Journal of Computational Physics 305 (2016) 557–574

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A low-rank approach to the computation of path integrals

Mikhail S. Litsarev [a], Ivan V. Oseledets [a,b]

[a] Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3, 143026 Moscow, Russia
[b] Institute of Numerical Mathematics of Russian Academy of Sciences, Gubkina St. 8, 119333 Moscow, Russia

ARTICLE INFO

Article history:
Received 25 April 2015
Received in revised form 3 November 2015
Accepted 5 November 2015
Available online 10 November 2015

Keywords:
Low-rank approximation
Feynman–Kac formula
Path integral
Multidimensional integration
Skeleton approximation
Convolution

ABSTRACT

We present a method for solving the reaction–diffusion equation with general potential in free space. It is based on the approximation of the Feynman–Kac formula by a sequence of convolutions on sequentially diminishing grids. For computation of the convolutions we propose a fast algorithm based on the low-rank approximation of the Hankel matrices. The algorithm has complexity of $\mathcal{O}(nrM\log M + nr^2M)$ flops and requires $\mathcal{O}(Mr)$ floating-point numbers in memory, where n is the dimension of the integral, $r \ll n$, and M is the mesh size in one dimension. The presented technique can be generalized to the higher-order diffusion processes.

1. Introduction

Path integrals [1–3] play a dominant role in description of a wide range of problems in physics and mathematics. They are a universal and powerful tool for condensed matter and high-energy physics, theory of stochastic processes and parabolic differential equations, financial mathematics, quantum chemistry and many others. Different theoretical and numerical approaches have been developed for their computation, such as the perturbation theory [4], the stationary phase approximation [5,6], the functional renormalization group [7,8], various Monte Carlo [9] and sparse grids methods [10,11]. The interested reader can find particular details in the original reviews and books [12–14].

In this paper we focus on the one-dimensional reaction–diffusion equation with initial distribution $f(x) : \mathbb{R} \to \mathbb{R}^+$ and a constant diffusion coefficient σ

$$\begin{cases} \dfrac{\partial}{\partial t}u(x,t) = \sigma \dfrac{\partial^2}{\partial x^2}u(x,t) - V(x,t)u(x,t), \\ u(x,0) = f(x) \end{cases} \qquad t \in [0,T], \quad x \in \mathbb{R}. \tag{1}$$

This equation may be treated in terms of a Brownian particle motion [15–17], where the solution $u(x,t) : \mathbb{R} \times [0,T] \to \mathbb{R}^+$ is the density distribution of the particles. The potential (or the dissipation rate) $V(x,t)$ is bounded from below. We do not consider the drift term $\rho \frac{\partial}{\partial x}u(x,t)$ because it can be easily excluded by a substitution $u(x,t) \to \tilde{u}(x,t)e^{-\rho x}$ [18].

E-mail address: m.litsarev@skoltech.ru (M.S. Litsarev).

http://dx.doi.org/10.1016/j.jcp.2015.11.009
0021-9991/

The solution of (1) can be expressed by the Feynman–Kac formula [18–20]

$$u(x,T) = \int\limits_{\mathcal{C}\{x,0;T\}} f(\xi(T)) e^{-\int_0^T V(\xi(\tau),T-\tau)d\tau} \mathcal{D}_\xi, \tag{2}$$

where the integration is done over the set $\mathcal{C}\{x,0;T\}$ of all continuous paths $\xi(T):[0,T]\to\mathbb{R}$ from the Banach space $\Xi([0,T],\mathbb{R})$ starting at $\xi(0)=x$ and stopping at arbitrary endpoints at time T. $\mathcal{D}_\xi$ is the Wiener measure, and $\xi(t)$ is the Wiener process [21,22]. One of the advantages of the formulation (2) is that it can be directly applied for the unbounded domain without any additional (artificial) boundary conditions.

Path integral (2) corresponding to the Wiener process is typically approximated by a *finite* multidimensional integral with the Gaussian measure (details are given in Section 2.1). The main drawback is that this integral is a high-dimensional one and its computation requires a special treatment. Several approaches have been developed to compute the multidimensional integrals efficiently. The sparse grid method [23,24] has been applied to the computation of path integrals in [25], but only for dimensions up to ~ 100, which is not enough in some applications. The main disadvantage of the Monte Carlo simulation is that it does not allow to achieve a high accuracy [26,27] for some cases (highly oscillatory functions, functions of sum of all arguments).

The multidimensional *integrand* can be represented numerically as a multidimensional array (*a tensor*), which contains values of a multivariate function on a fine uniform grid. For the last decades several approaches have been developed to efficiently work with tensors. They are based on the idea of *separation of variables* [28–31] firstly introduced in [32,33]. It allows to present a tensor in the *low-rank* or *low-parametric* format [34–36], where the number of parameters used for the approximation is almost linear (with respect to dimensionality). To construct such decompositions numerically the very efficient algorithms have been developed recently: two-dimensional *incomplete cross approximation*[1] for the skeleton decomposition, three-dimensional cross approximation [37] for the Tucker format [38–41] in 3D, *tt-cross* [42] approximation for the tensor train decomposition [43,44], which can be also considered as a particular case of the hierarchical Tucker format [45–47] for higher dimensional case. For certain classes of functions commonly used in the computational physics (multiparticle Schrödinger operator [48–53], functions of a discrete elliptic operator [54–59], Yukawa, Helmholtz and Newton potentials [60–63], etc.) there exist low-parametric representations in separated formats and explicit algorithms [64,65] to obtain and effectively work with them (especially *quantized tensor train* (QTT) format [66–73]). In many cases it is very effective to compute the multidimensional integrals [74] using separated representations [75], particularly for multidimensional convolutions [76–79] and highly oscillatory functions [80].

Our approach presented here is based on the *low-rank approximation* of matrices used in an essentially different manner. We formulate the Feynman–Kac formula as an iterative sequence of convolutions defined on grids of diminishing sizes. This is done in Section 3.2. To reduce the complexity of this computation, in Section 3.3 we find a *low-rank* basis set by applying the *cross approximation* (see Appendix A) to a matrix constructed from the values of a one-dimensional function on a very large grid. That gives reduction of computational time and memory requirements, resulting in fast and efficient algorithm presented in Section 3.4. The numerical examples are considered in Section 4. The most interesting part is that we are able to treat non-periodic potentials without any artificial boundary conditions (Section 4.3).

2. Problem statement

2.1. Time discretization

Equation (2) corresponds to the Wiener process. A standard way to discretize the path integral is to break the time range $[0,T]$ into n intervals by points

$$\tau_k = k\cdot\delta t, \qquad 0\le k\le n, \qquad n:\ \tau_n = T.$$

The average path of a Brownian particle $\xi(\tau_k)$ after k steps is defined as

$$\xi^{(k)} = \xi(\tau_k) = x + \xi_1 + \xi_2 + \ldots + \xi_k,$$

where every random step ξ_i, $1\le i\le k$, is independently taken from a normal distribution $\mathcal{N}(0,2\sigma\delta t)$ with zero *mean* and *variance* equal to $2\sigma\delta t$. By definition, $\xi^{(0)}=x$.

Application of a suitable quadrature rule on the uniform grid (i.e., trapezoidal or Simpson rules) with the weights $\{w_i\}_{i=0}^n$ to the time integration in (2) gives

$$\Lambda(T) = \int\limits_0^T V(\xi(\tau),T-\tau)d\tau \approx \sum_{i=0}^n w_i V_i^{(n)}\delta t, \qquad V_i^{(n)} \equiv V(\xi(\tau_i),\tau_{n-i}), \tag{3}$$

[1] Because the low-rank representation of large matrices based on the adaptive cross approximation is directly related to the manuscript we summarize the basics of the method in Appendix A.

and transforms the exponential factor to the approximate expression

$$e^{-\Lambda(T)} \approx \prod_{i=0}^{n} e^{-w_i V_i^{(n)} \delta t}.$$

The Wiener measure, in turn, transforms to the ordinary n-dimensional measure

$$\mathcal{D}_{\xi}^{(n)} = \left(\frac{\lambda}{\pi}\right)^{\frac{n}{2}} \prod_{k=1}^{n} e^{-\lambda \xi_k^2} \, d\xi_k, \qquad \lambda = \frac{1}{4\sigma\delta t},$$

and the problem reduces to an n-dimensional integral over the Cartesian coordinate space. Thus, we can approximate the exact solution (2) by $u^{(n)}(x,t)$

$$u(x,T) = \lim_{n\to\infty} u^{(n)}(x,T)$$

written in the following compact form

$$u^{(n)}(x,T) = \int_{-\infty}^{\infty} \mathcal{D}_{\xi}^{(n)} \, f\left(\xi^{(n)}\right) \prod_{i=0}^{n} e^{-w_i V_i^{(n)} \delta t}. \tag{4}$$

The integration sign here denotes an n-dimensional integration over the particle steps ξ_k, and $V_i^{(n)}$ is defined in (3). The convergence criterion in terms of n for the sequence (4) is discussed and proven in [17, p. 33]. The limit of (4) exists if it is a Cauchy sequence.

Our goal is to compute the integral (4) numerically in an efficient way.

3. Computational technique

3.1. Notations

In this paper vectors (written in columns) are denoted by boldface lowercase letters, e.g., $\mathbf{a}$, matrices are denoted by boldface capital letters, e.g., $\mathbf{A}$. The i-th element of a vector $\mathbf{a}$ is denoted by a_i, the element (i,j) of a matrix $\mathbf{A}$ is denoted by A_{ij}. A set of vectors $\mathbf{a}_m$, $m_0 \le m \le m_1$ is denoted by $\{\mathbf{a}_m\}_{m=m_0}^{m_1}$, and the i-th element of a vector $\mathbf{a}_m$ is denoted by a_{mi}.

Definition 1. Let $\mathbf{a} \in \mathbb{R}^k$ and $\mathbf{b} \in \mathbb{R}^m$ be vectors and $k \ge m$. We say that a vector $\mathbf{c} \in \mathbb{R}^{m+k-1}$ is a *convolution* of two ordered vectors $\mathbf{a}$ and $\mathbf{b}$ and write

$$\mathbf{c} = \mathbf{a} \circ \mathbf{b},$$

if $\mathbf{c}$ has the following components

$$c_i = \sum_{j=0}^{m-1} a_{i+j} b_j, \qquad a_i = 0, \forall i : \{i \mid i < 0 \vee i >= k\}.$$

The computation of the convolution can be naturally carried out as a multiplication by the *Hankel matrix*.

Definition 2. We say that the *Hankel matrix* $\mathbf{A} \in \mathbb{R}^{k\times k}$ is generated by row $\mathbf{a}^T \in \mathbb{R}^k$ and column $\mathbf{b} \in \mathbb{R}^{k-1}$, and denote this by

$$\mathbf{A} = [\mathbf{a}^T, \mathbf{b}]_H,$$

if

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{k-2} & a_{k-1} \\ a_1 & a_2 & a_3 & \cdots & a_{k-1} & b_0 \\ a_2 & a_3 & a_4 & \cdots & b_0 & b_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k-2} & a_{k-1} & b_0 & \cdots & b_{k-4} & b_{k-3} \\ a_{k-1} & b_0 & b_1 & \cdots & b_{k-3} & b_{k-2} \end{pmatrix},$$

$$\mathbf{a}^T = \left(a_0, a_1, a_2, \ldots, a_{k-2}, a_{k-1}\right), \qquad \mathbf{b} = \left(b_0, b_1, b_2, \ldots, b_{k-2}\right)^T. \tag{5}$$

This compact notation will be used to compute convolutions (when they are written as a Hankel matrix-vector products). As it can be directly verified, $\forall \alpha \in \mathbb{R}$

$$\alpha \cdot \mathbf{A} = [\alpha \cdot \mathbf{a}^T, \alpha \cdot \mathbf{b}]_H. \tag{6}$$

Definition 3. For two vectors $\mathbf{a}$ and $\mathbf{b}$ from (5) for the case $a_i = b_i$, $\forall i : 0 \le i < k-1$, we will also write

$$\mathbf{a} = \begin{pmatrix} \mathbf{b} \\ a_{k-1} \end{pmatrix}. \tag{7}$$

This notation will be used when vector $\mathbf{b}$ is a subvector of $\mathbf{a}$.

