = \max(\exp(X_T) - K, 0)$.

For the result of the quick SWIFT method, we refer to Figure 3a. Immediately, we notice that the result of scheme D does not improve when increasing the number of time-steps. This is due to the discontinuity of the terminal value of Z which creates a significant error. For the mixed SWIFT method, shown in Figure 3b, the error from the discontinuity has been removed. The approximate values $\hat{y}_0^{\Delta}(x_0)$ and $\hat{z}_0^{\Delta}(x_0)$ converge with approximately order 1 with respect to Δt for schemes A, B, and C and about order 2 for scheme D.

Since the driver function in Equation (6.6) depends on $\bar{\mu}$, the approximation error also depends on $\bar{\mu}$, even though the final result $v(0, x_0)$ is unrelated to the drift. For the same number of time-steps P, the error increased with the increase of $\bar{\mu}$, as shown in Figure 4.

The approximation algorithm can be further improved by applying antireflective boundary conditions in the recursive time-steps. In Figure 5 we see results of adding an antireflective step in a mixed SWIFT algorithm. The approximations near the middle of computational range are almost identical with the reference value, but the approximations with antireflective adjustment near both ends of the interval appear to be much better than the ones without.

6.3 Example 3: Bid-ask spread for interest rate

We next consider a financial model introduced in [2], where we have distinct borrowing and leading interest rates. The resulting market is imperfect and the driver function is non-linear.

Suppose that an agent can invest in bonds with risk-free return rate r and borrow money at rate $R > r$. For any European-type derivative with payoff $g(X_T)$ at time T, where the underlying asset $S_t = \log(X_t)$ follows a geometric Brownian motion, its price at time 0 can be obtained by solving the FBSDE:

$$\begin{cases} dX_t = \left(\bar{\mu} - \frac{1}{2}\bar{\sigma}^2\right) dt + \bar{\sigma} d\omega_t, \\ dY_t = -\left(-rY_t - \frac{\bar{\mu}-r}{\bar{\sigma}} Z_t - (R-r)\min\left(Y_t - \frac{Z_t}{\bar{\sigma}}, 0\right)\right) dt + Z_t d\omega_t, \end{cases}$$

with the payoff as the terminal condition. We use the example studied in both [1] and [18]. The payoff function is given by

$$g(X_T) = (e^{X_T} - K_1)^+ - 2(e^{X_T} - K_2)^+,$$

which equals a combination of a long call with strike $K_1 = 95$ and two short calls with strike $K_2 = 105$. We use the parameter values

$$S_0 = 100, r = 0.01, \bar{\mu} = 0.05, \bar{\sigma} = 0.2, T = 0.25, K_1 = 95, K_2 = 105, R = 0.06.$$

We notice that $\hat{z}_0^{\Delta}(x_0)$ fails to converge to the reference value with scheme D in Figure 6a. The reference values, $Y_0 = 2.9584544$ and $Z_0 = 0.55319$, are obtained by the BCOS method with a large number of time-steps P. Switching to the mixed SWIFT method, whose results are shown in Figure 6b, the approximated error for Y converges to zero with order of about 1 for schemes A, B and C and converges with order $\frac{3}{2}$ for scheme D. For schemes B and C, we also have a first-order convergence for Z but the convergence order is higher for schemes A and D.

6.4 Example 4

This example is taken from [19]. For the forward process, the drift and diffusion coefficients are time- and state-dependent. We aim to solve the following FBSDE:

$$\begin{cases} dX_t = \frac{1}{1+2\exp(t+X_t)}dt + \frac{\exp(t+X_t)}{1+\exp(t+X_t)}d\omega_t, \\ dY_t = -\left(-\frac{2Y_t}{1+2\exp(t+X_t)} - \frac{1}{2}\left(\frac{Y_t Z_t}{1+\exp(t+X_t)} - Y_t^2 Z_t\right)\right)dt + Z_t d\omega_t, \end{cases}$$

with the terminal condition $Y_T = g(X_T) = \frac{\exp(T+X_T)}{1+\exp(T+X_T)}$.

The exact solutions are given by

$$(Y_t, Z_t) = \left(\frac{\exp(t+X_t)}{1+\exp(t+X_t)}, \frac{(\exp(t+X_t))^2}{(1+\exp(t+X_t))^3}\right). \tag{6.7}$$

We choose terminal time $T = 1$ and initial condition $x_0 = 1$.

For the results of the quick SWIFT method, we refer to Figure 7. While the total error is different for each scheme, the approximated values $\hat{y}_0^{\Delta}(x_0)$ and $\hat{z}_0^{\Delta}(x_0)$ converge with $\mathrm{O}(\Delta t)$ for all schemes, as expected. Here the weak order of the Euler scheme plays a prominent role as the drift and volatility are state- and time- dependent.

6.5 Discussion

Compared with the BCOS method, the computational time for the SWIFT-type method is slightly lower when the number of basis functions used is the same and the forward process is independent of time. The most time-consuming portion is the matrix-vector multiplication used to calculate the value of $\hat{z}_p^{\Delta}$ and $\hat{h}$. We acknowledge that for the same error range, the BCOS and SWIFT-type methods may require different numbers of basis functions.

