Eur. Phys. J. Plus (2024) 139:394
https://doi.org/10.1140/epjp/s13360-024-05184-1

THE EUROPEAN PHYSICAL JOURNAL PLUS

Regular Article

Fast Fourier transforms and fast Wigner and Weyl functions in large quantum systems

C. Lei[a], **A. Vourdas**[b]

Department of Computer Science, University of Bradford, Bradford BD7 1DP, UK

Received: 25 March 2024 / Accepted: 16 April 2024

Abstract Two methods for fast Fourier transforms are used in a quantum context. The first method is for systems with dimension of the Hilbert space $D = d^n$ with d an odd integer, and is inspired by the Cooley-Tukey formalism. The 'large Fourier transform' is expressed as a sequence of n 'small Fourier transforms' (together with some other transforms) in quantum systems with d-dimensional Hilbert space. Limitations of the method are discussed. In some special cases, the n Fourier transforms can be performed in parallel. The second method is for systems with dimension of the Hilbert space $D = d_0...d_{n-1}$ with d_0, ..., d_{n-1} odd integers coprime to each other. It is inspired by the Good formalism, which in turn is based on the Chinese reminder theorem. In this case also the 'large Fourier transform' is expressed as a sequence of n 'small Fourier transforms' (that involve some constants related to the number theory that describes the formalism). The 'small Fourier transforms' can be performed in a classical computer or in a quantum computer (in which case we have the additional well known advantages of quantum Fourier transform circuits). In the case that the small Fourier transforms are performed with a classical computer, complexity arguments for both methods show the reduction in computational time from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$. The second method is also used for the fast calculation of Wigner and Weyl functions, in quantum systems with large finite dimension of the Hilbert space.

1 Introduction

The fast implementation of large Fourier transforms is very important for many technological applications. Roughly speaking in this paper we express the Fourier transform in a Hilbert space of large dimension, as a combination of many Fourier transforms in Hilbert spaces of small dimension. This is a fast Fourier transform, because performing many 'small' Fourier transforms instead of one 'large' Fourier transform, is computationally beneficial. The 'small' Fourier transforms can be performed in a classical computer or as quantum Fourier transforms in a quantum computer. In the latter case, we will also have an additional well known reduction of the computational time by quantum Fourier transform circuits (e.g., [1, 2]). Our methodology (and the associated reduction of computational time) is applicable to the calculation of other quantities also, like the Wigner and Weyl functions.

Two important approaches are the Cooley-Tukey formalism [3, 4], and the Good formalism [5–7] which is based on the Chinese remainder theorem. There are also many variations of these schemes (reviewed in [8–10]). In this paper we study the implementation of fast Fourier transforms in quantum systems with large dimension of the Hilbert space. We also study the fast calculation of the Wigner and Weyl functions. This is an important application of the physics of quantum systems with finite -dimensional Hilbert space(e.g. [11]).

We consider a finite quantum system $\Sigma(D)$ with variables in $\mathbb{Z}(D)$ (the ring of integers modulo d) where D is an odd integer. This system is described by the D-dimensional Hilbert space $H(D)$. There are well known technical differences between quantum systems with odd dimension D and even dimension D (e.g., [12–14]). In this paper we consider systems with odd dimension D. We discuss the fast implementation of the Fourier transform F in $\Sigma(D)$, using two methods described briefly below.

1.1 First method for the case $D = d^n$ with d an odd integer

The fast implementation of the Fourier transform F in $\Sigma(D)$, is using a sequence of n Fourier transforms (together with some other transforms) in a multipartite system $\Sigma_n(d)$ comprised of n components each of which is described with variables in $\mathbb{Z}(d)$. Positions and momenta in $\Sigma_n(d)$ take values in $[\mathbb{Z}(d)]^n = \mathbb{Z}(d) \times ...\mathbb{Z}(d)$ and the corresponding Hilbert space is $\mathfrak{H}_A = H(d) \otimes ... \otimes H(d)$. The

[a] e-mail: c.lei1@bradford.ac.uk
[b] e-mail: a.vourdas@bradford.ac.uk (corresponding author)

Published online: 08 May 2024

Hilbert spaces $H(D)$ and $\mathfrak{H}_A$ are isomorphic (they have the same dimension), and in this sense $\Sigma(D)$ and $\Sigma_n(d)$ are two different descriptions of the same system. However, Fourier transforms and other phase space methods are different in these two cases [15].

Mathematically, this approach is inspired by the Cooley-Tukey formalism [3] for fast Fourier transforms (see also [8–10]), and is used here in a quantum context. But we note that the most popular Cooley-Tukey algorithm is for $D = 2^n$, whilst in our approach D is a power of an odd number.

A quantum circuit for the implementation of this fast Fourier transform is given in Fig. 1. In some special cases, the various operations can be performed in parallel (parallel computing).

We discuss the complexity of this method and show that the computational time is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$. We also present numerical work that supports this.

A limitation of the method is the fact that the ring $\mathbb{Z}(D)$ (with $D = d^n$) is not isomorphic to the ring $[\mathbb{Z}(d)]^n$. Although there is a bijective map between them, sum and products do not correspond to sums and products (Sect. 2.A). The implications of this are discussed in Sect. 4.C. For example, this method cannot be used in Eq. (63) below, for the fast calculation of the Wigner and Weyl functions.

1.2 Second method for the case $D = d_0 ... d_{n-1}$ with d_0, ..., d_{n-1} odd integers coprime to each other

The fast implementation of the Fourier transform F, is using a multipartite system $\Sigma(d_0, ..., d_{n-1})$ comprised of n components, which are described with variables in $\mathbb{Z}(d_0)$, ..., $\mathbb{Z}(d_{n-1})$. Positions and momenta in $\Sigma(d_0, ..., d_{n-1})$ take values in $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ and the corresponding Hilbert space is $\mathfrak{H}_B = H(d_0) \otimes ... \otimes H(d_{n-1})$. The Hilbert spaces $H(D)$ and $\mathfrak{H}_B$ are isomorphic (they have the same dimension), and in this sense $\Sigma(D)$ and $\Sigma(d_0, ..., d_{n-1})$ are two different descriptions of the same system.

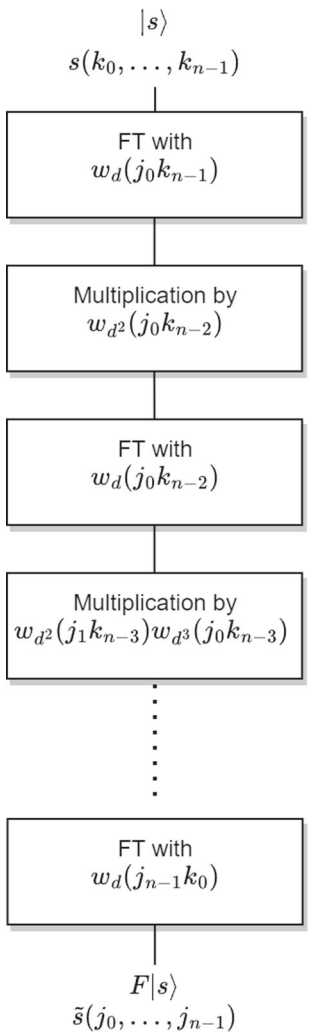

Fig. 1 Circuit for the fast calculation of the Fourier transform F, using Eqs. (31)–(34). Here $D = d^n$