3.2. Multidimensional integration via the sequence of one-dimensional convolutions

Multidimensional integral (4) can be represented in terms of an iterative sequence of one-dimensional convolutions. Indeed, for a one-dimensional function $F_k^{(n)}(x)$, such that

$$F_k^{(n)}(x) = \sqrt{\frac{\lambda}{\pi}} \int_{-\infty}^{\infty} \Phi_{k+1}^{(n)}(x+\xi)\, e^{-\lambda \xi^2} d\xi, \qquad x \in \mathbb{R}, \qquad k = n, n-1, \ldots, 1, \tag{8}$$

with

$$\Phi_{k+1}^{(n)}(x) = F_{k+1}^{(n)}(x)\, e^{-w_k V(x, \tau_{n-k})\delta t}, \tag{9}$$

and the initial condition

$$F_{n+1}^{(n)}(x) = f(x), \tag{10}$$

the solution (4) reads

$$u^{(n)}(x, T) = F_1^{(n)}(x)\, e^{-w_0 V(x,T)\,\delta t}. \tag{11}$$

The iteration starts from $k = n$ and goes down to $k = 1$. Since the function $\Phi_k^{(n)}(x)$ is bounded and the convolution (8) contains the exponentially decaying Gaussian, the integral has finite lower and upper bounds. Consider

$$F_k^{(n)}(x) \approx \tilde{F}_k^{(n)}(x) = \sqrt{\frac{\lambda}{\pi}} \int_{-a_x}^{a_x - h_x} \tilde{\Phi}_{k+1}^{(n)}(x+\xi)\, e^{-\lambda \xi^2} d\xi. \tag{12}$$

We suppose that the product $\Phi_{k+1}^{(n)}(x+\xi)\, e^{-\lambda \xi^2}$ rapidly decays, so that for a_x large enough, we can approximate the integral $F_k^{(n)}(x)$ in (8) by $\tilde{F}_k^{(n)}(x)$ and assume that this approximation has an error ε in some norm

$$\left\| F_k^{(n)}(x) - \tilde{F}_k^{(n)}(x) \right\| < \varepsilon.$$

This approximation has an important drawback: as soon as $F_1^{(n)}(x)$ has to be computed on the semi-open interval $[-a_x, a_x)$, the domain of $F_n^{(n)}(x)$ should be taken larger, i.e. $[-na_x, na_x)$ for n steps, because of the convolution structure of the integral (12). Indeed, if we suppose, that the function $F_k^{(n)}(x)$ is computed on the uniform mesh

$$x_i^{(k)} = -ka_x + ih_x, \qquad 0 \le i < kM, \qquad h_x = a_x / N_x, \qquad M = 2N_x, \tag{13}$$

and the integration mesh is chosen to be nested in (13) with the *same step* h_x

$$\xi_j = -a_x + jh_x, \qquad 0 \le j < M, \tag{14}$$

then the function $F_{k+1}^{(n)}(x)$ is defined on the mesh

$$x_i^{(k+1)} = -(k+1)a_x + ih_x, \qquad 0 \le i < (k+1)M, \tag{15}$$

and

$$x_{i+j}^{(k+1)} = x_i^{(k)} + \xi_j. \tag{16}$$

The last equality follows from definitions (13) and (14). This is illustrated in Fig. 1.

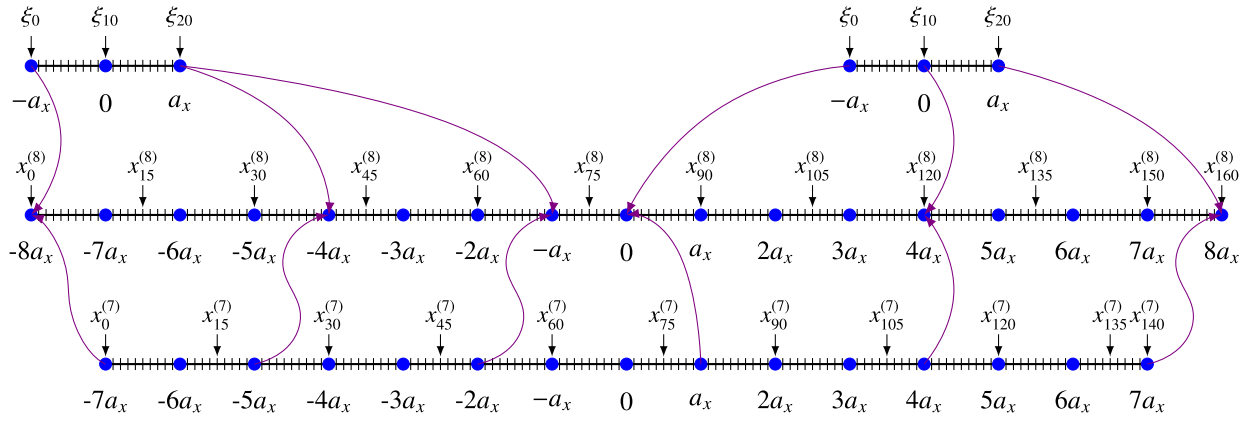

Fig. 1. A correspondence of meshes for two nearest iterations $x_{i+j}^{(k+1)} = x_i^{(k)} + \xi_j$ from equation (16) for $k = 7$. Blue filled circles separate the ranges corresponding to different steps m, $1 \le m \le k$ in time $[-ma_x, ma_x)$. Ticks on the axes label the mesh points. Violet curved lines show correspondence (16) between the two meshes for nearest iterations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The integral (12) can be calculated for every fixed $x_i^{(k)}$ of the mesh (13) as the quadrature sum with the weights $\{\mu_j\}_{j=0}^{M-1}$

$$\tilde{F}_k^{(n)}\left(x_i^{(k)}\right) \approx \sum_{j=0}^{M-1} \mu_j \tilde{\Phi}_{k+1}^{(n)}\left(x_{i+j}^{(k+1)}\right) p(\lambda, \xi_j), \qquad p(\lambda,\xi) = \sqrt{\frac{\lambda}{\pi}} e^{-\lambda\xi^2} \tag{17}$$

$$\tilde{\Phi}_{k+1}^{(n)}\left(x_i^{(k+1)}\right) = \tilde{F}_{k+1}^{(n)}\left(x_i^{(k+1)}\right) e^{-w_k V(x_i^{(k+1)}, \tau_{n-k})\delta t} \tag{18}$$

The complexity of the computation of $\tilde{F}_k^{(n)}\left(x_i^{(k)}\right)$ for all i is $\mathcal{O}(kN_x^2)$ flops. It can be reduced to $\mathcal{O}(kN_x \log N_x)$ by applying the Fast Fourier transform (FFT) for convolution (17). Full computation of $\tilde{F}_1^{(n)}\left(x_i^{(1)}\right)$ costs $\mathcal{O}(n^2 N_x \log N_x)$ operations and $\mathcal{O}(nN_x)$ floating-point numbers. This complexity becomes prohibitive for large n (i.e., for small time steps), but can be reduced. Below we present a fast approximate method for the calculation of $\tilde{F}_1^{(n)}\left(x_i^{(1)}\right)$ in $\mathcal{O}(nrN_x \log N_x + nr^2 N_x)$ flops and $\mathcal{O}(rN_x)$ memory cost with $r \ll n$, by applying low-rank decompositions.

3.3. Low-rank basis set for the convolution array

In this section we provide a theoretical justification for our approach. Consider a sequence of matrices $\mathbf{A}^{(k)} \in \mathbb{R}^{kM \times M}$ corresponding to the iterative process (17) and constructed in the following way

$$A_{ij}^{(k)} = a_{i+j}^{(k)} \equiv \tilde{\Phi}_k^{(n)}\left(x_{i+j}^{(k)}\right), \qquad 0 \le j < M, \qquad 0 \le i < kM, \tag{19}$$

where k is the iteration number.

Let us now consider iteration (17) at the step $k = k_0$ and for simplicity omit the index $k_0 + 1$ in the matrix and mesh notations (19). Let us also denote the sum (17) for x_i taken from the grid (13) by $s_i = \tilde{F}_{k_0}^{(n)}(x_i)$ and set $p_j \equiv \mu_j p(\lambda, \xi_j)$. Then

$$s_i = \sum_{j=0}^{M-1} A_{ij}\, p_j \qquad \Leftrightarrow \qquad \mathbf{s} = \mathbf{A}\mathbf{p}. \tag{20}$$

The equality (20) establishes the recurrence relation between iterations at the step k (the right-hand side) and the step $k-1$ (the left-hand side) according to (17) and (18), see Fig. 2.

The matrix $\mathbf{A}$ is a Hankel matrix, as follows from definition (19), and consists of k square blocks $\mathbf{H}_m$, $0 \le m < k$, such that

$$\mathbf{A}^T = \left(\mathbf{H}_0, \mathbf{H}_1 \ldots \mathbf{H}_{k-1}\right). \tag{21}$$

Here, every block $\mathbf{H}_m$ is a Hankel matrix as well *generated* by the upper row $\mathbf{l}_m^T$ and the right column $\mathbf{r}_{m+1}$ correspondingly:

$$\mathbf{H}_m = \left[\mathbf{l}_m^T, \mathbf{r}_{m+1}\right]_H$$

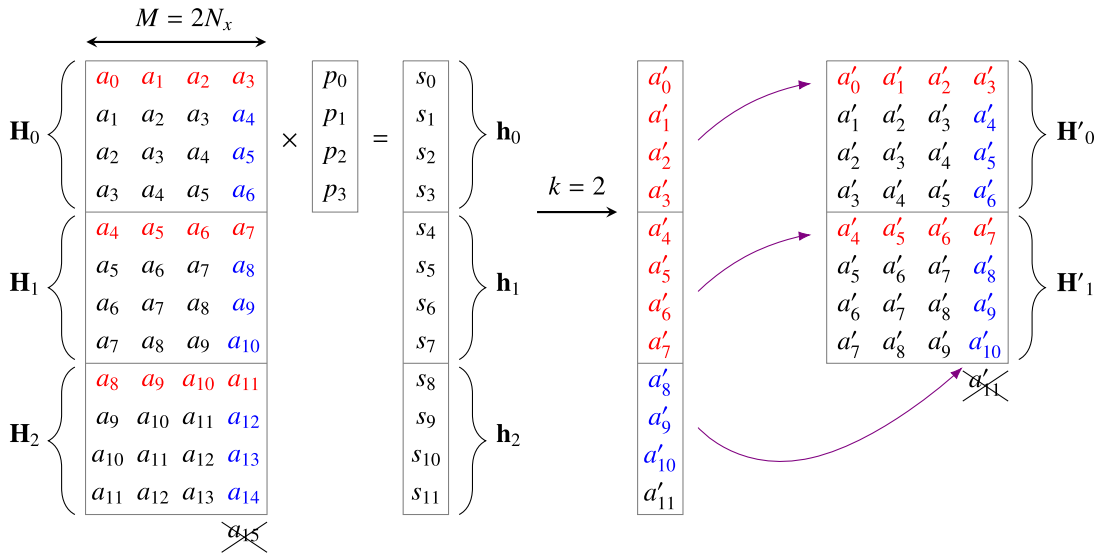

Fig. 2. Transition between two neighbour iterations is illustrated. The left-hand-side matrix $\mathbf{A}$ is multiplied by vector $\mathbf{p}$ in a resulting vector $\mathbf{s}$ according to (20). Explicit structure of matrix blocks $\mathbf{H}_m$ in (21) and vector blocks $\mathbf{h}_m$ in (24) is shown. Then entries of vector $\mathbf{s}$ are multiplied by corresponding factor $e^{-w_k V(x_i,\tau_{n-k})\delta t}$ to produce the next iteration step (9). From a new vector $\mathbf{a}'$ there formed a new matrix $\mathbf{A}'$ according to (19). The last point from the previous iteration is not needed and is thrown out. Then the steps repeat for the next iteration.

(the notation $[\mathbf{a},\mathbf{b}]_H$ is introduced in Section 3.1, Definition 2), where

$$\begin{aligned}\mathbf{l}_m^T &= (a_{i_0}, a_{i_0+1}, \ldots, a_{i_0+M-2}, a_{i_0+M-1}),\\ \mathbf{r}_m^T &= (a_{i_0}, a_{i_0+1}, \ldots, a_{i_0+M-2}),\end{aligned} \qquad i_0 = m\cdot M, \qquad 0 \le m < k, \tag{22}$$

by the definition, see Fig. 2. It can also be represented as a sum of two anti-triangular[2] Hankel matrices

$$\mathbf{H}_m = \mathbf{L}_m + \mathbf{R}_m, \qquad \mathbf{L}_m = \left[\mathbf{l}_{m\cdot M}^T, \mathbf{0}\right]_H, \qquad \mathbf{R}_m = \left[\mathbf{0}^T, \mathbf{r}_{(m+1)\cdot M}\right]_H, \tag{23}$$

where the upper-left $\mathbf{L}_m$ has nonzero anti-diagonal and the bottom-right $\mathbf{R}_m$ has zero anti-diagonal according to (22).

Equation (20) may be rewritten in the block form (see again Fig. 2)

$$\mathbf{h}_m = \mathbf{H}_m\mathbf{p}, \qquad \mathbf{h}_m^T = (s_{m\cdot M}, s_{m\cdot M+1} \ldots s_{m\cdot M+M-1}). \tag{24}$$

Here every block $\mathbf{H}_m$ is multiplied by the same vector $\mathbf{p}$. The number of matrix-vector multiplications can be reduced, if the dimension d of the linear span $\mathcal{H} = \{\mathbf{H}_m\}_{m=0}^{k-1}$ is less than k. Before estimation of the dimension we formulate some auxiliary lemmas (proven in the Appendix C).

