Functional Tensor-Train Chebyshev Method for Multidimensional Quantum Dynamics Simulations

Micheline B. Soley, Paul Bergold, Alex A. Gorodetsky, and Victor S. Batista*

Cite This: https://doi.org/10.1021/acs.jctc.1c00941

ABSTRACT: Methods for efficient simulations of multidimensional quantum dynamics are essential for theoretical studies of chemical systems where quantum effects are important, such as those involving rearrangements of protons or electronic configurations. Here, we introduce the functional tensor-train Chebyshev (FTTC) method for rigorous nuclear quantum dynamics simulations. FTTC is essentially the Chebyshev propagation scheme applied to the initial state represented in a continuous analogue tensor-train format. We demonstrate the capabilities of FTTC as applied to simulations of proton quantum dynamics in a 50-dimensional model of hydrogen-bonded DNA base pairs.

1. INTRODUCTION

Quantum dynamics simulations are essential for rigorous theoretical studies of quantum reaction dynamics, including applications to structural and dynamical problems with critical rearrangements of protons or electronic configurations. A variety of approaches have been developed, including time-dependent Hartree methods and other methods based on short-time approximations of the time-evolution operator, such as the Trotter expansion and finite difference methods. Here, we focus on the Chebyshev method for simulation of quantum wavepacket dynamics, which enables computation of the time-evolved quantum state at the final time without having to compute intermediate states at earlier times. Thus, contrary to methods based on short-time propagators, the Chebyshev propagation scheme can be implemented without error accumulation. Chebyshev propagation is currently one of the foremost approaches for simulations of quantum dynamics in low dimensionality, as demonstrated for nuclear quantum dynamics simulations of molecular systems with up to six dimensions. However, applications to higher-dimensional systems have been hindered by the exponential scaling of memory and computational cost with dimensionality, due to its reliance on full-grid representations. Here, we introduce a viable solution to the exponential scaling with dimensionality by applying the Chebyshev propagation scheme to the initial state represented in functional tensor-train (FT) format—i.e., the continuous analogue of the tensor-train/matrix product state decomposition. The resulting functional tensor-train Chebyshev (FTTC) method is demonstrated as applied to simulations of proton dynamics in a high-dimensional model of hydrogen-bonded adenine–thymine DNA base pairs, where photo-induced proton transfer has long been thought to have important biological implications (e.g., photoinduced mutations).

Tensor networks have been successfully applied to a wide range of studies, including applications to solving partial differential equations, quantum dynamics methods, machine learning, electronic structure calculations, and calculations of vibrational states.

Here, we build upon the tensor-train split-operator Fourier transform (TT-SOFT) method, and we develop the FTTC method which is essentially a functional tensor-train implementation of the Chebyshev propagation scheme, popularized by Kosloff and co-workers. FTTC expands the initial state as a functional tensor train and evolves it by applying the Chebyshev expansion of the time-evolution operator

$$e^{-\frac{2\pi}{N} t} \approx \sum_{k=0}^{N-1} (2 - \delta_{k,0}) (-i)^{k} J_{k}(t) T_{k}(\mathbf{H})$$

(1)

where $T_{k}(\mathbf{H})$ are Chebyshev polynomials of the Hamiltonian $\mathbf{H}$ in functional tensor-train format, $J_{k}(t)$ are the Bessel functions of the first kind, and $t$ is the final propagation time. In practice, a finite number $N \geq 1$ of polynomials is employed.
and the expansion is applied iteratively in time. Important advantages of the proposed FTTC algorithm when compared to other propagation methods based on matrix product states are: (i) the error need not accumulate with time since the state at time $t$ can be obtained directly without having to compute earlier intermediate states and (ii) the uniform character of the Chebyshev expansion decreases the error exponentially with $N$.

We focus on functional tensor trains (FT),$^{66-69}$ which are continuous analogues of tensor trains/matrix product states and have yet to be demonstrated as applied to simulations of quantum nuclear dynamics. Functional tensor-train representations of time-dependent states allow for efficient computations of gradients of multidimensional tensors, so they are expected to be particularly valuable for a variety of applications, including studies of the quantum control of molecular systems.$^{70}$ Therefore, we can implement eq 1 as expected to be particularly valuable for a variety of applications.

The FTTC method can also be implemented using discrete tensor trains,$^{71,72}$ instead of their continuous analogues, allowing for efficient representation and manipulation of matrix product states.$^{73}$

The article is organized as follows. Section 2 describes the Chebyshev methodology. Section 2.1 introduces the Chebyshev polynomials. Section 2.2 describes how to generate Chebyshev expansions of complex-valued functions, and Section 2.3 describes Chebyshev propagation based on discrete space representations. Section 3 describes the functional tensor-train decomposition as a continuous analogue tensor-train format for nonlinear algebra manipulations of high-dimensional tensors. Section 4 describes our functional tensor-train Chebyshev propagation method (FTTC) as implemented for numerical integration of the time-dependent Schrödinger equation. Section 6 demonstrates the capabilities of FTTC as applied to simulations of proton quantum dynamics in a 50-dimensional model of DNA base pairs with highly anharmonic modes. The Supporting Information demonstrates that tensor-train Chebyshev dynamics can entail lower computational cost relative to the state-of-the-art split operator Fourier transform method for long time steps. The results show that FTTC enables simulations of molecular systems far beyond the capabilities of the standard grid-based Chebyshev method.

