Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Superfast Wavelet Transform Using QTT Approximation. I: Haar Wavelets

(revised version: November 2013)

by

Boris N. Khoromskij and Sentao Miao

Preprint no.: 103 2013

Superfast Wavelet Transform Using QTT Approximation. I: Haar Wavelets

BORIS N. KHOROMSKIJ* SENTAO MIAO*◆

Abstract

We propose a superfast discrete Haar wavelet transform (SFHWT) as well as its inverse, using the QTT representation for the Haar transform matrices and input-output vectors. Though the Haar matrix itself does not have a low QTT-rank approximation, we show that factor matrices used at each step of the traditional multilevel Haar wavelet transform algorithm have explicit QTT representations of low rank. The SFHWT applies to a vector representing a signal sampled on a uniform grid of size $N = 2^d$. We develop two algorithms which roughly require square logarithmic time complexity with respect to the grid size, $O(\log^2 N)$, hence outperforming the traditional fast Haar wavelet transform (FHWT) of linear complexity, $O(N)$. Our approach also applies to the FHWT inverse as well as to the multidimensional wavelet transform. Numerical experiments demonstrate that the SFHWT algorithm is robust in keeping low rank of the resulting output vector and it outperforms the traditional FHWT for grid size larger than a certain value depending on the spacial dimension.

AMS Subject Classification: 65F30, 65F50, 65N35, 65F10
Key words: Tensor-structured methods, fast wavelet transform, multilevel methods, canonical tensor decomposition, quantized tensor approximation (QTT), data compression

1 Introduction

In the past decades a number of numerical techniques for efficient data-sparse representation of large data arrays have been developed based on the FFT, wavelet transform, polynomial interpolation, exponential fitting, nonlinear approximation, and reduced basis methods. However, in the case of high dimensional data most of traditional approaches suffer from the so-called "curse of dimensionality".

Recently, the general approach to data compression with logarithmic scaling, the so-called quantics TT (QTT) tensor approximation method was introduced, and rigorously proved for a wide class of function generated vectors and multidimensional tensors [13, 14].

*Max-Planck-Institute for Mathematics in the Sciences, Inselstr. 22-26, D-04103 Leipzig, Germany (bokh@mis.mpg.de).

**Supervised by Prof. B. Khoromskij during the internship work at Max-Planck-Institute for Mathematics in the Sciences, Inselstr. 22-26, D-04103 Leipzig, Germany.

◆Supervised by Prof. P. Oswald in Jacobs University Bremen, Campus Ring 1, Bremen, Germany(s.miao@jacobs-university.de).

The quantics-TT approximation method efficiently reduces the storage size of large N-vector to logarithmic size $O(\log_q N)$, where $N = q^d$ $(q = 2, 3, ...)$, by using compressed low-rank tensor representations of the q-reshaped d-dimensional $q \times q \times ... \times q$ image [14].

The theoretical substantiation of the QTT representation on a class of discretized differential and integral operators was first presented in [16] and [8], and later in [1, 9, 4]. Some numerics on TT reshaping applied to square matrices (without theoretical analysis) was considered in [22]. Various successful approaches for application of data-sparse QTT approximation method to the problems of electronic structure calculations were developed in [11, 12, 10]. Applications to high-dimensional dynamics including chemical master equation were described in [5, 3].

The QTT approximation method allows a low-parametric representation for large function related vectors or discrete operators (matrices) and, thus, an efficient way of data compression. Besides the reduced storage size, certain multilinear operations with the primal vectors can also be done under QTT images. Among others, one of the most important operations is matrix-vector multiplication, which can be carried out more efficiently on quantized images. Therefore, implementing basic operations by using QTT approximation, one can also apply more complicated algorithms and transforms, e.g. Fourier transform, convolution, etc. The Fourier transform in QTT format was introduced in [1], where a new algorithm called superfast Fourier transform was analyzed implemented in Matlab. In particular, the QTT approximation was efficiently applied to an input vector exploiting the multilevel structure of the Cooley-Tukey FFT algorithm. Then by applying the TT-rounding procedure the QTT-rank of resulting vector after each level in Cooley-Tukey algorithm could be kept low and stable, making the whole algorithm robust. Numerical tests also showed that superfast FFT demonstrates much better performance (especially in high dimension) than the traditional FFT.

Certainly, Fourier transform is not the only algorithm which can be optimized by the QTT approach, and in this paper, we develop a QTT-based algorithm for the wavelet transform, in particular, for one dimensional discrete Haar wavelets. It is known that discrete Haar wavelet transform has a similar multilevel structure to the traditional fast Fourier transform in form of the Cooley-Tukey FFT scheme, thus making it reasonable to guess that there also exists a "superfast" algorithm like superfast Fourier transform in QTT representation [1]. Also, it was recently shown that transforming a vector into QTT format shares certain similar features with wavelet transform [23], though, the QTT wavelet transform was not presented there.

In this paper, we introduce a so-called superfast (discrete) Haar wavelet transform (SFHWT), as well as its inverse transform, based on QTT representation (approximation) of Haar transform matrices and input-output vectors. Although the Haar matrix itself does not have a low QTT-rank approximation, we show that factor matrices used in each step of the traditional multilevel fast Haar wavelet transform have efficient explicit QTT representation. The superfast Haar wavelet transform is applied to a vector which is a function sampled on uniform grid of size $N = 2^d$. We introduce two algorithms which require roughly square logarithmic time complexity with respect to the grid size, $O(\log^2 N)$, hence outperforming the traditional fast Haar wavelet transform (FHWT) on large input vectors. Our approach also applies to inverse wavelet transform and to multidimensional case. Numerical experiments demonstrate that the SFHWT is robust in keeping low rank of resulting output vector and

it outperforms traditional FWHT for grid size larger than certain value depending on the number of spacial dimensions.

We know that the wavelet transform has wide applications in scientific computing and engineering, e.g. solving differential equations [24, 17]. In particular, when the size of a sampled signal is very large, our new superfast algorithm has a big advantage in enabling considerable reduction of the computing time. Although in numerical tests we restrict ourselves only to some specific class of input vectors, further research will be carried out for more general cases and we hope to demonstrate that the wavelet-QTT method can be of practical significance in real life applications.

The paper is organized as follows. In Section 2, we recall the notions of discrete Haar wavelet transform, as well as the basic concept of QTT format and its related notations. In Section 3, we first describe the traditional fast Haar wavelet transform, and then propose the one-level rank-truncated version of this algorithm by using the explicit QTT representation for each Haar transform matrix used in the recursion. In addition, we introduce the multilevel approach which improves the one-level "superfast" algorithm and analyze both SFHWT approaches in comparison. Section 4 discusses the SFHWT inverse, while Section 5 presents numerical examples demonstrating the advantage of our fast QTT-based algorithms implemented in Matlab. Conclusions outline possible generalizations of the approach.

2 Notations and Definitions

2.1 Discrete Haar Wavelet Transform

Similar to discrete Fourier transform, discrete wavelet transform (DWT) ([19, 20]) can be interpreted as spectral analysis using a set of basis functions in both time and frequency. In particular, a wavelet decomposition involves two families of functions, the *scaling functions* and the *wavelets*. These two sets of functions together perform a *multiresolution* analysis, see [19]. The essential thing in multiresolution analysis is to decompose $L^2(\mathbb{R})$ into a nested family of subspaces, i.e.

$$\{0\} \subseteq ...V_{j+2} \subseteq V_{j+1} \subseteq V_j \subseteq V_{j-1} \subseteq V_{j-2}... \subseteq L^2(\mathbb{R})$$

where $j \in \mathbb{Z}$ and V_j is the subspace of resolution 2^{-j}. Moreover, let W_j be the orthogonal complement of V_j in V_{j-1}:

$$V_j \oplus W_j = V_{j-1},$$

where W_j is called the *detail space*.