Lemma 1. *Let $\{\mathbf{u}_i\}_{i=0}^{r_1-1}$ be a basis set of span $\{\mathbf{l}_m\}_{m=0}^{k-1}$, $r_1 \le k$, and let matrix $\mathbf{U}_i = \left[\mathbf{u}_i^T, \mathbf{0}\right]_H$. Then $\{\mathbf{U}_i\}_{i=0}^{r_1-1}$ is a basis set of span $\{\mathbf{L}_m\}_{m=0}^{k-1}$ from (23).*

Lemma 2. *Let $\{\mathbf{w}_i\}_{i=0}^{r_2-1}$ be a basis set of span $\{\mathbf{r}_m\}_{m=1}^{k}$, $r_2 \le k$, and let matrix $\mathbf{W}_i = \left[\mathbf{0}^T, \mathbf{w}_i\right]_H$. Then $\{\mathbf{W}_i\}_{i=0}^{r_2-1}$ is a basis set of span $\{\mathbf{R}_m\}_{m=0}^{k-1}$ from (23).*

Lemma 3. *Let $\{\mathbf{u}_i\}_{i=0}^{r_1-1}$ be a basis set of span $\{\mathbf{l}_m\}_{m=0}^{k}$, such that $\mathbf{u}_i^T = (\mathbf{w}_i^T, u_{i,(M-1)})$ according to (7). Then $\{\mathbf{w}_i\}_{i=0}^{r_1-1}$ is a basis set of span $\{\mathbf{r}_m\}_{m=1}^{k}$.*

Let us define a basis set $\{\mathbf{Q}\}_{i=0}^{2r-1}$ as follows

$$\mathbf{Q}_i = \begin{cases}\mathbf{U}_i, & 0 \le i < r\\ \mathbf{W}_{i-r}, & r \le i < 2r\end{cases} \tag{25}$$

An obvious corollary of the previous lemma is the following theorem.

[2] By anti-triangular matrix we call a matrix which is triangular with respect to the anti-diagonal of the matrix.

Theorem 1. *The dimension of the linear span of matrices* $\{\mathbf{H}_m\}_{m=0}^{k-1}$ *is equal to* $2r$. *Moreover, it is contained in the linear span of the matrices* $\{\mathbf{Q}_i\}_{i=0}^{2r-1}$ *defined in* (25).

Proof. The matrix $\mathbf{H}_m$ can be written as a sum (23), $\mathbf{H}_m = \mathbf{L}_m + \mathbf{R}_m$. According to *Lemma 1*, the set $\{\mathbf{U}_i\}_{i=0}^{r-1}$ is a basis set of the span $\{\mathbf{L}_m\}_{m=0}^{k-1}$. By *Lemma 3*, $\{\mathbf{w}_i\}_{i=0}^{r-1}$ is a basis set of the span $\{\mathbf{r}_m\}_{m=0}^{r-1}$, and by *Lemma 2*, $\{\mathbf{W}\}_{i=0}^{r-1}$ is a basis set of $\{\mathbf{R}\}_{m=0}^{k-1}$. The subspaces $\{\mathbf{U}_i\}_{i=0}^{r-1}$ and $\{\mathbf{W}_i\}_{i=0}^{r-1}$ contain only zero matrix in common, so the dimension of the basis is $2r$. □

Lemma 4. *Let* $\{\mathbf{u}_i\}_{i=0}^{r-1}$ *be a basis set of span* $\{\mathbf{l}_m\}_{m=0}^{k}$. *Then for basis matrices* $\{\mathbf{Q}_i\}_{i=0}^{2r-1}$ *defined in* (25) *the computation of the matrix-by-vector products*

$$\mathbf{k}_i = \mathbf{U}_i\mathbf{p}, \qquad \mathbf{t}_i = \mathbf{W}_i\mathbf{p}, \tag{26}$$

costs $\mathcal{O}(M \log M)$ *flops for a fixed* $0 \le i < r$.

Proof. Consider a Hankel matrix

$$\mathbf{G}_i = \begin{pmatrix} \mathbf{W}_i \\ \mathbf{U}_i \end{pmatrix}.$$

A product $\mathbf{G}_i\mathbf{p}$ is a result of the convolution $\mathbf{u}_i \circ \hat{\mathbf{p}}$, which can be done by the FFT [81,82] procedure in $\mathcal{O}(M \log M)$ flops for a fixed $0 \le i < r$. The vector $\hat{\mathbf{p}} = (p_{M-1}, \ldots, p_1, p_0)^T$ is taken in the reverse order. □

Once the basis $\{\mathbf{Q}_i\}_{i=0}^{d-1}$ for the span of $\mathcal{H}$ is found, the complexity of the multiplication $\mathbf{Ap}$ in (20) can be estimated as follows.

Theorem 2. *Let the set* $\{\mathbf{Q}_i\}_{i=0}^{2r-1}$ *defined in* (25) *be a basis set of the linear span* $\mathcal{H}$ *generated by the set of Hankel matrices* $\mathbf{H}_m$ *defined in* (21). *Then the computation of any* K_s *elements* s_i *of the vector* $\mathbf{s}$ (20) *costs* $\mathcal{O}(rM \log M + r^2 M)$ *flops for* $K_s = \mathcal{O}(rM)$.

Proof. Indeed, by the assumption $\mathbf{H}_m = \sum_{i=0}^{2r-1} c_{mi}\mathbf{Q}_i$ for each m, $0 \le m < k$. The complexity of the product $\mathbf{Q}_i\mathbf{p}$, $0 \le i < 2r$ for a fixed i is $\mathcal{O}(M \log M)$ flops by Lemma 4. The computation of such products for all i takes $\mathcal{O}(rM \log M)$ flops.

The vector $\mathbf{h}_m$, which is a subvector of $\mathbf{s}$, is represented via few matrix-by-vector products (26) as follows

$$\mathbf{h}_m = \mathbf{H}_m\mathbf{p} = \mathbf{L}_m\mathbf{p} + \mathbf{R}_m\mathbf{p} = \sum_{i=0}^{r} \alpha_{mi}\mathbf{U}_i\mathbf{p} + \beta_{mi}\mathbf{W}_i\mathbf{p} = \sum_{i=0}^{r} \alpha_{mi}\mathbf{k}_i + \beta_{mi}\mathbf{t}_i. \tag{27}$$

The computation of its i-th component h_{mi} takes $\mathcal{O}(r)$ flops for any m. Computation of $\mathcal{O}(rM)$ components s_j of the vector $\mathbf{s}$, which are also the components of the particular vector $\mathbf{h}_m$ (for $m = \left\lfloor \frac{j}{M} \right\rfloor$), costs, in turn, $\mathcal{O}(r^2 M)$ flops. Finally, $\mathcal{O}(rM \log M + r^2 M)$. □

Remark 1. Each component of the resulting vector can be computed by the formula

$$s_j = h_{m_j l_j} = \sum_{i=0}^{r} \alpha_{m_j i} k_{i l_j} + \beta_{m_j i} t_{i l_j}, \qquad m_j = \left\lfloor \frac{j}{M} \right\rfloor, \qquad l_j = j \bmod M. \tag{28}$$

Here $k_{i l_j}$ is the l_j-th component of the vector $\mathbf{k}_i$ and $t_{i l_j}$ is the l_j-th component of the vector $\mathbf{t}_i$.

Remark 2. It follows from Lemma 3 that $\alpha_{i+1,j} = \beta_{ij}$ in (27).

3.4. Final algorithm

To compute $\tilde{F}_1^{(n)}(x_i)$, which defines the final solution (11) on the mesh (13), one needs to carry out iterations (17) starting from $k = n$ down to 1. At each iteration step k we construct a function $f_k\left(x_i^{(k)}\right)$, which *approximates* the entries s_i^k in equation (28) as follows. Suppose, that the function $f_{k+1}\left(x_i^{(k+1)}\right)$ has been already constructed at the previous step $k+1$. Then, to compute $f_k\left(x_i^{(k)}\right)$ at the current iteration k, we consider[3] the matrix $\mathbf{\Phi}^{(k+1)}$ with the entries

[3] But do not compute all its elements.

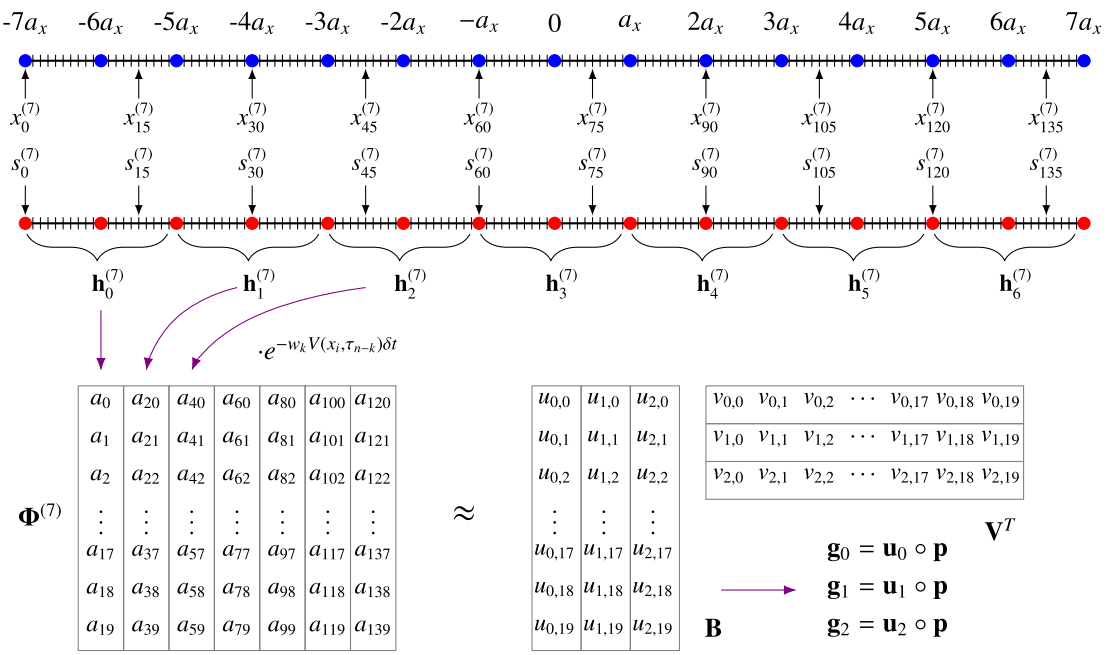

Fig. 3. Construction of matrix $\mathbf{\Phi}^{(k)}$ from a one-dimensional convolution (20) according to algorithm in Section 3.4. On a spatial homogeneous mesh (13) the corresponding entries of vector $\mathbf{s}$ (20) are calculated. By definition, vector $\mathbf{s}$ is composed from vectors $\mathbf{h}_m$ (24). Each column of the matrix $\mathbf{\Phi}^{(k)}$ is composed of $\mathbf{h}_m$ multiplied by a corresponding factor $e^{-w_k V(x_i,\tau_{n-k})\delta t}$. Then this matrix is decomposed by a *cross approximation* $\mathbf{\Phi}^{(k)} = \mathbf{B}\mathbf{V}^T$ (30). For the approximation there needed only some elements of matrix $\mathbf{\Phi}^{(k)}$, which are chosen adaptively and computed on-the-fly. Then convolutions $\mathbf{g}_i = \mathbf{u}_i \circ \mathbf{p}$ are calculated via fast Fourier transform and saved in the memory. Particular values of $s_i^{(k)}$ for the next iteration step $k-1$ can be computed by formula (28) then.

$$\Phi_{ij}^{(k+1)} = f_{k+1}\left(y_{ij}\right) e^{-w_k V(y_{ij},\tau_{n-k})\delta t}, \qquad y_{ij} = x_{i+j\cdot M}^{(k+1)}, \tag{29}$$

and apply the *cross approximation* (A.2) to this matrix. The columns of this matrix are vectors $\mathbf{h}_m^{(k+1)}$ element-wise multiplied by the corresponding exponential factor with the potential (29), see Fig. 3. The algorithm of the cross approximation requires only $\mathcal{O}(rM)$ entries, which are being chosen adaptively. They are calculated by the function $f_{k+1}\left(y_{ij}\right)$ on-the-fly for the particular points y_{ij}. Thus,

$$\mathbf{\Phi}^{(k+1)} = \mathbf{B}\mathbf{V}^T, \qquad \mathbf{B} \in \mathbb{R}^{M\times r}, \qquad \mathbf{V} \in \mathbb{R}^{(k+1)\times r}, \qquad r \ll M, \tag{30}$$

where $\mathbf{B}$ and $\mathbf{V}$ are matrices of the rank r saved in memory. By construction, the m-th column of matrix $\mathbf{\Phi}^{(k+1)}$ is the vector $\mathbf{l}_m$ from (22) and the i-th column of matrix $\mathbf{B}$ is the basis vector $\mathbf{u}_i$ from Lemma 1. Hence, $\mathbf{V}^T$ is the matrix of coefficients of the decomposition (C.1). Once the cross approximation (30) is obtained, the memory allocated for all data structures related to $f_{k+1}\left(x_i^{(k+1)}\right)$ can be overwritten at the next iteration.

