Solving high-dimensional nonlinear filtering problems using a tensor train decomposition method

Sijing Li[a], Zhongjian Wang[a], Stephen S.T. Yau[b,*], Zhiwen Zhang[a,*]

[a]*Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China.*
[b]*Department of Mathematics, Tsinghua University, Beijing 100084, China.*

Abstract

In [19], Yau and Yau developed a novel algorithm to solve the path-wise robust DMZ equation, which laid down a solid foundation to study the nonlinear filtering problem (NLF). Since then many efforts have been devoted to solve the path-wise robust DMZ equation. However, it is still very challenging to solve high-dimensional problem in an effective fashion. In this paper, we propose an efficient numerical method to solve high-dimensional NLF problems. Specifically, we use the tensor train decomposition method to solve the forward Kolmogorov equation (FKE) arising from the NLF problem. We explore the low-dimensional structures in the solution space of the FKE and adopt the tensor train decomposition method to solve the FKE. Our method consists of offline and online stages. In the offline stage, we discrete the PDE operators involved in the FKE and approximate them using the tensor train method. Moreover, we approximate the evolution of the FKE operator using the tensor train method. In the online stage using the pre-computed low-rank approximation tensors, we can quickly solve the FKE given the new observation data. Therefore, we can solve the NLF problem in a real-time manner. Under some mild assumptions, we prove the convergence and effectiveness analysis of the proposed method. Finally, we design two online numerical experiments to show the efficiency and accuracy of the proposed method.

AMS subject classification: 15A69, 35R60, 60G35, 65M99.

Keywords: nonlinear filtering problems; Forward Kolmogorov equations; Duncan-Mortensen-Zakai (DMZ) equation; tensor train decomposition; real-time algorithm.

1. Introduction

Nonlinear filtering (NLF) problem is originated from the problem of tracking and signal processing. The fundamental problem in the NLF is to give the instantaneous and accurate estimation of the states based on the noisy observations [10]. In this paper, we consider the

*Corresponding author

Email addresses: lsj17@hku.hk (Sijing Li), ariswang@connect.hku.hk (Zhongjian Wang), yau@tsinghua.edu.cn (Stephen S.T. Yau), zhangzw@hku.hk (Zhiwen Zhang)

signal based nonlinear filtering problems as follows,

$$\begin{cases} dx_t = f(x_t, t)dt + g(x_t, t)dv_t, \\ dy_t = h(x_t, t)dt + dw_t, \end{cases} \tag{1}$$

where $x_t \in R^n$ is the states of the system at time t and $y_t \in R^m$ is the observations at time t, and v_t and w_t are vector Brownian motion processes. Some growth conditions on f, g and h are required to guarantee the existence and well-posedness of the NLF problems, which will be discussed later.

Particle filter method is the most popular method to solve (1); see [1, 7, 2] and references therein. However, the main drawback of the particle filter method is that it is hard to be implemented in a real-time manner due to its nature of the Monte Carlo simulation. In practice, the real-time manner means the running time of the numerical integrator in solving the state equation for x_t is much less than the time between any two observations of y_t.

Alternatively, one can solve the Duncan-Mortensen-Zakai (DMZ) equation, also known as Zakai equation, to study the NLF problems [5, 15, 21]. The DMZ equation computed the unnormalized conditional density function of the states x_t, which provides a powerful tool to study the NLF problem since one can estimate the statistical quantities of the states x_t based on the DMZ solution. In general, one cannot solve the DMZ equation analytically. Many efforts have been made to develop efficient numerical methods; see [16, 9, 8, 4, 12] and the references therein.

The DMZ equation allows one to study the statistical quantities of the states x_t. In practice, however, one can only get one realization of the states x_t (instead of thousands of repeated experiments), which motivates researchers to develop robust methods in solving the DMZ equation. Namely, the robust method should not be sensitive to the given observation paths. A novel algorithm was proposed to solve the path-wise robust DMZ equation [19]. In this approach, for each realization of the observation process denoted by y_t, one can make an invertible exponential transformation and transform the DMZ equation into a deterministic partial differential equation (PDE) with stochastic coefficient. Several efficient numerical methods were developed along this direction; see [13, 14, 18], which can be efficient when the dimension of the NLF problems is small. However, it becomes expensive as the dimension of the NLF problems increases.

In this paper, we propose to use the tensor train decomposition method to solve the high-dimensional FKEs. Our method consists of offline and online stages. In the offline stage, we discrete the PDE operators involved in the FKEs and approximate them using the tensor train method. Moreover, we approximate the evolution of the FKE operator using the tensor train method. In the online stage, we can quickly solve the FKE given new observation data using the pre-computed low-rank approximation tensors. By exploring the low-dimensional structures in the solution space of the FKE, we can solve the NLF problem in a real-time manner. Under some mild assumptions, we prove the convergence and effectiveness analysis of the proposed method. Finally, we present two numerical experiments to show the efficiency and accuracy of the proposed method. We find that the tensor train method is scalable in

solving the NLF problem.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction of the NLF problem and DMZ equation. In Section 3, we introduce the basic idea of the tensor train decomposition method. In section 4, we propose our fast method to compute the high-dimensional FKEs. More details about the offline and online computing will be discussed. Some convergence analysis and computational complexity of the proposed method will also be discussed. In Section 5, we present numerical results to demonstrate the accuracy and efficiency of our method. Concluding remarks are made in Section 6.

2. Some basic results of the NLF problems

In this section, we shall introduce some basic results of the NLF problems. To start with, we consider the signal based model as follows,

$$\begin{cases} dx_t = f(x_t, t)dt + g(x_t, t)dv_t, \\ dy_t = h(x_t, t)dt + dw_t, \end{cases} \tag{2}$$

where $x_t \in R^n$ is a vector of the states of the system at time t, the initial state x_0 satisfying an initial distribution, $y_t \in R^m$ is a vector of the observations at time t with y_0, and v_t and w_t are vector Brownian motion processes with covariance matrices $E[dv_t dv_t^T] = Q(t)dt \in R^{n\times n}$ and $E[dw_t dw_t^T] = S(t)dt \in R^{m\times m}$, respectively. Moreover, x_0, dw_t and dv_t are assumed to be independent. Some growth conditions on f and h are required to guarantee the existence and uniqueness of the pathwise-robust DMZ equation [19]. In this paper, f, h, and g are C^2 in spatial variable and C^1 in the temporal variable

The most popular method so far to solve (2) is the particle filter, see [1, 7, 2] and references therein. However, the main drawback of the particle filter method is that it is hard to be implemented in a real-time manner due to its nature of the Monte Carlo simulation.