2. CHEBYSHEV APPROACH

The Chebyshev propagation method integrates numerically the time-dependent Schrödinger equation

$$i \hbar \frac{\partial \Psi}{\partial t} = \hat{H} \Psi$$

(2)

where we have used atomic units ($\hbar = 1$). For simplicity, we consider a system described by the Hamiltonian

$$\hat{H} = \frac{\hat{p} \cdot \hat{p}}{2m} + \hat{V} = -\frac{1}{2m} \Delta \Psi + \hat{V}$$

(3)

where $m > 0$ is the mass of the system, $\hat{p} = -i \nabla_x$ is the momentum operator, and $V: \mathbb{R}^d \rightarrow \mathbb{R}$ is a given potential energy surface (PES) describing interactions that rule the underlying dynamics of the system. Using the unitary evolution propagator $\hat{U}(t) = e^{-i \hat{H} t}$, the solution of eq 2, $\Psi(t)$, corresponding to the initial state $t = 0$ is

$$\Psi(t) = \hat{U}(t) \Psi_0$$

(4)

The Chebyshev propagation method approximates the propagator $\hat{U}(t) = e^{-i \hat{H} t}$ for a fixed final time $t$ in terms of a linear combination of the first $N \geq 1$ Chebyshev polynomials of the Hamiltonian $T_0(\hat{H}), \ldots, T_{N-1}(\hat{H})$, as discussed in Section 2.1.

2.1. Chebyshev Polynomials. For all integers $k \geq 0$ and all $x \in [-1, 1]$, the $k$th Chebyshev polynomial is defined as follows

$$T_k(x) = \cos(k \arccos(x))$$

(5)

where $\arccos$ is the inverse of the cosine (i.e., $\arccos(\cos(x)) = x$. We note that the Chebyshev polynomials are defined only for input values $x \in [-1, 1]$, since the cosine function attains values only in that limited range, and satisfy the following recurrence relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$

(6)

with $T_0(x) = 1$ and $T_1(x) = x$ defining the subsequent Chebyshev polynomials, so the first four polynomials are

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1,$$

$$T_3(x) = 4x^3 - 3x$$

(7)

Figure 1. Plot of the first four Chebyshev polynomials, defined only in the limited interval $[-1, 1]$, since the possible values of the cosine function are limited to that interval.

We note that Chebyshev polynomials have a number of remarkable properties and are therefore an important tool in the field of approximation theory.$^{74,75}$ For instance, let us remark that they satisfy the following orthogonality relation for all $j, k \geq 1, j \neq k$

$$\int_{-1}^{1} \frac{dx}{\sqrt{1 - x^2}} T_j(x) T_k(x) = \frac{\pi}{2} \delta_{j,k}$$

(8)

showing that the Chebyshev polynomials are orthogonal with respect to the weighted inner product defined by the left hand side of eq 8.

2.2. Chebyshev Expansion of Complex-Valued Functions. Chebyshev polynomials can be used to approximate a given complex-valued function $f$ via its Fourier series...
representation. To show that, we introduce the 2π-periodic function
\[ g(x) = f(\cos(x)) \] (9)
which can be represented in the interval \((-\pi < x < \pi)\) in terms of its Fourier series as follows
\[ g(x) = \sum_{k=0}^{\infty} (2 - \delta_{k,0}) a_k \cos(kx), \]
\[ a_k = \frac{1}{\pi} \int_{0}^{\pi} g(x) \cos(kx) \, dx \] (10)
Therefore
\[ f(y) = g(\arccos(y)) \] (11)
can be represented in terms of the Chebyshev polynomials as follows
\[ f(y) = \sum_{k=0}^{\infty} (2 - \delta_{k,0}) c_k T_k(y), \]
\[ c_k = \frac{1}{\pi} \int_{-1}^{1} \frac{dy}{\sqrt{1 - y^2}} f(y) T_k(y) \] (12)
for \(y \in [-1, 1]\). Equation 12 is called the Chebyshev expansion of \(f\) and it can be used to approximate \(f\) as the linear combination of the first \(N\) Chebyshev polynomials as follows
\[ f(y) \approx S_N f(y) = \sum_{k=0}^{N-1} (2 - \delta_{k,0}) c_k T_k(y) \] (13)
The coefficients \(c_k\), defined by eq 12, are essentially the Fourier coefficients of the function \(g\) that decay exponentially with \(N\) for analytic functions76 (i.e., smooth in the complex domain) and thus provide fast convergence of the partial sums \(S_N f\). The resulting Chebyshev approximant \(S_N f\) is a polynomial of degree \(N\), which is known to be close to the polynomial of the same degree with minimal error in the interval \([-1, 1]\).77