Computation of the circulant matrix-vector products (26) is done according to Lemma 4 by the convolution $\mathbf{g}_i = \mathbf{b}_i \circ \hat{\mathbf{p}}$, where $\mathbf{b}_i$ is a column of the matrix $\mathbf{B}$. The vectors $\mathbf{g}_i = (\mathbf{t}_i, \mathbf{k}_i)^T$ are also saved in the memory. Then $f_k\left(x_i^{(k)}\right)$ is calculated by equation (28), and the algorithm proceeds to the next iteration.

At some iteration step k the rank of the decomposition (30) will reach the number of columns and from this iteration it will be more efficient to carry out the convolution (20) without low-rank approximation. Complexity of one iteration of the presented algorithm is estimated in Theorem 2. Finally, for all n steps it is $\mathcal{O}(nrM\log M + nr^2M)$ flops, $r \ll n$. The standard FFT based algorithm applied to the whole array without any low-rank compression at each step gives complexity for all n steps equal to $\mathcal{O}(n^2 M\log M)$ flops. We illustrate these theoretical estimations by the example from Section 4.2 in Fig. 4.

Basically, the asymptotic complexity, proven in Theorem 2 is practically useful for $r \ll n$. This is the main assumption for the matrix from (29). Existence of such an approximation (and the properties of the initial problem) is in general still an open question. Some particular cases were studied in [83]. It was shown, that the cross approximation converges for matrices having singular vectors satisfying the *coherence* property. Some estimations can be found in [84–86] also. There is a theoretical idea how to identify the existence of the low-rank structure of a given matrix generated by a one-dimensional target function *a priori* (see [87] and Appendix B for details).

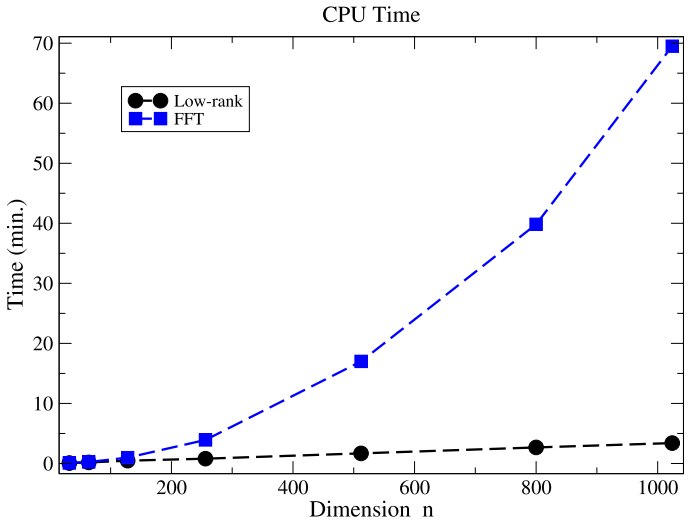

Fig. 4. A numerical illustration of theoretical estimations for the example from Section 4.2. For a standard FFT based algorithm applied to the whole array the $\mathcal{O}(n^2 M \log M)$ flops complexity is labeled by square points. The low-rank complexity $\mathcal{O}(nrM \log M + nr^2 M)$ flops is labeled by circles. The time is scaled in minutes, n is the number of dimensions (iteration steps), $M = 8000$, $r = 10$.

4. Numerical experiments and discussions

4.1. Harmonic oscillator

As a first example, let us consider a model system, which can be solved analytically, with the initial condition $f_{ho}(x)$ and the dissipation rate $V_{ho}(x,t)$ defined as

$$f_{ho}(x) = p(\beta, x) = \sqrt{\frac{\beta}{\pi}} e^{-\beta x^2}, \qquad V_{ho}(x,t) = \frac{x^2}{t+1}. \tag{31}$$

According to equation (11) the exact solution $u_{ho}^{(n)}(x,t)$ for the particular case (31) has the following form (see Appendix D for derivation)

$$u_{ho}^{(n)}(x,t) = \Psi_1^{(n)}(x)\, e^{-w_0 V(x,t)\, \delta t}. \tag{32}$$

Comparison of the numerical low-rank solution with the exact one (32) gives the relative error

$$\epsilon = \left\| \tilde{\mathbf{u}} - \mathbf{u} \right\| \Big/ \left\| \mathbf{u} \right\|, \tag{33}$$

which in the order of magnitude is equal to the machine precision, where $\tilde{\mathbf{u}}$ is an approximate solution on the final mesh and $\mathbf{u}$ is the exact one on the same mesh. For our example

$$\tilde{u}_i = \tilde{F}_1^{(n)}(x_i)\, e^{-w_0 V_{ho}(x_i, T)\, \delta t}, \qquad u_i = \Psi_1^{(n)}(x_i)\, e^{-w_0 V_{ho}(x_i, T)\, \delta t}.$$

Here $\sigma = 0.25$, $T = 10$, $n = 100$, and the mesh is a uniform one on $[-2, 2]$ with $M = 2N_x = 8000$ points. It is interesting that the scheme is exact for this case.

4.2. Cauchy distribution

The second example is taken from [25] and is interesting because it can be solved analytically as well. For $V_c(x,t)$ and initial condition $f_c(x)$ such that

$$V_c(x,t) = -\frac{1}{t+1} + 2\sigma \frac{3x^2 - 1}{(x^2+1)^2}, \qquad f_c(x) = \frac{1}{\pi} \frac{1}{x^2+1}, \tag{34}$$

the exact solution is

$$u_c(x,t) = \frac{1}{\pi} \frac{t+1}{x^2+1}.$$

Table 1
Convergence rate for system (34). Accuracy of the cross approximation $\varepsilon = 10^{-10}$. Direct convolutions start from $n = 20$, $\sigma = 0.5$, range of final spatial domain is $[-2, 2)$, $N_x = 4000$. Dimension of the integral (4) is labeled by n, δt is a time step, T is a final time for solution $u(x, T)$, ϵ is an error estimated by the Richardson extrapolation, and p is the order of the scheme for δt. Ranks of the matrix $\mathbf{\Phi}^{(k)}$ from (30) are presented in column labeled by r. The CPU time for computation of the integral (4) in *all points* of the mesh is reported in the last column.

T	n	δt	p	ϵ	r	*CPU time* (min.)
1.0	32	$3.1 \cdot 10^{-2}$	–	$2.8 \cdot 10^{-4}$	10	0.1
	64	$1.6 \cdot 10^{-2}$	–	$7.0 \cdot 10^{-5}$	10	0.2
	128	$7.8 \cdot 10^{-3}$	1.997	$1.8 \cdot 10^{-5}$	10	0.4
	256	$3.9 \cdot 10^{-3}$	1.999	$4.4 \cdot 10^{-6}$	10	0.9
	512	$2.0 \cdot 10^{-3}$	2.0	$1.1 \cdot 10^{-6}$	10	1.8
	1024	$9.8 \cdot 10^{-4}$	2.0	$2.8 \cdot 10^{-7}$	10	3.8
20.0	32	$6.3 \cdot 10^{-1}$	–	$4.1 \cdot 10^{-1}$	10	0.1
	64	$3.1 \cdot 10^{-1}$	–	$1.6 \cdot 10^{-1}$	10	0.2
	128	$1.6 \cdot 10^{-1}$	1.10	$4.8 \cdot 10^{-2}$	10	0.4
	256	$7.8 \cdot 10^{-2}$	1.68	$1.2 \cdot 10^{-2}$	10	0.9
	512	$3.9 \cdot 10^{-2}$	1.93	$3.1 \cdot 10^{-3}$	10	1.9
	1024	$2.0 \cdot 10^{-2}$	1.98	$7.9 \cdot 10^{-4}$	10	4.0
	2048	$9.8 \cdot 10^{-3}$	1.995	$2.0 \cdot 10^{-4}$	10	8.0
	4096	$4.9 \cdot 10^{-3}$	1.999	$4.9 \cdot 10^{-5}$	10	16.8
	8192	$2.4 \cdot 10^{-3}$	2.0	$1.2 \cdot 10^{-5}$	10	37.5

In Table 1 we present numerical results demonstrating the numerical order of scheme by the Runge formula

$$p = \log_2 \frac{\left\| \mathbf{u}_n - \mathbf{u}_{n/2} \right\|}{\left\| \mathbf{u}_{n/2} - \mathbf{u}_{n/4} \right\|},$$

with respect to δt and the timings for the whole computation. Here $\mathbf{u}_n$ is the computed solution at the final step in time.

Using our approach, it becomes possible to calculate $u^{(n)}(x, t)$ for large values of final time T due to the *low-rank* approximation of matrices $\mathbf{\Phi}^{(k)}$ composed from the columns of the integrand values (see Section 3.4). That significantly reduces the computational cost. For an example, for the last row of Table 1 iterations start from the calculation of the convolution on the range $[-16\,386, 16\,386)$ with 32 772 000 mesh points. This is reduced to the calculation of 10 (the rank) convolutions of two arrays with 8000 elements.

As it can be seen from our results, the scheme has the second order in time. It can be improved to higher orders by Richardson extrapolation on fine meshes [88,89]. Another way is to use other path integral formulations with high-order propagators [90,91].

4.3. Nonperiodic potential with impurity

The dissipation rate $V(x, t)$ causes the creation and annihilation of diffusing particles, as it follows from the main equation (1). Without the Laplacian, which is responsible for the free diffusion, we have

$$\frac{\partial}{\partial t} u(x, t) = -V(x, t) u(x, t).$$

It can be seen, that the density of particles increases over time for $V(x, t) < 0$ and decreases for $V(x, t) > 0$ correspondingly. The case $V(x, t) < 0$ may lead to an instability in the solution, because the integral

$$\int_{-\infty}^{\infty} f(x + \xi) e^{-w_i V(x+\xi, \tau_{n-i})\delta t} e^{-\lambda \xi^2} d\xi, \tag{35}$$

may diverge (see Eq. (4)). Therefore, when choosing $V(x, t) < 0$, one should make sure that the integral in (35) converges.

Consider the following problem (see Fig. 5)

$$V_i(x) = a + \sin^2\left(\pi\left(\frac{x}{a} + 1\right)\right) - \frac{1}{1 + \left(\frac{x}{a} + 1\right)^8}, \qquad f_i(x) = \sqrt{\frac{\beta}{\pi}} e^{-\beta(x-a)^2}, \qquad a = 0.5, \quad \beta = 0.5. \tag{36}$$

It can be interpreted as a nonperiodic system with an impurity. The term $V(x)$ does not decay in the spatial domain and it is not periodic. Therefore the reduction of this problem to a bounded domain is not a trivial task and would require sophisticated artificial boundary conditions.

In Table 2 we present results of numerical calculations, which show the order of the numerical scheme. In Fig. 6 we also present the computed solutions for different values of n. Even in this case, the solution converges with the order $p = 2$. We also used the Richardson extrapolation of $u(x, T)$ for different n to get higher order schemes in time. (See also Fig. 7.)

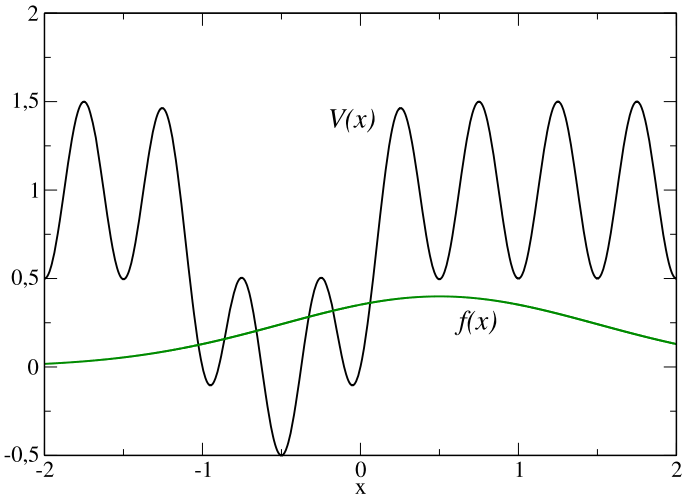

Fig. 5. Potential $V(x)$ and initial distribution $f(x)$ for periodic system with impurity (36). Potential oscillates on a free space. Functions $V(x)$ and $f(x)$ are relatively shifted to break the symmetry.