From the numerical experiments, we conclude that the computation with scheme D often fails to converge with Δt when the time-step is small when using the quick SWIFT method. This is due to the discontinuity of $z_P^{\Delta}(x)$ in x. Schemes A, B and C behave similarly for the quick SWIFT and

mixed SWIFT methods in our examples. However, scheme D often has the best performance in the mixed SWIFT method. This means that damping the discontinuity in our scheme is beneficial. The graphs also demonstrate that with a proper choice of SWIFT parameters, the approximation error itself will be dominated by the discretization error and decreases with respect to the increase of parameter P. It implies that the error of calculating expectation with SWIFT is relatively small.

7 Conclusion

A new probabilistic method for solving FBSDEs numerically has been proposed in this article. It is derived from a time-discretization of the forward and backward stochastic differential equations, taking conditional expectations to get an $\mathcal{F}_{t_p}$-adapted approximation and calculating the conditional expectations with the quick variant of the SWIFT formula.

We have shown that in order to apply the quick variant of the SWIFT formula, the continuity of the target function has to be ensured. While applying the quick variant of SWIFT formula instead of the original version introduces an extra error, it is of the same order of the original version when the target function is continuous and drops quickly with respect to J, due to the exponential convergence of the characteristic function for a smooth density. The error of applying the SWIFT method is relatively minor compared to the discretization error for the stochastic process. However, the quick variant of the SWIFT formula can greatly reduce the difficulties of our algorithm and increase the computational speed. So we believe that the mixed SWIFT method provides a good balance between efficiency and accuracy.

We have discussed the different approximation errors in detail in the paper. Additional attention is needed for the error of the SWIFT formula near the computational boundary, as we explained in Section 5. We also demonstrated how to improve our algorithm with the anti-reflective boundary conditions. Finally, the applicability and the effectiveness of our numerical algorithm have been tested with various FBSDEs, which all give positive results with the mixed SWIFT method.

Overall, applying the SWIFT method to solve discretized BSDEs retains the high accuracy of Fourier inversion techniques, although the computations involved are greatly simplified. We also gain additional freedom in adjusting the approximation values at each time point.

References

[1] Christian Bender and Jessica Steiner. Least-squares Monte Carlo for backward SDEs. In René A. Carmona, Pierre Del Moral, Peng Hu, and Nadia Oudjane, editors, *Numerical Methods in Finance: Bordeaux, June 2010*, pages 257–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[2] Yaacov Z. Bergman. Option pricing with differential interest rates. *Rev Financ Stud*, 8(2):475–500, 1995.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *J Polit Econ*, 81(3):637–654, 1973.

[4] Bruno Bouchard and Nizar Touzi. Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their Applications*, 111(2):175 – 206, 2004.

[5] Philippe Briand and Cline Labart. Simulation of BSDEs by Wiener chaos expansion. *Ann. Appl. Probab.*, 24(3):1129–1171, 06 2014.

[6] D. Crisan and K. Manolarakis. Solving backward stochastic differential equations using the cubature method: Application to nonlinear pricing. *SIAM J. Finan. Math.*, 3(1):534–571, 2012.

[7] Marco Donatelli and Stefano Serra-Capizzano. Antireflective boundary conditions for deblurring problems. *J. Elect. Comput. Eng*, 2010, 2010.

[8] F. Fang and C. W. Oosterlee. A novel pricing method for european options based on fourier-cosine series expansions. *SIAM J. Sci. Comput*, 31(2):826–848, 2009.

[9] Bernd Fischer and Jürgen Prestin. Wavelets based on orthogonal polynomials. *Math Comput*, 66(220):1593–1618, 1997.

[10] Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. A regression-based Monte Carlo method to solve backward stochastic differential equations. *Ann. Appl. Probab.*, 15(3):2172–2202, 08 2005.

[11] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 2nd edition, 2008. Cambridge Books Online.

[12] Jean-Philippe Lemor, Emmanuel Gobet, and Xavier Warin. Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations. *Bernoulli*, 12(5):889–916, 10 2006.

[13] S. C. Maree, L. Ortiz-Gracia, and C. W. Oosterlee. Pricing early-exercise and discrete barrier options by Shannon wavelet expansions. 2016. (working paper).

[14] Luis Ortiz-Gracia and Cornelis W. Oosterlee. A highly efficient Shannon wavelet inverse Fourier technique for pricing european options. *SIAM J. Sci. Comput*, 38(1):B118–B143, 2016.

[15] E. Pardoux and S. G. Peng. Adapted solution of a backward stochastic differential equation. *Syst Control Lett*, 14(1):55 – 61, 1990.

[16] E. Pardoux and S. G. Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In Boris L. Rozovskii and Richard B. Sowers, editors, *Stochastic Partial Differential Equations and Their Applications: Proceedings of IFIP WG 7/1 International Conference University of North Carolina at Charlotte, NC June 6–8, 1991*, pages 200–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[17] Allan Pinkus and Samy Zafrany. *Fourier Series and Integral Transforms*. Cambridge University Press, 1997. Cambridge Books Online.

[18] M. J. Ruijter and C. W. Oosterlee. A fourier cosine method for an efficient computation of solutions to BSDEs. *SIAM J. Sci. Comput*, 37(2):A859–A889, 2015.