2.3. Chebyshev Propagation in Discrete Representations. We obtain an approximation of the operator \(U = e^{-itH}\) at time \(t\) by considering the function \(f(y) = e^{-iy}\) for which the coefficients \(c_k\) defined according to eq 12 can be expressed in terms of the Bessel functions \(J_k\) (of the first kind) as follows78
\[ c_k = (-i)^k J_k(t) \] (14)
yielding the following approximation
\[ e^{-iy} \approx \sum_{k=0}^{N-1} (2 - \delta_{k,0}) (-i)^k J_k(t) T_k(y) \] (15)
with \(y \in [-1, 1]\). Using a linear transformation of the argument \(y\), we can restate eq 15 for an arbitrary Hermitian matrix \(H \in \mathbb{C}^{D \times D}\) (where \(D > 1\) is a positive integer) with eigenvalues contained in a finite interval \([a, b]\) as follows
\[ e^{-itH} \approx e^{-it^*} \sum_{k=0}^{N-1} (2 - \delta_{k,0}) (-i)^k J_k(t^*) T_k(H_0) \] (16)
where we have introduced the rescaled variables \(t^* = \frac{t}{2} (b-a)\) and \(H_0 = \frac{2}{b-a} (H - b + a I_D)\) (17)
where \(I_D\) is the \(D \times D\) identity matrix.

Fast convergence is typically obtained for \(e^{-iy}\) since it is a smooth function, although the number of required polynomials increases with \(t\) since \(e^{-iy}\) is oscillatory. Thus, a sufficiently large number \(N\) of Chebyshev polynomials is needed to resolve the oscillations. In fact, it has been shown that the error falls like the \(N\)th order in \(t^* / (2N)\) for sufficiently large \(N\).

It is important to note that eq 16 can be used more generally than in the current implementation to approximate the solution to any linear system of the form \(iu = Hu\). Such linear systems typically arise in space discretization methods, including the Fourier collocation method, the Fourier Galerkin method, or the Hermite Galerkin method.79 So, we anticipate that the FTTC method should also be valuable for solving high-dimensional linear systems in a wide range of applications beyond the solution of the time-dependent Schrödinger equation.

2.4. Discrete Tensor-Train Implementation. Discrete tensor-train approximations of \(e^{-it^*H}\) are obtained by discretizing the \(d\)-dimensional space with a uniform grid of size \(\Delta x_j > 0\) for the \(j\)th nuclear coordinate, spanning the range \(x_{j,\text{min}}\) to \(x_{j,\text{max}}\) with \(n_j > 1\) points for each dimension \(j = 1, \ldots, d\). Analogously, discrete tensor-train representations of wave functions are obtained as low-rank \(d\)-dimensional complex-valued tensor trains approximating
\[ W[k_1, \ldots, k_d] = \Psi(x_k) \] (18)
where \(k = (k_1, \ldots, k_d)\) are the indices of tensor-train entries corresponding to nuclear coordinate values \(x_k = (x_{1,k_1}, \ldots, x_{d,k_d})\). The discrete tensor-train representation of \(W[k_1, \ldots, k_d]\) is defined as follows
\[ W[k_1, k_2, \ldots, k_d] = W[k_1] W[k_2] \cdots W[k_d], \]
\[ 1 \leq k_j \leq n_j \text{ for all } j \] (19)
where \(W[k_j] \in \mathbb{C}^{n_j \times M_j}\) are matrices and \(n_j\) are the number of grid points in the \(j\)th coordinate direction.

The action of the Hamiltonian \(\hat{H}\) on a wave function \(\Psi\) is represented by the Hermitian operator \(\hat{H} = \hat{\mathcal{J}} + \mathcal{V}\). The real-valued "potential energy tensor"
\[ \mathcal{V}[k_1, \ldots, k_d] = V(x_{1,k_1}, \ldots, x_{d,k_d}) \] (20)
acts on \(W[k_1, \ldots, k_d]\) as an element-wise multiplication operator (Hadamard product). The discrete kinetic energy operator
\[ (\hat{\mathcal{J}} W)[k_1, \ldots, k_d] \approx -\frac{1}{2m} \Delta \Psi(x_k) \] (21)
is defined by the Laplacian \(\Delta\), that acts as a multiplication operator in momentum space. Therefore, we apply the kinetic energy operator in momentum space by exploiting the highly efficient (linear scaling with dimensionality) implementation of the multidimensional discrete Fourier transform of tensor trains to switch between position and momentum space. With the help of the fast Fourier transform (FFT), we therefore obtain a very efficient and accurate implementation of the discretized kinetic energy operator.
The discrete Hamiltonian is rescaled, according to eq 17, as follows