Table 2
Convergence rate for system (36). Accuracy of the cross approximation $\varepsilon = 10^{-12}$. Direct convolutions start from $n = 20$, $\sigma = 0.25$, final domain is $[-2, 2)$, $N_x = 8000$. Dimension of the integral (4) is labeled by n, δt is the time step, $T = 20$ is the final time. The order of the scheme p_2 for δt and the relative error ϵ_2 (33) are estimated from the original data computed by the algorithm from Section 3.4. The next values p_4 and ϵ_4 are estimated by the Richardson extrapolation. As it can be seen, the scheme has the fourth order in time after the extrapolation. The ranks of the matrix $\mathbf{\Phi}^{(k)}$ from (30) are given in the column labeled by r. The CPU time for computation of the integral (4) in *all points* of the mesh is reported in the last column.

n	δt	p_2	ϵ_2	p_4	ϵ_4	r	*CPU time* (min.)
64	$3.1 \cdot 10^{-1}$	–	–	–	–	9	0.2
128	$1.6 \cdot 10^{-1}$	–	$8.3 \cdot 10^{-2}$	–	–	9	0.3
256	$7.8 \cdot 10^{-2}$	1.47	$3.3 \cdot 10^{-2}$	–	$2.8 \cdot 10^{-3}$	9	0.8
512	$3.9 \cdot 10^{-2}$	1.62	$1.1 \cdot 10^{-2}$	2.00	$7.0 \cdot 10^{-4}$	9	1.7
1024	$2.0 \cdot 10^{-2}$	1.84	$3.1 \cdot 10^{-3}$	3.04	$8.6 \cdot 10^{-5}$	9	3.6
2048	$9.8 \cdot 10^{-3}$	1.95	$8.1 \cdot 10^{-4}$	3.66	$6.8 \cdot 10^{-6}$	9	7.0
4096	$4.9 \cdot 10^{-3}$	1.988	$2.0 \cdot 10^{-4}$	3.85	$4.7 \cdot 10^{-7}$	9	14.7
8192	$2.4 \cdot 10^{-3}$	1.997	$5.1 \cdot 10^{-5}$	3.98	$3.0 \cdot 10^{-8}$	9	33.0

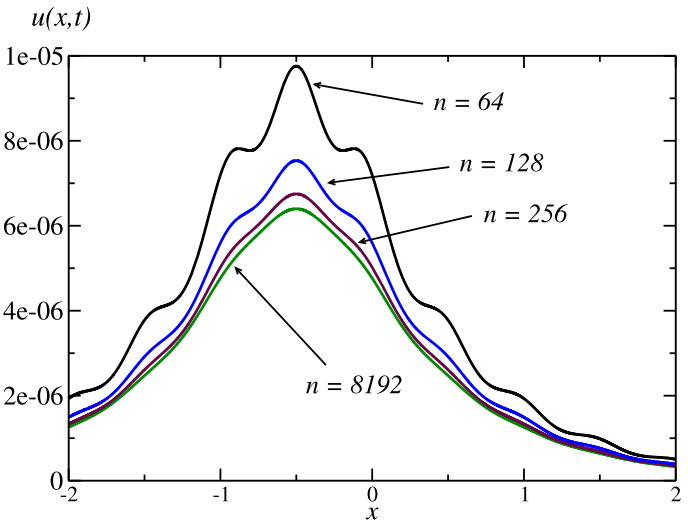

Fig. 6. Convergence of solution $u(x, t)$ for nonperiodic potential with impurity (36) for different n. These results correspond to the data presented in Table 2. The number of spacial mesh points $M = 2N_x = 8000$ in the final range $[-2, 2)$. The dissipation rate (36) leads to a decrease in the norm of the distribution density. As seen in the picture, the solution is far from the correct one for the dimensions $n = 64, 128, 256$.

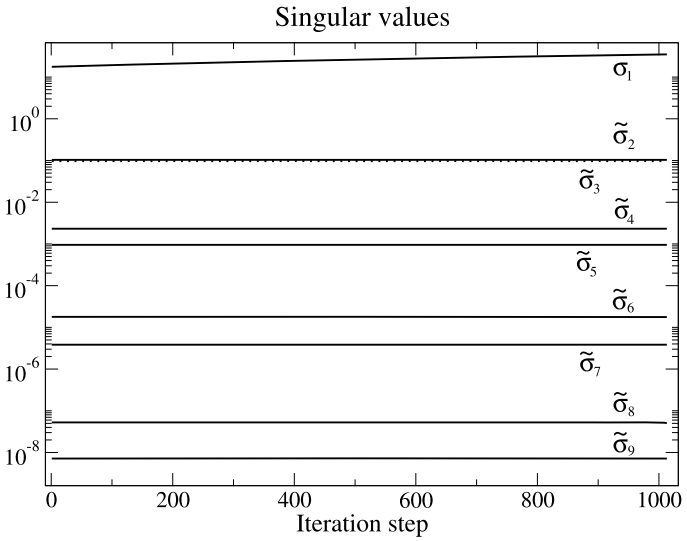

Fig. 7. The first few singular values (s.v.) of the matrix (30) for system (34) at each iteration step. The first s.v. σ_1 is presented in the absolute value. The other ones are given in the relative values as σ_i/σ_1. The values below the cross accuracy $\varepsilon = 10^{-10}$ are thrown out. As it can be seen, approximate SVD-rank is similar to the cross rank (in the sense of criterion (A.3)).

Table 3
Timings for system (34). Accuracy of the cross approximation $\varepsilon = 10^{-10}$. Direct convolutions start from $n = 30$, $\sigma = 0.5$, range of final spatial domain is $[-2, 2)$, $N_x = 4000$. Dimension of the integral (4) is labeled by n, δt is a time step, T is a final time for solution $u(x_0, T)$ computed in a fixed point x_0. Here $x_0 = 0$, $T = 1$. The relative error $\epsilon = |\tilde{u}(x_0, T) - u(x_0, T)|/|u(x_0, T)|$ is computed in one point x_0. Time for one point calculation is presented for Monte Carlo approach (37) and is estimated for the whole mesh array consisting of $M = 2N_x = 8000$ points (the last column). For the low-rank computation the total timings are presented as well. Monte Carlo simulation has been done with $K = 10^9$ samples. The low-rank results are labeled by LR, while the Monte Carlo results are labeled by MC.

n	δt	$u(x_0, T)$	ϵ	*CPU time (1 point)*	*CPU time (total)*
32	$3.1 \cdot 10^{-2}$	0.6369899_{MC}	$5.8 \cdot 10^{-4}$	40.2 min	$5.3 \cdot 10^3$ h (est.)
		0.6369792_{LR}	$5.6 \cdot 10^{-4}$		6 s
64	$1.6 \cdot 10^{-2}$	0.6367165_{MC}	$1.5 \cdot 10^{-4}$	79.1 min	$1.0 \cdot 10^4$ h (est.)
		0.6367099_{LR}	$1.4 \cdot 10^{-4}$		13 s
128	$7.8 \cdot 10^{-3}$	0.6366653_{MC}	$7.2 \cdot 10^{-5}$	171 min	$2.2 \cdot 10^4$ h (est.)
		0.6366423_{LR}	$3.5 \cdot 10^{-5}$		26 s
256	$3.9 \cdot 10^{-3}$	0.6366388_{MC}	$3.0 \cdot 10^{-5}$	355 min	$4.7 \cdot 10^4$ h (est.)
		0.6366254_{LR}	$8.9 \cdot 10^{-6}$		53 s
512	$2.0 \cdot 10^{-3}$	0.6366218_{MC}	$3.2 \cdot 10^{-6}$	705 min	$9.4 \cdot 10^4$ h (est.)
		0.6366212_{LR}	$2.2 \cdot 10^{-6}$		1.8 min
		0.6366198_{exact}			

4.4. Monte Carlo experiments

In this section we present results of Monte Carlo simulation. To estimate the solution in *a fixed point* x_0 the following formula is used

$$u^{(n)}_{MC}(x_0, T) = \frac{1}{K} \sum_{k=1}^{K} f\left(\xi_{\langle k \rangle}(n)\right) \prod_{i=0}^{n} e^{-w_i V(\xi_{\langle k \rangle}(i), \tau_{n-i})\delta t},$$
$$\xi_{\langle k \rangle}(i) = \xi_{\langle k \rangle 1} + \ldots + \xi_{\langle k \rangle i}, \tag{37}$$

where each component of the vector $\boldsymbol{\xi}_{\langle k \rangle} = (\xi_{\langle k \rangle 1}, \ldots, \xi_{\langle k \rangle n})^T$ is independently taken from the normal distribution $\mathcal{N}(0, 2\sigma\delta t)$ at each trial step $k : 1 \le k \le K$, where K being the number of trials.

Results for the exactly solvable model (34) are presented in Table 3. We compare accuracy and timings for Monte Carlo and low-rank calculations. It should be emphasized that in the Monte Carlo approach only one point of $u(x_0, T)$ is calculated for a fixed x_0 in one simulation, while our approach allows to compute the whole array $u(x_i, T)$ on the whole mesh simultaneously. This numerical experiments have been done on a single CPU core without parallelization of the Monte Carlo algorithm just to estimate the speedup of the low-rank computation. More advanced realization such as quasi Monte Carlo methods can be used. As it can be seen, the low-rank algorithm presented in Section 3.4 is much faster.

5. Conclusion and future work

The presented results show that the proposed method is an efficient approach for solving diffusion equations in a free space without artificial boundary conditions (ABC). Instead of standard solvers based on the ABC designed for certain cases [92,93], our method is more universal one and is applicable to a wide class of potentials as a unified approach. It needs a constant memory size, which depends only on the final mesh size M and the rank r of the matrix of solution from (30) at each iteration step. Its complexity, then, is similar to the classical time-stepping schemes for the solution of the reaction–diffusion equations in a bounded domain. It also shows a favourable scaling.

It is natural to extend the approach presented in the current work to higher dimensions. Then, instead of one-dimensional convolutions we will have to work with d-dimensional convolutions, where d is the dimension of the problem. The extended domain will be $[-na, na]^d$, where n is the number of time steps (equal to the dimension of the path integral). Thus, for higher dimensions the solution can be treated as a $(d+1)$-dimensional tensor of size $M \times n \times \ldots \times n$. Instead of the matrix low-rank approximation, stable low-rank factorization based on the tensor train decomposition [44] could be used, with the final cost approximately equal to the cost of the computation the convolutions on the small domain.

Finally, the most intriguing part of the work to be done is to apply the similar techniques to the Schrödinger equation. There, the convolution is no longer a convolution with a Gaussian function. Thus, the problem is much more difficult and our approach requires modifications. The presented method can also be applied to path integrals arising in other application areas, including the financial mathematics. The main requirement is that the integrand depends on the sum of variables multiplied by a separable function.

Acknowledgement

This work was partially supported by Russian Science Foundation grant 14-11-00659.

Appendix A. The cross approximation of matrices

Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\hat{I} = \{i_1, i_2, \ldots, i_r\}$, $\hat{J} = \{j_1, j_2, \ldots, j_r\}$ be subsets of $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$, respectively, $r \le \min(n, m)$. By $\hat{\mathbf{A}} = \mathbf{A}(\hat{I}, \hat{J})$ we denote a submatrix of $\mathbf{A}$ formed by the entries of $\mathbf{A}$ at the intersections of rows $i \in \hat{I}$ and columns $j \in \hat{J}$. In this paper we use the following concept of *the skeleton decomposition* [94–97]. For any matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ of rank r there exists its decomposition

$$\mathbf{A} = \mathbf{B}\hat{\mathbf{A}}^{-1}\mathbf{C}^T, \tag{A.1}$$

where $\mathbf{B} = \mathbf{A}(I, \hat{J})$, $\mathbf{C}^T = \mathbf{A}(\hat{I}, J)$, and $\hat{\mathbf{A}} = \mathbf{A}(\hat{I}, \hat{J}) \in \mathbb{R}^{r\times r}$ is a certain submatrix of $\mathbf{A}$, such that $det\hat{\mathbf{A}} \neq 0$. For the numerical reasons it is more effective to work with orthogonal matrices. The decomposition (A.1) can be rewritten by the factorization of the matrices $\mathbf{B} = \mathbf{Q}_B\mathbf{R}_B$ and $\mathbf{C}^T = \mathbf{R}_C^T\mathbf{Q}_C^T$ by the *QR-decomposition*, and by further factorization of the rank-r square matrix $\mathbf{R}_B\hat{\mathbf{A}}^{-1}\mathbf{R}_C^T = \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^T$ by the *singular value decomposition* (SVD) [98,99]. Thus, we will use the dyadic representation of (A.1)

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T, \qquad A_{ij} = \sum_{q=1}^{r} X_{iq}Y_{jq}, \qquad \mathbf{X} = \mathbf{Q}_B\mathbf{U}_A\mathbf{\Sigma}_A^{1/2}, \qquad \mathbf{Y}^T = \mathbf{\Sigma}_A^{1/2}\mathbf{V}_A^T\mathbf{Q}_C^T. \tag{A.2}$$

If the rank of the matrix $\mathbf{A}$ is greater than r, in practice instead of exact equation (A.1) we consider approximation in some norm. To obtain the decomposition (A.2) in this case we use the *cross approximation* algorithm [100,101] based on the concept of the maximum volume submatrix (*maxvol*) introduced in [102,103]. We have implemented our version of the algorithm available at [104,105]. Example of the usage of our code can be found in [106]. The new version of the code for complex and real matrices will be available soon at [107].