Since we have such a decomposition of $L^2(\mathbb{R})$, we can construct an orthonormal basis of L^2 by first finding an orthonormal basis of V_0. By definition of multiresolution analysis ([19]) such an orthonormal basis can be constructed by translating a scaling function $\phi(t)$. In particular, $\{\phi_{0,n}(t) : \phi_{0,n}(t) = \phi(t+n)\}_{n\in\mathbb{Z}}$ forms an orthonormal basis of V_0. Then, under the structure of multiresolution analysis, we can find an orthonormal basis of each V_j by dilating and translating the scaling function, i.e. $\{\phi_{j,k}(t) : \phi_{j,k}(t) = 2^{-j/2}\phi(2^{-j}t + k)\}_{k\in\mathbb{Z}}$ forms an orthonormal basis for V_j.

By this construction, we then obtain an orthonormal basis of L^2: $\{\phi_{j,k}(t) : \phi_{j,k}(t) = 2^{-j/2}\phi(2^{-j}t + k)\}_{k\in\mathbb{Z},j\in\mathbb{Z}}$. Indeed, we can do even more; namely, getting known the scaling

function ϕ, we can associate another function called wavelet ψ ([19]). By similar dilation and translation as above, we have $\{\psi_{j,k}(t) : \psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t + k)\}_{k\in\mathbb{Z}}$ an orthonormal basis for detail space W_j. In particular, for Haar wavelet transform, the scaling function and wavelet are defined as

$$\phi(t) = \begin{cases} 1 & 0 \leqslant t < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\psi(t) = \begin{cases} 1 & 0 \leqslant t < \frac{1}{2} \\ -1 & \frac{1}{2} \leqslant t < 1 \\ 0 & \text{otherwise} \end{cases}$$

To understand how we apply the wavelet transform, consider the projection operator P_{V_j} and P_{W_j} onto V_j and W_j:

$$P_{V_j} f = \sum_{k=-\infty}^{\infty} \langle f, \phi_{j,k}\rangle \phi_{j,k}$$

$$P_{W_j} f = \sum_{k=-\infty}^{\infty} \langle f, \psi_{j,k}\rangle \psi_{j,k}$$

for a signal or function $f(t)$ in L^2. As by multiresolution analysis we realize that $L^2 = \oplus_{j=-\infty}^{\infty} W_j$ and $L^2 = \oplus_{j=-\infty}^{m} W_j \oplus V_m$, we have

$$f(t) = \sum_j P_{W_j} f(t) = \sum_j \sum_k a_{j,k}\psi_{j,k}(t)$$

and

$$f(t) = P_{V_M} f(t) + \sum_{j=-\infty}^{m} P_{W_j} f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_{m,k}(t) + \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{m} d_{j,k}\psi_{j,k}(t)$$

where the expansion coefficients $a_{j,k} = \langle f, \psi_{j,n}\rangle$ are called *discrete wavelet transform* (DWT) of $f(t)$. This expansion usually has the feature that "energy" of the function is well represented by a few $a_{j,k}$ (see [19, 20]).

Since in real application our signal is represented by sampling on some grid, we have to consider the Haar wavelet transform in discretized space $\mathbb{Z}$. Note that scaling function and wavelet of Haar wavelet transform are also discretized, hence the original L^2 inner product, $\langle f, \psi_{j,n}\rangle = \int f(t)\psi_{j,n}(t)dt$ transforms to the discrete version, that is

$$\sum_{i\in\mathbb{Z}/2} f(i)\psi_{j,n}(i) = f(n) - f(n + \frac{1}{2}).$$

Similarly, for scaling function, we have $\sum_{i\in\mathbb{Z}/2} f(i)\phi_{j,n}(i) = f(n) + f(n + \frac{1}{2})$.

Now, if we have a signal or function sampled on uniform grid, we substitute the infinite space $\mathbb{Z}$ by final grid. After normalization of scaling function and wavelet to that grid, (finite) Discrete Haar wavelet transform of this signal can be represented by matrix vector multiplication. Suppose $f(t)$ is uniformly sampled on interval $[a, b]$ with grid size $N = 2^d$ as vector $\mathbf{x} \in \mathbb{R}^N$, then one level of discrete Haar wavelet transform reads:

$$\mathbf{y} = H_N \mathbf{x} \in \mathbb{R}^N,$$

such that

$$\mathbf{y}_k = (H_N\mathbf{x})_k = \frac{1}{\sqrt{2}}(\mathbf{x}_{2k} + \mathbf{x}_{2k-1}),$$
$$\mathbf{y}_{2^{d-1}+k} = (H_N\mathbf{x})_{2^{d-1}+k} = \frac{1}{\sqrt{2}}(\mathbf{x}_{2k-1} - \mathbf{x}_{2k}),\ k = 1, ..., 2^{d-1}$$

where $\mathbf{y}$, H_N, and $\mathbf{x}$ represent respectively the *coefficient*, *Haar transform matrix*, and *input vector*. Although there are different names for $\mathbf{y}_k \in V_1$ and $\mathbf{y}_{2^{d-1}+k} \in W_1$, we simply call them *low frequency* part and *high frequency* part (details), respectively. As an example,

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & & 1 & -1 \end{pmatrix}\begin{pmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} x(1)+x(2) \\ x(3)+x(4) \\ x(1)-x(2) \\ x(3)-x(4) \end{pmatrix},$$

or

$$H_4\mathbf{x} = \mathbf{y} \text{ and } \mathbf{x} = H_4^T\mathbf{y}.$$

¿From this example it is easy to see that Haar transform matrices are orthogonal. Therefore, the inverse transform of one-level Haar wavelet transform is simply $\mathbf{x} = H_N^T\mathbf{y}$. Further discussion concerning the fast QTT Haar transform inverse will be postponed to Section 4.

For the purpose of normalization, the Haar transform matrices are always multiplied by the factor $\frac{1}{\sqrt{2}}$, but since scaler multiplication has no effect on QTT structure, we will dismiss the $\frac{1}{\sqrt{2}}$ for brevity. Therefore we have the following representation of the non-normalized Haar transform matrices,

$$H_N = \begin{pmatrix} H_N^+ \\ H_N^- \end{pmatrix},$$

where

$$H_N^+ = \begin{pmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & \ddots & \\ & & & & 1 & 1 \end{pmatrix}, \quad H_N^- = \begin{pmatrix} 1 & -1 & & & & \\ & & 1 & -1 & & \\ & & & & \ddots & \\ & & & & 1 & -1 \end{pmatrix},$$

are $N/2 \times N$ matrices. Note that here $N = 2^d$ is the dimension of the input vector. In above example we have, $H_4 = \begin{pmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & & 1 & -1 \end{pmatrix}$, $H_4^+ = \begin{pmatrix} 1 & 1 & & \\ & & 1 & 1 \end{pmatrix}$, and $H_4^- = \begin{pmatrix} 1 & -1 & & \\ & & 1 & -1 \end{pmatrix}$.

The previous example repeats only one recursion of the standard Haar wavelet transform. The complete procedure for $N = 4$ should be the following: first, compute H_4x, then apply H_2 to the low frequency part. Indeed, this is exactly the idea of FHWT. But before we proceed the description of the SFHWT, let us introduce the notations of the matrices which

will be used in the description of fast Haar wavelet transform (we call them the *modified Haar transform matrices*),

$$H_{N,k} = \begin{pmatrix} H_{2^{(d-k)}} & & & & \\ & I_{2^{(d-k)}} & & & \\ & & I_{2^{(d-k+1)}} & & \\ & & & \ddots & \\ & & & & I_{2^{d-1}} \end{pmatrix}, \quad \text{where } k = 1, ..., d-1, \tag{2.1}$$

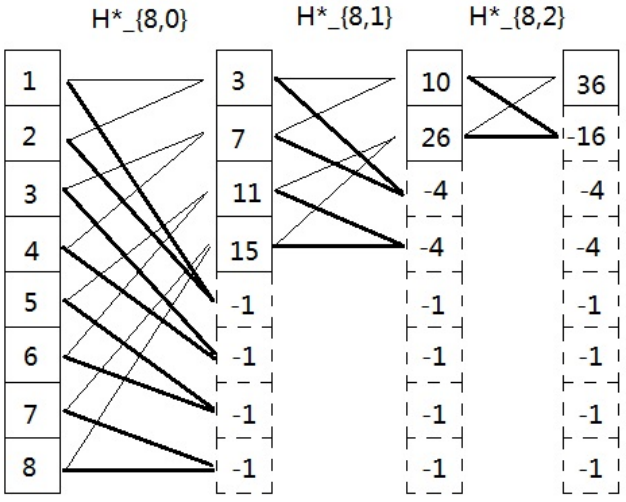
and for $k = 0$, we set $H_{N,0} = H_N$.