\[
\hat{H}_0 = \frac{2}{E_{\text{max}} - E_{\text{min}}} \left( \hat{H} - \frac{E_{\text{max}} + E_{\text{min}}}{2} \hat{I} \right) \tag{22}
\]

where \( \hat{I} \) denotes the identity on the tensor space. The bounds for the eigenvalues \( E_{\text{min}} \) and \( E_{\text{max}} \) depend on the extension of the grid and are given by

\[
E_{\text{min}} = \min_k \mathcal{V}[k_1, ..., k_d],
\]

\[
E_{\text{max}} = \max_k \mathcal{V}[k_1, ..., k_d] + \frac{n^2}{2m} \left( \frac{1}{\Delta x_1^2} + \cdots + \frac{1}{\Delta x_d^2} \right) \tag{23}
\]

where we used \( \Delta p_j = 2\pi/(x_{j,\text{max}} - x_{j,\text{min}}) \) for the grid size in momentum space of the \( j \)-th coordinate, giving the maximum kinetic energy

\[
\frac{1}{2m} p_j^2,_{\text{max}} = \frac{\pi^2}{2m \Delta x_j^2} \tag{24}
\]

Consequently, the solution \( \Psi(t) \) is approximated with \( N \) Chebyshev polynomials as follows

\[
\Psi(t) = e^{-it\hat{H}_0} \Psi(0) \approx e^{-it\hat{H}_0} \sum_{k=0}^{N-1} (2 - \delta_{k,0})(-i)^k f_k(t) T_k(\hat{H}_0) \mathcal{W}_0 \tag{25}
\]

where \( t^2 = E^2/2, E^2 = E_{\text{max}} + E_{\text{min}} \) and \( \mathcal{W}_0 \) samples the initial wave function.

We implement eq 25 as a one-step propagator to compute \( \Psi(t) \) directly from the initial data using the Clenshaw algorithm \(^80\) (see Appendix A). Alternatively, we can obtain the time-dependent states \( T_k(\hat{H}_0) \mathcal{W}_0 \) according to the recurrence relation eq 6

\[
T_0(\hat{H}_0) \mathcal{W}_0 = \mathcal{W}_0 \]

\[
T_{k+1}(\hat{H}_0) \mathcal{W}_0 = 2\hat{H}_0 T_k(\hat{H}_0) \mathcal{W}_0 - T_{k-1}(\hat{H}_0) \mathcal{W}_0 \tag{26}
\]

for \( k \geq 1 \)

The same Chebyshev propagation scheme described in this section for discrete tensor-train (TT) decompositions\(^32\) can be readily implemented using the continuous analogue functional tensor-train decomposition,\(^66\) as described in the following section.

3. FUNCTIONAL TENSOR-TRAIN DECOMPOSITION

3.1. Continuous Analogue of the Tensor-Train Decomposition. Following refs 68 and 69, here we give a brief overview of the functional tensor-train (FT) format, which offers an efficient data compression scheme\(^66,67\)

\[
f(x_1, x_2, ..., x_d) = \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_d=1}^{r_d} f_{1}(i_1,x_1)f_{2}(i_2,x_2) \cdots f_{d}(i_d,x_d) \tag{27}
\]

where \( f_{k}(\cdot), \mathcal{X}_k \to \mathbb{R}, \mathcal{X}_k \) denotes the domain of the \( k \)-th physical dimension, with \( r_0 = r_d = 1 \) for single-output functions such as polynomials or linear elements. The FT decomposition, introduced by eq 27, is a low-rank decomposition of multivariate functions in an analogous way that the TT decomposition, eq 19, is a low-rank decomposition of multivariate arrays. A more compact expression for the FT analogue is obtained by viewing a function value as a set of products of matrix-valued functions

\[
f(x_1, x_2, ..., x_d) = f(x_1; \mathcal{F}_1(x_2) ... \mathcal{F}_d(x_d) \tag{28}
\]

where each matrix-valued function \( \mathcal{F}_k: \mathcal{X}_k \to \mathbb{R}^{r_k \times r_k} \) is called a core and can be visualized as an array of the univariate functions

\[
\mathcal{F}_k(x_k) = \begin{bmatrix} f_k^{(1)}(x_k) & \cdots & f_k^{(r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_k-1)}(x_k) & \cdots & f_k^{(r_k-r_k)}(x_k) \end{bmatrix} \tag{29}
\]

If each univariate function is represented with \( p \) parameters (for example, coefficients of a polynomial) and \( r_k \leq r \) for all \( k \), then the storage complexity scales as \( O(d p^r) \). Comparing this representation with eq 19, we see a very close resemblance between TT cores and FT cores. Indeed, they are both matrices when indexed by a discrete index \( i_k \) for the TT or a continuous index \( x_k \) for the FT.