The DMZ equation or Zakai equation [5, 15, 21] asserts that the unnormalized conditional density function of the states x_t, denoted by $\sigma(x, t)$, satisfies the following stochastic partial differential equation (SPDE):

$$\begin{cases} d\sigma(x, t) = \mathcal{L}\sigma(x, t)dt + \sigma(x, t)h^T(x, t)S^{-1}dy_t, \\ \sigma(x, 0) = \sigma_0(x), \end{cases} \tag{3}$$

where $\sigma_0(x)$ is the density of the initial states x_0, and

$$\mathcal{L}(\cdot) := \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}\big((gQg^T)_{ij}\,\cdot\,\big) - \sum_{i=1}^{n} \frac{\partial(f_i\cdot)}{\partial x_i} \tag{4}$$

The DMZ equation laid down a solid foundation to study the NLF problem. However, one cannot solve the DMZ equation analytically in general. Many efforts have been made to develop efficient numerical methods. One of the commonly used method is the splitting-up method originated from the Trotter product formula, which was first introduced in [4] and

has been extensively studied later, see [16, 9, 8]. In [12], the so-called S^3 algorithm was developed based on the Wiener chaos expansion. By separating the computations involving the observations from those dealing only with the system parameters, this approach gives rise to a new numerical scheme for NLF problems. However, the limitation of their method is that the drifting term f and the observation term h in (2) should be bounded.

To overcome this restriction, Yau and Yau [19] developed a novel algorithm to solve the path-wise robust DMZ equation. Specifically, for each realization of the observation process denoted by y_t, they make an invertible exponential transformation

$$\sigma(x,t) = \exp\big(h^T(x,t)S^{-1}(t)y_t\big)u(x,t), \tag{5}$$

and transform the DMZ equation (3) into a deterministic partial differential equation (PDE) with stochastic coefficient

$$\begin{cases} \dfrac{\partial}{\partial t}u(x,t) + \dfrac{\partial}{\partial t}(h^T S^{-1})y_t u(x,t) = \\ \qquad\qquad \exp\big(-h^T(x,t)S^{-1}(t)y_t\big)\big(\mathcal{L} - \dfrac{1}{2}h^T S^{-1}h\big)\big(\exp\big(-h^T(x,t)S^{-1}(t)y_t\big)u(x,t)\big), \\ u(x,0) = \sigma_0(x). \end{cases} \tag{6}$$

Equation (6) is called the pathwise robust DMZ equation [12, 19]. Compared with the DMZ equation (3), the pathwise robust DMZ equation (6) is easier to solve, since the stochastic term has been transformed into the coefficients.

The existence and uniqueness of (6) has been investigated by many researchers. The well-posedness is guaranteed when the drift term $f \in C^1$ and the observation term $h \in C^2$ are bounded in [17]. Later on, similar results can be obtained under weaker conditions. For instance, the well-posedness results on the pathwise-robust DMZ equation with a class of unbounded coefficients were obtained in [3, 6], but the results were for one-dimensional case. In [19], the third author of this paper and his collaborator established the well-posedness result under the condition that f and g have at most linear growth. In [13], a well-posedness result was obtained for time-dependent pathwise-robust DMZ equation under some mild growth conditions on f and h.

Let us assume that the observation time sequences $0 = t_0 < t_1 < \cdots < t_{N_t} = T$ are given. In each time interval $t_{j-1} \le t < t_j$, one freezes the stochastic coefficient y_t to be $y_{t_{i-1}}$ in Eq.(6) and makes the exponential transformation

$$u_j(x,t) = \exp\big(h^T(x,t)S^{-1}(t)y_{t_{j-1}}\big)u(x,t). \tag{7}$$

It is easy to deduce that u_j satisfies the FKE

$$\frac{\partial}{\partial t}u_j(x,t) = \big(\mathcal{L} - \frac{1}{2}h^T S^{-1}h\big)u_j(x,t), \tag{8}$$

where the operator $\mathcal{L}$ is defined in (4). In [14], Luo and Yau investigated the Hermite spectral method to numerically solve the 1d FKE (8) and analyzed the convergence rate

of the proposed method. In their algorithm, the main idea is to shift part of the heavy computations off-line, so that only computations involved observations are performed on-line and synchronized with off-line data. The numerical method based on the Hermite polynomial approximation is efficient though, it is extremely hard to extend to solve high-dimensional FKEs in the real-time manner, since the number of the Hermite polynomial basis functions grows fast for high-dimensional problems. Namely, it suffers from the curse of dimensionality.

In a very recent result [18], we proposed to use the proper orthogonal decomposition (POD) method to numerically solve the 2D FKE. By exploring the low-dimensional structures in the solution space of the FKE and building POD basis, our method provides considerable savings over the Hermite polynomial approximation method that was used in [14]. The POD method helps us alleviate the curse of dimensionality to a certain extent though, it is still very challenging to solve high-dimensional NFL problems. The reason is that in the POD method one needs solution snapshots to construct POD basis. However, to obtain solution snapshots is extremely expensive for high-dimensional NFL problems. We shall address this challenge using the Tensor Train decomposition method in this paper.

3. The Tensor Train decomposition method

We shall introduce the tensor train (TT) decomposition method for approximating solutions of high-dimensional NLF problems. Let us assume the dimension of the NLF problem is d. For any fixed time t, if we discretize the solution $u(x,t)$, $x \in R^d$ of the FKE (8) using numerical methods, such as finite difference method, we obtain a d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ tensor $U(i_1, i_2, \cdots, i_d)$, which is a multidimensional array. The number of unknowns in this representation grows fast as d increases and is subject to the curse of dimensionality. To attack this challenge, one should explore potential low-dimensional structures in the tensor and approximate the tensor in a certain data-sparse way.

The TT decomposition method is a recently developed method for tensor approximation [22]. A brief introduction of the TT-format is given below. If a d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ tensor $U(i_1, i_2, \cdots, i_d)$ can be written as the element-wise form

$$U(i_1, i_2, \cdots, i_d) = G_1(i_1) G_2(i_2) \cdots G_d(i_d), \quad 1 \le i_k \le n_k, \tag{9}$$

where $G_k(i_k)$ is a $r_{k-1} \times r_k$ matrix for each fixed i_k, $1 \le k \le d$ and $r_0 = r_d = 1$. We call the tensor U is in the TT-format, if it is represented in the form (9). Furthermore, each element G_k can be regarded as an 3-dimensional tensor of the size $r_{k-1} \times n_k \times r_k$. In the representation (9), $G_1, G_2, \cdots, G_d$ are called the cores of the TT-format tensor U, numbers r_k are called TT-ranks, and numbers $n_1, n_2, \cdots, n_d$ are called mode sizes. With these definitions, the representation (9) can be rewritten as

$$U(i_1, i_2, \cdots, i_d) = \sum_{\alpha_1, \cdots, \alpha_{d-1}} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d, \alpha_d) \tag{10}$$

where $\alpha_0 = \alpha_d = 1, 1 \le \alpha_k \le r_k$ for $1 \le k \le d-1$. In practice, one only needs to store all the cores G_k in the TT-format, in order to save a tensor. Thus, if all the TT-ranks r_k are bounded by a constant r and the mode sizes n_k are bounded by n, the storage of the d-dimensional tensor U is $\mathcal{O}(dnr^2)$ in the TT-format. Recall that the storage of the tensor U is about $\mathcal{O}(n^d)$, if no approximation is used.

To further reduce the storage of the TT-format, a quantized tensor train (QTT) format was introduced in [23, 24, 25]. The QTT format is derived by introducing virtual dimensions along each real dimension of a tensor. Specifically, suppose each one-dimensional size of the tensor U is a power of 2, i.e. $n_1 = n_2 = \cdots = n_d = 2^L$. The d-dimensional tensor U can be reshaped to a D-dimensional tensor with $D = dL$, while each mode size is equal to 2. The QTT-format is the TT-format of the reshaped tensor, which has a larger number of dimension but much smaller mode sizes (here is 2) than the TT-format. The concepts of cores, QTT-ranks and mode sizes (all are equal to 2) of QTT-format are defined similarly as the TT-format. The storage of the QTT-format is further reduced to $\mathcal{O}(d\log(n)r^2)$.