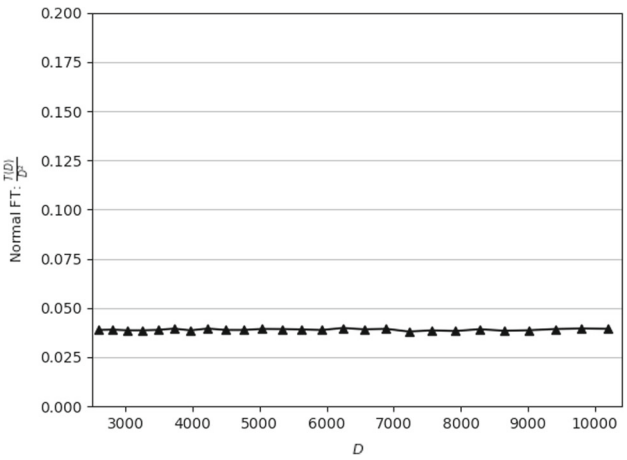

Fig. 2 $\frac{T(D)}{D^2}$ for the 'normal' Fourier transform of a random vector (Eq. (38)), as a function of $D = d^2$ where $d = 51, 53, ..., 101$ is an odd integer. The result is a horizontal line, and this confirms that the computational time for the normal Fourier transform is $\mathcal{O}(D^2)$. The Fourier transform of different random vectors give similar results

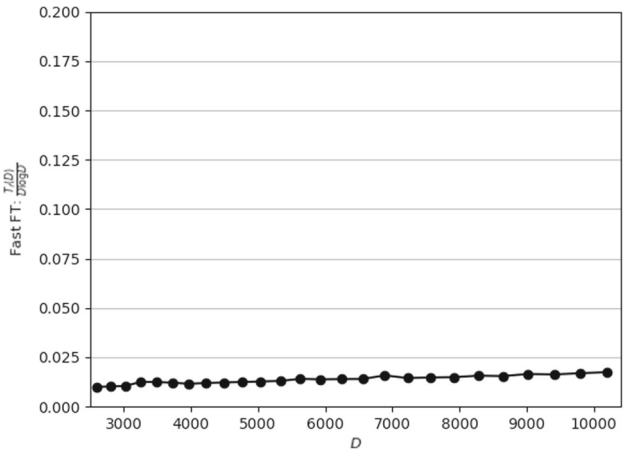

Fig. 3 $\frac{T_f(D)}{D \log D}$ for the fast Fourier transform of a random vector(Eqs. (39) and (40)), as a function of $D = d^2$ where $d = 51, 53, ..., 101$ is an odd integer. The result is a slightly ascending line, and this confirms that a lower bound for the computational time for the fast Fourier transform is approximately $\mathcal{O}(D \log D)$. The Fourier transform of different random vectors give similar results

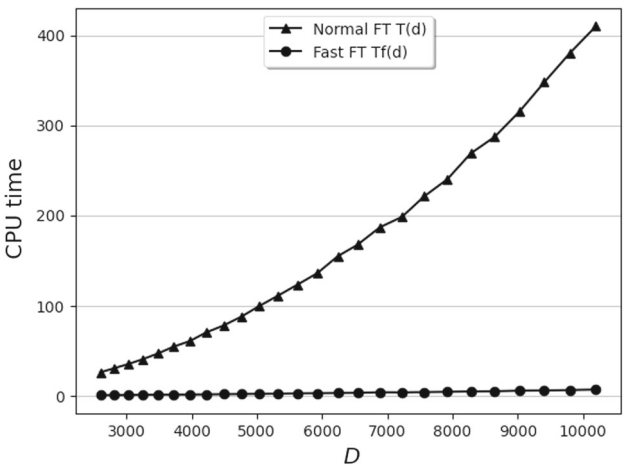

Fig. 4 The CPU times $T(D)$ and $T_f(D)$ for the normal Fourier transform (Eq. (38)) and the fast Fourier transform (Eqs. (39), (40)) correspondingly, of a random vector, as a function of $D = d^2$ where $d = 51, 53, ..., 101$ is an odd integer. It is seen that $T_f(D)$ is much smaller than $T(D)$. Different random vectors give similar results

Mathematically, this approach is inspired by the Good formalism [5–7] for fast Fourier transforms (see also [8–10]), which in turn is based on the Chinese remainder theorem, and is used here in a quantum context. A quantum circuit for the implementation of this fast Fourier transform is given in Fig. 5.

The complexity of the method is discussed, and it is shown that the computational time is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$. This is supported with numerical work.

A strength of the method is the fact that the ring $\mathbb{Z}(D)$ is isomorphic to the ring $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ (Sect. 2.B). Because of this the method is used for the fact calculation of Wigner and Weyl functions in Sect. 6.

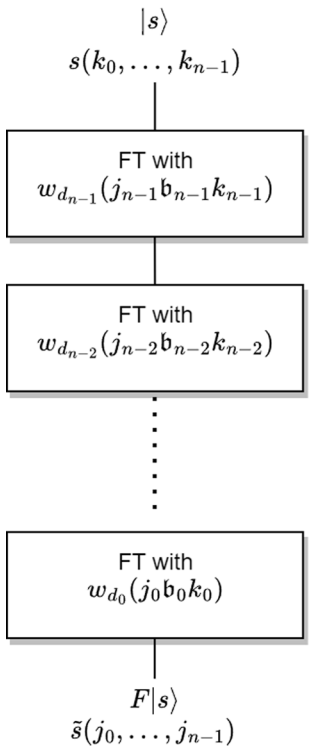

Fig. 5 Circuit for the fast implementation of the Fourier transform$f$ using Eqs. (58)–(60). Here $D = d_0 \ldots d_{n-1}$ with coprime d_0, ..., d_{n-1}. The constants $\mathfrak{b}_\nu$ are defined in Eq. (9)

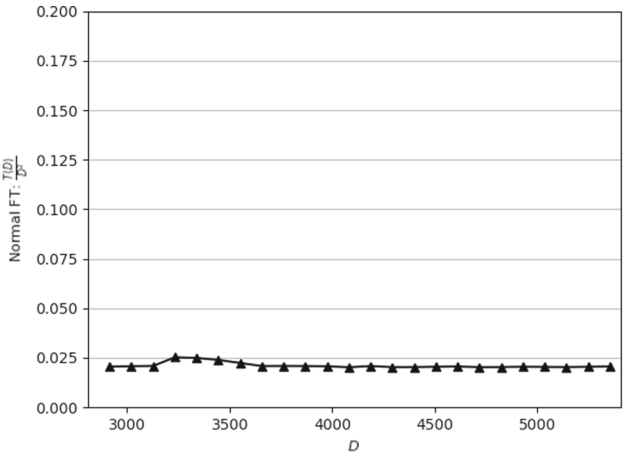

Fig. 6 $\frac{T(D)}{D^2}$ for the 'normal' Fourier transform of a random vector (Eq. (38)), as a function of $D = d_1 d_2$ where $d_1 = 53$ and $d_2 = 55, 57, \ldots, 101$ (the d_1, d_2 are coprime). The result is a horizontal line, and this confirms that the computational time for the normal Fourier transform is $\mathcal{O}(D^2)$. The Fourier transform of different random vectors give similar results

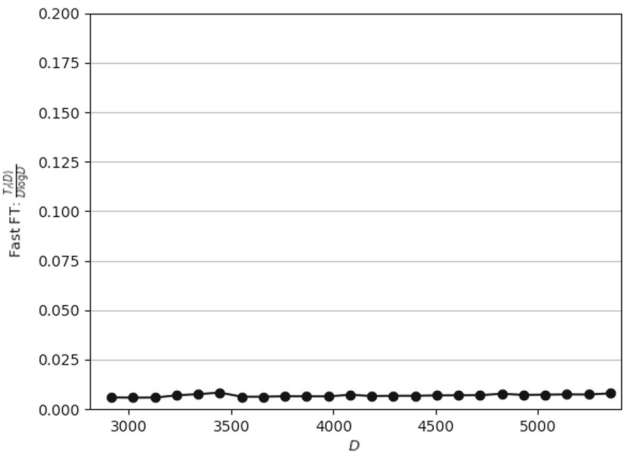

Fig. 7 $\frac{T_f(D)}{D \log D}$ for the fast Fourier transform of a random vector(Eqs. (39) and (40)), as a function of $D = d_1 d_2$ where $d_1 = 53$ and and $d_2 = 55, 57, \ldots, 101$ (the d_1, d_2 are coprime). The result is approximately a horizontal line, and this confirms that the computational time for the fast Fourier transform is approximately $\mathcal{O}(D \log D)$. The Fourier transform of different random vectors give similar results