3.2. Parameterizations of Low-Rank Functions. The finer structure of the FT format is described by FT cores comprised of \( d \) sets of univariate functions \( (\mathcal{F}_k)_{i_k=1}^{r_k} \). Each set could be different for different dimensions (\( e.g., 2\pi \)-periodic functions could represent physical dimensions corresponding to torsional angles, while Hermite polynomials could represent stretching modes). As a result, the full FT is parameterized through the parameterization of the set of univariate functions of each dimension, as chosen for the optimal representation of physical coordinates. Considering that \( p_{kij} \in \mathbb{Z}_+ \) denotes the number of parameters describing \( f_k^{(i)} \) and \( \theta \in \mathbb{R}^l \) the vector of parameters of all of the univariate functions, then there are a total of \( \frac{p_k}{r_k-1} \mathcal{F}_k \) parameters describing the FT representation.

The parameter vector \( \theta \) is indexed by a multi-index \( \alpha = (k, i, j, l) \) where \( k = 1, ..., d \) corresponds to an input variable, \( i = 1, ..., r_k-1 \), \( j = 1, ..., r_k \) correspond to a univariate function within the \( k \)-th core, and \( l = 1, ..., r_k \) corresponds to a specific parameter within that univariate function. In other words, we adopt the convention that \( \theta_k = \theta_{kij} \) refers to the \( l \)-th parameter of the univariate function in the \( i \)-th row and \( j \)-th column of the \( k \)-th core.

The additional flexibility of the representation allows both linear and nonlinear parameterizations of univariate functions. In particular, the linear parameterization represents a univariate function as an expansion of basis functions \( (\phi_k^{(i)}; \mathcal{X}_k \to \mathbb{R})_{i=1}^{r_k} \) according to

\[
f_k^{(i)}(x_k; \theta) = \sum_{l=1}^{r_k} \theta_{kij} \phi_k^{(i)}(x_k) \tag{30}
\]

3.3. Low-Rank Functions vs Low-Rank Coefficients. For greater versatility, the FT can be used by independently parameterizing the univariate functions of each core, and both linear and nonlinear parameterizations are possible. As described below, the advantage of this representation includes a naturally sparse storage scheme for the cores.
An advantage of this type of structure is that it readily enables adaptivity when performing common multilinear algebraic operations with functions in low-rank format. For example, taking the product of two functions in low-rank format requires computing products between univariate functions in corresponding cores of the two functions. In particular, it requires computing the product between every combination of the functions. Because we store each univariate function separately, this product can accurately account for the complexity of the resulting univariate function. For instance, if the product of two third-order polynomials is considered, then a sixth-order polynomial will be stored. However, if a third-order polynomial is multiplied by a first-order polynomial, then only a fourth-order polynomial needs to be stored. In contrast, traditional tensor-based storage schemes would require storing a fourth-order polynomial, as the building blocks of low-rank representations.

Another advantage is the availability of efficient computational algorithms for multilinear algebra that can adapt the representation of each univariate function individually as needed in the spirit of continuous computation pioneered by Chebfun. The salient point is that we consider univariate functions, rather than matrices or lower-order arrays, as the building blocks of low-rank representations. Another advantage is the availability of efficient computational algorithms for multilinear algebra that can adapt the representation of each univariate function individually as needed in the spirit of continuous computation pioneered by Chebfun.

The TT/ MPS format is a particular case of the general FT decomposition, naturally arising when two simplifying assumptions are made:

1. linear parameterization of each $f_i^{(k)}$;
2. identical basis for the functions within each FT core, i.e., $P_{kj} = p_k$ and $\phi_{kl}^{(i)} = \phi_{kl}^{k'}$ for all $i = 1, \ldots, r_k$, $k' = 1, \ldots, r_k$, and $l = 1, \ldots, p_k$.

These assumptions transform the problem of storing low-rank functions to the problem of storing low-rank coefficients, allowing the use of discrete TT algorithms and theory. Both representations store the coefficients of a tensor-product basis $\phi_{kl}$ for all $k$ and $l$.

Function evaluations can be obtained from the coefficients defining the tensor $F_k \in \mathbb{R}^{p_1 \times \ldots \times p_d}$ of the following form

$$F_k[i, : , l, :] = \begin{bmatrix}
\theta_{k1l1} & \cdots & \theta_{k1lr_k} \\
\vdots & \ddots & \vdots \\
\theta_{kr_kl1} & \cdots & \theta_{kr_klr_k}
\end{bmatrix}$$

for $l = 1, \ldots, p_k$, by performing the following summation

$$f(x_1, \ldots, x_d) = \sum_{l_1=1}^{p_1} \ldots \sum_{l_d=1}^{p_d} F_k[i, : , l_1, : ] \ldots F_k[i, : , l_d, : ] \phi_{kl}(x_i) \phi_{kl}(x_i)$$

(32)

From eqs 28, 31, and 32 we can see that the relationship between the TT cores $F_k$ and the FT cores $F_k$ is

$$F_k(x_i) = \sum_{l=1}^{p_k} F_k[i, : , l, : ] \phi_{kl}(x_i)$$

(33)

where the basis function multiplies every element of the tensor. In other words, FT cores represent a TT decomposition of the $p_1 \times p_2 \times \cdots \times p_d$ coefficient tensor of a tensor-product basis and inherit the properties of the TT decomposition.