One can also simply formulate the TT-format of d-dimensional matrices [23][25]. The TT-format of a d-dimensional $(m_1 \times \cdots \times m_d) \times (n_1 \times \cdots \times n_d)$ matrix B can be written as

$$B(i_1, \cdots, i_d, j_1, \cdots, j_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1(1, i_1, j_1, \alpha_1) G_2(\alpha_1, i_2, j_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d, j_d, 1), \tag{11}$$

where the index 1 in G_1 and G_d is due to the TT-ranks $\alpha_0 = \alpha_d = 1$. The definitions of cores, ranks and mode sizes are similar as tensors. The QTT representation of matrices are preferred in this paper.

Simple calculations show that in the TT-format the computational complexity of addition (together with TT-rounding after addition) is $\mathcal{O}(dnr^3)$, matrix-by-vector multiplication is $\mathcal{O}(dn^2r^4)$, and Hadamard product is $\mathcal{O}(dnr^4)$. Therefore, the TT/QTT-format allows lower complexity of algebraic operation than dense matrix or tensor form. In practice, especially in solving time-dependent problems, one needs to apply the TT-rounding procedure in the computation. The computational complexity of the TT-rounding is $\mathcal{O}(dnr^3)$. The purpose of TT-rounding is to decrease TT/QTT-rank of a matrix or tensor already in the TT/QTT-format while preserving a given accuracy ϵ. The QTT-format can even reduce the factor n of complexity to $\log(n)$.

4. The fast algorithm and its implementation

In this section, we present the fast algorithm to solve the NLF problem. Let us assume that the observation time sequences $0 = t_0 < t_1 < \cdots < t_{N_t} = T$ are given. But the observation data $\{y_{t_i}\}$ at each observation time t_i, $i = 0, ..., N_t$ are unknown until the on-line experiment runs. For simplicity, we assume $t_i - t_{i-1} = \Delta T$. We shall study how to solve the FKE (8) within each time interval $[t_{i-1}, t_i]$ and compute the exponential transformation (7) using the QTT-format. If we use an explicit scheme to discretize the time derivative in the FKE (8),

we get

$$u_i^{k\tau}(x,t) = \Big\{\tau(\mathcal{L} - \frac{1}{2}h^T S^{-1}h) + I\Big\}u_i^{(k-1)\tau}(x,t), \quad k = 1, \cdots, \frac{\Delta T}{\tau}, \tag{12}$$

where τ is the time step in discretizing the FKE (8) and ΔT is the time interval between two sequential observations.

Our algorithm consists of an offline procedure and an online procedure. In the offline procedure, we compute the matrices of spatial discretization of operators in (12) and convert them into the QTT-format in advance, which will reduce the computational time in the online procedure. In the online procedure, we solving the FKE and update the solution with new observation data. We shall show that the online procedure can be finished in real time manner since all the computations are done in the QTT-format.

4.1. Spatial discretization and low-rank approximation

We shall discuss how to discretize the FKE (8) and represent the spatial discretization in the QTT-format. To simply the notaions and illustrate the main idea of our algorithm, we choose a squared domain and uniform grid. Specifically, we take a large $a > 0$ and let $[-a, a]^d$ denote physical domain of the FKE. An uniform grid is set on each dimension with $N = 2^L$ points and mesh size $h = \frac{2a}{(N-1)}$. We use $x_k(i_k)$, $i_k = 1, \cdots, N$ to record the coordinates of grid points on k-th dimension.

First, the Laplace operator in (8) is discretized by using a finite difference scheme. The resulting matrix is of the form

$$\Delta_d = \Delta_1 \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes \Delta_1, \tag{13}$$

where

$$\Delta_1 = \frac{1}{h^2}\text{tridiag}(1, -2, 1),$$

and I is an $N \times N$ identity matrix. In particular, when $d = 3$ the Laplace operator has the form

$$\Delta_3 = \Delta_1 \otimes I \otimes I + I \otimes \Delta_1 \otimes I + I \otimes I \otimes \Delta_1. \tag{14}$$

The QTT-format of matrix Δ_d has a low-rank representation that is bounded by 4 (see Corollary 5.3 of [23]).

Second, the convection operator $\sum_{i=1}^d \frac{\partial(f_i \cdot)}{\partial x_i}$ in (8) (see also Eq.(4)) is discretized by using a central difference scheme. Thus, the corresponding d-dimensional matrix has the form

$$C_d = (C \otimes I \otimes \cdots \otimes I)F_1 + (I \otimes C \otimes \cdots \otimes I)F_2 + \cdots + (I \otimes I \otimes \cdots \otimes C)F_d, \tag{15}$$

where F_k's are diagonal matrices associated with the diagonalization of tensor discretization of the drift functions f_k, i.e.,

$$F_k(i_1, i_2, \cdots, i_d, i_1, i_2, \cdots, i_d) = f_k(x_1(i_1), x_2(i_2), \cdots, x_d(i_d)),$$

for all $1 \leq k \leq d$, and C is an one-dimensional central difference operator,

$$C = \frac{1}{h}\text{tridiag}(-\frac{1}{2}, 0, \frac{1}{2}).$$

Under certain conditions for the drift terms f_k, the QTT-format of matrix C_d has a bounded low-rank representation. We summary the result into the following lemma and the proof can be found in [24].

Lemma 4.1. *Suppose that the QTT-ranks of the functions f_k on a tensor grid are bounded by r. Then, the QTT-rank of the matrix C_d in (15) is bounded by $5dr$.*

Although an exact TT-decomposition of any tensor is feasible [22], it rarely has a low-rank structure. Therefore, one should apply TT-rounding procedures in order to decrease the TT/QTT-ranks while preserving a given accuracy ϵ. Let us consider the drift terms f_k as an example. In order to construct the QTT-format of functions f_k with low ranks, one can use the TT-SVD algorithm [22]. QTT-ranks of these QTT-format tensors are guaranteed to be smaller than a prescribed approximation error ϵ in the sense of Frobenius norm [22].

Lemma 4.2 (see Theorem 2.2, [27])**.** *For any tensor A with size $n_1 \times n_k \times \cdots \times n_d$, there exists a tensor B in the TT-format with TT-ranks r_k such that*

$$||A - B||_F \leq \sqrt{\sum_{k=1}^{d-1} \epsilon_k^2},$$

where ϵ_k is the distance from A_k to its best rank-r_k approximation in the Frobenius norm,

$$\epsilon_k = \min_{\text{rank}C \leqslant r_k} ||A_k - C||_F,$$

where A_k is the k-th unfolding matrix of tensor A

$$A_k = \text{reshape}\left(A, \prod_{s=1}^{k} n_s, \prod_{s=k+1}^{d} n_s\right).$$

Remark 4.1. The Lemma 4.2 allows us to control the accuracy and TT/QTT-ranks, when we compute the approximation of any tensor in the TT/QTT-format.