[19] Weidong Zhao, Yu Fu, and Tao Zhou. New kinds of high-order multistep schemes for coupled forward backward stochastic differential equations. *SIAM J. Sci. Comput*, 36(4):A1731–A1751, 2014.

[20] Weidong Zhao, Yang Li, and Guannan Zhang. A generalized θ-scheme for solving backward stochastic differential equations. *Discrete Continuous Dyn Syst Ser B*, 17(5):1585–1603, 2012.

A Appendix

A.1 Pointwise convergence of orthogonal projection

We shall demonstrate that under some mild assumptions on a square integrable function in $(-2^{-m}J, 2^{-m}J]$, its orthogonal projection on V_J converges to the original function in a pointwise fashion, therefore bounding our approximation error. It is an adaptation of the standard Dirichlet kernel argument to our setting, a similar proof can be found in standard Fourier series textbook, like [17].

Theorem A.1. *Let g be a square integrable function defined on the set $(-2^{-m}J, 2^{-m}J]$ and the left- and right-side derivatives exist everywhere for its alternating extension $\tilde{g}$. If $\tilde{g}$ is continuous in a neighborhood around point x, the following result holds:*

$$\lim_{J\to\infty} H_{V_J} g(x) = \tilde{g}(x).$$

Proof. By direct calculation,

$$\begin{aligned}
&H_{V_J} g(x)\\
=&\frac{2^m}{J}\sum_{k=1}^{J}\left(\int_{-2^{-m}J}^{2^{-m}J} g(\varsigma)\cos\left(2^m C_k\varsigma\right)d\varsigma\cos\left(2^m C_k x\right)+\int_{-2^{-m}J}^{2^{-m}J} g(\varsigma)\sin\left(2^m C_k\varsigma\right)d\varsigma\sin\left(2^m C_k x\right)\right)\\
=&\frac{2^m}{J}\sum_{k=1}^{J}\left(\int_{-2^{-m}J}^{2^{-m}J}\tilde{g}(\varsigma)\cos\left(2^m C_k(\varsigma-x)\right)d\varsigma\right)=\frac{2^m}{J}\sum_{k=1}^{J}\left(\int_{-2^{-m}J}^{2^{-m}J}\tilde{g}(\varpi+x)\cos\left(2^m C_k\varpi\right)d\varpi\right)\\
=&\frac{2^m}{J}\int_{-2^{-m}J}^{2^{-m}J}\tilde{g}(\varpi+x)\Upsilon_J(\varpi)d\varpi, \qquad (A.1)
\end{aligned}$$

where

$$\Upsilon_J(x) := \sum_{k=1}^{J}\cos(2^m C_k x) = \frac{\sin(2^{m-1}\pi x)}{\sin(2^{m-1}\pi x/J)}. \qquad (A.2)$$

It can be shown that

$$\frac{2^m}{J}\int_0^{2^{-m}J}\Upsilon_J(\varsigma)d\varsigma = \frac{2^m}{J}\int_{-2^{-m}J}^{0}\Upsilon_J(\varsigma)d\varsigma = \frac{2}{\pi}\sum_{k=1}^{J}\frac{(-1)^{k+1}}{2k-1} =: \frac{2}{\pi}G_J. \qquad (A.3)$$

In fact, G_J is the famous Gregory-Leibniz series and we have $\lim_{J\to\infty} G_J = \frac{\pi}{4}$.

Based on our assumption, at point x, $\lim_{h\to 0+}\tilde{g}(x+h) = \lim_{h\to 0-}\tilde{g}(x+h) = \tilde{g}(x)$. Also, $\frac{g(s+x)-g(x)}{s}\mathbb{1}_{\{s\in(0,2^{-m}J)\}}$ and $\frac{g(s+x)-g(x)}{s}\mathbb{1}_{\{s\in(-2^{-m}J,0)\}}$ are integrable functions on $(2^{-m}J, 2^{-m}J]$. Note that in our construction, $2^{-m}J$ is a positive constant (a) and m is a function of J.

Therefore,

$$
\begin{aligned}
&H_{V_J}g(x) - \frac{4}{\pi}G_J\tilde{g}(x)\\
=&\frac{2^m}{J}\int_0^a (\tilde{g}(\varpi+x)-\tilde{g}(x))\Upsilon_J(\varpi)d\varpi + \frac{2^m}{J}\int_{-a}^0 (\tilde{g}(\varpi+x)-\tilde{g}(x))\Upsilon_J(\varpi)d\varpi\\
=&\frac{2}{\pi}\int_0^a \frac{\tilde{g}(\varpi+x)-\tilde{g}(x)}{\varpi}\frac{\pi\varpi/2a}{\sin(\pi\varpi/2a)}\sin(2^{m-1}\pi\varpi)d\varpi\\
&+\frac{2}{\pi}\int_{-a}^0 \frac{\tilde{g}(\varpi+x)-\tilde{g}(x)}{\varpi}\frac{\pi\varpi/2a}{\sin(\pi\varpi/2a)}\sin(2^{m-1}\pi\varpi)d\varpi. \qquad \text{(A.4)}
\end{aligned}
$$