The rank in the cross approximation technique is determined adaptively. The algorithm starts from the guess rank r_0 and at each iteration step k the subspace of vectors of $\mathbf{B}$ and $\mathbf{C}^T$ is doubled (they are chosen by the *maxvol* subroutine, which returns a set of $2r_k$ row (column) indices of a submatrix of (almost) maximum volume). The next value of the rank r_{k+1}, $r_{k+1} \le 2r_k$ is chosen from the singular values of the matrix $\mathbf{\Sigma}_A$ of size $2r_k \times 2r_k$ according to the following criterion

$$r_{k+1} = \min_{1\le s\le 2r_k} \{s \mid \zeta(s) < \varepsilon_c\}, \qquad \zeta(s) = \sqrt{\frac{\sum_{i=s+1}^{2r_k}\sigma_i^2}{\sum_{i=1}^{2r_k}\sigma_i^2}}, \qquad \zeta(2r_k) \equiv 0. \tag{A.3}$$

The algorithm stops when $||\mathbf{\Sigma}_A^{(k)} - \mathbf{\Sigma}_A^{(k+1)}||_2 < \varepsilon_c\mathbf{\Sigma}_A^{(k+1)}$ for the relative accuracy ε_c.

Approximation (A.2) can be obtained by the SVD decomposition of the whole matrix $\mathbf{A}$ with $\mathcal{O}((n^2+m^2)m)$ complexity, which is prohibitively slow. In contrast, the rank-r cross approximation requires only $\mathcal{O}((n+m)r)$ evaluations of the elements and $\mathcal{O}((n+m)r^2)$ additional operations. This becomes crucial in practice, when the matrix element A_{ij} is a time-consuming function to be calculated in a point (i, j) for a finite time or the given matrix is very large. Existence of such an approximation and convergence of the cross algorithm are discussed in Section 3.4 and Appendix B.

Appendix B. Numerical investigation of the low-rank structure of the solution basis set

Suppose, a function $t(x) \in L_2(\mathbb{R})$ can be expanded into a series

$$t(x) = \sum_{l=0}^{\infty} c_l \phi_l(x), \qquad c_l = \int_{-\infty}^{\infty} t(x)\, \phi_l(x) dx, \tag{B.1}$$

where

$$\phi_l(x) = \left(\frac{1}{2^l l! \sqrt{\pi}} \right)^{\frac{1}{2}} e^{-x^2/2} H_l(x), \qquad H_0(x) = 1, \quad H_1(x) = 2x, \quad H_2(x) = 4x^2 - 2, \tag{B.2}$$

and $H_l(x)$, are Hermite polynomials, with fast decaying coefficients c_l, such that for a given accuracy ε_1

$$\exists l_0 : \chi(l_0) < \varepsilon_1 \, \chi(0), \qquad \chi(l') = \sum_{l=l'+1}^{\infty} c_l^2.$$

And the approximated function

$$\tilde{t}(x) = \sum_{l=1}^{l_0} c_l \phi_l(x), \qquad ||t(x) - \tilde{t}(x)|| < \varepsilon_1$$

is of a canonical ε_1-rank l_0. Then, the question is about the rank structure of the matrix $\mathbf{\Phi}_l$ constructed as a reshape of a corresponding one-dimensional basis vector $\phi_l(x_i)$ defined on the uniform mesh (15)

$$(\Phi_l)_{ij} = \phi_l(x_{i+j \cdot M}). \tag{B.3}$$

If the matrices $\{\mathbf{\Phi}_l\}_{l=1}^{l_0}$ are the low-rank ones then the matrix of a target function $\tilde{t}(x)$

$$\mathbf{T} = \sum_{l=1}^{l_0} c_l \, \mathbf{\Phi}_l, \qquad T_{ij} = t(x_{i+j \cdot M}), \tag{B.4}$$

is of low-rank as well, which does not exceed the upper bound $l_0 \cdot r_{\max}$, where $r_{\max} = \max_{1 \le l \le l_0}(rank(\mathbf{\Phi}_l))$, but practically, it is of order $r_{\max}$. In the Table 4 we present first several singular values of matrix $\mathbf{\Phi}_l$. As it can be seen, each matrix has the low-rank structure. It would be nice to prove this numerical fact theoretically.

Finally, to estimate the rank of the approximation (B.4) one needs only to compute the coefficients c_l in the expansion (B.1) and investigate their behaviour. This idea is similar to the QTT approach applied to the Laplace and its inverse operators in [108].

Appendix C. Proof of the lemmas

Proof of Lemma 1. For the basis set $\{\mathbf{u}_i\}_{i=0}^{r_1-1}$ the following equality holds

$$\mathbf{l}_m = \sum_{i=0}^{r_1-1} \alpha_{mi} \mathbf{u}_i. \tag{C.1}$$

Then, according to (6)

$$\left[\mathbf{l}_m^T, \mathbf{0}\right]_H = \left[\sum_{i=0}^{r_1-1} \alpha_{mi} \mathbf{u}_i^T, \mathbf{0} \right]_H = \sum_{i=0}^{r_1-1} \alpha_{mi} \left[\mathbf{u}_i^T, \mathbf{0}\right]_H, \qquad \Leftrightarrow \qquad \mathbf{L}_m = \sum_{i=0}^{r_1-1} \alpha_{mi} \mathbf{U}_i, \qquad \forall m \in [0, k-1]. \quad \square \tag{C.2}$$

Proof of Lemma 2. From the equality $\mathbf{r}_m = \sum_{i=0}^{r_2-1} \beta_{mi} \mathbf{w}_i$ it follows that

$$\left[\mathbf{0}^T, \mathbf{r}_m\right]_H = \left[\mathbf{0}^T, \sum_{i=0}^{r_2-1} \beta_{mi} \mathbf{w}_i \right]_H = \sum_{i=0}^{r_2-1} \beta_{mi} \left[\mathbf{0}^T, \mathbf{w}_i\right]_H, \qquad \Leftrightarrow \qquad \mathbf{R}_m = \sum_{i=0}^{r_2-1} \beta_{mi} \mathbf{W}_i, \qquad \forall m \in [0, k-1]. \quad \square$$

Table 4
Singular values (s.v.) of the matrix (B.3) composed from the discretized basis (B.2). The order of polynomials is labeled by l, σ_1 is the first (absolute) singular value, then σ_i/σ_1 corresponds to the following relative singular values. The size of the matrix is 8000×1024. As it can be seen (numerical) ranks do not exceed the value of 8 and grow from small to bigger l.

l	σ_1	σ_2/σ_1	σ_3/σ_1	σ_4/σ_1	σ_5/σ_1	σ_6/σ_1	σ_7/σ_1	σ_8/σ_1	σ_9/σ_1
0	32.2	0.96	$9.0 \cdot 10^{-5}$	$8.7 \cdot 10^{-5}$	$8.9 \cdot 10^{-16}$	$8.7 \cdot 10^{-16}$	$7.4 \cdot 10^{-16}$	$6.4 \cdot 10^{-16}$	$5.5 \cdot 10^{-16}$
1	35.5	0.77	$0.6 \cdot 10^{-3}$	$0.6 \cdot 10^{-3}$	$1.1 \cdot 10^{-14}$	$1.1 \cdot 10^{-14}$	$3.7 \cdot 10^{-16}$	$3.4 \cdot 10^{-16}$	$2.8 \cdot 10^{-16}$
2	40.3	0.48	$2.2 \cdot 10^{-3}$	$1.6 \cdot 10^{-3}$	$8.6 \cdot 10^{-14}$	$8.3 \cdot 10^{-14}$	$5.6 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$
3	38.0	0.62	$0.8 \cdot 10^{-3}$	$0.7 \cdot 10^{-3}$	$6.2 \cdot 10^{-13}$	$5.1 \cdot 10^{-13}$	$5.0 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$
4	37.0	0.68	$2.1 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$3.9 \cdot 10^{-12}$	$3.8 \cdot 10^{-12}$	$6.6 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$	$4.5 \cdot 10^{-16}$
5	35.5	0.77	$5.0 \cdot 10^{-2}$	$4.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-11}$	$1.5 \cdot 10^{-11}$	$5.0 \cdot 10^{-16}$	$4.0 \cdot 10^{-16}$	$4.0 \cdot 10^{-16}$
6	38.3	0.59	$9.2 \cdot 10^{-2}$	$5.8 \cdot 10^{-2}$	$9.2 \cdot 10^{-11}$	$8.8 \cdot 10^{-11}$	$5.3 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$
7	33.9	0.83	0.18	0.13	$4.8 \cdot 10^{-10}$	$3.7 \cdot 10^{-10}$	$3.8 \cdot 10^{-16}$	$3.8 \cdot 10^{-16}$	$3.4 \cdot 10^{-16}$
8	36.2	0.68	0.25	0.09	$1.9 \cdot 10^{-9}$	$1.8 \cdot 10^{-9}$	$4.1 \cdot 10^{-16}$	$4.0 \cdot 10^{-16}$	$3.5 \cdot 10^{-16}$
9	38.2	0.47	0.29	0.26	$7.3 \cdot 10^{-9}$	$7.0 \cdot 10^{-9}$	$4.7 \cdot 10^{-16}$	$3.8 \cdot 10^{-16}$	$3.6 \cdot 10^{-16}$
10	28.7	0.85	0.60	0.58	$3.6 \cdot 10^{-8}$	$2.9 \cdot 10^{-8}$	$7.9 \cdot 10^{-16}$	$4.7 \cdot 10^{-16}$	$4.5 \cdot 10^{-16}$
11	31.1	0.73	0.65	0.35	$1.1 \cdot 10^{-7}$	$9.4 \cdot 10^{-8}$	$5.5 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$4.7 \cdot 10^{-16}$
12	33.1	0.65	0.58	0.25	$3.7 \cdot 10^{-7}$	$3.1 \cdot 10^{-7}$	$5.3 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$
13	30.1	0.69	0.68	0.51	$1.3 \cdot 10^{-6}$	$1.2 \cdot 10^{-6}$	$7.0 \cdot 10^{-16}$	$5.7 \cdot 10^{-16}$	$5.3 \cdot 10^{-16}$
14	26.2	0.95	0.74	0.67	$3.9 \cdot 10^{-6}$	$3.2 \cdot 10^{-6}$	$7.2 \cdot 10^{-16}$	$6.9 \cdot 10^{-16}$	$6.1 \cdot 10^{-16}$
15	30.2	0.74	0.71	0.38	$8.0 \cdot 10^{-6}$	$8.0 \cdot 10^{-6}$	$6.2 \cdot 10^{-16}$	$6.0 \cdot 10^{-16}$	$5.3 \cdot 10^{-16}$
16	30.8	0.82	0.63	0.21	$2.0 \cdot 10^{-5}$	$1.7 \cdot 10^{-5}$	$5.3 \cdot 10^{-16}$	$4.9 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$
17	28.4	0.92	0.67	0.43	$5.5 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	$5.1 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$	$4.6 \cdot 10^{-16}$
18	28.5	0.88	0.68	0.48	$1.3 \cdot 10^{-4}$	$1.1 \cdot 10^{-4}$	$5.9 \cdot 10^{-16}$	$5.5 \cdot 10^{-16}$	$5.4 \cdot 10^{-16}$
19	30.0	0.86	0.51	0.47	$2.6 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	$5.0 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$	$4.5 \cdot 10^{-16}$
20	31.6	0.80	0.56	0.24	$5.3 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$	$5.3 \cdot 10^{-16}$	$4.7 \cdot 10^{-16}$	$4.5 \cdot 10^{-16}$
21	28.9	0.90	0.72	0.23	$1.12 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$	$4.6 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$	$4.3 \cdot 10^{-16}$
22	27.0	0.98	0.76	0.47	$2.5 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$6.2 \cdot 10^{-16}$	$5.6 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$
23	30.3	0.67	0.62	0.59	$4.03 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$6.0 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$	$4.7 \cdot 10^{-16}$
24	28.0	0.82	0.78	0.53	$8.3 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$	$8.2 \cdot 10^{-16}$	$7.6 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$
25	30.0	0.77	0.76	0.23	$1.4 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$	$2.4 \cdot 10^{-15}$	$1.7 \cdot 10^{-15}$	$5.8 \cdot 10^{-16}$
26	29.4	0.91	0.67	0.16	$2.4 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$7.7 \cdot 10^{-15}$	$6.8 \cdot 10^{-15}$	$6.3 \cdot 10^{-16}$
27	27.4	0.88	0.87	0.34	$4.2 \cdot 10^{-2}$	$3.7 \cdot 10^{-2}$	$2.6 \cdot 10^{-14}$	$2.4 \cdot 10^{-14}$	$4.5 \cdot 10^{-16}$
28	29.4	0.75	0.71	0.50	$5.8 \cdot 10^{-2}$	$5.7 \cdot 10^{-2}$	$6.8 \cdot 10^{-14}$	$5.6 \cdot 10^{-14}$	$5.6 \cdot 10^{-16}$
29	28.4	0.81	0.71	0.56	$8.9 \cdot 10^{-2}$	$8.4 \cdot 10^{-2}$	$2.2 \cdot 10^{-13}$	$1.5 \cdot 10^{-13}$	$5.5 \cdot 10^{-16}$
30	27.8	0.86	0.79	0.44	0.14	$8.8 \cdot 10^{-2}$	$6.2 \cdot 10^{-13}$	$5.6 \cdot 10^{-13}$	$5.1 \cdot 10^{-16}$
31	28.0	0.96	0.74	0.20	0.17	0.12	$1.2 \cdot 10^{-12}$	$1.2 \cdot 10^{-12}$	$6.1 \cdot 10^{-16}$
32	26.3	0.99	0.89	0.27	0.19	0.14	$5.4 \cdot 10^{-12}$	$5.3 \cdot 10^{-12}$	$7.8 \cdot 10^{-16}$