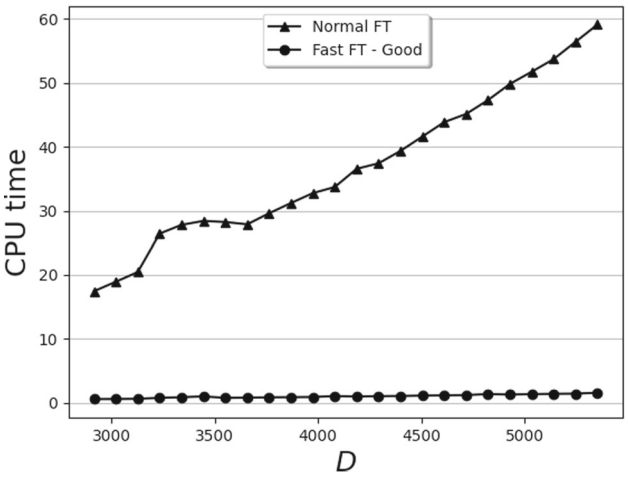

Fig. 8 The CPU times $T(D)$ and $T_f(D)$ for the normal Fourier transform (Eq. (38)) and the fast Fourier transform (Eqs. (39) and (40)) correspondingly, of a random vector, as a function of $D = d_1 d_2$ where $d_1 = 53$ and and $d_2 = 55, 57, ..., 101$ (the d_1, d_2 are coprime). It is seen that $T_f(D)$ is much smaller than $T(D)$. Different random vectors give similar results

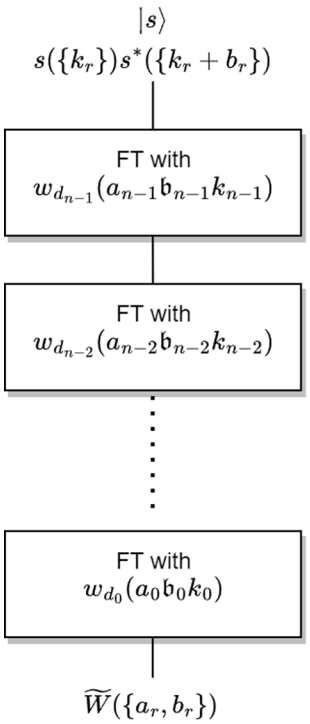

Fig. 9 Circuit for the fast calculation of the Weyl function, using Eqs. (67)–(69). Here $D = d_0...d_{n-1}$ with coprime d_0, ..., d_{n-1}. The constants $\mathfrak{b}_\nu$ are defined in Eq. (9)

1.3 Contents

The work is complementary to the work on quantum Fourier transforms. It reduces a large Fourier transform to many small Fourier transforms, and this reduces the computational time. The small Fourier transforms can be preformed with a classical computer, or (if available) with a quantum computer so that we have the additional (and well known) advantages of quantum Fourier transforms [1, 2].

In Sect. 2 we discuss the number theory related to the two methods. In Sect. 3 we consider a quantum system $\Sigma(D)$ with variables in $\mathbb{Z}(D)$ where D is an odd integer, described by the D-dimensional Hilbert space $H(D)$. In Sect. 4 we present the first method for the case where $D = d^n$. In Sect. 5 we present the second method for the case where $D = d_0...d_{n-1}$ with d_0, ..., d_{n-1} odd integers coprime to each other. In Sect. 6 we use the second method for the fast calculation of the Wigner and Weyl functions. We conclude in Sect. 7 with a discussion of our results.

2 Number theory for the two fast Fourier transforms

2.1 A bijective map between the non-isomorphic rings $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$ when $D = d^n$

$\mathbb{Z}(D)$ is the ring of integers modulo D, where D is an odd integer. We take $D = d^n$ (where d is an odd integer) and consider a bijective map between $[\mathbb{Z}(d)]^n = \mathbb{Z}(d) \times ... \times \mathbb{Z}(d)$ and $\mathbb{Z}(D)$. We use upper case letters for elements in $\mathbb{Z}(D)$, and lower case letters for elements in $\mathbb{Z}(d)$. We also take $j_r \in \mathbb{Z}(d)$ and $J \in \mathbb{Z}(D)$ in the 'periods'

$$\left[-\frac{d-1}{2}, \frac{d-1}{2}\right]; \quad \left[-\frac{D-1}{2}, \frac{D-1}{2}\right], \tag{1}$$

correspondingly.

We introduce the following bijective map between the sets $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$

$$(j_0, ..., j_{n-1}) \leftrightarrow J = j_0 + j_1 d + ... + j_{n-1} d^{n-1}. \tag{2}$$

Given J, we can find the j_0, ..., j_{n-1} as the remainders in the following sequence of divisions:

- We divide J by d and we get $j_1 + j_2 d + ... j_{n-1} d^{n-2}$ and remainder j_0.
- We divide $j_1 + j_2 d + ... j_{n-1} d^{n-2}$ by d and we get $j_2 + j_3 d + ... j_{n-1} d^{n-3}$ and remainder j_1.
- e.t.c.

We note that the $[\mathbb{Z}(d)]^n$ as a ring (with addition and multiplication componentwise), is not isomorphic to the ring $\mathbb{Z}(D)$ because addition and multiplication is different [15]. Indeed

$$(j_0, ..., j_{n-1}) + (k_0, ..., k_{n-1}) = (j_0 + k_0, ..., j_{n-1} + k_{n-1}) \tag{3}$$

does not correspond to $J + K$. The sum in $\mathbb{Z}(D)$ has the 'carry' rule and the r-component might be $j_r + k_r + 1$ rather than $j_r + k_r$. In contrast, there is no 'carry' rule in $[\mathbb{Z}(d)]^n$. Also the multiplication in $\mathbb{Z}(D)$

$$JK = j_0 k_0 + d(j_1 k_0 + k_1 j_0) + ... + d^{n-1}(j_0 k_{n-1} + ... + j_{n-1} k_0), \tag{4}$$

does not correspond to the componentwise multiplication in $[\mathbb{Z}(d)]^n$

$$(j_0, ..., j_{n-1}) \cdot (k_0, ..., k_{n-1}) = (j_0 k_0, ..., j_{n-1} k_{n-1}). \tag{5}$$

Due to the non-isomorphism of the rings $\mathbb{Z}(D)$ and $[\mathbb{Z}(d)]^n$, there is a limitation (see subsection 4.C) of the corresponding fast Fourier transform method in Sect. 4.

We use the notation

$$\omega_r(s) = \exp\left(i\frac{2\pi s}{r}\right). \tag{6}$$

For later use, we use Eq. (4) and we get

$$\omega_D(JK) = \omega_{d^n}(j_0 k_0)\omega_{d^{n-1}}(j_1 k_0 + k_1 j_0)...\omega_d(j_0 k_{n-1} + ... + j_{n-1} k_0). \tag{7}$$

Example 2.1 We consider the bijective map between the sets $[\mathbb{Z}(3)]^2$ and $\mathbb{Z}(9)$:

$$(j_0, j_1) \leftrightarrow J = j_0 + 3j_1; \quad j_\nu = -1, 0, 1; \quad J = -4, ..., 4. \tag{8}$$

Then (1, 1) corresponds to $4 \in \mathbb{Z}(9)$. Addition in $[\mathbb{Z}(3)]^2$ gives $(1, 1) + (1, 1) = (-1, -1)$ which corresponds to $-4 \in \mathbb{Z}(9)$. The corresponding addition in $\mathbb{Z}(9)$ gives $4 + 4 = -1$.