Finally, the approximation of the function $h^T S^{-1} h$ in the FKE (8) in the QTT-format can be obtained using the same approach as f_k in the convection operator. Specifically, we discretize the function $h^T S^{-1} h$ on the spatial tensor grid and diagonalize it to a matrix. Then, we approximate it by a low-rank QTT-format using the TT-SVD algorithm.

4.2. The offline procedure

In the offline procedure, we first assemble the operators involved in the FKE, including the Laplace operator, the convection operator and the multiplication operator associated with the function $h^T S^{-1} h$, into a combined operator A. In this paper, we assume the drift and observation functions are time-independent. Thus, the numerical scheme (12) becomes

$$u_i^{k\tau}(x,t) = \big(\tau A + I\big) u_i^{(k-1)\tau}(x,t), \quad k = 1, \cdots, \frac{\Delta T}{\tau}. \tag{16}$$

Recall that the discretizations of the Laplace operator, the convection operator and the multiplication operator associated with the function $h^T S^{-1} h$ all have low-rank approximations. Moreover, addition of matrices or tensors in the QTT-format only causes addition of QTT-ranks. Therefore, the spatial discretization of the operator A has a low rank QTT-format approximation with given maximal QTT-rank r or with certain given precision ϵ in the sense of Frobenius norm.

Notice that in the NLF problem, there will be no observation available during the time period with length ΔT. Thus, we directly compute the operator $(\tau A + I)^{\frac{\Delta T}{\tau}}$ and approximate it in the QTT-format. Then, we rewrite the scheme (16) as

$$u_i^{\Delta T}(x,t) = (\tau A + I)^{\frac{\Delta T}{\tau}} u_i^0(x,t). \tag{17}$$

Exact addition of τA and I in the QTT format only increases the rank by one. However, exact multiplication of matrices in the QTT-format will lead to a significant growth of QTT-ranks. In our algorithm, we apply TT-rounding to control the growth of the QTT-rank caused by matrix-matrix multiplication, which can be easily achieved and maintain accuracy [22, 24].

4.3. Online procedure

In this section, we shall demonstrate that using the tensor train decomposition method and the precomputed time integration results we can achieve fast computing in the online stage.

At first, we set an initial probability density function according to initial state x_0, and solve the forward Kolmogorov equation with such initial condition. At each observing time t_j, when a new observation y_{t_j} arrives, we do the exponential transformation to get the initial condition of equation (8). We then solve the equation (8) by our algorithm (17). All of these operations are done in QTT-format, thus we need to do TT-rounding operation after both exponential transformation and solving (8).

Proposition 4.3. *Suppose the QTT-ranks of all functions required in online procedure on a tensor grid, including $u(x,t_i)$ and $\exp[h^T(x,t_i)S^{-1}(t_i)(y_{t_i} - y_{t_{i-1}})]$, are bounded by r. The accuracy ϵ of TT-rounding is properly specified to ensure QTT-ranks of $u(x,t)$ are also bounded by r after any TT-rounding procedure. Then the total complexity of online procedure are $\mathcal{O}(N^d r^2 + d\log(N) r^6)$.*

Proof. The complexity of constructing the QTT-format of the function $\exp[h^T(x,t_i)S^{-1}(t_i)(y_{t_i} - y_{t_{i-1}})]$ from a full multidimensional array is $\mathcal{O}(N^d r^2)$ by Theorem2.1 in [25]. The exponential

transformation is essentially a Hadamard product in the QTT-format whose complexity is $\mathcal{O}(d\log(N)r^4)$ [22]. Solving forward Kolmogorov equation (8) is practically a matrix-vector multiplication (17) in the QTT-format whose complexity is $\mathcal{O}(d\log(N)r^4)$ [22]. Requirement of TT-rounding through standard TT-SVD algorithm is $\mathcal{O}(d\log(N)r^6)$ [22]. □

Remark 4.2. A lower complexity of the QTT-format than the TT-format is obtained by introducing virtual dimensions along each real dimension. $d\log_2 N$ is the dimension of the QTT-format. The logarithmic dependence of QTT operations on the total degree of freedom N^d is the key of computational improvement.

Remark 4.3. Note that the term $N^d r^2$ is a result of construting a QTT-format tensor from a full tensor. In the case of 3-d problems of our experiments, it has less effect on the whole computation than the term $d\log(N)r^6$. Actually the TT-rounding does cost most computational time as a dominating term.

4.4. The complete algorithm of the NLF problem

In this subsection, we give the complete algorithm of the NLF problem. The on- and off-line computing stages in our algorithm are summarized in the Algorithm 1 and Algorithm 2, respectively. The performance of our numerical method will be demonstrated in Section 5.

Algorithm 1 Offline computing

1: Compute matrices of spatial discretization of operators mentioned in the Section 4.1, including the Laplace operator, i.e., Eq.(13), the convection operator, i.e., Eq.(15), and the multiplication operator associated with the function $h^T S^{-1} h$.
2: Convert these matrices into the QTT-format.
3: Compute the addition of operator matrices in the QTT-format by taking into account the time stepping τ, i.e. compute $\tau A + I$ in Eq.(16).
4: Compute the power of the operator $\tau A + I$ in the QTT-format, i.e. compute $(\tau A + I)^{\frac{\Delta T}{\tau}}$ in Eq.(17).

4.5. Some convergence analysis

Proposition 4.4. *Let δ_1 denote a given precision in the construction of QTT-format and TT-rounding, and δ_2 denote the error(in the sense of Frobenius norm) of operator $(\tau A+I)^{\frac{\Delta T}{\tau}}$ between the finite difference matrix and QTT-format approximation matrix, respectively. Then, the proposed QTT method (Algorithm 2) converges to finite difference method as $\delta_1, \delta_2 \to 0$.*

Proof. Let $u_i^{FD,-}$ and u_i^{FD} denote the finite difference solutions at time t_i before and after the exponential transformation, respectively. Similarly, let $u_i^{QTT,-}$ and u_i^{QTT} denote the counterparts obtained by using the QTT method.

Notice that the online procedure (i.e., the Algorithm 2) is divided into two main parts of computation. The first part is assimilating the observation data into the priori solution

Algorithm 2 Online computing

1: Set up the initial data $u(x,0) = \sigma_0(x)$ of the FKE according to the distribution of the initial state x_0, convert the $u(x,0)$ into the QTT-format, and apply the propagator operator (17) to get the priori solution (prediction) at time t_1, denoted by $u^-(x,t_1)$.

2: **for** $i = 1 \to N_t - 1$ **do**

3: Convert the term $\exp[h^T(x,t_i)S^{-1}(t_i)(y_{t_i} - y_{t_{i-1}})]$ into the QTT-format.

4: Assimilate the new observation data y_{t_i} into the priori solution $u^-(x,t_i)$ using a QTT-format Hadamard product:

$$u(x,t_i) = \exp[h^T(x,t_i)S^{-1}(t_i)(y_{t_i} - y_{t_{i-1}})]u^-(x,t_i).$$

5: Compute the priori solution (prediction) at time t_{i+1} using a matrix-vector multiplication in the QTT-format:

$$u^-(x,t_{i+1}) = (\tau A + I)^{\frac{\Delta T}{\tau}} u(x,t_i).$$

6: Calculate related statistics of prediction by using $u^-(x,t_{i+1})$ as the unnormalized density function at time t_{i+1}.