### 3.4. Operations in the FT Format

Performing continuous multilinear algebra is one of the main advantages of the continuous framework. The operations of addition, multiplication, differentiation, integration, and inner products are easily performed for functions in the FT format as follows.

Addition and multiplication of two functions are performed similarly to addition and multiplication of tensors in the TT format. For addition, the cores of $g(x) = f(x) + h(x)$ are

$$G_k(x) = \left[ F_k(x), H_k(x) \right], \quad \mathcal{G}_k(x) = \begin{bmatrix} F_k(x) & 0 \\ 0 & H_k(x) \end{bmatrix}$$

(34)

for $k = 2, \ldots, d$. For multiplication, $g(x) = f(x) h(x)$, we have

$$G_k(x) = F_k(x) \otimes H_k(x) \quad \text{for } k = 1, \ldots, d$$

(35)

For both of these operations, the continuous functional decomposition has an important advantage compared to operations based on the discretized representation. Primarily, the advantage comes from the ability to add functions of differing discretization levels, e.g., functions represented with bases of different orders. In the discrete case, one can only add functions with identical discretizations.

The continuous nature of the FT also allows us to perform differentiation, as necessary to the implementation of the Laplacian, by differentiating scalar-valued functions that make up the corresponding core. For example, consider the partial derivative of a $d$-dimensional function $f$

$$\frac{\partial f}{\partial x_k} = \frac{d_{k}}{d x_{k}} F_{1} \ldots F_{k-1} : \ldots : F_{k+1} \ldots F_{d}$$

(36)

$$\frac{d_{k}^{(i+1)}}{d x_{k}} \frac{d_{k}^{(i)}}{d x_{k}} \frac{d_{k}^{(i-1)}}{d x_{k}} \frac{d_{k}^{(i-r)}}{d x_{k}}$$

When the univariate functions are expressed in, for example, a basis of orthonormal polynomials, then this operation is unique, well-defined, and computationally inexpensive.

Integration is widely used in Section 6 to compute expectation values. Integrating the multivariate functions scales linearly with dimensionality since it requires integrating over the one-dimensional functions in each core and then performing matrix-vector multiplication $d - 1$ times as follows

$$\int f(x) \ dx = \int F_1(x_1) F_2(x_2) \ldots F_d(x_d) \ dx_1 \ldots \ dx_d$$

$$\left( \int F_1(x_1) \ dx_1 \right) \left( \int F_2(x_2) \ dx_2 \right) \ldots \left( \int F_d(x_d) \ dx_d \right)$$

$$= \Gamma_1 \ldots \Gamma_d$$

(37)

where $\Gamma_k = \int F_k(x_k) \ dx_k$ contains entries $\Gamma_k[i, j] = \int f^{(i)}(x_k) \ dx_k$ and the integral stands for an integral over an arbitrary domain. Furthermore, since each of the univariate functions is typically represented on a known basis, the integral is well defined, unique, and computationally inexpensive to obtain.
The inner product between two functions is another important operation essential for quantum dynamics simulations and computations of correlation functions. Naively, the inner product can be implemented by first computing the product $g(x) = f(x)h(x)$ and then integrating $g(x)$, requiring $O(dr^4)$ operations. However, this operation can be made more efficient by combining the operations needed for integration and multiplication. For example, Algorithm 1 uses an efficient computation of $v^T(A \otimes B)$ to perform the inner product in $O(dr^3)$, where $v$ is a vector and $A$ and $B$ are matrices.

Algorithm 1: Inner product between two functions in FT format.

```plaintext
Require: Functions $f$ with ranks $r_f^D$ and $g$ with ranks $r_g^D$ in FT format.
Ensure: $y = \int f(x)g(x)dx$
1: $y = f \otimes g \otimes 1$
2: for $k = 2$ to $d$
3: $Y = \text{reeshap}(y, r_f^D, r_g^D, r_y^D)$
4: $T = F_I(x)^T Y$
5: $A = G_I(x) T^T (x)$
6: $Y = A \otimes 1$
7: $y = \text{reeshap}(Y, 1, r_y^D)$
end for
8: $y = y[1]
```

The second derivative is implemented core-by-core in the space of univariate functions. The second derivatives of univariate functions are computed only once, which exploits the benefits of the continuous representation and avoids the need for explicit calculation of the Fourier transform required by grid-based methods.