Since $\frac{x}{\sin(x)}$ is integrable on $[-\frac{\pi}{2},\frac{\pi}{2}]$ and m tends to infinity whenever J tends to infinity, the last two terms go to 0, when J tends to infinity. This is due to the Riemann-Lebesgue Lemma. So, we have

$$\lim_{J\to\infty} H_{V_J}g(x) = \lim_{J\to\infty}\frac{4}{\pi}G_J\tilde{g}(x) = \tilde{g}(x), \qquad \text{(A.5)}$$

and complete the proof. □

A.2 Proof of Equations (4.6) and (4.7)

Proof. Using Equations (2.3b), (4.1), (4.4), (4.3), and standing assumption (A3), we have

$$
\begin{aligned}
&|\hat{z}_p^\Delta(x) - z_p^\Delta(x)|\\
\le&\frac{1-\theta_2}{\theta_2}|\mathbb{E}_p^x[z_{p+1}^\Delta(X_{t_{p+1}}^\Delta)] - \hat{\mathbb{E}}[z_{p+1}^\Delta(X_{t_{p+1}}^\Delta)|X_{t_p}^\Delta = x]|\\
+&\frac{1}{\theta_2\Delta t}|\mathbb{E}_p^x[y_{p+1}^\Delta(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}] - \hat{\mathbb{E}}[y_{p+1}^\Delta(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}|X_{t_p}^\Delta = x]|\\
+&\frac{1-\theta_2}{\theta_2}|\mathbb{E}_p^x[f(t_{p+1},X_{t_{p+1}}^\Delta,y_{p+1}^\Delta(X_{t_{p+1}}^\Delta),z_{p+1}^\Delta(X_{t_{p+1}}^\Delta))\Delta\omega_{p+1}]\\
&\qquad - \hat{\mathbb{E}}[f(t_{p+1},X_{t_{p+1}}^\Delta,y_{p+1}^\Delta(X_{t_{p+1}}^\Delta),z_{p+1}^\Delta(X_{t_{p+1}}^\Delta))\Delta\omega_{p+1}|X_{t_p}^\Delta = x]|\\
\le&\frac{1-\theta_2}{\theta_2}|\zeta_{z_{p+1}^\Delta}^{m,J}(t_p,x)| + \frac{1}{\theta_2\Delta t}|\zeta_{y_{p+1}^\Delta}^{m,J,\omega}(t_p,x)| + \frac{1-\theta_2}{\theta_2}|\zeta_{f_{p+1}}^{m,J,\omega}(t_p,x)|\\
+&\frac{1-\theta_2}{\theta_2}\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \rho_{z_{p+1}^\Delta}\left(\frac{r}{2^m}\right)\right||\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]|\\
+&\frac{1}{\theta_2\Delta t}\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \rho_{y_{p+1}^\Delta}\left(\frac{r}{2^m}\right)\right||\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|\\
+&\frac{1-\theta_2}{\theta_2}\frac{1}{J}\sum_{r=1-J}^{J}\left|f\left(t_{p+1},\frac{r}{2^m},y_{p+1}^\Delta\left(\frac{r}{2^m}\right),z_{p+1}^\Delta\left(\frac{r}{2^m}\right)\right) - \rho_{f_{p+1}}\left(\frac{r}{2^m}\right)\right||\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|\\
\le&\frac{1-\theta_2}{\theta_2}|\zeta_{z_{p+1}^\Delta}^{m,J}(t_p,x)| + \frac{1}{\theta_2\Delta t}|\zeta_{y_{p+1}^\Delta}^{m,J,\omega}(t_p,x)| + \frac{1-\theta_2}{\theta_2}|\zeta_{f_{p+1}}^{m,J,\omega}(t_p,x)|\\
+&\frac{(1-\theta_2)(1+M)}{\theta_2}\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^\Delta\left(\frac{r}{2^m}\right) - \rho_{z_{p+1}^\Delta}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)]| + \mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^\Delta)\Delta\omega_{p+1}]|)
\end{aligned}
$$

$$+\left(\frac{1}{\theta_2\Delta t}+\frac{(1-\theta_2)M}{\theta_2}\right)\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|$$

Note that we have added some extra terms in the last inequality to simplify the expression. Taking the maximum over $\left\{\frac{(1-\theta_2)(1+M)}{\theta_2},\frac{1}{\theta_2\Delta t}+\frac{(1-\theta_2)M}{\theta_2}\right\}$ finishes the proof for Equation (4.6).

Using Equations (4.5), (4.6), (3.5), (3.7), (4.1),(4.3), and standing assumption (A3), we have