Therefore, a p level decomposition of the discrete Haar wavelet transform is given by $H_{2^d,p-1}...H_{2^d,0}x$ where $p = 1, ..., d$. In this paper we only discuss the case of complete d level decomposition, i.e. $p = d$, thus, the complete matrix transform $\mathcal{H}_{2^d} = H_{2^d,d-1}...H_{2^d,0}$ is called the *Haar matrix*. The traditional FHWT can be then described by Algorithm 1.

Algorithm 1: Fast Haar Wavelet Transform

Data: Input vector x of size $N = 2^d$; Modified Haar transform matrices $H_{N,k}$ where $k = 0, ..., d-1$.
Result: Output vector y of same size as x.
begin
  for $k = 0$ **to** $d-1$ **do**
    $x = H_{N,k} * x$;
  $y = x$;
end

In case of $N = 8$, the algorithm can be understood via the following graph.



Remark 2.1 *Note that Algorithm 1 serves more for representation rather than for real implementation. Although in our algorithm we use matrix-vector multiplication, the real*

computation complexity is much smaller than that. The reason is that one has to compute $x_{2i+1} \pm x_{2i+2}$ *only. Hence, to implement this algorithm, we cannot simply use the matrix-vector multiplication because the dummy operations* $0 * x_i$ *will be counted.*

Therefore, the actual complexity of FHWT is $O(N)$ *rather than the complexity of matrix-vector multiplication,* $O(N^2)$*.*

The multilevel factorization in FHWT scheme will be also used in the superfast QTT-HWT algorithm to be introduced in the following.

2.2 Quantized Tensor Train (QTT) approximation

A *tensor* is an array with d indices (or modes)

$$\mathbf{X} = [\mathbf{X}(i_1, ..., i_d)], \quad i_p = 1, ..., n_p, \quad \text{for} \quad p = 1, 2, ..., d.$$

Then the *tensor train* format [21] reads $\mathbf{X}(i_1, ..., i_d) = \mathbf{X}^1_{i_1}\mathbf{X}^2_{i_2}...\mathbf{X}^d_{i_d}$, where each $\mathbf{X}^p_{i_p}$ is an $r_{p-1} \times r_p$ matrix and $r_0 = r_d = 1$. Values $r_1, ..., r_d$ are referred to as TT-ranks, and the 3-tensors $\mathbf{X}^p$ are referred to as TT-cores. This is the tensor train format for vectors. In the community of quantum physics and chemistry this format is known since longer as the matrix product states (MPS) representation, see for example [28, 26, 25, 27]. We refer to literature surveys on multilinear algebra in tensor formats [18, 15, 7, 6].

A quantics tensor approximation method of N-vectors was introduced in [14] and called quantics-TT (QTT) approximation. The QTT approximation method applies to a vector of size $N = q^d$ by approximating its quantics image in the canonical or TT form. The quantics image is a dth order tensor of size $q^{\otimes d}$ obtained by the standard reshaping of the target q^d-vector (the *q-adic folding*). Quantics tensor approximation method of $\log N$-complexity was justified by the rigorous proof of the uniform rank bound for a wide class of function generated vectors/tensors [14]. Basic multi-linear algebra in QTT format and its generalization [15, 4] can be implemented with logarithmic complexity scaling in N.

For a matrix, the only difference is that the TT-cores are not 3-tensors but 4-tensors. In particular, for a matrix $\mathbf{M}$ of size $N \times N$, elements can be indexed by $2d$-tuples $(i_1, ..., i_d, j_1, ..., j_d)$, where $(i_1, ..., i_d)$ enumerate the rows of M and $(j_1, ..., j_d)$ enumerate its columns. Hence, we say the matrix M is in TT-format if its elements are defined as

$$\mathbf{M}(i_1, ..., i_d, j_1, ..., j_d) = \mathbf{M}^1_{i_1,j_1}...\mathbf{M}^d_{i_d,j_d},$$

where $\mathbf{M}^p_{i_p,j_p}$ is $r_{p-1} \times r_p$ matrix.

We call the vector or matrix which can be folded (quantized) and then represented in TT form the *QTT vector or matrix*, respectively. First explicit examples on QTT matrix representation for Laplace-type operators and the respective theory can be found in [16, 8], see also further results in [2, 4, 1, 9].

We will use the definition of a tensor operation $\bowtie$ introduced in that paper.

Definition 2.2 *For two TT cores* U *and* V *of sizes* $p \times n \times m \times q$ *and* $q \times k \times l \times r$ *respectively, consisting of TT blocks* $A_{\alpha\gamma}$, $\alpha = 1, ..., p, \gamma = 1, ..., q$ *and* $B_{\gamma\beta}$, $\gamma = 1, ..., q, \beta = 1, ..., r$ *respectively, let us define their inner core product* $U \bowtie V$ *as a TT core of size* $p \times nk \times ml \times r$*, comprising TT blocks* $\sum_{\gamma=1}^{q} A_{\alpha\gamma}B_{\gamma\beta}$, $\alpha = 1, ..., p, \beta = 1, ..., r$*.*

In other words, $U \bowtie V$ is defined as a regular matrix product of the two core matrices, their elements (TT blocks) being multiplied by means of tensor product ([8]). For example,

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \bowtie \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11} \otimes B_{11} + A_{12} \otimes B_{21} & A_{11} \otimes B_{12} + A_{12} \otimes B_{22} \\ A_{21} \otimes B_{11} + A_{22} \otimes B_{21} & A_{21} \otimes B_{12} + A_{22} \otimes B_{22} \end{pmatrix}.$$

3 Superfast Haar Wavelet Transform

3.1 Low rank QTT format of modified Haar transform matrices

In the last section, we have seen that the Haar matrix $\mathcal{H}_N$ is the product of $H_{N,k}$, $k = 0, 1, .., d-1$. From the numerical tests, we saw that it does not have an efficient QTT decomposition, i.e. it has nearly full QTT rank. However, each factor $H_{N,k}$ has an efficient QTT approximation. Indeed, they have exact low rank QTT representations, and the next lemma proves this fact.

Lemma 3.1 *Let $H_{N,k}$ ($k = 0, 1, ..., d-1$) be defined in (2.1), then we have:*

(a) If $k = 0$, then $H_{N,0} = H_N = \begin{pmatrix} K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ & I_2 \end{pmatrix}^{\bowtie(d-2)} \bowtie \begin{pmatrix} H_2^+ \\ H_2^- \end{pmatrix}$.

(b) If $k = d-1$, then $H_{N,d-1} = \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ J_{22} & J_{11} \end{pmatrix}^{\bowtie(d-2)} \bowtie \begin{pmatrix} I_2 \\ H_2 \end{pmatrix}$.

(c) For $k = 1, ..., d-2$,

$$H_{N,k} = \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ J_{22} & J_{11} \end{pmatrix}^{\bowtie(k-1)} \bowtie \begin{pmatrix} K_3 & K_4 & & \\ & & K_1 & K_2 \end{pmatrix}$$
$$\bowtie \begin{pmatrix} J_{11} & J_{21} & & \\ J_{12} & J_{22} & & \\ & & I_2 & \\ & & & I_2 \end{pmatrix}^{\bowtie(d-k-2)} \bowtie \begin{pmatrix} I_2^+ \\ I_2^- \\ H_2^+ \\ H_2^- \end{pmatrix}.$$

Here I_m is the $m \times m$ identity matrix; $I_m^{\pm}$ is $\begin{pmatrix} I_{\frac{m}{2}} & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & I_{\frac{m}{2}} \end{pmatrix}$, respectively; J_{ij} is the 2×2 matrix with ij-th position to be 1 and rest to be 0, and K_i ($i = 1, 2, 3, 4$), define matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \text{ and } \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix},$$

respectively.