4. FUNCTIONAL TENSOR-TRAIN CHEBYSHEV PROPAGATION

Wavepackets and operators are efficiently represented in terms of low-rank functional tensor trains (FT) or discrete tensor trains (TT). The decompositions are constructed analytically or interpolated with the cross approximation as implemented in the Compressed Continuous Computation (C³) library in terms of linear element expansions or the TT-Toolbox. Operations are computed in the position-space representation, including the kinetic energy operator in the FT representation, which is computed analytically from the Laplacian. In the discrete TT representation, the kinetic energy operator is computed numerically in momentum space. Algebraic manipulations are followed by rounding schemes to avoid an artificial growth of the rank.

Furthermore, once in FT format, many other familiar operators may be applied to a function with relative ease. Consider the Laplacian $\Delta f(x) = g(x) = \sum_{k=1}^{d} \frac{\partial^2 f(x)}{\partial x_k^2}$, necessary for implementation of the kinetic energy operator without having to rely on the Fourier transform. Written in this form, one can consider the Laplacian as the summation of $d$ functions $g_k(x)$ in function-train format, where

$$g_k(x) = \frac{\partial^2 f(x)}{\partial x_k^2}$$

The second derivative is implemented core-by-core in the space of univariate functions. The second derivatives of univariate functions are computed only once, which exploits the benefits of the continuous representation and avoids the need for explicit calculation of the Fourier transform required by grid-based methods.
The minimal and maximal potential energy surface values required for rescaling the Hamiltonian in the Chebyshev scheme are determined either analytically or through constrained nonlinear optimization to avoid calculation of the multidimensional potential energy surface at all position-space grid points considered. Individual Chebyshev polynomials are determined as either tensor trains or function trains via the recurrence relation eq 6 or the action of the propagator on the wave function is determined directly from the Clenshaw algorithm, see Appendix A. The resultant dynamics is analyzed via calculation of survival amplitudes and wavepackets.

### 5. CHEMICAL MODEL

We simulate the dynamics of protons in a 50-dimensional model of hydrogen-bonded DNA adenine–thymine base pairs, described by the model potential energy surface\(^8^7,^8^8\)

\[
V(x_1, x_2, ..., x_d) = \sum_{i=1}^{D} \alpha(0.429x_i - 1.126x_i^2 - 0.143x_i^3 + 0.563x_i^4) + \sum_{i>1}^{D} \alpha\beta(x_{i-1})
\]

Figure 3. Comparison of two-dimensional slices of the 50-dimensional time-dependent wavepacket obtained from FTTC simulations (red line) and its discrete TT implementation (blue line) as compared to benchmark TT-SOFT (black) simulations of tautomerization quantum dynamics for uncoupled ($\beta = 0$ au) DNA base pairs.

Figure 4. Comparison of survival amplitudes from simulations of the time-dependent wavepacket evolving on a 50-dimensional potential energy surface for the tautomerization dynamics of uncoupled ($\beta = 0$ au) DNA base pairs, including the real (left) and imaginary (right) parts, obtained with FTTC (red line) and its discrete TT implementation (blue line) as compared to benchmark TT-SOFT (black).
where $\alpha = 0.1 \text{ au}$ determines the energy scaling of the model potential and $\beta$ is the hydrogen-bond coupling parameter (see Figure 2). Each $x_i$ describes the coordinate of proton motion in an individual adenine–thymine ($A$–$T$) pair as it tautomerizes from the energetically favored amino–keto $A$–$T$ form to the isomeric imino–enol $A^*$$-T^*$ form. The coupling term parameterized by $\beta$ provides a model of interaction between base pairs.

The resulting 50-dimensional model potential involves strongly anharmonic modes, which are beyond the reach of the grid-based Chebyshev approach or other quantum dynamic methods based on full-grid representations. The molecular system also provides a challenging test case for low-rank methods based on full-grid representations. The molecular model systems since they require computational resources that scale exponentially with dimensionality. Even the standard grid-based implementation of the Chebyshev method, renowned for its ability to achieve accuracy within machine precision, has been limited in applications to nuclear quantum dynamics to model systems with no more than four atoms. Here, we have shown how to extend the capabilities of the Chebyshev propagation scheme to high-dimensional systems in terms of the FTTC algorithm. We anticipate that the resulting FTTC methodology will be useful not only for simulations of quantum reaction dynamics in general but also as a general method to obtain numerical solutions of linear systems in high dimensionality, typically arising from space discretization in many other types of applications. Furthermore, the functional train decomposition should also find wide applicability in studies requiring computations of gradients, integrals, and correlation functions of systems with high dimensionality.

With regards to the basis functions, we note that the functional tensor-train representation can implement suitable choices of univariate basis functions that could be ideal for data compression in chemistry, for example, waveforms or Gaussian functions, which are common to both wavepacket propagation methods and electronic structure calculations alike. In general, representations that require $O(n^d)$ data points in a $d$-dimensional grid with $n$ points for each direction would require at most $O(dn^2r^2)$ data points for a maximum rank $r$ in a discrete tensor-train representation and only $O(dr^2)$ data points in functional tensor-train format where $r$ is the number of parameters, which represents a significant reduction in computational cost ideal for modeling molecular systems.