$$\begin{aligned}
&|\hat{y}_p^{\Delta,I}-y_p^{\Delta}(x)|\\
\leq&\frac{1+\xi}{1-\xi}\varepsilon_p^{Picard}+\frac{\xi}{1-\xi}|\hat{z}_p^{\Delta}(x)-z_p^{\Delta}(x)|+\frac{1}{1-\xi}|\hat{h}(t_p,x)-h(t_p,x)|\\
\leq&\frac{1+\xi}{1-\xi}\varepsilon_p^{Picard}+\frac{M_1\cdot\xi}{1-\xi}(|\zeta^{m,J}_{z_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{y_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{f_{p+1}}(t_p,x)|)\\
+&\frac{M_1\cdot\xi}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{z_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|)\\
+&\frac{M_1\cdot\xi}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|\\
+&\frac{1}{1-\xi}|\mathbb{E}_p^x[y_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta})]-\hat{\mathbb{E}}[y_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta})|X_{t_p}^{\Delta}=x]|\\
+&\frac{\Delta t(1-\theta_1)}{1-\xi}|\mathbb{E}_p^x[f(t_{p+1},X_{t_{p+1}}^{\Delta},y_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta}),z_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta})]-\hat{\mathbb{E}}[f(t_{p+1},X_{t_{p+1}}^{\Delta},y_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta}),z_{p+1}^{\Delta}(X_{t_{p+1}}^{\Delta}))|X_{t_p}^{\Delta}=x]|\\
\leq&\frac{1+\xi}{1-\xi}\varepsilon_p^{Picard}+\frac{M_1\cdot\xi}{1-\xi}(|\zeta^{m,J}_{z_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{y_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{f_{p+1}}(t_p,x)|)\\
+&\frac{1}{1-\xi}|\zeta^{m,J}_{y_{p+1}^{\Delta}}(t_p,x)|+\frac{\Delta t(1-\theta_1)}{1-\xi}|\zeta^{m,J}_{f_{p+1}}(t_p,x)|\\
+&\frac{M_1\cdot\xi}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{z_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|)\\
+&\frac{M_1\cdot\xi}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|\\
+&\frac{1}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|\\
+&\frac{\Delta t(1-\theta_1)}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|f\left(t_{p+1},\frac{r}{2^m},y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right),z_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)\right)-\rho_{f_{p+1}}\left(\frac{r}{2^m}\right)\right|\,|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|\\
\leq&\frac{1+\xi}{1-\xi}\varepsilon_p^{Picard}+\frac{M_1\cdot\xi}{1-\xi}(|\zeta^{m,J}_{z_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{y_{p+1}^{\Delta}}(t_p,x)|+|\zeta^{m,J,\omega}_{f_{p+1}}(t_p,x)|)\\
+&\frac{1}{1-\xi}|\zeta^{m,J}_{y_{p+1}^{\Delta}}(t_p,x)|+\frac{\Delta t(1-\theta_1)}{1-\xi}|\zeta^{m,J}_{f_{p+1}}(t_p,x)|\\
+&\frac{M_1\cdot\xi+M\Delta t(1-\theta_1))}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{z_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|)
\end{aligned}$$

$$+\frac{M_1\cdot\xi+1+M\Delta t(1-\theta_1)}{1-\xi}\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|).$$

This concludes the proof if $M_2 := \frac{M_1\cdot\xi+1+M\Delta t(1-\theta_1)}{1-\xi}$ and $M_3 := \frac{1+\xi}{1-\xi}\frac{\varepsilon_p^{Picard}}{M_2}$. Again, extra terms are included in the expression to simplify the formula. □

A.3 A simple example for deriving the error formula

In this section, we would use the result in Section 4.4 to derive an error formula for the approximation of Example 2 in Section 6.2. In addition to the parameters provided in Section 6.2, we let $\theta_1 = 0$, $\theta_2 = 1$ and $P = 2$. Note that $P = 2$ is merely used here to simplify our expression.

It is clear that driver function f and terminal function g satisfy all standing assumptions with the Lipschitz coefficient of f with respect to y and z being 0.4. We have $\Delta t = 0.05$, $\xi = \varepsilon_p^{Picard} = 0$ for $p = 0, 1$ in our setting. Using the derivation in Appendix B, Equations (4.6) and (4.7) can be simplified as follows:

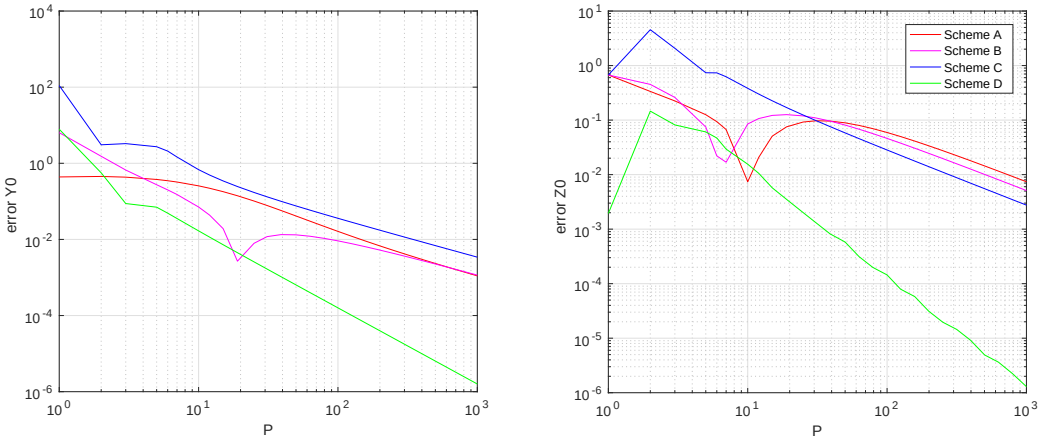
$$\frac{1}{20}|\hat{z}_p^{\Delta}(x)-z_p^{\Delta}(x)|\leq|\zeta_{y_{p+1}^{\Delta}}^{m,J,\omega}(t_p,x)|+\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right||\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|$$

$$\begin{aligned}\frac{1}{1.02}|\hat{y}_p^{\Delta,I}(x)-y_p^{\Delta}(x)|\leq&|\zeta_{y_{p+1}^{\Delta}}^{m,J}(t_p,x)|+|\zeta_{f_{p+1}}^{m,J}(t_p,x)|\\&+\frac{1}{J}\sum_{r=1-J}^{J}\left|z_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{z_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|)\\&+\frac{1}{J}\sum_{r=1-J}^{J}\left|y_{p+1}^{\Delta}\left(\frac{r}{2^m}\right)-\rho_{y_{p+1}^{\Delta}}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})]|+\mathbb{E}_p^x[\varphi_{J,r}(X_{t_{p+1}}^{\Delta})\Delta\omega_{p+1}]|),\end{aligned}$$