Proof. (a) Note that $H_{N,0} = H_N = K_1 \otimes H_{N/2}^+ + K_2 \otimes H_{N/2}^-$, then we have

$$H_N = \begin{pmatrix} K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} H_{N/2}^+ \\ H_{N/2}^+ \end{pmatrix}.$$

While $H^{\pm}_{N/2} = I_2 \otimes H_{N/4}$, we obtain

$$H_N = \begin{pmatrix} K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ & I_2 \end{pmatrix} \bowtie \begin{pmatrix} H^+_{N/4} \\ H^+_{N/4} \end{pmatrix}.$$

If we decompose in this fashion, the result in (a) follows.

(b) Note that $H_{N,d-1} = J_{22} \otimes I_{2^{d-1}} + J_{11} \otimes H_{N/2,d-2}$, then

$$H_{N,d-1} = \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_{2^{d-1}} \\ H_{N/2,d-2} \end{pmatrix}.$$

Also, $I_{2^{d-1}} = I_2 \otimes I_{2^{d-2}}$ and $H_{N/2,d-2} = J_{22} \otimes I_{2^{d-2}} + J_{11} \otimes H_{N/4,d-3}$, then we get

$$H_{N,d-1} = \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_{2^{d-2}} \\ H_{N/4,d-3} \end{pmatrix}.$$

We keep decomposing $\begin{pmatrix} I_{2^{d-2}} \\ H_{N/4,d-3} \end{pmatrix}$ in this fashion, and at the end get the resultant factorization.

(c) Note that $H_{N,k} = J_{22} \otimes I_{2^{d-1}} + J_{11} \otimes H_{N/2,k}$; hence we have

$$\begin{aligned} H_{N,k} &= \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_{2^{d-1}} \\ H_{N/2,k-1} \end{pmatrix} \\ &= \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & & \\ & J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_{2^{d-2}} \\ I_{2^{d-2}} \\ H_{N/4,k-2} \end{pmatrix} \\ &= \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_{2^{d-2}} \\ H_{N/4,k-2} \end{pmatrix} \\ &\vdots \\ &= \begin{pmatrix} J_{22} & J_{11} \end{pmatrix} \bowtie \begin{pmatrix} I_2 & \\ J_{22} & J_{11} \end{pmatrix}^{\bowtie(k-1)} \bowtie \begin{pmatrix} I_{2^{d-k}} \\ H_{2^{d-k},0} \end{pmatrix}. \end{aligned} \tag{3.1}$$

Now we will proceed with the further decomposition of $\begin{pmatrix} I_{2^{d-k}} \\ H_{2^{d-k},0} \end{pmatrix}$, taking into account that

$$I_{2^{d-k}} = K_3 \otimes I^+_{2^{d-k-1}} + K_4 \otimes I^-_{2^{d-k-1}},$$

$$H_{2^{d-k},0} = K_1 \otimes H^+_{2^{d-k-1}} + K_2 \otimes H^-_{2^{d-k-1}}.$$

Moreover,

$$\begin{aligned} I^+_{2^j} &= J_{11} \otimes I^+_{2^{j-1}} + J_{21} \otimes I^-_{2^{j-1}} \\ I^-_{2^j} &= J_{12} \otimes I^+_{2^{j-1}} + J_{22} \otimes I^-_{2^{j-1}}. \\ H^{\pm}_{2^j} &= I_2 \otimes H^{\pm}_{2^{j-1}}. \end{aligned}$$

Therefore, we obtain one step recursion

$$\begin{pmatrix} I_{2^{d-k}} \\ H_{2^{d-k},0} \end{pmatrix} = \begin{pmatrix} K_3 & K_4 & & \\ & & K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} I^+_{2^{d-k-1}} \\ I^-_{2^{d-k-1}} \\ H^+_{2^{d-k-1}} \\ H^-_{2^{d-k-1}}, \end{pmatrix}$$

and then proceed

$$\begin{aligned} \begin{pmatrix} I_{2^{d-k}} \\ H_{2^{d-k},0} \end{pmatrix} &= \begin{pmatrix} K_3 & K_4 & & \\ & & K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} J_{11} & J_{21} & & \\ J_{12} & J_{22} & & \\ & & I_2 & \\ & & & I_2 \end{pmatrix} \bowtie \begin{pmatrix} I^+_{2^{d-k-2}} \\ I^-_{2^{d-k-2}} \\ H^+_{2^{d-k-2}} \\ H^-_{2^{d-k-2}} \end{pmatrix} \\ &\vdots \\ &= \begin{pmatrix} K_3 & K_4 & & \\ & & K_1 & K_2 \end{pmatrix} \bowtie \begin{pmatrix} J_{11} & J_{21} & & \\ J_{12} & J_{22} & & \\ & & I_2 & \\ & & & I_2 \end{pmatrix}^{\bowtie(d-k-2)} \bowtie \begin{pmatrix} I^+_2 \\ I^-_2 \\ H^+_2 \\ H^-_2 \end{pmatrix} \end{aligned} \tag{3.2}$$

Now combining equations (3.1), (3.2) leads to the result. ∎

Remark 3.2 *¿From Lemma 3.1 we can see that each $H_{N,k}$ has low-rank QTT representation with the ranks bounded by 4. However, if we directly decompose $H_{N,k}$ using the TT toolbox in Matlab, we do not get an exact low-rank decomposition. Indeed numerical experiments show that the QTT rank of $H_{N,k}$ is bounded by 5, which is still an acceptable low QTT rank.*

Nonetheless, if we compute the exact QTT representation of $H_{N,k} * x$, the rank will be at least multiplied by a factor 2 each time, and the ultimate rank will be 2^d which is a full rank, making the QTT representation completely ineffective. As a consequence, after each step, we have to truncate the QTT-ranks, i.e. approximate with a lower QTT-rank tensor.

Factorization in Lemma 3.1 leads to the useful rank truncation algorithm by TT-rounding applied after multiplication by each individual factor. Wavelet transform has similar orthogonal structure as discrete Fourier transform; therefore similar to superfast Fourier transform in [1], the TT-rounding can also be effectively accomplished with our superfast Haar wavelet transform. With this problem solved, we formally get our superfast Haar wavelet transform procedure sketched in Algorithm 2. Recall SFHWT is equivalent to recursively multiplying $\mathbf{x}$ with $H_{N,k}$ $(k = 0, 1, ..., d-1)$, and Algorithm 2 simply implements the $H_{N,k}\mathbf{x}$ in QTT format with the rank optimization.

3.1.1 Complexity Analysis

We define

$$r = max(r_H, r_x),$$

where r_H is the maximal QTT rank of the modified Haar transform matrices in each recursion of Algorithm 2, and r_x is the maximal QTT rank of input vector x and all intermediate vectors $y = H_{N,k} * x$ $(k = 0, 1, ..., d-1)$.

Algorithm 2: Superfast Haar Wavelet Transform

Data: Vector x in the QTT-format with cores $X_l(j_l)$, and matrices $H_{N,k}$ in the QTT-format with cores $H_l(i_l, j_l)$ and $k = 0, ..., d-1$

Result: Output vector y in QTT-format with cores Y_l.

begin

  for $k = 0$ **to** $d-1$ **do**

    for $l = 1$ **to** d **do**

      $Y_l(i_l) = \sum_{j_l} H_l(i_l, j_l) \otimes X_l(j_l)$;

    $y = QTT - rounding(y, accuracy)$;

    $x = y$;

end

Lemma 3.3 *The complexity of Algorithm 2 is estimated by* $O(d^2r^3)$.

Proof. Note that complexity of matrix-vector multiplication in QTT format is $O(dn^2r^4)$, where n is the dth root of N. However, since we know that r_H is always bounded by 4 and $n = 2$, the complexity of one step recursion in our algorithm can be reduced to $O(dr^2)$.