**Appendix A**

*Clenshaw Algorithm.* The direct computation of the Chebyshev expansion eq 25 based on the usual summation
algorithm has two disadvantages: (1) all summands have to be kept in the memory of the computer, which can be very expensive in practical applications since the tensors $T_k(\hat{\mathcal{H}}_0)W_0$ (and also their low-rank approximations) are typically large objects, and (2) it is known that the worst-case error generated by the floating point operations grows proportionally to the number $N$ of summands.\textsuperscript{89} We therefore use the Clenshaw algorithm,\textsuperscript{80} which offers a stable alternative to evaluate linear combinations of polynomials that satisfy a linear recurrence relation such as the Chebyshev polynomials.\textsuperscript{75}

Assuming that for given coefficients $c_0, c_1, ..., c_{N-1} \in \mathbb{C}$ we are interested in the value of the partial Chebyshev sum eq 13, the Clenshaw algorithm replaces the summation by the evaluation of the following backward recurrence system

$$
\begin{align*}
B_0(y) &= 2yB_{+1}(y) - B_{+2}(y) + c_r, \\
B_N(y) &= 0, \\
B_{N+1}(y) &= 0
\end{align*}
$$

and then expresses the partial Chebyshev sum as

$$
\sum_{k=0}^{N-1} (2 - \delta_{k,0})c_kT_k(y) = B_0(y) - B_2(y)
$$

To obtain the approximation of the wave function $\Psi(t)$, we adapted the Clenshaw algorithm by first solving the backward recurrence system

$$
\begin{align*}
\mathcal{B}_r &= 2\hat{\mathcal{H}}_0\mathcal{B}_{r+1} - \mathcal{B}_{r+2} + (-i)^r\int r^{-1}\mathcal{H}_0\mathcal{B}_0, \\
\mathcal{B}_N &= 0, \\
\mathcal{B}_{N-1} &= 0
\end{align*}
$$

and then computing the approximant

$$
\Psi(t) \approx e^{-i\mathcal{E}t} (\mathcal{B}_0 - \mathcal{B}_2)
$$

We note that this numerically stable procedure needs to keep only three tensors in memory.

Figure 6. Comparison of survival amplitudes from simulations of the time-dependent wavepacket evolving on a 50-dimensional potential energy surface for the tautomerization dynamics of coupled ($\beta = -2$ au) DNA base pairs, including the real (left) and imaginary (right) parts, obtained with FTTC (red line), its discrete $TT$ implementation (blue line), and benchmark TT-SOFT (black).

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.1c00941.

fttcs.zip (ZIP)

Accuracy of the computed wavefunction (Figure S1) and autocorrelation function (Figure S2) in TTC and TT-SOFT dynamics and $L^2$-norm error of the wavefunction in TTC as a function of the number of Chebyshev polynomials (Figure S3) for an analytically solvable benchmark system (PDF)

AUTHOR INFORMATION

Corresponding Author

Victor S. Batista – Yale Quantum Institute, Yale University, New Haven, Connecticut 06520-8263, United States; Department of Chemistry, Yale University, New Haven, Connecticut 06520, United States; Energy Sciences Institute, Yale University, West Haven, Connecticut 06516-7394, United States; orcid.org/0000-0002-3262-1237; Email: victor.batista@yale.edu

Authors

Micheline B. Soley – Yale Quantum Institute, Yale University, New Haven, Connecticut 06520-8263, United States; Department of Chemistry, Yale University, New Haven, Connecticut 06520, United States; orcid.org/0000-0001-7973-2842

Paul Bergold – Zentrum Mathematik, Technical University of Munich, 85748 Garching, Germany

Alex A. Gorodetsky – Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109-2140, United States

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.1c00941
Notes
The authors declare no competing financial interest.

ACKNOWLEDGMENTS
M.B.S. acknowledges financial support from the Yale Quantum Institute Postdoctoral Fellowship. V.S.B. acknowledges support from the NSF Grant no. CHE-1900160 and high-performance computing time from NERSC and the Yale High-Performance Computing Center. A.A.G. was supported by the AFOSR from the NSF Grant no. CHE-1900160 and high-performance Institute Postdoctoral Fellowship. V.S.B. acknowledges support from the Yale Quantum Institute Postdoctoral Fellowship. A.A.G. was supported by the AFOSR from the NSF Grant no. CHE-1900160 and high-performance Institute Postdoctoral Fellowship. V.S.B. acknowledges support from the Yale Quantum Institute Postdoctoral Fellowship. A.A.G. was supported by the AFOSR from the NSF Grant no. CHE-1900160 and high-performance Institute Postdoctoral Fellowship.

REFERENCES
(40) Greene, S. M.; Batista, V. S. Tensor-Train Split-Operator Fourier Transform (TT-SOFT) Method: Multidimensional Non-
Quantum Dynamics Simulations. https://github.com/michelinesoley/FTTC.