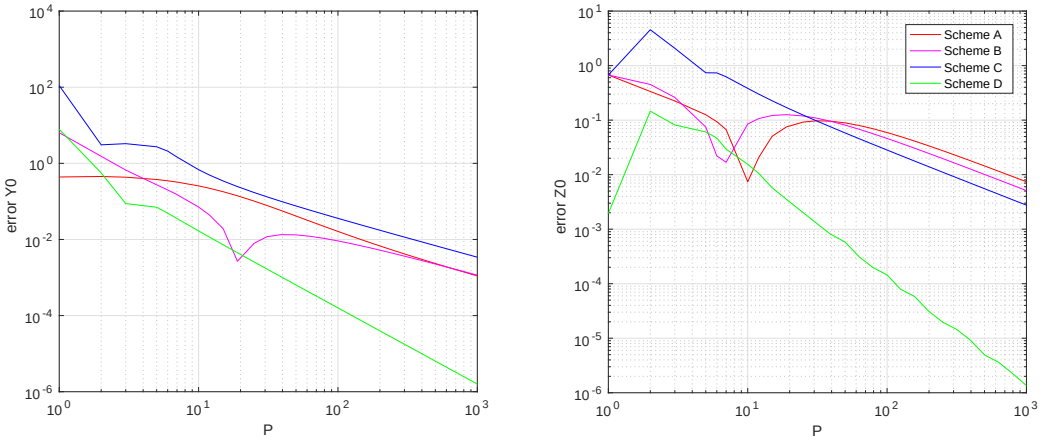
for $p = 0, 1$. Therefore, the error of applying the quick SWIFT scheme to the discretizated system reads:

$$\begin{aligned}&|\hat{y}_0^{\Delta,I}(x_0)-y_1^{\Delta}(x_0)|\\\leq&1.02|\zeta_{y_1^{\Delta}}^{m,J}(0,x_0)|+1.02|\zeta_{f_1}^{m,J}(0,x_0)|\\&+\frac{1.02}{J}\sum_{r=1-J}^{J}\left|z_1^{\Delta}\left(\frac{r}{2^m}\right)-\hat{z}_1^{\Delta}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})]|+\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})\Delta\omega_1]|)\\&+\frac{1.02}{J}\sum_{r=1-J}^{J}\left|y_1^{\Delta}\left(\frac{r}{2^m}\right)-\hat{y}_1^{\Delta,I}\left(\frac{r}{2^m}\right)\right|(|\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})]|+\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})\Delta\omega_1]|)\\\leq&1.02|\zeta_{y_1^{\Delta}}^{m,J}(0,x_0)|+1.02|\zeta_{f_1}^{m,J}(0,x_0)|\\&+\frac{1}{J}\sum_{r=1-J}^{J}\left(20.4\left|\zeta_{y_2^{\Delta}}^{m,J,\omega}\left(t_1,\frac{r}{2^m}\right)\right|+1.02^2\left|\zeta_{y_2^{\Delta}}^{m,J}\left(t_1,\frac{r}{2^m}\right)\right|+1.02^2\left|\zeta_{f_2}^{m,J}\left(t_1,\frac{r}{2^m}\right)\right|\right)\\&\qquad(|\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})]|+\mathbb{E}_0^{x_0}[\varphi_{J,r}(X_{t_1}^{\Delta})\Delta\omega_1]|).\end{aligned}$$

Note that there is no recurring error at t_2 as we know the exact terminal condition.