Hence, at ith recursion, the complexity of $H_{2^{d+1-i}} * x$ in QTT format is bounded by $O(dr^2)$. While complexity of TT rounding algorithm is bounded by $O(dr^3)$, we conclude that the complexity of each recursion is bounded by $O(dr^3)$.

Finally, since in total we have d recursions, the overall complexity is estimated by $O(d^2r^3)$, where r depends on the input vector and approximation accuracy. ∎

In view of above considerations, it is important to keep r low for sake of minimizing the computing time. Numerical experiments show that image rank usually keeps relatively low and stable, indicating that the rounding algorithm in SFHWT is effective. The rank of wavelet image can be an interesting topic to discuss: although we cannot prove rigorously the rank bound, numerical results do show that it remains low and increases slowly (even may decrease!) as d increases. However, this uncertainty is a flaw of our SFHWT algorithm. In general, the algorithm may fail if the given input vector is not "structurally regular" enough, i.e. it has large QTT rank. Therefore, it is important to find another way to alleviate this problem.

Furthermore, to practically implement this algorithm, we, first, have to obtain the QTT approximation of Haar transform matrices $H_{N,k}$. To this end, there are basically two ways: either to directly utilize the explicit format we proved in Lemma 3.1, or to apply the reshaping and TT matrix procedures to numerically obtain the QTT matrix.

However, both of them have certain drawbacks. The second approach is limited by the high cost of numerically obtaining QTT format of $H_{N,k}$, hence restricting our computations to relatively small vector size. Concerning the first approach, there is a problem with the format of wavelet images. More precisely, we illustrate the point by taking an exponential input vector as an example.

Given vector x in the QTT-format with cores $X_l(j_l)$ which consists of a single vector block of size 2 by 1, and matrices $H_{N,k}$ in the QTT-format with cores $H_l(i_l, j_l)$. ¿From the algorithm, we know that the cores of wavelet image $\mathbf{y} = H_{N,k}\mathbf{x}$ are given by $Y_l(i_l) = \sum_{j_l} H_l(i_l, j_l) \otimes X_l(j_l)$. However, in Lemma 3.1, the last core of matrix $H_{N,k}$ consists of

blocks of size 1×2 and some core blocks K_i have size 4×2. Hence, when multiplying $X_l(j_l)$, which is a single vector block of size 2 by 1, with those matrix cores above, we obtain that $Y_l(i_l) = \sum_{j_l} H_l(i_l, j_l) \otimes X_l(j_l)$ is a single vector block of size 1×1 or 4×1, thus changing the shape of the whole QTT vector $\mathbf{y}$. As a result, if we use this $\mathbf{y}$ for the next recursion, we will encounter the error of multiplying matrices, which have row size 2, with vectors of size 1 and 4.

In order to keep vector $\mathbf{y}$ suitable for next recursion, we apply TT reshape for making all sizes of blocks in the QTT cores of vector $\mathbf{y}$ to be 2×1 again. Although the reshaping algorithm is not formally counted for computing complexity, it does require certain computational time. This problem may be solved if we can find another way for explicit QTT representation of $H_{d,k}$, but till now it is still an open problem.

3.2 A Multilevel Algorithm for SFHWT

In the previous section, we have described Algorithm 2 which is prior to traditional FHWT, but with certain drawbacks mentioned in the previous discussion.

Hence, we need another algorithm to implement SFHWT more efficiently. The idea is the following. Instead of expecting the whole wavelet image vector in QTT format, we make the output as a sequence of QTT tensors in decreasing size. More explicitly, let us analyze in more detail how fast Haar wavelet transform works on the input vector: after each recursion, we only keep the low frequency part of image for next recursion but store the high frequency part. Now the new algorithm reduces to application of QTT approximation to high frequency part only, accomplished with simple algebraic operations. In particular, consider one loop of FHWT, $\mathbf{y} = H_{2^d}\mathbf{x}$, which has low frequency part $\mathbf{y}^+$ and high frequency part $\mathbf{y}^-$; then the next recursion reduces to $H_{2^{(d-1)}}\mathbf{y}^+$. Based upon the factorization in FHWT, the new algorithm represents only "short" vectors of type $\mathbf{y} = H_{2^{(d-k)}}\mathbf{y}^+$ in the QTT format. This reduces the considerable part of the computing time and now the output consists of image vectors of different frequencies over all levels.

But the problem of compatibilities of format dimensions after TT rounding mentioned in the last section may also appear. Thus, we are looking for another explicit QTT representation of modified Haar transform matrices. This time we do not need application of $H_{N,k}$ for all k, because the new algorithm only makes use of $H_{N,0} = H_N$ for dyadic decreasing size N, i.e., we are applying H_N on different levels.

Lemma 3.4 *Instead of QTT representations in Lemma 3.1 (a), we have*

$$H_N = \begin{pmatrix} J_{11} & J_{12} & J_{21} & J_{22} \end{pmatrix} \bowtie \begin{pmatrix} J_{11} & J_{12} & & \\ J_{21} & J_{22} & & \\ & & J_{11} & J_{12} \\ & & J_{21} & J_{22} \end{pmatrix}^{\bowtie(d-2)} \bowtie \begin{pmatrix} H_2^{+,1} \\ H_2^{+,2} \\ H_2^{-,1} \\ H_2^{-,2} \end{pmatrix},$$

where $H_2^{+,1} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$, $H_2^{+,2} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$, $H_2^{-,1} = \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$, *and* $H_2^{-,2} = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix}$.

Proof. Let us introduce

$$H_N^{\pm,1} = \begin{pmatrix} H_N^{\pm} \\ 0 \end{pmatrix}, \quad H_N^{\pm,2} = \begin{pmatrix} 0 \\ H_N^{\pm} \end{pmatrix},$$

where $H_N^{\pm}$ are defined as in Section 2.1. We have

$$H_N = J_{11} \otimes H_{N/2}^{+,1} + J_{12} \otimes H_{N/2}^{+,2} + J_{21} \otimes H_{N/2}^{-,1} + J_{22} \otimes H_{N/2}^{-,2}.$$

Therefore,

$$H_N = \begin{pmatrix} J_{11} & J_{12} & J_{21} & J_{22} \end{pmatrix} \bowtie \begin{pmatrix} H_{N/2}^{+,1} \\ H_{N/2}^{+,2} \\ H_{N/2}^{-,1} \\ H_{N/2}^{-,2} \end{pmatrix}.$$

Since $H_{N/2}^{\pm,1} = J_{11} \otimes H_{N/4}^{\pm,1} + J_{12} \otimes H_{N/4}^{\pm,2}$ and $H_{N/2}^{\pm,2} = J_{21} \otimes H_{N/4}^{\pm,1} + J_{22} \otimes H_{N/4}^{\pm,2}$, we obtain

$$\begin{pmatrix} H_{N/2}^{+,1} \\ H_{N/2}^{+,2} \\ H_{N/2}^{-,1} \\ H_{N/2}^{-,2} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & & \\ J_{21} & J_{22} & & \\ & & J_{11} & J_{12} \\ & & J_{21} & J_{22} \end{pmatrix} \bowtie \begin{pmatrix} H_{N/4}^{+,1} \\ H_{N/4}^{+,2} \\ H_{N/4}^{-,1} \\ H_{N/4}^{-,2} \end{pmatrix}.$$

Then in the next decomposition, we can similarly get

$$\begin{pmatrix} H_{N/4}^{+,1} \\ H_{N/4}^{+,2} \\ H_{N/4}^{-,1} \\ H_{N/4}^{-,2} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & & \\ J_{21} & J_{22} & & \\ & & J_{11} & J_{12} \\ & & J_{21} & J_{22} \end{pmatrix} \bowtie \begin{pmatrix} H_{N/8}^{+,1} \\ H_{N/8}^{+,2} \\ H_{N/8}^{-,1} \\ H_{N/8}^{-,2} \end{pmatrix}.$$

We proceed decomposing $\begin{pmatrix} H_{N/8}^{+,1} \\ H_{N/8}^{+,2} \\ H_{N/8}^{-,1} \\ H_{N/8}^{-,2} \end{pmatrix}$ in this fashion and get the desired result. ■

Having at hand the explicit QTT representation for H_N, the problem of virtual reshaping is solved as shown in Algorithm 3.