7: **end for**

with a TT-rounding procedure afterwards. By triangular inequality and stability of finite difference scheme, we easily obtain

$$\begin{aligned}
||u_i^{QTT} - u_i^{FD}||_2 &\leq ||\exp[h^T(x,t_i)S^{-1}(t_i)(y_{t_i} - y_{t_{i-1}})]||_2 \Big(||u_i^{QTT,-} - u_i^{FD,-}||_2 + \delta_1 ||u_i^{QTT,-}||_2 \Big) + \delta_1 ||u_i^{QTT}||, \\
&\leq C(1+\delta_1)||u_i^{QTT,-} - u_i^{FD,-}||_2 + C\delta_1 ||u_i^{FD,-}||_2 + \delta_1 ||u_i^{FD}||_2 + \delta_1 ||u_i^{QTT} - u_i^{FD}||_2, \\
&\leq C\frac{1+\delta_1}{1-\delta_1}||u_i^{QTT,-} - u_i^{FD,-}||_2 + C\frac{\delta_1}{1-\delta_1}, \qquad (18)
\end{aligned}$$

where C is a generic constant that does not depend on the mesh size and time step in the finite difference scheme. The second part of computation is a matrix-vector multiplication in the QTT-format also with a TT-rounding procedure. By triangular inequality and the fact $||M||_2 \leq ||M||_F$ for any matrix M, we easily obtain

$$\begin{aligned}
||u_{i+1}^{QTT,-} - u_{i+1}^{FD,-}||_2 &\leq ||(\tau A + I)^{\frac{\Delta T}{\tau}}||_2 \Big(||u_i^{QTT} - u_i^{FD}||_2 + \delta_2 ||u_i^{QTT}||_2 \Big) + \delta_1 ||u_{i+1}^{QTT,-}||_2, \\
&\leq C(1+\delta_2)||u_i^{QTT} - u_i^{FD}||_2 + C\delta_2 ||u_i^{FD}||_2 + \delta_1 ||u_{i+1}^{FD,-}||_2 + \delta_1 ||u_{i+1}^{QTT,-} - u_{i+1}^{FD,-}||_2, \\
&\leq C\frac{1+\delta_2}{1-\delta_1}||u_i^{QTT} - u_i^{FD}||_2 + C\frac{\delta_1+\delta_2}{1-\delta_1}. \qquad (19)
\end{aligned}$$

Combining the above two parts, we get

$$||u_{i+1}^{QTT,-} - u_{i+1}^{FD,-}||_2 \leq C\frac{(1+\delta_1)(1+\delta_2)}{(1-\delta_1)^2}||u_i^{QTT,-} - u_i^{FD,-}||_2 + C\frac{\delta_1+\delta_2}{(1-\delta_1)^2}. \qquad (20)$$

Finally, we obtain by induction that

$$\begin{aligned} ||u_{N_t}^{QTT,-} - u_{N_t}^{FD,-}||_2 &\leq e^{CN_t}\Big(||u_0^{QTT} - u_0^{FD}||_2 + C(\delta_1 + \delta_2)\Big), \\ &\leq Ce^{CN_t}(\delta_1 + \delta_2), \end{aligned} \tag{21}$$

where we have used the fact $||u_0^{QTT} - u_0^{FD}||_2 \leq \delta_1$, because at the beginning we have done a QTT-format construction at the step 1 of the Algorithm 2. □

5. Numerical results

In this section, we are interested in investigating the approximation properties of our method and computational savings over the existing methods, such as the finite difference method and particle filter (PF) method. We shall carry out numerical experiments on two 3D NLF problems. The definitions of these two NLF problems are given as follows,

Example 1: Almost linear problem This problem is modeled by a SDE in the Ito form as follows,

$$\begin{cases} dx_1 = -0.3x_1 + dv_1, \\ dx_2 = -0.3x_2 + dv_2, \\ dx_3 = -0.3x_3 + dv_3, \\ dy_1 = (x_2 + \sin(x_1))dt + dw_1, \\ dy_2 = (x_3 + \sin(x_2))dt + dw_2, \\ dy_3 = (x_1 + \sin(x_3))dt + dw_3. \end{cases} \tag{22}$$

where $E[d\mathbf{v}_t d\mathbf{v}_t^T] = I_3 dt$ with $\mathbf{v} = [v_1, v_2, v_3]^T$, $E[d\mathbf{w}_t d\mathbf{w}_t^T] = 0.5 I_3 dt$ with $\mathbf{w} = [w_1, w_2, w_3]^T$, and I_3 is the identity matrix of size 3×3. The noise are independent Brownian motions. The initial state is $\mathbf{x}(0) = [x_1(0), x_2(0), x_3(0)]^T = [0, 0, 0]^T$ with $\mathbf{x}(t) = [x_1(t), x_2(t), x_3(t)]^T$.

Example 2: Cubic sensor problem This problem is modeled by a SDE in the Ito form as follows,

$$\begin{cases} dx_1 = (-0.6x_1 - 0.1x_2)dt + dv_1, \\ dx_2 = (-0.5x_2 + 0.1x_3)dt + dv_2, \\ dx_3 = (-0.6x_3 + 0.1x_1)dt + dv_3, \\ dy_1 = x_2^3 dt + dw_1, \\ dy_2 = x_3^3 dt + dw_2, \\ dy_3 = x_1^3 dt + dw_3, \end{cases} \tag{23}$$

where $E[d\mathbf{v}_t d\mathbf{v}_t^T] = 1.5 I_3 dt$ with $\mathbf{v} = [v_1, v_2, v_3]^T$, $E[d\mathbf{w}_t d\mathbf{w}_t^T] = 1.5 I_3 dt$ with $\mathbf{w} = [w_1, w_2, w_3]^T$, I_3 is the identity matrix of size 3×3. The initial state is $\mathbf{x}(0) = [x_1(0), x_2(0), x_3(0)]^T = [0, 0, 0]^T$ with $\mathbf{x}(t) = [x_1(t), x_2(t), x_3(t)]^T$.

The total experimental time is $T = 20s$ for both examples. The cubic sensor problem has higher nonlinearity than the almost linear one. Thus, it is more difficult.

5.1. QTT-ranks in the spatial discretization

We consider the spatial discretization of operators on the 3D domain in the QTT-format. We have showed that the exact QTT-decomposition of the discretized Laplace operator via standard finite difference scheme and first partial derivative operator via central difference scheme have low QTT-ranks. Hence we mainly concentrate on the QTT-ranks of the discretization of velocity field $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})]^T$ and function $\mathbf{h}^T S^{-1}\mathbf{h}$.

Let us define the effective QTT-rank of a QTT-format as

$$r_{er} := \frac{\sqrt{(r_0 n_1 + r_d n_d)^2 + 4(\sum_{k=2}^{d-1} n_k) \sum_{k=1}^{d} r_{k-1} n_k r_k} - (r_0 n_1 + r_d n_d)}{2\sum_{k=2}^{d-1} n_k}, \tag{24}$$

where n_k, r_k representing mode sizes and QTT-ranks. Notice that $n_k = 2, k = 1, \cdots, d$, in the QTT-format case. In Table 1 and Table 2, we show the effective QTT-rank for the Example 1 and Example 2, respectively. We observe a slow grow in the effective QTT-rank with respect to the degree of freedom in the spatial discretization.