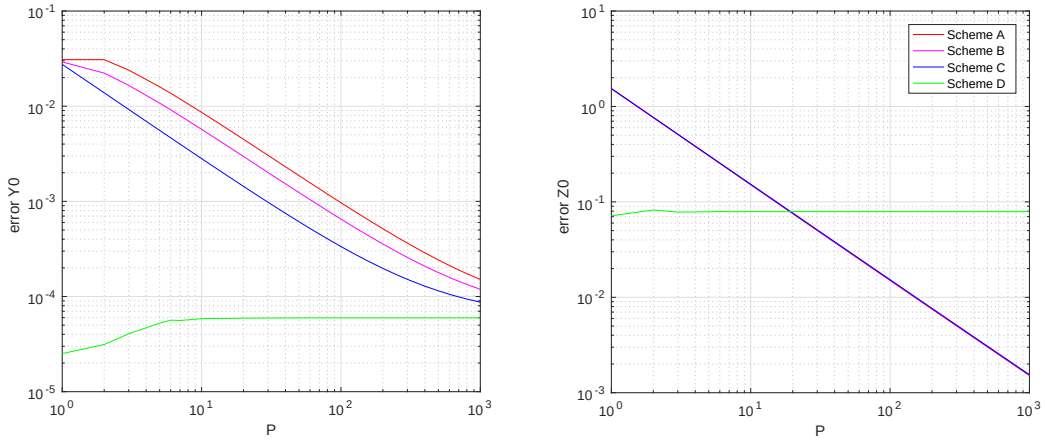


(a) Quick SWIFT with $J = 2^9$

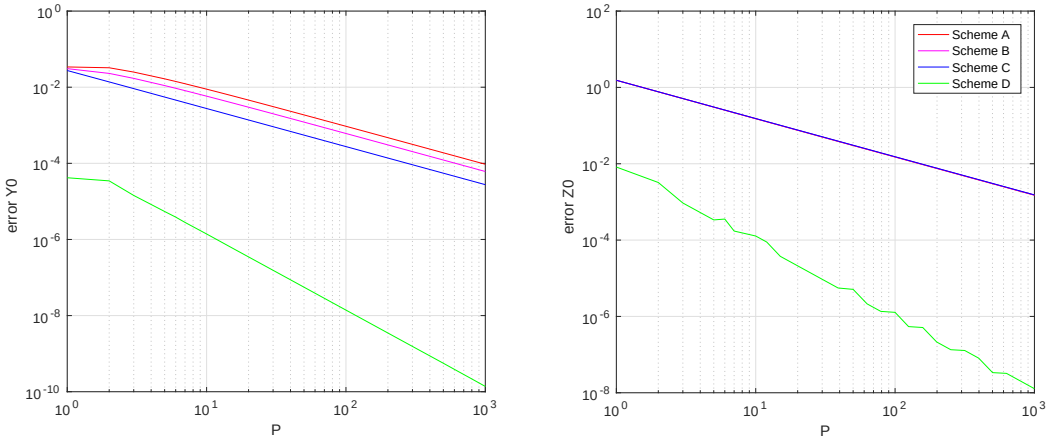


(b) Mixed SWIFT with $J = 2^9$

Figure 2: Results example 1, left: error in $y(0, x_0)$, right: error in $z(0, x_0)$



(a) Quick SWIFT with $J = 2^9$



(b) Mixed SWIFT with $J = 2^9$

Figure 3: Results example 2, left: error in $y(0, x_0)$, right: error in $z(0, x_0)$

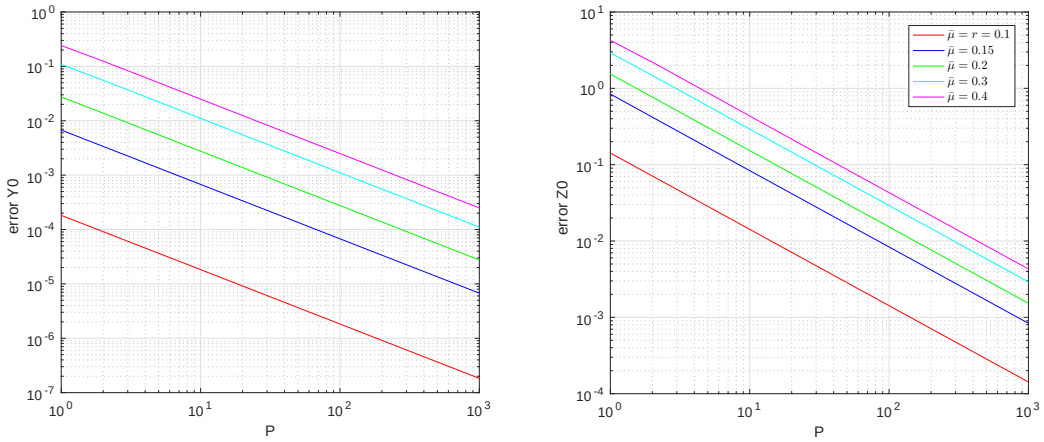


Figure 4: Results example 2 for different values of $\bar{\mu}$ (Scheme C)