Algorithm 3: Multilevel Superfast Haar Wavelet Transform

Data: Vector x in the QTT-format with cores $X_l(j_l)$, and matrices $H_{2^k,0}$ in the QTT-format with cores $H_l^k(i_l, j_l)$ and $k = 0, ..., d-1$.
Result: Array of wavelet images W in QTT representation.
begin
  for $k = 0$ **to** $d-1$ **do**
    for $l = 1$ **to** d **do**
      $Y_l(i_l) = \sum_{j_l} H_l^k(i_l, j_l) \otimes X_l(j_l)$;
    $y = TT - rounding(y, accuracy)$;
    $x = reshape(y(:, ..., :, 1), 2 * ones(1, d-1-k))$;
    $W(k+1) = y(:, ..., 2)$;
end

3.2.1 Complexity Analysis and Rank Control

Next Lemma proves the complexity of Algorithm 3.

Lemma 3.5 *Let r be as in Lemma 3.3. Then the complexity of Algorithm 3 is of order $O(\frac{d^2}{2}r^3)$.*

Proof. Similar to Algorithm 2, the complexity of each recursion is of order $O(dr^3)$. Since at each recursion step the coefficient size is half of the previous one, i.e. it reduces from 2^d to 2^{d-1}, etc., the total cost is bounded by $O(dr^3) + O((d-1)r^3) + ... + O(r^3) = O(\frac{d^2}{2}r^3)$. ∎

The main difference between Algorithms 2 and 3 is that for the latter the parameter r represents the maximum rank of all Haar transform matrices (but not of the modified Haar transform matrices as in Algorithm 2), intermediate results, and the input vector. Notice that we still have TT reshaping procedure involved in Algorithm 3, but with a decreasing size of QTT vectors. Consequently, Algorithm 3 is faster than Algorithm 2 when using the explicit matrix representation as in Lemma 3.1.

Algorithm 3 undermines the uncertainty of image rank in Algorithm 2 because the structure of coefficients in each recursion remains similar. For example, the QTT ranks of exponential and trigonometric vectors are provec to be exactly 1 and 2, respectively [14]. Then we can prove that QTT ranks of low frequency part at each recursion step of these two inputs are always 1 and 2. For uniformly sampled polynomial vector, the QTT rank is bounded by $m+1$, where m is the degree of polynomial [14]. Again, we are able to prove that the rank of low frequency coefficient at each recursion step is bounded by $m+1$. The following theorem states the final result on the uniform rank bounds for multilevel algorithm applied to the special class of functional vectors.

Theorem 3.6 *Let the input in Algorithm 3 is given by (a) exponential, (b) trigonometric or (c) degree-m polynomial vector sampled on the uniform grid. Then the QTT rank of low-frequency part at each recursion step does not exceed* 1, 2 *and* $m+1$*, respectively.*

Proof. (a) Without loss of generality, we assume the input vector is of form $(z^h, ..., z^{2^d h})$. We claim that in k-th recursion, the wavelet image is still an exponential vector for any $k = 1, ..., d$. This claim can be proved by induction. For $k = 1$, the image vector is given by

$$\begin{pmatrix} z^h + z^{2h} \\ \vdots \\ z^{(2^d-1)h} + z^{2^d h} \end{pmatrix} = z^{-h}(1+z^h) \begin{pmatrix} z^{2h} \\ z^{2h\cdot 2} \\ \vdots \\ z^{2h\cdot 2^{d-1}} \end{pmatrix},$$

which is again an exponential vector. Now assume that for $k = n-1$ the image is an exponential vector

$$C \begin{pmatrix} z^{h'} \\ z^{2h'} \\ \vdots \\ z^{2^{d-n+1}h'} \end{pmatrix},$$

then by the same procedure as in the basic case, the image of next recursion is still exponential since the constant h' does not affect our result.

(b) For trigonometric vector, we note that $\sin x = \frac{e^{ix}-e^{-ix}}{2i}$, $\cos x = \frac{e^{ix}+e^{-ix}}{2}$ and the result follows from (a).

(c) It can be proved using induction again. Let $P_m(x)$ be the degree m polynomial uniformly sampled on the interval with mesh size h. Therefore the input vector is $(P_m(h), P_m(2h), ..., P_m(2^d h)$. We claim that in k-th recursion, the wavelet image is still a degree-m polynomial vector, where $k = 1, ..., d$. Indeed, for $k = 1$, the image vector is

$$\begin{pmatrix} P_m(h) + P_m(2h) \\ \vdots \\ P_m((2^d-1)h) + P_m((2^d)h) \end{pmatrix} = \begin{pmatrix} P'_m(h) \\ P'_m(3h) \\ \vdots \\ P'_m((2^d-1)h) \end{pmatrix},$$

where

$$P'_m(x) = P_m(x) + P_m(x+h),$$

which is again a polynomial of degree m. Therefore, the coefficient is still a uniformly sampled polynomial vector of degree m. Now assume that for $k = n-1$ the image is a polynomial vector

$$\begin{pmatrix} P_m(h') \\ P_m(2h') \\ \vdots \\ P_m(2^{d-n+1}h') \end{pmatrix},$$

then by the same argument as in the basic case, the coefficient of next recursion is still a polynomial vector. ■

4 Haar Wavelet Transform Inverse

In above sections, we have described superfast QTT-based algorithms of logarithmic complexity for Haar wavelet transform. In this section the similar algorithm will be described for the inverse transform. Note that Haar transform matrices and modified Haar transform matrices are all orthogonal, i.e.

$$H_N^T H_N = I_N \quad \text{and} \quad H_{N,k}^T H_{N,k} = I_N.$$

Since inverse Haar wavelet transform has very similar structure with the direct transform, a superfast algorithm for the inverse transform can be designed analogously to the previous schemes as in Algorithms 2 and 3, respectively. First, we focus on FHWT and note that the modified Haar transform matrices $H_{N,k}$ are all orthogonal (after normalization), then the inverse FHWT can obviously be described by Algorithm 4 which has exactly the same complexity as Algorithm 1 (FHWT).

Then the superfast inverse HWT (SFIHWT) corresponding to Algorithm 2 is constructed by simply representing (approximating) vectors and matrices in Algorithm 4 in QTT format. For inverse transform associated with Algorithm 3, it is a bit more complicated because at

Algorithm 4: Inverse Fast Haar Wavelet Transform

Data: Wavelet image vector y of size $N = 2^d$; Modified Haar transform matrices $H_{N,k}$ where $k = 0, ..., d-1$.

Result: Input vector x of same size as y.

begin

  for $k = 0$ **to** $d-1$ **do**

    $x = H_{N,k} * y$;

  $x = y$;

end

Algorithm 5: Inverse Multilevel Superfast Haar Wavelet Transform

Data: Array of wavelet images W in QTT form, matrices $H^T_{2^k,0} = H^T_{2^k}$ in the QTT-format, $k = 0, ..., d-1$.

Result: Input vector x in QTT format

begin

  for $k = 1$ **to** $d-1$ **do**

    $m = reshape(W(d-k), 2 * ones(1, k))$;

    if *k=1* **then**

      $x = reshape([H^T_{2^k} * W(d), m], 2 * ones(1, k+1))$;

    else

      $x = reshape([H^T_{2^k} * x, m], 2 * ones(1, k+1))$;

    $x = TT - rounding(x, accuracy)$;

  $x = H^T_{2^d} * x$;

  $x = TT - rounding(x, accuracy)$

end

each recursion step we have to concatenate QTT tensors representing different frequencies. Algorithm 5 describes this procedure.

This algorithm is more time consuming compared with Algorithm 3 because of the concatenation of tensors. Here we again use notation $*$ for the matrix-vector multiplication in QTT format.