2.2 The isomorphic rings $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ and $\mathbb{Z}(D)$ when $D = d_0...d_{n-1}$ and the d_0, ..., d_{n-1} are coprime

A different method for Fast Fourier transforms is the Good method [5–7] which is based on the Chinese remainder theorem. In a quantum context it has been used in [11, 16].

If d_0, ..., d_{n-1} are coprime, then the ring $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ is isomorphic to $\mathbb{Z}(D)$ where $D = d_0 \times ... \times d_{n-1}$. We first define the integers

$$\mathfrak{a}_\nu = \frac{D}{d_\nu}; \quad \mathfrak{a}_\nu \mathfrak{b}_\nu = 1 (\text{mod } d_\nu) \tag{9}$$

$\mathfrak{b}_\nu$ is the inverse of $\mathfrak{a}_\nu$ within $\mathbb{Z}(d_\nu)$, and it exists because the $\mathfrak{a}_\nu$, d_ν are coprime. We also define the $\mathfrak{c}_\nu = \mathfrak{a}_\nu \mathfrak{b}_\nu$ as an element of $\mathbb{Z}(D)$, which is an integer multiple of d_ν plus one $(Nd_\nu + 1)$.

Lemma 2.2

$$\mathfrak{a}_\nu \mathfrak{a}_\mu = \mathfrak{a}_\nu^2 \delta_{\mu\nu} (mod\ D); \quad \mathfrak{c}_\nu \mathfrak{c}_\mu = \mathfrak{c}_\nu \delta_{\mu\nu} (mod\ D); \quad \mathfrak{a}_\nu \mathfrak{c}_\mu = \mathfrak{a}_\nu \delta_{\mu\nu} (mod\ D). \tag{10}$$

Proof In the first relation, for $\nu \neq \mu$ we get a multiple of D, which is 0(mod D).

In the second relation, we get

$$\begin{aligned} \mathfrak{c}_\nu \mathfrak{c}_\mu &= \mathfrak{a}_\nu \mathfrak{b}_\nu \mathfrak{a}_\mu \mathfrak{b}_\mu = (\mathfrak{a}_\nu \mathfrak{b}_\nu)^2 \delta_{\nu\mu} = \mathfrak{c}_\nu^2 \delta_{\nu\mu} = \mathfrak{c}_\nu (N d_\nu + 1) \delta_{\nu\mu} = \mathfrak{c}_\nu \delta_{\nu\mu} + N \mathfrak{b}_\nu (\mathfrak{a}_\nu d_\nu) \\ &= \mathfrak{c}_\nu \delta_{\nu\mu} + N \mathfrak{b}_\nu D = \mathfrak{c}_\nu \delta_{\nu\mu} \ (\text{mod } D). \end{aligned} \tag{11}$$

In the third relation, we get

$$\begin{aligned} \mathfrak{a}_\nu \mathfrak{c}_\mu =& \mathfrak{a}_\nu \mathfrak{a}_\mu \mathfrak{b}_\mu = \mathfrak{a}_\nu^2 \mathfrak{b}_\nu \delta_{\nu\mu} = \mathfrak{a}_\nu \mathfrak{c}_\nu \delta_{\nu\mu} = \mathfrak{a}_\nu (N d_\nu + 1) \delta_{\nu\mu} = \mathfrak{a}_\nu \delta_{\nu\mu} + N D \\ =& \mathfrak{c}_\nu \delta_{\nu\mu} \ (\text{mod } D). \end{aligned} \tag{12}$$

□

We define a bijective map between $\mathbb{Z}(d_1) \times ... \times \mathbb{Z}(d_n)$ and $\mathbb{Z}(D)$ as follows:

$$(j_0, ..., j_{n-1}) \ \leftrightarrow \ J; \quad j_\nu = J (\text{mod } d_\nu) \in \mathbb{Z}(d_\nu); \quad J = \sum j_\nu \mathfrak{c}_\nu \in \mathbb{Z}(D). \tag{13}$$

The Chinese remainder theorem ensures that this map is bijective. Using Eq. (12), we prove that

$$\begin{aligned} &(j_0 + j_0', ..., j_{n-1} + j_{n-1}') \ \leftrightarrow \ J + J'; \\ &(j_0 j_0', ..., j_{n-1} j_{n-1}') \ \leftrightarrow \ J J'. \end{aligned} \tag{14}$$

and therefore the ring $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ is isomorphic to $\mathbb{Z}(D)$.

We also define a different bijective map

$$(\widehat{j_0}, ..., \widehat{j_{n-1}}) \ \leftrightarrow \ J; \quad \widehat{j_\nu} = J \mathfrak{b}_\nu (\text{mod } d_\nu) \in \mathbb{Z}(d_\nu); \quad J = \sum \widehat{j_\nu} \mathfrak{a}_\nu \in \mathbb{Z}(D). \tag{15}$$

From Eqs. (13) and (15) we find the relationship between j_ν and $\widehat{j_\nu}$:

$$\widehat{j_\nu} = j_\nu \mathfrak{b}_\nu (\text{mod } d_\nu); \quad j_\nu = \widehat{j_\nu} \mathfrak{a}_\nu (\text{mod } d_\nu). \tag{16}$$

Using Eqs. (12), (13) and (15) we prove that

$$J K = \widehat{j_0} k_0 \mathfrak{a}_0 + ... + \widehat{j_{n-1}} k_{n-1} \mathfrak{a}_{n-1}. \tag{17}$$

It then follows the important relation:

$$\begin{aligned} \omega_D (J K) &= \omega_{d_0} (\widehat{j_0} k_0) ... \omega_{d_{n-1}} (\widehat{j_{n-1}} k_{n-1}) \\ &= \omega_{d_0} (j_0 \mathfrak{b}_0 k_0) ... \omega_{d_{n-1}} (j_{n-1} \mathfrak{b}_{n-1} k_{n-1}) \end{aligned} \tag{18}$$

Example 2.3 Let $d_0 = 3$ and $d_1 = 5$. Then $D = 15$ and

$$\begin{aligned} &\mathfrak{a}_0 = 5; \quad \mathfrak{b}_0 = 2; \quad \mathfrak{c}_0 = 10 \\ &\mathfrak{a}_1 = 3; \quad \mathfrak{b}_1 = 2; \quad \mathfrak{c}_1 = 6. \end{aligned} \tag{19}$$

Then

$$J = 10 j_0 + 6 j_1 = 5 \widehat{j_0} + 3 \widehat{j_1} \tag{20}$$

As an example, we take $J = 11$ and we find the corresponding $(j_0,\ j_1) = (2,\ 1)$ and $(\widehat{j_0},\ \widehat{j_1}) = (4,\ 2)$. We confirm Eq. (16):

$$\begin{aligned} &j_0 \mathfrak{b}_0 = 2 \times 2 = 4 = \widehat{j_0}; \quad j_1 \mathfrak{b}_1 = 1 \times 2 = \widehat{j_1} \\ &\widehat{j_0} \mathfrak{a}_0 = 4 \times 5 = 2 (mod\ 3) = j_0; \quad \widehat{j_1} \mathfrak{a}_1 = 2 \times 3 = 1 (mod\ 5) = j_1. \end{aligned} \tag{21}$$

3 A quantum system $\Sigma(D)$ with variables in $\mathbb{Z}(D)$

We consider a quantum system $\Sigma(D)$ with variables in the ring $\mathbb{Z}(D)$, where D is an odd integer. $H(D)$ is the D-dimensional Hilbert space describing this system.