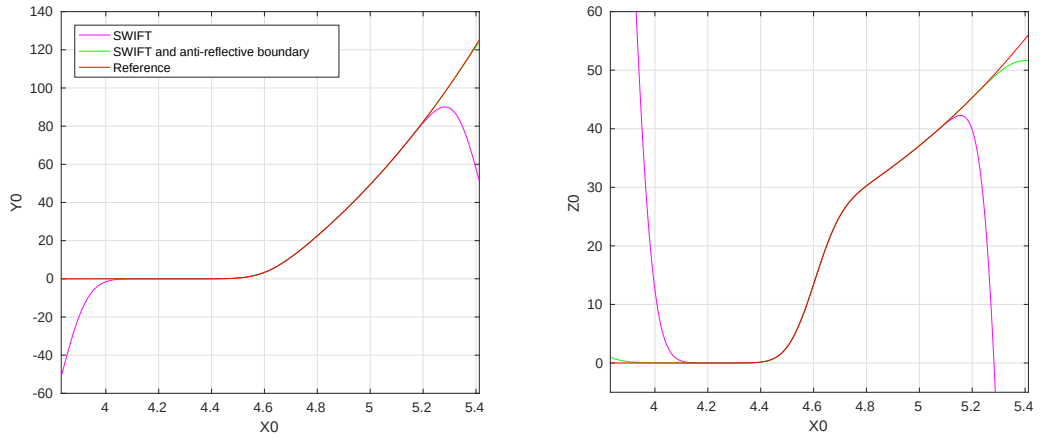
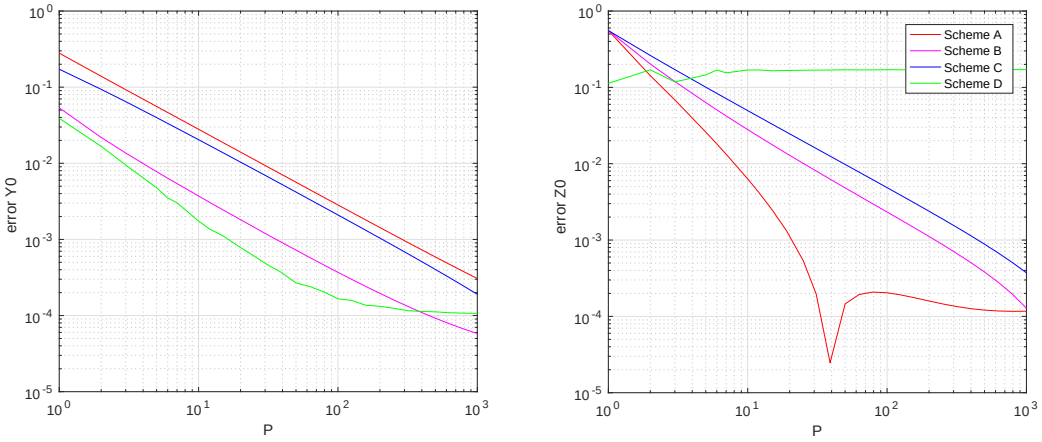
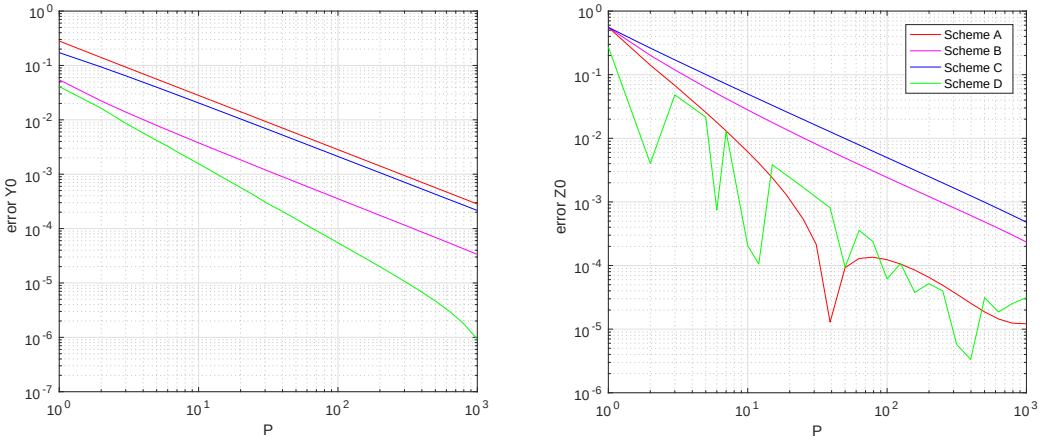


Figure 5: Results example 2 with and without applying anti-reflective boundary technique (Scheme D, $P = 1000$ and $J = 2^9$)



(a) Quick SWIFT with $J = 2^9$



(b) Mixed SWIFT with $J = 2^9$

Figure 6: Results example 3, left: error in $y(0, x_0)$, right: error in $z(0, x_0)$

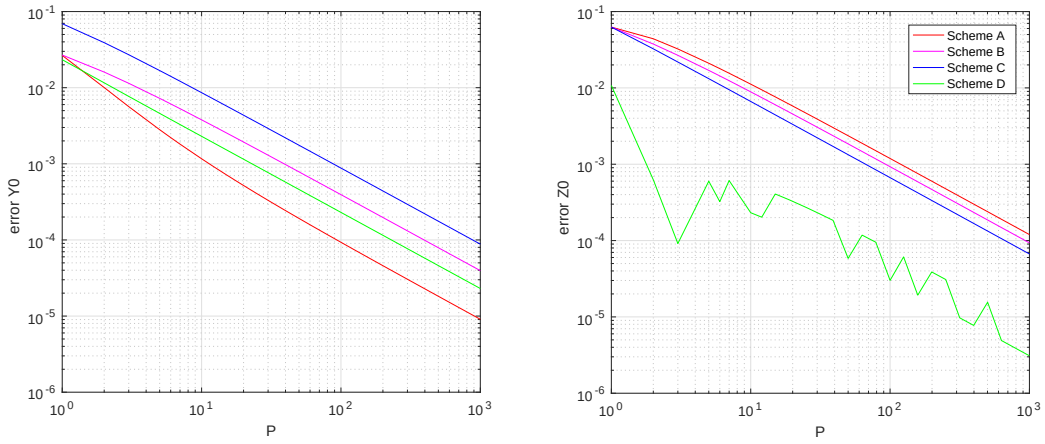


Figure 7: Results example 4, ($J = 2^9$), left: error in $y(0, x_0)$, right: error in $z(0, x_0)$