5 Numerical Tests

5.1 Illustrations to direct SFHWT Algorithms

As it was noticed in Section 3.2.1, Algorithm 3 beats Algorithm 2 in computing time if the QTT matrix representations in Algorithm 2 are given as in Lemma 3.1. However, if $H_{N,k}$ is approximated numerically (even though this procedure is very expensive, we assume it is given for free) and use the result as input in Algorithm 2, the CPU time will be reduced since the TT reshaping is not needed any more. In particular, given the low QTT-rank input vector, Algorithm 3 needs slightly more time than Algorithm 2 because of the reshaping algorithm involved.

Moreover, Algorithm 3 sometimes even beats Algorithm 2 in computing time.

In some cases, the image ranks of Algorithm 2 could be high, thus taking more time to execute. The point is that we have no proofs that the rank of ultimate wavelet images in Algorithm 2 could be controlled, even though the numerical tests give often the positive results. On the contrary, Theorem 3.6 guaranties rank stability for Algorithm 3 on a wide class of input data. In general, since in each recursion the input vector has almost same structure, and decreasing size, the ranks could be kept low (and stable) and even decreasing.

Following tables give the numerical comparison between Algorithms 2 and 3 in computing time and effective ranks of wavelet images at all recursion steps. Uniform sampling vectors for the following functions on unit interval $[0, 1]$ are taken as examples, $e^x, \sin(x), \sin(100x), (x+1)^{10}$. The approximation accuracy is taken as $\epsilon = 10^{-6}$. Here $\overline{r}$ is the average rank described in [14], and $\tilde{r}$ is the maximum of $\overline{r}$ oven all recursions.

	e^x				$\sin(x)$			
d	$time_{Alg2}$	$\overline{r}$	$time_{Alg3}$	$\tilde{r}$	$time_{Alg2}$	$\overline{r}$	$time_{Alg3}$	$\tilde{r}$
8	22	3.44	41	1	25	3.75	44	2
9	27	3.52	48		30	3.79	52	
10	32	3.47	55		36	3.81	60	
11	38	3.42	65		43	3.75	70	
12	45	3.38	74		50	3.68	80	
13	51	3.28	83		59	3.57	90	

	$\sin(100x)$				$(x+1)^{10}$			
d	$time_{Alg2}$	$\overline{r}$	$time_{Alg3}$	$\tilde{r}$	$time_{Alg2}$	$\overline{r}$	$time_{Alg3}$	$\tilde{r}$
8	27	5.43	44	2	27	4.29	45	$\leqslant$3.33
9	34	6.23	51		33	4.40	53	
10	44	7.12	60		41	4.42	62	
11	57	7.95	68		49	4.48	72	
12	76	8.72	80		57	4.44	82	
13	94	9.32	90		66	4.40	93	

¿From these tables we can see that when r is low, Algorithm 2 usually outperforms Algorithm 3 in trivial amount of time, but for r becoming higher this advantage gradually disappears. For instance, in the case of $\sin(100x)$, r increases when d grows in Algorithm 2, making time growing significantly. While r is always the same for Algorithm 3, we can see that already for $N = 2^{13}$, i.e. for $d = 13$, Algorithm 3 already outperforms Algorithm 2.

For the comparison between FHWT and SFHWT, we take Algorithm 3 for SFHWT since Algorithm 2 cannot be practically implemented to higher dimensions d as mentioned in the previous sections. In the following table, we show CPU time for SFHWT (in milliseconds) with respect to vector size $N = 2^d$ and accuracy $\epsilon = 10^{-6}$. Our input vectors are generated by e^x, $\sin(100x)$, and $(x+1)^{10}$, using their uniform sampling on the unit interval. Here $time_{FHWT}$ means the runtime of the FHWT.

d	$time_{FHWT}$	e^x $time_{SFHWT}$	$\sin(100x)$ $time_{SFHWT}$	$(x+1)^{10}$ $time_{SFHWT}$
13	6	84	97	99
14	11	94	108	112
15	22	104	120	123
16	45	115	132	136
17	90	125	144	150
18	180	137	157	164
19	367	151	172	180
20	730	163	185	192
21	1500	177	199	210
22	2900	190	214	226

All experiments are performed for fixed $\epsilon = 10^{-6}$ since it was proved that for exponential, trigonometric, and polynomial vectors, we have exact-rank QTT representation for each coefficient. Thus, theoretically the running time should not depend on ϵ for these three inputs. ¿From the above table we see that the running time of FHWT exceeds those of SFHWT for $d \geq 18$.

Furthermore, we considered the Gaussian e^{-px^2} on the interval $[-1, 1]$ for $p = 1, 100, 1000$. The purpose of this numerical experiment is to test the wavelet transform of functions with sharp behaviour and with local support. Compared with the Fourier transform which is only localized in frequency domain, the wavelet transform is also localized in time. As a result, when dealing with signals having almost singular behavior, the wavelet transform has certain advantages for giving more information about particular location of singularities. Based upon these facts, we can choose Gaussian with large parameter p, to obtain a function

which is almost singular at 0. By testing these examples, we can see that Algorithm 3 also has a good performance for this class of functions. Accuracy here is again fixed to $\epsilon = 10^{-6}$.

d	e^{-px^2} $time_{FHWT}$	$p = 1$ $time_{SFHWT}$	$p = 100$ $time_{SFHWT}$	$p = 1000$ $time_{SFHWT}$
13	6	103	104	107
14	11	115	121	115
15	22	125	134	133
16	45	138	146	148
17	90	149	164	157
18	180	167	175	174
19	367	179	195	196
20	730	195	212	210
21	1500	214	228	229
22	2900	231	244	251

Concerning data compression, recall that $O(\log N)$ estimate on storage size of QTT approximation [14], will lead to quite outstanding results. The following table shows the storage saving for input functions e^x, $\sin(100x)$, and $(x+1)^{10}$. The percentage in the table is computed via storage size of output of Algorithm 3 divided by full vector size $N = 2^d$. Here the accuracy ϵ is again 10^{-6}.

d	$e^x(\%)$	$\sin(100x)(\%)$	$(x+1)^{10}(\%)$	$e^{-x^2}(\%)$	$e^{-100x^2}(\%)$	$e^{-1000x^2}(\%)$
13	2.08	7.35	15.87	21.70	37.96	31.32
14	1.20	4.30	9.13	12.51	22.36	18.86
15	0.69	2.48	5.19	7.11	12.93	10.94
16	0.39	1.42	2.90	3.98	7.36	6.26
17	0.22	0.81	1.61	2.20	4.12	3.53
18	0.12	0.45	0.89	1.21	2.28	1.97
19	0.07	0.25	0.49	0.66	1.25	1.09
20	0.04	0.14	0.26	0.36	0.68	0.60
21	0.02	0.08	0.14	0.19	0.37	0.33
22	0.01	0.04	0.08	0.10	0.20	0.18

5.2 Numerical Tests of Inverse Transform

Here we don not consider the computing time of inverse transform, but just test the accuracy in the original input of SFHWT being approximated by using the inverse transform. The rule is as following.

Given an input vector x, applying SFHWT (Algorithm 3) with accuracy ϵ, we obtain a chain of wavelet images W of different frequencies. At the end, we apply inverse transform by Algorithm 5 to W with accuracy ϵ, obtaining an approximating input vector $\hat{x}$. We are interested in the bound on the relative approximation error $\frac{||x-\hat{x}||}{||x||}$.

In accordance with Theorem 3.6, exponential and trigonometric vectors have exact low-rank QTT representations in each recursion step; therefore for these two input vectors, we always obtain the exact original input no matter which ϵ is given, i.e. $||x - \hat{x}|| = 0$.