Proof of Lemma 3. From definition (22) and the decomposition $\mathbf{l}_m = \sum_{i=0}^{r_1-1} \gamma_{mi}\mathbf{u}_i$, $\forall j_m \in [0,k]$, it follows that

$$\begin{pmatrix} \mathbf{r}_m \\ l_{m,(M-1)} \end{pmatrix} = \mathbf{l}_m = \sum_{i=0}^{r_1-1} \gamma_{mi}\mathbf{u}_i = \sum_{i=0}^{r_1-1} \gamma_{mi} \begin{pmatrix} \mathbf{w}_i \\ u_{m,(M-1)} \end{pmatrix}, \qquad \Rightarrow \qquad \mathbf{r}_m = \sum_{i=0}^{r_1-1} \gamma_{mi}\mathbf{w}_i. \qquad \square$$

Appendix D. Solution for the harmonic oscillator potential

In this section we analytically integrate equations (8), (9), (10) with the initial condition (31).
Let us define $F_k^{(n)}(x) \equiv \Psi_k^{(n)}(x)$ for harmonic potential (31). Starting from $k = n$,

$$\Psi_k^{(n)}(x) = \sqrt{\frac{\lambda}{\pi}}\sqrt{\frac{\beta}{\pi}} \int_{-\infty}^{\infty} e^{-\beta(x+\xi)^2} e^{-w_n(x+\xi)^2\delta t}\, e^{-\lambda\xi^2} d\xi,$$

and making use of the integral

$$P(\alpha,\beta,y) = \int_{-\infty}^{\infty} p(\beta, y+\xi)p(\alpha,\xi)d\xi = p\left(\tfrac{\alpha\beta}{\alpha+\beta}, y\right) = \sqrt{\tfrac{\alpha\beta}{\pi(\alpha+\beta)}} e^{-\frac{\alpha\beta}{\alpha+\beta}y^2}, \qquad y \in \mathbb{R}, \qquad \alpha,\beta > 0,$$

where $p(\alpha, x)$ is defined in (17), we obtain

$$\Psi_n^{(n)}(x) = \sqrt{\frac{\beta_n}{\gamma_n}}\sqrt{\frac{\beta_{n-1}}{\pi}} e^{-\beta_{n-1}x^2}, \qquad \beta_{n-1} = \frac{\lambda\gamma_n}{\lambda+\gamma_n}, \qquad \gamma_n = \beta_n + w_n\delta t, \qquad \beta_n = \beta.$$

For the next $k = n-1$, we have

$$\Psi_{n-1}^{(n)}(x) = \sqrt{\frac{\lambda}{\pi}}\sqrt{\frac{\beta_n}{\gamma_n}}\sqrt{\frac{\beta_{n-1}}{\pi}}\int\limits_{-\infty}^{\infty} e^{-\beta_{n-1}(x+\xi)^2} e^{-w_{n-1}\frac{(x+\xi)^2}{1+\delta t}\delta t}\, e^{-\lambda\xi^2} d\xi,$$

$$\Psi_{n-1}^{(n)}(x) = \sqrt{\frac{\beta_n}{\gamma_n}}\sqrt{\frac{\beta_{n-1}}{\gamma_{n-1}}}\sqrt{\frac{\beta_{n-2}}{\pi}} e^{-\beta_{n-2}x^2}, \qquad \beta_{n-2} = \frac{\lambda\gamma_{n-1}}{\lambda+\gamma_{n-1}}, \qquad \gamma_{n-1} = \beta_{n-1} + w_{n-1}\frac{\delta t}{1+\delta t}.$$

By induction, we conclude that

$$\Psi_k^{(n)}(x) = \Gamma_k^{(n)}\sqrt{\frac{\beta_{k-1}}{\pi}} e^{-\beta_{k-1}x^2}, \qquad \beta_{k-1} = \frac{\lambda\gamma_k}{\lambda+\gamma_k}, \qquad \gamma_k = \beta_k + w_k\frac{\delta t}{1+(n-k)\delta t},$$

$$\Gamma_k^{(n)} = \sqrt{\frac{\beta_n}{\gamma_n}}\sqrt{\frac{\beta_{n-1}}{\gamma_{n-1}}}\cdot\ldots\cdot\sqrt{\frac{\beta_k}{\gamma_k}}, \qquad 1 \le k \le n.$$

References

[1] R.P. Feynman, Space–time approach to non-relativistic quantum mechanics, Rev. Mod. Phys. 20 (1948) 367–387, http://dx.doi.org/10.1103/RevModPhys.20.367.
[2] R. Feynman, A. Hibbs, Quantum Mechanics and Path Integrals, Int. Ser. Pure Appl. Phys., McGraw–Hill, 1965.
[3] C. Garrod, Hamiltonian path-integral methods, Rev. Mod. Phys. 38 (1966) 483–494, http://dx.doi.org/10.1103/RevModPhys.38.483.
[4] A.A. Abrikosov, I. Dzyaloshinskii, L.P. Gorkov, Methods of Quantum Field Theory in Statistical Physics, Dover, 1975, https://cds.cern.ch/record/107441.
[5] G. Mahan, Many-Particle Physics. Physics of Solids and Liquids, Springer, 2000, http://books.google.co.uk/books?id=xzSgZ4-yyMEC.
[6] R.A.H. Norman Bleistein, Asymptotic Expansions of Integrals, Dover Edition, 2010.
[7] J. Zinn-Justin, Quantum Field Theory and Critical Phenomena, 3rd edition, Clarendon Press, Oxford, 1996.
[8] J. Polonyi, Lectures on the functional renormalization group method, Cent. Eur. J. Phys. 1 (1) (2003), http://dx.doi.org/10.2478/BF02475552.
[9] J. Dick, F. Kuo, G. Peters, I. Sloan, Monte Carlo and Quasi-Monte Carlo Methods, Springer, 2012.
[10] M. Holtz, Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance, Lect. Notes Comput. Sci. Eng., vol. 77, Springer, 2011.
[11] J. Garcke, M. Griebel, Sparse Grids and Applications, Springer, 2013.
[12] K.Y. Wong, Review of Feynman's path integral in quantum statistics: from the molecular Schrodinger equation to Kleinert's variational perturbation theory, Commun. Comput. Phys. 15 (4) (2014), http://dx.doi.org/10.4208/cicp.140313.070513s.
[13] M. Masujima, Path Integral Quantization and Stochastic Quantization, Springer, 2008.
[14] H. Kleinert, Path Integrals in Quantum Mechanics, Statistics, Polymer Physics, and Financial Markets, 5th edition, World Scientific, 2009.
[15] J. Crank, The Mathematics of Diffusion, Clarendon Press, Oxford, 1975.
[16] R.F. Bass, Diffusions and Elliptic Operators, Springer, 1998.
[17] A. Chaichian, A. Demichev, Path Integrals in Physics, vol. I, IoP Publishing, 2001.
[18] A.N. Borodin, P. Salminen, Handbook of Brownian Motion. Facts and Formulae, Springer, 2002.
[19] M. Kac, On some connections between probability theory and differential and integral equations, http://projecteuclid.org/euclid.bsmsp/1200500229, 1951.
[20] I. Karatzas, S.E. Shreve, Brownian Motion and Partial Differential Equations, Springer, 1991.
[21] R.M. Mazo, Brownian Motion. Fluctuations, Dynamics, and Applications, Clarendon Press, Oxford, 2002.
[22] E. Nelson, Dynamical Theories of Brownian Motion, Princeton University Press, 2001.
[23] S.A. Smolyak, Quadrature and interpolation formulas for tensor products of certain class of functions, Dokl. Akad. Nauk SSSR 148 (5) (1964) 1042–1053; Transl.: Sov. Math. Dokl. 4 (1963) 240–243.
[24] T. Gerstner, M. Griebel, Dimension-adaptive tensor-product quadrature, Computing 71 (2003) 65–87, http://dx.doi.org/10.1007/s00607-003-0015-5.
[25] T. Gerstner, M. Griebel, Numerical integration using sparse grids, Numer. Algorithms 18 (3–4) (1998) 209–232, http://dx.doi.org/10.1023/A:1019129717644.
[26] N. Makri, W.H. Miller, Monte Carlo integration with oscillatory integrands: implications for Feynman path integration in real time, Chem. Phys. Lett. 139 (1) (1987) 10–14, http://dx.doi.org/10.1016/0009-2614(87)80142-2.
[27] V.N. Gorshkov, S. Tretiak, D. Mozyrsky, Semiclassical Monte-Carlo approach for modelling non-adiabatic dynamics in extended molecules, Nat. Commun. 4 (2013) 1–8, http://dx.doi.org/10.1038/ncomms3144.
[28] L. de Lathauwer, A survey of tensor methods, in: IEEE International Symposium on Circuits and Systems, 2009, pp. 2773–2776.
[29] A. Smilde, R. Bro, P. Geladi, Multi-way Analysis with Applications in the Chemical Sciences, Wiley, 2004.
[30] B.N. Khoromskij, Introduction to tensor numerical methods in scientific computing, Preprint, Lecture Notes 06-2011, University of Zürich, 2010, http://www.math.uzh.ch/fileadmin/math/preprints/06_11.pdf.
[31] B.N. Khoromskij, Tensor-structured numerical methods in scientific computing: survey on recent advances, Chemom. Intell. Lab. Syst. 110 (1) (2012) 1–19, http://dx.doi.org/10.1016/j.chemolab.2011.09.001.
[32] F.L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. Phys. 6 (1) (1927) 164–189.
[33] F.L. Hitchcock, Multiple invariants and generalized rank of a p-way matrix or tensor, J. Math. Phys. 7 (1) (1927) 39–79.
[34] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitt. 36 (1) (2013) 53–78, http://dx.doi.org/10.1002/gamm.201310004.
[35] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500, http://dx.doi.org/10.1137/07070111X.
[36] L. Grasedyck, H. Wolfgang, An introduction to hierarchical h-rank and tt-rank of tensors with examples, Comput. Methods Appl. Math. 11 (3) (2011) 291.
[37] I.V. Oseledets, D.V. Savostianov, E.E. Tyrtyshnikov, Tucker dimensionality reduction of three-dimensional arrays in linear time, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 939–956, http://dx.doi.org/10.1137/060655894.
[38] L.R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311, http://dx.doi.org/10.1007/BF02289464.
[39] L. Tucker, Implications of factor analysis of three-way matrices for measurement of change, Probl. Meas. Change (1963) 122–137.