Let $|X;J\rangle$ where $J \in \mathbb{Z}(D)$ be an orthonormal basis in $H(D)$. The X in the notation is not a variable, it simply indicates 'position states'. The finite Fourier transform F is given by [17]

$$F = \frac{1}{\sqrt{D}} \sum_{J,K} \omega_D(JK)|X;J\rangle\langle X;K|; \quad A, J, K \in \mathbb{Z}(D)$$
$$F^4 = \mathbf{1}; \quad FF^\dagger = \mathbf{1}. \tag{22}$$

We act with $F^\dagger$ on position states and get the dual basis

$$|P;J\rangle = F^\dagger|X;J\rangle = \frac{1}{\sqrt{D}} \sum_K \omega_D(-JK)|X;K\rangle. \tag{23}$$

The P in the notation is not a variable, it simply indicates 'momentum states'. A state $|s\rangle$ in $H(D)$ can be written as

$$|s\rangle = \sum s(J)|X;J\rangle = \sum \widetilde{s}(J)|P;J\rangle$$
$$\widetilde{s}(J) = \frac{1}{\sqrt{D}} \sum_K \omega_D(JK)s(K) \tag{24}$$

Below we study the fast implementation of this Fourier transform.

4 The case $D = d^n$

4.1 A multipartite system $\Sigma_n(d)$ with variables in $[\mathbb{Z}(d)]^n$

We consider a multipartite system $\Sigma_n(d)$ comprised of n components each of which is described with variables in $\mathbb{Z}(d)$. Positions and momenta take values in $[\mathbb{Z}(d)]^n$. This system is described with the d^n-dimensional Hilbert space $\mathfrak{H}_A = H(d) \otimes ... \otimes H(d)$. We consider the basis

$$|X;j_0,...,j_{n-1}\rangle = |X;j_0\rangle \otimes ... \otimes |X;j_{n-1}\rangle; \quad j_r \in \mathbb{Z}(d). \tag{25}$$

An arbitrary state is written as

$$|s\rangle = \sum s(j_0,...,j_{n-1})|X;j_0,...,j_{n-1}\rangle; \quad \sum |s(j_0,...,j_{n-1})|^2 = 1. \tag{26}$$

Fourier transforms are defined as:

$$\mathfrak{F}_A = \mathcal{F} \otimes ... \otimes \mathcal{F}; \quad \mathcal{F} = \frac{1}{\sqrt{d}} \sum_{j,k} \omega_d(jk)|X;j\rangle\langle X;k|$$
$$\mathfrak{F}_A^4 = \mathbf{1}; \quad \mathfrak{F}_A\mathfrak{F}_A^\dagger = \mathbf{1}; \quad j,k \in \mathbb{Z}(d). \tag{27}$$

We assume that $D = d^n$ and compare and contrast the systems $\Sigma_n(d)$ and $\Sigma(D)$. Then $\mathfrak{H}_A$ is isomorphic to $H(D)$ (because they both have the same dimension), and therefore $\Sigma(D)$ and $\Sigma_n(d)$ are two different descriptions of the same system. However as discussed in ref [15], Fourier transforms and phase space methods (displacement operators, Wigner and Weyl functions, etc) are different in $\Sigma(D)$ and $\Sigma_n(d)$ (F is different from $\mathfrak{F}_A$). This is because in these techniques we use addition and multiplication and as we explained above, the rings $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$ are not isomorphic to each other. Furthermore (proposition 4.4 in ref [15]), depending on the d, n, the Fourier transforms in $\Sigma_n(d)$ and $\Sigma(D)$ are unitarily inequivalent or unitarily equivalent.

Below we explain how the equivalent of the Fourier transform F in $\Sigma(D)$, is a sequence of transformations in $\Sigma_n(d)$ that involve Fourier transforms in the various components together with some other transformations. The latter is a fast Fourier transform in a quantum context.

4.2 Fast Fourier transform F in $\Sigma(D)$ as a sequence of transformations in $\Sigma_n(d)$ with $D = d^n$

We use the following dual notation for functions and states in $\Sigma(D)$, based on the bijective map in Eq. (2):

$$s(K) = s(k_0,...,k_{n-1}). \tag{28}$$

The matrix elements of the Fourier transform F in $\Sigma(D)$ (Eq. (22)) as:

$$\begin{aligned} &F(j_0, ..., j_{n-1}|k_0, ..., k_{n-1}) = \langle j_0, ..., j_{n-1}|F|k_0, ..., k_{n-1}\rangle \\ &= \frac{1}{\sqrt{d^n}}\omega_{d^n}[j_0k_0 + d(j_1k_0 + k_1j_0) + ... + d^{n-1}(j_0k_{n-1} + ... + j_{n-1}k_0)] \\ &= \mathcal{A}(k_{n-1})\mathcal{A}(k_{n-2})\mathcal{A}(k_{n-3})...\mathcal{A}(k_0) \end{aligned} \tag{29}$$

where

$$\begin{aligned} \mathcal{A}(k_{n-1}) &= \frac{1}{\sqrt{d}}\omega_d(j_0k_{n-1}) \\ \mathcal{A}(k_{n-2}) &= \frac{1}{\sqrt{d}}\omega_d(j_1k_{n-2})\omega_{d^2}(j_0k_{n-2}) \\ \mathcal{A}(k_{n-3}) &= \frac{1}{\sqrt{d}}\omega_d(j_2k_{n-3})\omega_{d^2}(j_1k_{n-3})\omega_{d^3}(j_0k_{n-3}) \\ &..... \\ \mathcal{A}(k_0) &= \frac{1}{\sqrt{d}}\omega_d(j_{n-1}k_0)\omega_{d^2}(j_{n-2}k_0)....\omega_{d^n}(j_0k_0) \end{aligned} \tag{30}$$

Using this we implement the Fourier transform in Eq. (24), as a sequence of transforms in the system $\Sigma_n(d)$. It involves the following steps (shown also in the quantum circuit in Fig. 1):

- A Fourier transform of $s(K) = s(k_0, ..., k_{n-1})$ with $\omega_d(j_0k_{n-1})$ that involves summation over k_{n-1}:

$$s_1(j_0|k_0, .., k_{n-2}) = \frac{1}{\sqrt{d}}\sum_{k_{n-1}}\omega_d(j_0k_{n-1})s(k_0, ..., k_{n-1}) \tag{31}$$

- We first multiply $s_1(j_0|k_0, .., k_{n-2})$ by $\omega_{d^2}(j_0k_{n-2})$ (this is the analogue of 'twiddle factors' [18] in the present context). Then we perform a Fourier transform of $\omega_{d^2}(j_0k_{n-2})s_1(j_0|k_0, .., k_{n-2})$ with $\omega_d(j_1k_{n-2})$, that involves summation over k_{n-2}:

$$s_2(j_0, j_1|k_0, .., k_{n-3}) = \frac{1}{\sqrt{d}}\sum_{k_{n-2}}\omega_d(j_1k_{n-2})[\omega_{d^2}(j_0k_{n-2})s_1(j_0|k_0, .., k_{n-2})] \tag{32}$$

- We first multiply $s_2(j_0, j_1|k_0, .., k_{n-2})$ by $\omega_{d^2}(j_1k_{n-3})\omega_{d^3}(j_0k_{n-3})$. Then we perform a Fourier transform of $\omega_{d^2}(j_1k_{n-3})\omega_{d^3}(j_0k_{n-3})s_2(j_0, j_1|k_0, .., k_{n-2})$ with $\omega_d(j_2k_{n-3})$, that involves summation over k_{n-3}:

$$s_3(j_0, j_1, j_2|k_0, .., k_{n-4}) = \frac{1}{\sqrt{d}}\sum_{k_{n-3}}\omega_d(j_2k_{n-3})[\omega_{d^2}(j_1k_{n-3})\omega_{d^3}(j_0k_{n-3})s_2(j_0, j_1|k_0, .., k_{n-2})] \tag{33}$$

- We continue in this way and the n-step is a Fourier transform with $\omega_d(j_{n-1}k_0)$ that involves summation over k_0:

$$\widetilde{s}(J) = \widetilde{s}(j_0, ..., j_{n-1}) = \frac{1}{\sqrt{d}}\sum_{k_0}\omega_d(j_{n-1}k_0)[\omega_{d^2}(j_{n-2}k_0)....\omega_{d^n}(j_0k_0)s_{n-1}(j_0, ..., j_{n-2}|k_0)] \tag{34}$$