In the following table, we present the results for input vector obtained by sampling function $(x+1)^{10}$, and all results in this table should be multiplied with 10^{-7}.

d	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
13	4.7646	1.2077	1.2069	1.2069
14	4.7651	1.2086	1.2078	1.2078
15	4.7651	1.2087	1.2080	1.2080
16	4.7896	1.2088	1.2080	1.2080
17	4.7979	1.2088	1.2080	1.2080
18	4.8000	1.2087	1.2080	1.2080
19	4.8006	1.2087	1.2080	1.2080
20	4.9614	1.7411	1.7406	1.7406
21	5.0008	1.8504	1.8499	1.8499
22	5.0106	1.8767	1.8762	1.8762

The next table represents the results for Gaussian e^{-100x^2} sampled on $[-1, 1]$.

d	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
13	5.7171	0.7815	0.7803	0.7803
14	5.7163	0.8150	0.8138	0.8138
15	6.2941	2.7596	2.7596	2.7596
16	5.7542	1.0561	1.0552	1.0552
17	5.7607	1.1182	1.1179	1.1179
18	5.7625	1.1347	1.1338	1.1338
19	5.7630	1.1386	1.1378	1.1378
20	5.7631	1.1396	1.1388	1.1388
21	5.7631	1.1395	1.1390	1.1390
22	5.7631	1.1399	1.1391	1.1391

We can see that the relative errors are small and stable, which mean that our algorithm is robust in d.

6 Conclusions

In this paper, we proved that the Haar transform matrices have low QTT-rank r, which is a prerequisite for developing superfast algorithms for Haar wavelet transform of logarithmic complexity scaling. We introduced two algorithms implementing our QTT-based superfast Haar wavelet transform both of asymptotic complexity $O(d^2r^3)$, thus outperforming the traditional fast Haar wavelet transform which scales as $O(N)$, where $N = 2^d$. The first algorithm applies to modified HWT matrix, while the second one only treats the low frequency part of a vector on the current recursion step, e.g. the output formats of these algorithms are different. Comparing the ranks of intermediate wavelet images and the respective computing times by numerical experiments, we draw the conclusion that Algorithm 3 is better than Algorithm 2 concerning the rank control, and it even outperforms Algorithm 2 if the image rank in Algorithm 2 is getting higher. Also from the numerical experiments we observe that

our SFHWT outperforms the traditional FHWT when dimension d is higher than some fixed value (say, for $d \geq 18$).

Future research can be carried out in several directions. First goal is to develop an analogous algorithm for higher dimensional Haar wavelet transform, say, 2D and 3D Haar wavelets. This can be done similarly to 1D case since higher dimensional wavelet transform is the tensor product of univariate ones. The second option can be developing superfast algorithms in QTT representation for different families of wavelet transforms and their application in data processing.

Acknowledgements. The first author would like to thank Dr. Venera Khoromskaia (Max-Planck-Institute for Mathematics in the Sciences, Leipzig) for useful comments on the presentation.

References

[1] S.V. Dolgov, B.N. Khoromskij, and D. Savostianov. *Superfast Fourier transform using QTT approximation.* J. Fourier Anal. Appl., v. 18(5):915–953, 2012.

[2] S. Dolgov, and B.N. Khoromskij. *Tensor-product approach to global time-space-parametric discretization of chemical master equation.* Preprint 68/2012, MPI MiS, Leipzig 2012, submitted.

[3] S. Dolgov, and B.N. Khoromskij. *Simultaneous state-time approximation of the chemical master equation using tensor product formats.* arXiv:1311.3143, 2013; http://arxiv.org/abs/1311.3143, (submitted).

[4] S.V. Dolgov, and B.N. Khoromskij. *Two-level Tucker-TT-QTT format for optimized tensor calculus.* SIAM J. on Matr. Anal. Appl., **34**(2), 2013, pp. 593-623.

[5] I.V. Gavrilyuk, and B.N. Khoromskij. *Quantized-TT-Cayley transform to compute dynamics and spectrum of high-dimensional Hamiltonians.* Comp. Meth. in Applied Math., v.11 (2011), No. 3, 273-290.

[6] L. Grasedyck, D. Kressner and C. Tobler. *A literature survey of low-rank tensor approximation techniques.* arXiv:1302.7121v1, 2013.

[7] W. Hackbusch. *Tensor spaces and numerical tensor calculus.* Springer, 2012.

[8] V. Kazeev, and B.N. Khoromskij. *Explicit low-rank QTT representation of Laplace operator and its inverse.* SIAM Journal on Matrix Anal. and Appl., 33(3), 2012, 742-758.

[9] V. Kazeev, B.N. Khoromskij, and E.E. Tyrtyshnikov. *Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity.* SIAM J. Sci. Comput. 35-3 (2013), pp. A1511-A1536.

[10] V. Khoromskaia. *Black-Box Hartree-Fock Solver by the Tensor Numerical Methods.* Comp. Meth. in Applied Math., 2013, doi: 10.1515/cmam-2013-0023. (Preprint 90/2013, MPI MIS, Leipzig 2013.)

[11] V. Khoromskaia, B.N. Khoromskij, and R. Schneider. *Tensor-structured calculation of two-electron integrals in a general basis.* SIAM J. Sci. Comput., **35**(2), 2013, A987-A1010.

[12] V. Khoromskaia and B.N. Khoromskij. *Møller-Plesset (MP2) Energy Correction Using Tensor Factorizations of the Grid-based Two-electron Integrals.* Comp. Phys. Communications 185 (2014), pp. 2-10. Preprint 26/2013, MPI MiS, Leipzig 2013.

[13] B.N. Khoromskij. *$O(d \log N)$-Quantics Approximation of N-d Tensors in High-Dimensional Numerical Modeling.* Preprint 55/2009, Max-Planck Institute for Mathematics in the Sciences, Leipzig 2009; http://www.mis.mpg.de/publications/preprints/2009/prepr2009-55.html.

[14] B.N. Khoromskij. *$O(d \log N)$-Quantics Approximation of N-d Tensors in High-Dimensional Numerical Modeling.* J. Constr. Approx., v. 34(2), 257-289 (2011).

[15] B.N. Khoromskij. *Tensors-structured Numerical Methods in Scientific Computing: Survey on Recent Advances.* Chemometr. Intell. Lab. Syst. 110 (2012), 1-19.

[16] B.N. Khoromskij, and I. Oseledets. *Quantics-TT approximation of elliptic solution operators in higher dimensions.* Russ. J. Numer. Anal. Math. Modeling, v. 26(3), pp. 303-322 (2011).

[17] J. Ko, A. Kurdila, P. Oswald, *Scaling function and wavelet preconditioners for second order elliptic problems.* In: Multiscale Wavelet Methods for PDEs, W. Dahmen, A. Kurdila, P. Oswald (eds.), Academic Press, 1997, 413-438.

[18] T. G. Kolda and B. W. Bader. *Tensor Decompositions and Applications.* SIAM Rev. 51(3) (2009) 455–500.

[19] Mallat S. *A wavelet tour of signal processing.* Access Online via Elsevier, 1999.

[20] Mohlenkamp M.J., Pereyra M.C. *Wavelets: Their Friends and what They Can Do for You.* European Mathematical Society, 2008.

[21] I.V. Oseledets, and E.E. Tyrtyshnikov, *Breaking the Curse of Dimensionality, or How to Use SVD in Many Dimensions.* SIAM J. Sci. Comp., 31 (2009), 3744-3759.

[22] I.V. Oseledets, *Approximation of* $2^d \times 2^d$ *matrices using tensor decomposition.* SIAM J. Matrix Anal. Appl., 31(4):2130-2145, 2010.

[23] Oseledets I V, Tyrtyshnikov E E. *Algebraic wavelet transform via quantics tensor train decomposition.* SIAM J. Sci. Comput., 2011, 33(3): 1315-1328.

[24] P. Oswald, *Multilevel solvers for elliptic problems on domains.* In: Multiscale Wavelet Methods for PDEs, W. Dahmen, A. Kurdila, P. Oswald (eds.), Academic Press, 1997, 3-58.

[25] F. Verstraete, D. Porras, and J.I. Cirac, *DMRG and periodic boundary conditions: A quantum information perspective.* Phys. Rev. Lett., 93(22): 227205, Nov. 2004.

[26] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations.* Phys. Rev. Lett. 91(14), 2003, 147902-1 147902-4.

[27] H. Wang, and M. Thoss, *Multilayer formulation of the multiconfiguration time-dependent Hartree theory.* J. Chem. Phys. 119 (2003), 1289-1299.

[28] S.R. White, *Density-matrix algorithms for quantum renormalization groups.* Phys. Rev. B, v. 48(14), 1993, 10345-10356.