[40] L.R. Tucker, The extension of factor analysis to three-dimensional matrices, Contrib. Math. Psychol. (1964) 109–127.
[41] L. de Lathauwer, B. de Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (2000) 1253–1278, http://dx.doi.org/10.1137/s0895479896305696.
[42] I.V. Oseledets, E.E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra Appl. 432 (1) (2010) 70–88, http://dx.doi.org/10.1016/j.laa.2009.07.024.
[43] I.V. Oseledets, E.E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, SIAM J. Sci. Comput. 31 (5) (2009) 3744–3759, http://dx.doi.org/10.1137/090748330.
[44] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317, http://dx.doi.org/10.1137/090752286.
[45] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, J. Fourier Anal. Appl. 15 (5) (2009) 706–722, http://dx.doi.org/10.1007/s00041-009-9094-9.
[46] L. Grasedyck, Hierarchical singular value decomposition of tensors, SIAM J. Matrix Anal. Appl. 31 (4) (2010) 2029–2054, http://dx.doi.org/10.1137/090764189.
[47] W. Hackbusch, Tensor Spaces and Numerical Tensor Calculus, Springer-Verlag, Berlin, ISBN 978-3642280269, 2012.
[48] B.N. Khoromskij, V. Khoromskaia, S.R. Chinnamsetty, H.J. Flad, Tensor decomposition in electronic structure calculations on 3D Cartesian grids, J. Comput. Phys. 228 (16) (2009) 5749–5762, http://dx.doi.org/10.1016/j.jcp.2009.04.043.
[49] B.N. Khoromskij, V. Khoromskaia, Multigrid accelerated tensor approximation of function related multidimensional arrays, SIAM J. Sci. Comput. 31 (4) (2009) 3002–3026, http://dx.doi.org/10.1137/080730408.
[50] V. Khoromskaia, B. Khoromskij, Tensor numerical methods in quantum chemistry: from Hartree–Fock to excitation energies, Phys. Chem. Chem. Phys. (2015), http://dx.doi.org/10.1039/c5cp01215e, http://arxiv.org/abs/1504.06289.
[51] R. Schneider, T. Rohwedder, J. Blauert, Direct minimization for calculating invariant subspaces in density functional computations of the electronic structure, J. Comput. Math. 27 (2009) 360.
[52] Heinz-Jürgen Flad, Best n-term approximation in electronic structure calculations. II. Jastrow factors, ESAIM: M2AN 41 (2) (2007) 261–279, http://dx.doi.org/10.1051/m2an:2007016.
[53] B.N. Khoromskij, V. Khoromskaia, H.J. Flad, Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format, SIAM J. Sci. Comput. 33 (1) (2011) 45–65, http://dx.doi.org/10.1137/090777372.
[54] V.A. Kazeev, B.N. Khoromskij, Low-rank explicit QTT representation of the Laplace operator and inverse, SIAM J. Matrix Anal. Appl. 33 (3) (2012) 742–758, http://dx.doi.org/10.1137/100820479, http://www.mis.mpg.de/de/publications/preprints/2010/prepr2010-75.html.
[55] W. Hackbusch, B.N. Khoromskij, Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions, Computing 76 (3–4) (2006) 177–202, http://dx.doi.org/10.1007/s00607-005-0144-0.
[56] W. Hackbusch, B.N. Khoromskij, Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. II. HKT representation of certain operators, Computing 76 (3–4) (2006) 203–225, http://dx.doi.org/10.1007/s00607-005-0145-z.
[57] G. Beylkin, M.J. Mohlenkamp, Numerical operator calculus in higher dimensions, Proc. Natl. Acad. Sci. USA 99 (16) (2002) 10246–10251, http://dx.doi.org/10.1073/pnas.112329799.
[58] I.P. Gavrilyuk, W. Hackbusch, B.N. Khoromskij, Tensor-product approximation to the inverse and related operators in high-dimensional elliptic problems, Computing (74) (2005) 131–157.
[59] B.N. Khoromskij, Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$, Constr. Approx. (30) (2009) 599–620, http://dx.doi.org/10.1007/s00365-009-9068-9.
[60] B.N. Khoromskij, V. Khoromskaia, Low rank Tucker-type tensor approximation to classical potentials, Cent. Eur. J. Math. 5 (3) (2007) 523–550, http://dx.doi.org/10.2478/s11533-007-0018-0.
[61] W. Hackbusch, B.N. Khoromskij, Tensor-product approximation to operators and functions in high dimensions, J. Complex. 23 (4–6) (2007) 697–714, http://dx.doi.org/10.1016/j.jco.2007.03.007; festschrift for the 60th Birthday of Henryk Woźniakowski.
[62] W. Hackbusch, D. Braess, Approximation of $\frac{1}{x}$ by exponential sums in $[1, \infty]$, IMA J. Numer. Anal. 25 (4) (2005) 685–697.
[63] I.V. Oseledets, E.A. Muravleva, Fast orthogonalization to the kernel of discrete gradient operator with application to the Stokes problem, Linear Algebra Appl. 432 (6) (2010) 1492–1500, http://dx.doi.org/10.1016/j.laa.2009.11.010.
[64] I.V. Oseledets, Constructive representation of functions in low-rank tensor formats, Constr. Approx. 37 (1) (2013) 1–18, http://dx.doi.org/10.1007/s00365-012-9175-x, http://pub.inm.ras.ru/pub/inmras2010-04.pdf.
[65] G. Beylkin, M.J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, SIAM J. Sci. Comput. 26 (6) (2005) 2133–2159.
[66] S.V. Dolgov, B.N. Khoromskij, I.V. Oseledets, Fast solution of multi-dimensional parabolic problems in the tensor train/quantized tensor train–format with initial application to the Fokker–Planck equation, SIAM J. Sci. Comput. 34 (6) (2012) A3016–A3038, http://dx.doi.org/10.1137/120864210.
[67] B.N. Khoromskij, I.V. Oseledets, Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs, Comput. Methods Appl. Math. 10 (4) (2010) 376–394, http://dx.doi.org/10.2478/cmam-2010-0023.
[68] B.N. Khoromskij, I.V. Oseledets, DMRG + QTT approach to computation of the ground state for the molecular Schrödinger operator, Preprint 69, MPI MIS, Leipzig, 2010, http://www.mis.mpg.de/preprints/2010/preprint2010_69.pdf.
[69] I.V. Oseledets, Approximation of $2^d \times 2^d$ matrices using tensor decomposition, SIAM J. Matrix Anal. Appl. 31 (4) (2010) 2130–2145, http://dx.doi.org/10.1137/090757861.
[70] O.S. Lebedeva, Tensor conjugate-gradient-type method for Rayleigh quotient minimization in block QTT-format, Russ. J. Numer. Anal. Math. Model. 26 (5) (2011) 465–489, http://dx.doi.org/10.1515/rjnamm.2011.026.
[71] B.N. Khoromskij, I.V. Oseledets, QTT-approximation of elliptic solution operators in high dimensions, Russ. J. Numer. Anal. Math. Model. 26 (3) (2011) 303–322, http://dx.doi.org/10.1515/rjnamm.2011.017.
[72] D.V. Savostyanov, QTT-rank-one vectors with QTT-rank-one and full-rank Fourier images, Preprint 45, MPI MIS, Leipzig, 2011, http://www.mis.mpg.de/preprints/2011/preprint2011_45.pdf.
[73] B.N. Khoromskij, $\mathcal{O}(d \log n)$–quantics approximation of N–d tensors in high-dimensional numerical modeling, Constr. Approx. 34 (2) (2011) 257–280, http://dx.doi.org/10.1007/s00365-011-9131-1.
[74] V. Khoromskaia, B.N. Khoromskij, R. Schneider, Tensor-structured factorized calculation of two-electron integrals in a general basis, SIAM J. Sci. Comput. 35 (2) (2013) A987–A1010, http://dx.doi.org/10.1137/120884067.
[75] J. Ballani, Fast evaluation of singular BEM integrals based on tensor approximations, Numer. Math. 121 (3) (2012) 433–460, http://dx.doi.org/10.1007/s00211-011-0436-6.
[76] M.V. Rakhuba, I.V. Oseledets, Fast multidimensional convolution in low-rank tensor formats via cross approximation, SIAM J. Sci. Comput. 37 (2) (2015) A565–A582, http://dx.doi.org/10.1137/140958529.
[77] B.N. Khoromskij, Fast and accurate tensor approximation of multivariate convolution with linear scaling in dimension, J. Comput. Appl. Math. 234 (11) (2010) 3122–3139, http://dx.doi.org/10.1016/j.cam.2010.02.004.
[78] V. Kazeev, B.N. Khoromskij, E.E. Tyrtyshnikov, Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity, Tech. Rep. 36, MPI MIS, Leipzig, 2011, http://www.mis.mpg.de/publications/preprints/2011/prepr2011-36.html.
[79] S.V. Dolgov, B.N. Khoromskij, D.V. Savostyanov, Superfast Fourier transform using QTT approximation, J. Fourier Anal. Appl. 18 (5) (2012) 915–953, http://dx.doi.org/10.1007/s00041-012-9227-4.

[80] G. Beylkin, L. Monzón, Approximation by exponential sums revisited, Appl. Comput. Harmon. Anal. 28 (2) (2010) 131–149, http://dx.doi.org/10.1016/j.acha.2009.08.011.
[81] E.O. Brigham, The Fast Fourier Transform and Its Applications, Prentice Hall, 1988.
[82] H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, Springer, 1981.
[83] J. Chiu, L. Demanet, Sublinear randomized algorithms for skeleton decompositions, SIAM J. Matrix Anal. Appl. 34 (3) (2013) 1361–1383, http://dx.doi.org/10.1137/110852310.
[84] M. Bebendorf, R. Grzhibovskis, Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation, Math. Methods Appl. Sci. 29 (14) (2006) 1721–1747, http://dx.doi.org/10.1002/mma.759.
[85] M. Bebendorf, S. Kunis, Recompression techniques for adaptive cross approximation, J. Integral Equ. Appl. 21 (3) (2009) 331–357, http://dx.doi.org/10.1216/JIE-2009-21-3-331.
[86] M. Bebendorf, Adaptive cross approximation of multivariate functions, Constr. Approx. 34 (2) (2011) 149–179, http://dx.doi.org/10.1007/s00365-010-9103-x.
[87] E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations, Springer Ser. Comput. Math., Springer, 2006.
[88] C. Brezinski, M.R. Zaglia, Extrapolation Methods. Theory and Practice, Elsevier, 1991.
[89] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, 3rd edition, Springer, 2002.
[90] N. Makri, Numerical path integral techniques for long time dynamics of quantum dissipative systems, J. Math. Phys. 36 (5) (1995) 2430–2457, http://dx.doi.org/10.1063/1.531046.
[91] N. Makri, Blip decomposition of the path integral: exponential acceleration of real-time calculations on quantum dissipative systems, J. Chem. Phys. 141 (13) (2014) 134117, http://dx.doi.org/10.1063/1.4896736.
[92] E. Dubach, Artificial boundary conditions for diffusion equations: numerical study, J. Comput. Appl. Math. 70 (1) (1996) 127–144, http://dx.doi.org/10.1016/0377-0427(95)00140-9.
[93] X. Wu, Z.Z. Sun, Convergence of difference scheme for heat equation in unbounded domains using artificial boundary conditions, Appl. Numer. Math. 50 (2) (2004) 261–277, http://dx.doi.org/10.1016/j.apnum.2004.01.001.
[94] E.E. Tyrtyshnikov, Mosaic-skeleton approximations, Calcolo 33 (1) (1996) 47–57, http://dx.doi.org/10.1007/BF02575706.
[95] S.A. Goreinov, E.E. Tyrtyshnikov, N.L. Zamarashkin, A theory of pseudo-skeleton approximations, Linear Algebra Appl. 261 (1997) 1–21, http://dx.doi.org/10.1016/S0024-3795(96)00301-1.
[96] S.A. Goreinov, E.E. Tyrtyshnikov, N.L. Zamarashkin, Pseudo-skeleton approximations of matrices, Rep. Russ. Acad. Sci. 342 (2) (1995) 151–152.
[97] S.A. Goreinov, N.L. Zamarashkin, E.E. Tyrtyshnikov, Pseudo-skeleton approximations by matrices of maximum volume, Math. Notes 62 (4) (1997) 515–519, http://dx.doi.org/10.1007/BF02358985.
[98] G.H. Golub, C.F.V. Loan, Matrix Computations, 4th edition, 2012.
[99] J.W. Demmel, Applied Numerical Linear Algebra, 1997.
[100] E.E. Tyrtyshnikov, Incomplete cross approximation in the mosaic-skeleton method, Computing 64 (4) (2000) 367–380, http://dx.doi.org/10.1007/s006070070031.
[101] M. Bebendorf, Approximation of boundary element matrices, Numer. Math. 86 (4) (2000) 565–589, http://dx.doi.org/10.1007/pl00005410.
[102] S.A. Goreinov, E.E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, Contemp. Math. 280 (2001) 47–51.
[103] S.A. Goreinov, I.V. Oseledets, D.V. Savostyanov, E.E. Tyrtyshnikov, N.L. Zamarashkin, How to find a good submatrix, Research Report 08-10, ICM HKBU, Kowloon Tong, Hong Kong, 2008, http://www.math.hkbu.edu.hk/ICM/pdf/08-10.pdf.
[104] M.S. Litsarev, I.V. Oseledets, Cross2d C++ code: included in the DEPOSIT distribution, https://bitbucket.org/appl_m729/code-deposit, 2014.
[105] M.S. Litsarev, I.V. Oseledets, The DEPOSIT computer code based on the low rank approximations, Comput. Phys. Commun. 185 (10) (2014) 2801–2802, http://dx.doi.org/10.1016/j.cpc.2014.06.012.
[106] M.S. Litsarev, I.V. Oseledets, Fast low-rank approximations of multidimensional integrals in ion-atomic collisions modelling, Numer. Linear Algebra Appl. (2015), http://dx.doi.org/10.1002/nla.2008.
[107] M.S. Litsarev, Cross2d C++ code for the real and complex matrices, template version https://bitbucket.org/appl_m729/dzcross2d, 2015.
[108] I.P. Gavrilyuk, B.N. Khoromskij, Quantized-TT-Cayley transform to compute dynamics and spectrum of high-dimensional Hamiltonians, Comput. Methods Appl. Math. 11 (3) (2011) 273–290, http://dx.doi.org/10.2478/cmam-2011-0015, http://www.mis.mpg.de/de/publications/preprints/2011/prepr2011-43.html.