N on each direction	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	$\mathbf{h}^T S^{-1}\mathbf{h}$
2^4	1.32	1.32	1.32	4.77
2^5	1.34	1.34	1.34	5.41
2^6	2.20	1.35	2.04	5.82
2^7	2.23	2.34	2.29	6.08
2^8	2.40	2.35	2.30	6.27

Table 1: Effective QTT-rank of the discretized functions on the spatial grid with given precision 1×10^{-12} in almost linear problem.

N on each direction	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	$\mathbf{h}^T S^{-1}\mathbf{h}$
2^4	1.69	1.69	2.00	3.03
2^5	1.70	1.70	2.00	3.53
2^6	1.71	1.71	2.00	4.27
2^7	2.65	2.65	2.93	4.80
2^8	2.66	2.63	2.94	5.15

Table 2: Effective QTT-rank of the discretized functions on the spatial grid with given precision 1×10^{-12} in cubic problem.

In Table 3 and Table 4, we show the effective QTT-ranks of assembled operators $(\tau A + I)$ and $(\tau A + I)^{\frac{\Delta T}{\tau}}$ that are pre-computed in the offline procedure; see Eq.(16) and Eq.(17). In the experiments, we set $\Delta T = 0.25$ and $\frac{\Delta T}{\tau} = 100$ in Example 1 ($\frac{\Delta T}{\tau} = 200$ in Example 2). Recall the ΔT is the time between two observations and τ is the time step in discretizing

the FKE (8). The time step τ in Example 1 and Example 2 is chosen in such a way that the Courant-Friedrichs-Lewy (CFL) stability condition is satisfied.

If TT-rounding precision ϵ is given and far larger than approximation precision, for instance 1×10^{-12}, the accuracy of $(\tau A + I)^{\frac{\Delta T}{\tau}}$ is nearly $\frac{\Delta T}{\tau}\epsilon$. Actually, we have a rough estimate

$$\begin{aligned}
\delta_n :&= \frac{||(\tau A + I)_n - (\tau A + I)^n||_F}{||(\tau A + I)^n||_F}, \\
&\leq \frac{||(\tau A + I)_n - (\tau A + I)_{n-1} \times (\tau A + I)||_F}{||(\tau A + I)^n||_F} + \frac{||(\tau A + I)_{n-1} \times (\tau A + I) - (\tau A + I)^n||_F}{||(\tau A + I)^n||_F}, \\
&\approx \frac{||(\tau A + I)_n - (\tau A + I)_{n-1} \times (\tau A + I)||_F}{||(\tau A + I)^n||_F} + \frac{||(\tau A + I)_{n-1} - (\tau A + I)^{n-1}||_F}{||(\tau A + I)^{n-1}||_F}, \\
&\leq \frac{\epsilon}{1 - \delta_n} + \delta_{n-1} \lesssim n\epsilon, \qquad (25)
\end{aligned}$$

where $(\tau A + I)_n$ denotes the QTT-format approximation of the operator $(\tau A + I)^n$ after TT-rounding and $n = \frac{\Delta T}{\tau}$.

spatial N on each direction	Example 1	Example 2
2^4	15.56	15.42
2^5	16.65	16.31
2^6	19.56	17.25
2^7	22.17	22.37
2^8	22.96	22.87

Table 3: Effective QTT-rank of the assembled operator $(\tau A + I)$ with given precision 1×10^{-12}.

spatial N on each direction	Example 1	Example 2
2^4	8.28	9.04
2^5	9.63	12.96
2^6	12.94	17.46
2^7	17.28	21.88
2^8	38.17	23.47

Table 4: Effective QTT-rank of the assembled operator $(\tau A+I)^{\frac{\Delta T}{\tau}}$ with given TT-rounding precision 5×10^{-4} in Example 1 and 5×10^{-5} in Example 2.

From the results in Tables (1)(2) and (3)(4), we find that the QTT-ranks increase slowly while N increases exponentially fast. Hence, by exploring low-dimensional structures in the solution space, the QTT method helps us alleviate the curse of dimensionality to a certain extent.

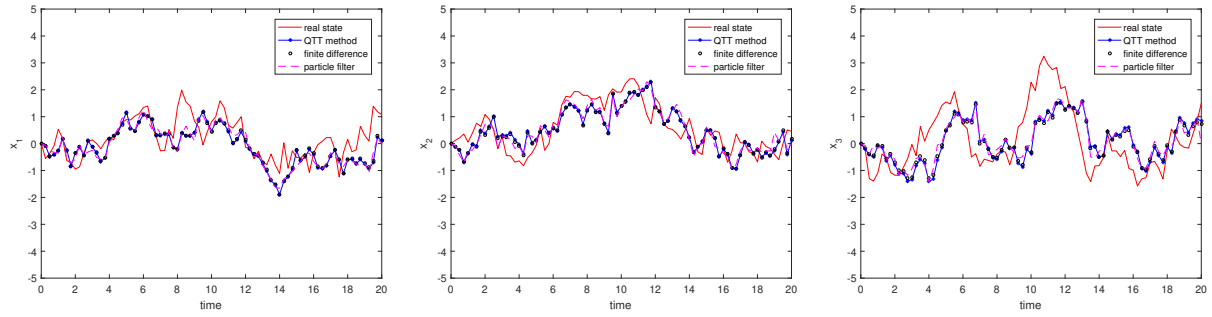
5.2. Comparison with existing methods

To obtain the reference solution, we solve Eqns.(22) and (23) using Euler-Maruyama scheme [11] with a fine time step, which generates two sequences X_{t_i} and Y_{t_i} of length $dt = 0.001$ as discrete real states at time $t_i = idt$, $i = 1, ..., 20000$. We feed the observation Y_{t_j} into the online procedure at each observation time $t_j = j\Delta T$, i.e. only a subsequence Y_{t_j} of Y_{t_i} is regarded as observation sequence and utilized.

To solve the NLF problem in a real time manner, one need to solve the path-wise robust DMZ equation associated with Eqns.(22) and (23). As such, one can solve the FKEs associated with Eqns.(22) and (23) using a finite difference method. However, the finite difference method becomes expensive when the dimension of the NLF problem increases. We shall show that the QTT method provides considerable savings over finite difference method.

For the FKE associated with the almost linear problem, we restrict the FKE on the domain $[-5, 5]^3$ and discretize the domain into 2^6 grids on each dimension. Thus the total degree of freedom is 2^{18}. The initial distribution is assumed to be Gaussian, while the corresponding unnormalized conditional density function is $\sigma_0(\mathbf{x}) = \exp(-4|\mathbf{x}|^2)$. The time step is chosen to be $\tau = \frac{\Delta T}{100}$ so that the CFL stability condition is satisfied. Based on the initial discretization of $\sigma_0(\mathbf{x})$, we implement the QTT method to solve the FKE simultaneously, where we fix the TT-rounding precision to be $\epsilon = 5 \times 10^{-4}$.