We note that:

- $$\sum|s(k_0, ..., k_{n-1})|^2 = \sum|s_1(j_0|k_0, .., k_{n-2})|^2 = ... = \sum|\widetilde{s}(j_0, ..., j_{n-1})|^2 = 1. \tag{35}$$

- Starting from $s_r(j_0, ..., j_{r-1}|k_0, .., k_{n-r-1})$ with a series of inverse Fourier transforms we get the original wavefunction $s(k_0, ..., k_{n-1})$. For example from $\widetilde{s}(j_0, ..., j_{n-1})$ we go to $s_{n-1}(j_0, ..., j_{n-2}|k_0)]$ as follows:

$$s_{n-1}(j_0, ..., j_{n-2}|k_0)] = [\omega_{d^2}(-j_{n-2}k_0)....\omega_{d^n}(-j_0k_0)]\frac{1}{\sqrt{d}}\sum_{j_0}\omega_d(-j_{n-1}k_0)\widetilde{s}(j_0, ..., j_{n-1}) \tag{36}$$

  In a similar way we go backwards in all above steps. Therefore all the $s_r(j_0, ..., j_{r-1}|k_0, .., k_{n-r-1})$ contain the same information as the original wavefunction $s(k_0, ..., k_{n-1})$.

Example 4.1 For $n = 2$, Eq. (29) becomes

$$F(j_0, j_1|k_0, k_1) = \frac{1}{\sqrt{d^2}}\omega_{d^2}[j_0k_0 + d(j_1k_0 + k_1j_0)]. \tag{37}$$

Acting on a vector $s(K) = s(k_0, k_1)$ we get

$$\widetilde{s}(J) = \widetilde{s}(j_0, j_1) = \frac{1}{\sqrt{d^2}}\sum_{k_0,k_1}\omega_{d^2}[j_0k_0 + d(j_1k_0 + k_1j_0)]s(k_0, k_1). \tag{38}$$

In this case the fast Fourier transform given above becomes

$$\widetilde{s}(J) = \widetilde{s}(j_0, j_1) = \frac{1}{\sqrt{d}} \sum_{k_0} \omega_d(j_1 k_0)[\omega_{d^2}(j_0 k_0) s_1(j_0|k_0)] \tag{39}$$

with

$$s_1(j_0|k_0) = \frac{1}{\sqrt{d}} \sum_{k_1} \omega_d(j_0 k_1) s(k_0, k_1) \tag{40}$$

4.3 Limitation of the method

We have calculated the Fourier transform of the function $s(K) = s(k_0, \ldots, k_{n-1})$. For other functions it is not easy to apply this method. For example in the Weyl function in Eq. (63) below, we want to calculate the Fourier transform of the function $s(K)s^*(B+K)$. Because the rings $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$ (with $D = d^n$) are not isomorphic to each other, if

$$\begin{aligned} (k_0, \ldots, k_{d-1}) &\leftrightarrow K = k_0 + k_1 d + \ldots + k_{n-1} d^{n-1} \\ (b_0, \ldots, b_{d-1}) &\leftrightarrow B = b_0 + b_1 d + \ldots + b_{n-1} d^{n-1}. \end{aligned} \tag{41}$$

the $(k_0 + b_0, \ldots, k_{n-1} + b_{n-1})$ does not correspond to $K + B$. It is then difficult to apply directly the above formalism to Eq. (63) for the fast calculation of the Weyl and Wigner functions.

In general, this fast Fourier transform is not directly applicable to functions which involve various sums and products of the variables. The fact that the rings $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$ are not isomorphic to each other, limits the practical use of the method.

4.4 Parallelism in the special case of factorisable states

We consider the factorisable state

$$s(K) = s(k_0, \ldots, k_{n-1}) = g_0(k_0) g_1(k_1) \ldots g_{n-1}(k_{n-1}); \quad \sum_{k_\nu} |g_\nu(k_\nu)|^2 = 1. \tag{42}$$

In this case

$$\begin{aligned} s_1(j_0|k_0, .., k_{n-2}) &= g_0(k_0) \ldots g_{n-2}(k_{n-2}) \widetilde{g}_{n-1}(j_0) \\ \widetilde{g}_{n-1}(j_0) &= \frac{1}{\sqrt{d}} \sum_{k_{n-1}} \omega_d(j_0 k_{n-1}) g_{n-1}(k_{n-1}) \end{aligned} \tag{43}$$

Also

$$\begin{aligned} s_2(j_0, j_1|k_0, .., k_{n-3}) &= g_0(k_0) \ldots g_{n-3}(k_{n-3}) \widetilde{G}_{n-2}(j_0, j_1) \widetilde{g}_{n-1}(j_0) \\ \widetilde{G}_{n-2}(j_0, j_1) &= \frac{1}{\sqrt{d}} \sum_{k_{n-2}} \omega_d(j_1 k_{n-2})[\omega_{d^2}(j_0 k_{n-2}) g_{n-2}(k_{n-2})] \end{aligned} \tag{44}$$

Also

$$\begin{aligned} s_3(j_0, j_1, j_2|k_0, .., k_{n-4}) &= g_0(k_0) \ldots g_{n-4}(k_{n-4}) \widetilde{G}_{n-3}(j_0, j_1, j_2) \widetilde{G}_{n-2}(j_0, j_1) \widetilde{g}_{n-1}(j_0) \\ \widetilde{G}_{n-3}(j_0, j_1, j_2) &= \frac{1}{\sqrt{d}} \sum_{k_{n-3}} \omega_d(j_2 k_{n-3})[\omega_{d^2}(j_1 k_{n-3}) \omega_{d^3}(j_0 k_{n-3}) g_{n-3}(k_{n-3})] \end{aligned} \tag{45}$$

etc. The last one is

$$\begin{aligned} \widetilde{s}(j_0, \ldots, j_{n-1}) &= \widetilde{G}_0(j_0, \ldots, j_{n-1}) \widetilde{G}_1(j_0, \ldots, j_{n-2}) \ldots \widetilde{G}_{n-2}(j_0, j_1) \widetilde{g}_{n-1}(j_0) \\ \widetilde{G}_0(j_0, \ldots, j_{n-1}) &= \frac{1}{\sqrt{d}} \sum_{k_0} \omega_d(j_{n-1} k_0)[\omega_{d^2}(j_{n-2} k_0) \ldots . \omega_{d^n}(j_0 k_0) g_0(k_0)] \end{aligned} \tag{46}$$

We note that for factorisable functions, we can calculate independently each of the n factors $\widetilde{G}_0(j_0, \ldots, j_{n-1})$, $\widetilde{G}_1(j_0, \ldots, j_{n-2})$, ..., $\widetilde{g}_{n-1}(j_0)$ and multiply them at the end. Therefore this scheme is suitable for parallel computation. The calculation in the previous subsection for general functions, needs to be done sequentially.

Example 4.2 For $n = 2$ we consider the factorisable state

$$s(K) = s(k_0, k_1) = g_0(k_0) g_1(k_1) \tag{47}$$

In this case

$$\widetilde{s}(J) = \widetilde{s}(j_0, j_1) = \widetilde{G}_0(j_0, j_1)\widetilde{g}_1(j_0)$$
$$\widetilde{g}_1(j_0) = \frac{1}{\sqrt{d}} \sum_{k_1} \omega_d(j_0 k_1) g_1(k_1)$$
$$\widetilde{G}_0(j_0, j_1) = \frac{1}{\sqrt{d}} \sum_{k_0} \omega_d(j_1 k_0)[\omega_{d^2}(j_0 k_0) g_0(k_0)] \tag{48}$$

The two factors $\widetilde{G}_0(j_0,\ j_1)$ and $\widetilde{g}_1(j_0)$ can be calculated in parallel.

Remark 4.3 The 'parallel formalism' of this section is limited to special cases where we know that the factorisation in Eq.(42) holds. Given $s(K) = s(k_0,\ ...k_{n-1})$, we give a necessary (but not sufficient) condition for the factorisation to hold.

We define the

$$|g(k_v)|^2 = \sum_{\neq k_v} |s(k_0, ...k_{n-1})|^2. \tag{49}$$

Here we have a summation over all indices, except one. A necessary (but not sufficient) condition for Eq. (42) to hold, is that

$$|s(k_0, ...k_{n-1})| = |g(k_0)|...|g(k_{n-1})| \tag{50}$$

4.5 Time complexity of the Fourier transform: counting the number of multiplications

The estimate of the computational time is usually based on the number of multiplications, because they require more computational time than additions. It is easily seen that the number of multiplications for 'normal' Fourier transform is $\mathcal{O}(D^2)$ (it is a multiplication of a $D \times D$ matrix with a D-dimensional vector). For the fast Fourier transform it is known that a lower bound for the computational time is $\mathcal{O}(D \log D)$, and we now give an approximate estimate for this.

In the fast transform in Sect. 4.2, the first step in Eq. (31) is a Fourier transform in a d-dimensional space and it requires d^2 multiplications. This needs to be repeated for all values of the $n-1$ variables k_0, .., k_{n-2} which take d values each, therefore the number of multiplications is $d^2 d^{n-1} = Dd$. The second step in Eq. (32) involves another Dd multiplications (plus some extra multiplications which we ignore because we are interested in a lower limit). In this way we find that a lower bound for the number of multiplications is

$$Dnd \geq Dn \log d = D \log D. \tag{51}$$

Many authors pointed out that this is a lower bound and that 'real' numerical fast Fourier transforms take a bit more time than that.

We consider a Hilbert space $H(D)$ with $D = d^2$, where d that takes all the odd values 51, ..., 101. Using a random vector $s(K) = s(k_0,\ k_1)$ (produced by qiskit [19]), we calculated $\widetilde{s}(J) = \widetilde{s}(j_0,\ j_1)$ using both Eq. (38) (that involves the multiplication of a $D \times D$ matrix times a D-dimensional vector) and also the fast Fourier transform in Eqs(39), (40). We call $T(D)$ the computational time for the calculation of all components $\widetilde{s}(j_0,\ j_1)$ with the 'normal' Fourier transform in Eq.(38), and $T_f(D)$ the computational time for the calculation with the fast Fourier transform in Eqs(39), (40). In Figs. 2 and 3 we plot

$$\frac{T(D)}{D^2};\quad \frac{T_f(D)}{D \log D};\quad D = d^2. \tag{52}$$

The result for $\frac{T(D)}{D^2}$ in Fig. 2 is a horizontal line and this confirms that the computational time for the normal Fourier transform is $\mathcal{O}(D^2)$.

The result for $\frac{T_f(D)}{D \log D}$ in Fig. 3 is a slightly ascending line and this confirms that a good lower bound for the computational time of the fast Fourier transform is approximately $\mathcal{O}(D \log D)$.

In Fig. 4 we compare $T(D)$ with $T_f(D)$. It is seen that $T_f(D)$ is much smaller than $T(D)$. We checked that the Fourier transform of different random vectors give similar results.

5 The case $D = d_0 ... d_{n-1}$ with coprime d_0, ..., d_{n-1}

5.1 A multipartite system $\Sigma(d_0,\ ...,\ d_{n-1})$ with variables in $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$.

In this section $D = d_0 ... d_{n-1}$ with d_0, ..., d_{n-1} odd integers coprime to each other. We consider a multipartite system $\Sigma(d_0,\ ...,\ d_{n-1})$ comprised of n components, which are described with variables in $\mathbb{Z}(d_0)$, ..., $\mathbb{Z}(d_{n-1})$. Positions and momenta in the multipartite system take values in $\mathbb{Z}(d_0) \times ... \times \mathbb{Z}(d_{n-1})$ and the corresponding Hilbert space is $\mathfrak{H}_B = H(d_0) \otimes ... \otimes H(d_{n-1})$. The Hilbert

spaces $H(D)$ and $\mathfrak{H}_B$ are isomorphic (they have the same dimension), and therefore $\Sigma(D)$ and $\Sigma(d_0, ..., d_{n-1})$ are two different descriptions of the same system.

We consider the basis

$$|X; j_0, ..., j_{n-1}\rangle = |X; j_0\rangle \otimes ... \otimes |X; j_{n-1}\rangle; \quad j_\nu \in \mathbb{Z}(d_\nu). \tag{53}$$

An arbitrary state is written as

$$|s\rangle = \sum s(j_0, ..., j_{n-1})|X; j_0, ..., j_{n-1}\rangle; \quad \sum |s(j_0, ..., j_{n-1})|^2 = 1. \tag{54}$$

Fourier transforms in $\Sigma(d_0, ..., d_{n-1})$ are defined as:

$$\mathfrak{F}_B = \mathcal{F}_0 \otimes ... \otimes \mathcal{F}_{n-1}; \quad \mathcal{F}_\nu = \frac{1}{\sqrt{d}} \sum_{j_\nu, k_\nu} \omega_{d_\nu}(j_\nu k_\nu)|X; j_\nu\rangle\langle X; k_\nu|$$
$$\mathfrak{F}_B^4 = \mathbf{1}; \quad \mathfrak{F}_B \mathfrak{F}_B^\dagger = \mathbf{1}; \quad j_\nu, k_\nu \in \mathbb{Z}(d_\nu). \tag{55}$$

Clearly F is very different from $\mathfrak{F}_B$.

5.2 Fast Fourier transform F in $\Sigma(D)$ as a sequence of transformations in $\Sigma(d_0, ..., d_{n-1})$ with $D = d_0...d_{n-1}$

We use the following dual notation for all functions and states in $\Sigma(D)$, based on the bijective map in Eq. (13):

$$s(K) = s(k_0, ..., k_{n-1}); \quad k_\nu \in \mathbb{Z}(d_\nu). \tag{56}$$

Using Eq. (18) we express the matrix elements of the Fourier transform F in $\Sigma(D)$ (Eq. (22)) as:

$$F(j_0, ..., j_{n-1}|k_0, ..., k_{n-1}) = \langle j_0, ..., j_{n-1}|F|k_0, ..., k_{n-1}\rangle$$
$$= \left[\frac{1}{\sqrt{d_0}}\omega_{d_0}(j_0\mathfrak{b}_0 k_0)\right]...\left[\frac{1}{\sqrt{d_{n-1}}}\omega_{d_{n-1}}(j_{n-1}\mathfrak{b}_{n-1}k_{n-1})\right] \tag{57}$$

The constants $\mathfrak{b}_\nu$ have been defined in Eq. (9). Using this we implement the Fourier transform in Eq. (24), as a sequence of transforms in the system $\Sigma(d_0, ..., d_{n-1})$. It involves the following steps (shown also in the quantum circuit in Fig. 5):

- A Fourier transform of $s(K) = s(k_0, ..., k_{n-1})$ with $\omega_{d_{n-1}}(j_{n-1}\mathfrak{b}_{n-1}k_{n-1})$ (we note here the constant $\mathfrak{b}_{n-1}$) and summation over k_{n-1}:

$$s_1(j_{n-1}|k_0, .., k_{n-2}) = \frac{1}{\sqrt{d_{n-1}}} \sum_{k_{n-1}} \omega_{d_{n-1}}(j_{n-1}\mathfrak{b}_{n-1}k_{n-1})s(k_0, ..., k_{n-1}) \tag{58}$$

- A Fourier transform of $s_1(j_{n-1}|k_0, ..., k_{n-2})$ with $\omega_{d_{n-2}}(j_{n-2}\mathfrak{b}_{n-2}k_{n-2})$ (we note here the constant $\mathfrak{b}_{n-2}$) and summation over k_{n-2}:

$$s_2(j_{n-2}, j_{n-1}|k_0, .., k_{n-3}) = \frac{1}{\sqrt{d_{n-2}}} \sum_{k_{n-2}} \omega_{d_{n-2}}(j_{n-2}\mathfrak{b}_{n-2}k_{n-2})s_1(j_{n-1}|k_0, .., k_{n-2}), \tag{59}$$

- etc. The last step is

$$\widetilde{s}(J) = \widetilde{s}(j_0, ..., j_{n-1}) = \frac{1}{\sqrt{d_0}} \sum_{k_0} \omega_{d_0}(j_0\mathfrak{b}_0 k_0)s_{n-1}(j_1, ..., j_{n-1}|k_0) \tag{60}$$

Similarly to the previous method, for factorisable functions these n steps can be done in parallel. But for general functions, they need to be done sequentially.