In Fig.1, we show the estimation results of the almost linear problem in three coordinates separately. The CPU time of the finite difference method is $2052s$, but the QTT method only requires $15.19s$. A significant computational saving is achieved by our method. Because there is only logarithmic dependence of most of operations on degree of freedom, and polynomial dependence on QTT-ranks that can be restricted to be small. The CPU time of the PF is 17.36. The efficiency of the PF is closely related to the number of particles. In this example, we use 3000 particles to avoid explosion in the tracking, which might happen frequently if only 1000 particles are used. In Fig.2, we show the profile of the density function at time $t = 10$.



(a) x_1 component. (b) x_2 component. (c) x_3 component.

Figure 1: Comparison of a trajectory of the almost linear problem obtained using different methods.

For the FKE associated with the cubic sensor problem, we restrict the FKE on the domain $[-3, 3]^3$ and discretize the domain into 2^6 grids on each dimension. Thus the total

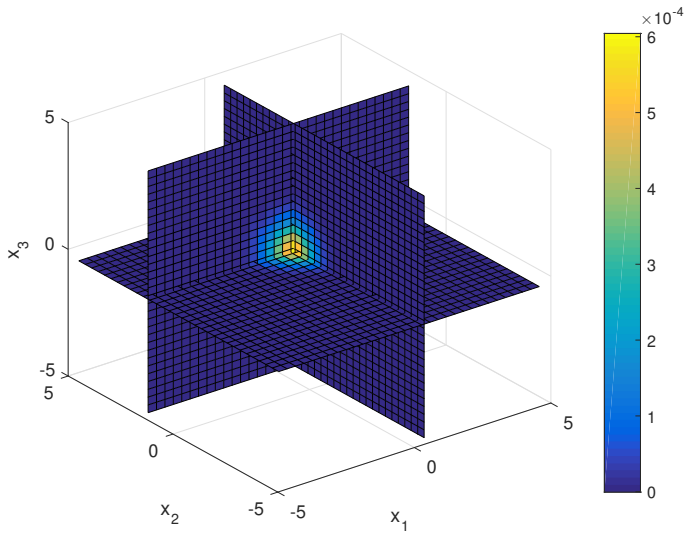
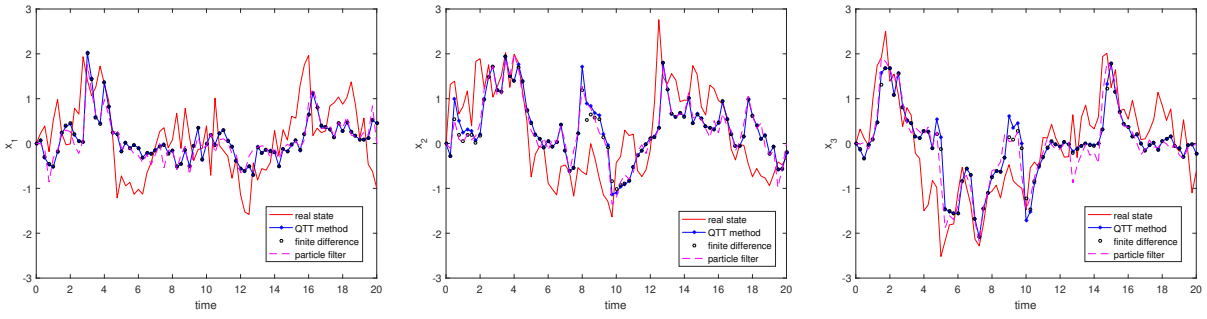


Figure 2: The estimations of density function at $t = 10s$ in almost linear problem

degree of freedom is 2^{18}. The unnormalized conditional density function of the initial state is $\sigma_0(\mathbf{x}) = \exp(-10(x_1^4 + x_2^4 + x_3^4))$. The time step is chosen to be $\tau = \frac{\Delta T}{200}$, which is smaller than the first example, in order to satisfy the CFL stability condition. We fix a higher TT-rounding precision $\epsilon = 5 \times 10^{-5}$ due to the higher nonlinearity of this example.

In Fig. 3, we show the estimation results of this almost linear problem in three coordinates separately. The CPU time of the finite difference method is $4079s$, while the QTT method is only $17.11s$. Even though the QTT-format tensor constructor has linear dependence on the degree of freedom, it takes up a minor part of computation. The CPU time of the PF is $29.45s$. In this example, we use 5000 particles to avoid explosion in the tracking. In Fig.4, we show the profile of the density function at time $t = 12s$.

We have shown through two numerical examples, especially the challenging cubic sensor problem that the QTT method gives very accurate approximation to the finite difference solution. In the meanwhile, the QTT method provides considerable savings over finite difference method.



(a) x_1 component. (b) x_2 component. (c) x_3 component.

Figure 3: Comparison of a trajectory of the cubic sensor problem obtained using different methods.

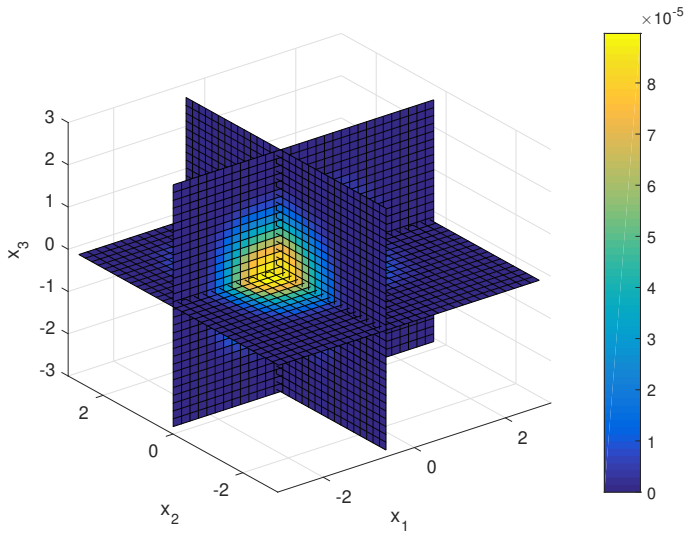


Figure 4: The estimations of density function at $t = 12s$ in cubic sensor problem

Remark 5.1. The PF method is a very popular method in solving NLF problem, which is a Monte Carlo method and it requires a certain amount of sample to compute statistical quantities. Since the PF method and our method are designed on totally different mechanics, we cannot reach a general conclusion about their performances for NLF problems.

5.3. Verification of the computational complexity

In this subsection, we intend to verify the computational complexity studied in the Proposition 4.3. Notice that the main computational load in the online procedure consists of two parts. The complexity of the first part is $\mathcal{O}(d\log(N)r^6)$, which comes from the QTT operations in solving the FKEs. It polynomially depends on QTT-ranks and logarithmically on the degree of freedom hence allows for higher dimensional setting. The second part is $\mathcal{O}(N^d r^2)$ and it comes from assimilating the observation data into the QTT solution. This part depends linearly on the degree of freedom in the spatial discretization. In our 3D numerical experiments, these two parts are comparable.

Let t_{FKE} denote the CPU time in solving the FKE and t_{EXP} is the CPU time in computing the exponential transformation, i.e., assimilating the observation data, respectively. In Table 5, we show the computational time of the QTT method and finite difference method in solving the Example 1. The QTT-rank is the averaged effective QTT-rank of solution u at every time. We find that the growth of computational complexity of the QTT method is significantly slower than that of finite difference method. When QTT operations part is dominant, i.e. $t_{FKE} > t_{EXP}$, the QTT method can achieve almost logarithmic complexity with respect to the degree of freedom.

In the experiment, the mean square error (MSE) between two methods keeps small. Namely, the conditional density function u has a low rank structure and such structure is approximated well in the QTT-format. In practice, the precision of TT-rounding in all of

	QTT method			Finite difference	
N	t_{FKE}	t_{EXP}	QTT-rank	t_{FKE}	t_{EXP}
2^4	1.87	0.85	6.83	9.62	0.03
2^5	4.15	1.36	7.88	169	0.08
2^6	11.18	5.20	8.78	2802	0.65
2^7	27.49	151.25	8.89	–	–

Table 5: CPU time(sec.) of the QTT method and finite difference method. N is the grid number in each dimension.

our experiments can be reduced more without loss much of the accuracy. So it is possible to get more efficiency by relaxing the precision. Furthermore, the TT-rounding algorithm also allows for prescribing an upper bound of QTT-ranks of TT-rounding operation, which is useful for assembling the operator matrix in (17). Note that it is reasonable that QTT-ranks of both u and operator matrix (see Table 3) increase with N, so that the growth rate is slightly larger than logarithm.

6. Conclusions

In this paper, we investigate the tensor train decomposition method to numerically solve the forward Kolmogorov equation (FKE), which has important application in solving nonlinear filtering problems. In [19], Yau and Yau developed a novel algorithm to solve the path-wise robust DMZ equation, which laid down the solid foundation to study the nonlinear filtering problem. Since then many efforts have been devoted to solve the path-wise robust DMZ equation. However, it is still very challenging to solve high-dimensional problem in an effective fashion.

We explore the low-dimensional structures in the solution space of the FKE and adopt the tensor train decomposition method to solve the FKE. Our method consists of offline and online stages. In the offline stage, we discrete the PDE operators involved in the FKE and approximate them using the tensor train method. Moreover, we approximate the evolution of the FKE operator using the tensor train method. In the online stage using the pre-computed low-rank approximation tensors, we can quickly solve the FKE given the new observation data. Therefore, we can solve the NLF problem in a real-time manner. Under some mild assumptions, we prove the convergence and effectiveness analysis of the proposed method. Finally, we design two online numerical experiments to show the efficiency and accuracy of the proposed method.

There are several directions we want to explore in our future work. First, we are interested in designing efficient numerical method for high-dimensional NLF problems (with $d \geq 3$), which will be reported in our subsequent work. Moreover, we shall conduct more numerical experiments to compare the performance of the tensor train method with the particle filter in solving NLF problems.

7. Acknowledgements

The research of S. Li is partially supported by the Doris Chen Postgraduate Scholarship. The research of Z. Wang is partially supported by the Hong Kong PhD Fellowship Scheme. The work of S. S.-T. Yau was supported by the National Natural Science Foundation of China (11471184). The research of Z. Zhang is supported by the Hong Kong RGC grants (Projects 27300616, 17300817, and 17300318), National Natural Science Foundation of China (Project 11601457), Seed Funding Programme for Basic Research (HKU), and the Hung Hing Ying Physical Sciences Research Fund (HKU).

References

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[2] A. Bain and D. Crisan. *Fundamentals of stochastic filtering*, volume 3. Springer, 2009.

[3] J. Baras, G. Blankenship, and W. Hopkins. Existence, uniqueness, and asymptotic behavior of solutions to a class of Zakai equations with unbounded coefficients. *IEEE Transactions on Automatic Control*, 28(2):203–214, 1983.

[4] A. Bensoussan, R. Glowinski, and A. Rascanu. Approximation of the Zakai equation by the splitting up method. *SIAM Journal on Control and Optimization*, 28(6):1420–1431, 1990.

[5] T. Duncan. Probability densities for diffusion processes with applications to nonlinear filtering theory and detection theory. Technical report, STANFORD UNIV CA STANFORD ELECTRONICS LABS, 1967.

[6] W. Fleming and S. Mitter. Optimal control and nonlinear filtering for nondegenerate diffusion processes. *Stochastics*, 8(1):63–77, 1982.

[7] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002.

[8] I. Gyöngy and N. Krylov. On the splitting-up method and stochastic partial differential equations. *The Annals of Probability*, 31(2):564–591, 2003.

[9] K. Ito. Approximation of the Zakai equation for nonlinear filtering. *SIAM Journal on Control and Optimization*, 34(2):620–634, 1996.

[10] G. Kallianpur. *Stochastic filtering theory*, volume 13. Springer Science & Business Media, 2013.

[11] P. E. Kloeden and E. Platen. *Numerical solution of Stochastic Differential Equations*. Springer-Verlag Berlin, Heidelberg, 1992.

[12] S. Lototsky, R. Mikulevicius, and B. L. Rozovskii. Nonlinear filtering revisited: a spectral approach. *SIAM Journal on Control and Optimization*, 35(2):435–461, 1997.

[13] X. Luo and Stephen S. T. Yau. Complete real time solution of the general nonlinear filtering problem without memory. *IEEE Transactions on Automatic Control*, 58(10):2563–2578, 2013.

[14] X. Luo and Stephen S. T. Yau. Hermite spectral method to 1-D forward Kolmogorov equation and its application to nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 58(10):2495–2507, 2013.

[15] R. Mortensen. Optimal control of continuous-time stochastic systems. Technical report, CALIFORNIA UNIV BERKELEY ELECTRONICS RESEARCH LAB, 1966.

[16] N. Nagase. Remarks on nonlinear stochastic partial differential equations: an application of the splitting-up method. *SIAM journal on control and optimization*, 33(6):1716–1730, 1995.

[17] E. Pardouxt. Stochastic partial differential equations and filtering of diffusion processes. *Stochastics*, 3:127–167, 1980.

[18] Z. Wang, X. Luo, Stephen S.-T. Yau, and Z. Zhang. Proper orthogonal decomposition method to nonlinear filtering problems in medium-high dimension. *submitted to IEEE Transactions on Automatic Control*, 2018.

[19] Shing-Tung Yau and Stephen S-T Yau. Real time solution of the nonlinear filtering problem without memory II. *SIAM Journal on Control and Optimization*, 47(1):163–195, 2008.

[20] M. Yueh, W. Lin, and S. T. Yau. An efficient numerical method for solving high-dimensional nonlinear filtering problems. *Communications in Information and Systems*, 14(4):243–262, 2014.

[21] M. Zakai. On the optimal filtering of diffusion processes. *Probability Theory and Related Fields*, 11(3):230–243, 1969.

[22] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295-2317, 2011.

[23] Vlanimir A. Kazeev, and Boris N. Khoromskij. Low-rank explicit QTT representation of the laplace operator and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 33(3):724–758, 2012.

[24] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker-Planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

[25] I. V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applicatoins*, 31(4):2130–2145, 2010.

[26] Boris N. Khoromskij. $\mathcal{O}(d\log n)$-Quantics approximation of $N - d$ tensors in high-dimensional nemerical modeling. *Constructive Approximation* , 34:257–280, 2011.

[27] I. V. Oseledets and E. E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432:70–88, 2010.