5.3 Time complexity of the Fourier transform: counting the number of multiplications

We first give an approximate estimate that a lower bound for the computational time in the present scheme, is $\mathcal{O}(D \log D)$.

In the fast transform in Sect.5.2, the first step in Eq. (58) is a Fourier transform in a d_{n-1}-dimensional space and it requires d_{n-1}^2 multiplications. This needs to be repeated for all values of the $n-1$ variables $k_0, .., k_{n-2}$, therefore the number of multiplications is $d_1...d_{n-2}d_{n-1}^2 = Dd_{n-1}$. The second step in Eq. (59) involves another Dd_{n-2} multiplications. In this way we find that a lower bound for the number of multiplications is

$$D(d_0 + ... + d_{n-1}) \geq D(\log d_0 + ... + \log d_{n-1}) = D \log D. \tag{61}$$

We consider Hilbert spaces $H(d_1 d_2)$ where $d_1 = 53$ and d_2 takes the odd values 55, 57, ..., 101. Since 53 is a prime number the d_1, d_2 are coprime. As in Sect. 4.5 we used a random vector $s(K) = s(k_0, k_1)$ (produced by qiskit [19]), we calculated $\widetilde{s}(J) = \widetilde{s}(j_0, j_1)$. In Figs. 6 and 7 we plot

$$\frac{T(D)}{D^2}; \quad \frac{T_f(D)}{D \log D}; \quad D = d_1 d_2. \tag{62}$$

The result for $\frac{T(D)}{D^2}$ in Fig. 6 is a horizontal line and this confirms that the computational time for the normal Fourier transform is $\mathcal{O}(D^2)$. The result for $\frac{T_f(D)}{D\log D}$ in Fig. 7 is also a horizontal line and this confirms that a good lower bound for the computational time of the fast Fourier transform is approximately $\mathcal{O}(D\log D)$.

In Fig. 8 we compare $T(D)$ with $T_f(D)$. It is seen that $T_f(D)$ is much smaller than $T(D)$. We checked that the Fourier transform of different random vectors give similar results.

6 Fast Wigner and Weyl functions using the second method

Phase space methods for the system $\Sigma(D)$ (Wigner and Weyl functions, etc) rely heavily on Fourier transforms. Therefore fast Fourier transforms can be used for the fast calculation of various quantities within the phase space formalism.

As an example, we consider the Weyl function $\widetilde{W}(A,\ B)$ and the Wigner function $W(A, B)$ for the state $|s\rangle = \sum_K s(K)|X; K\rangle$ of the system $\Sigma(D)$. They are given by the following Fourier transforms(e.g., [20]):

$$\widetilde{W}(A,B) = \omega_D(2^{-1}AB)\sum_K \omega_D(AK)s(K)s^*(B+K); \quad A,B\in\mathbb{Z}(D)$$
$$W(A,B) = \omega_D(2AB)\sum_K \omega_D(-2AK)s(K)s^*(2B-K) \tag{63}$$

The $2^{-1} = \frac{D+1}{2}(\text{mod } D)$ for odd D.

We explained in Sect. 4.3 that the first method for fast Fourier transforms (in the case $D = d^n$) is not directly applicable to Eq. (63), for the fast calculation of these functions. This is related to the fact that the rings $[\mathbb{Z}(d)]^n$ and $\mathbb{Z}(D)$ (with $D = d^n$) are not isomorphic to each other.

The second method for fast Fourier transforms (in the case $D = d_0...d_{n-1}$ with coprime $d_0, ..., d_{n-1}$) is directly applicable in the fast calculation of the Weyl and Wigner functions. We present in detail the fast Weyl function. We use the bijective map in Eq. (13), and express K, B as

$$\begin{aligned} K &\leftrightarrow (k_0,...,k_{n-1}); \quad k_\nu\in\mathbb{Z}(d_\nu)\\ B &\leftrightarrow (b_0,...,b_{n-1}); \quad b_\nu\in\mathbb{Z}(d_\nu)\\ A &\leftrightarrow (a_0,...,a_{n-1}); \quad a_\nu\in\mathbb{Z}(d_\nu)\end{aligned} \tag{64}$$

The rings $\mathbb{Z}(D)$ and $\mathbb{Z}(d_0)\times ...\times\mathbb{Z}(d_{n-1})$ are isomorphic and therefore

$$K+B\leftrightarrow(k_0+b_0,...,k_{n-1}+b_{n-1}). \tag{65}$$

Consequently

$$s(K)s^*(B+K) = s(k_0,...,k_{n-1})s^*(k_0+b_0,...,k_{n-1}+b_{n-1}). \tag{66}$$

We now give briefly the basic steps for the fast Weyl function (shown also in the quantum circuit in Fig 9).

- A Fourier transform of $s(\{k_r\})s^*(\{k_r+b_r\})$ with $\omega_{d_{n-1}}(a_{n-1}\mathfrak{b}_{n-1}k_{n-1})$ (we note here the constant $\mathfrak{b}_{n-1}$) and summation over k_{n-1}:
$$\widetilde{W}_1(a_{n-1}|k_0,..,k_{n-2}|\{b_r\}) = \sum_{k_{n-1}}\omega_{d_{n-1}}(a_{n-1}\mathfrak{b}_{n-1}k_{n-1})s(\{k_r\})s^*(\{k_r+b_r\}). \tag{67}$$
- A Fourier transform of $\widetilde{W}_1(a_{n-1}|k_0,\ ..,\ k_{n-2}|\{b_r\})$ with $\omega_{d_{n-2}}(a_{n-2}\mathfrak{b}_{n-2}k_{n-2})$ (we note here the constant $\mathfrak{b}_{n-2}$) and summation over k_{n-2}:
$$\widetilde{W}_2(a_{n-2},a_{n-1}|k_0,..,k_{n-3}|\{b_r\}) = \sum_{k_{n-2}}\omega_{d_{n-2}}(a_{n-2}\mathfrak{b}_{n-2}k_{n-2})\widetilde{W}_1(a_{n-1}|k_0,..,k_{n-2}|\{b_r\}), \tag{68}$$
- etc. The last step is
$$\widetilde{W}(A,B) = \widetilde{W}(\{a_r,b_r\}) = \omega_D(2^{-1}AB)\sum_{k_0}\omega_{d_0}(a_0\mathfrak{b}_0k_0)\widetilde{W}_{n-1}(a_1,...,a_{n-1}|k_0|\{b_r\}). \tag{69}$$

  Similarly to the previous methods, for factorisable functions $s(k_0, ..., k_{n-1})$ these n steps can be done in parallel. But for general functions, they need to be done sequentially.

Analogous algorithm can be given for the Wigner function.

We note that the Fourier transform requires $\mathcal{O}(D^2)$ multiplications, but it needs to be performed for all values of A, B. Therefore the complexity of the calculation of the Wigner or Weyl function is $\mathcal{O}(D^4)$, and with the fast Fourier transforms discussed above it is reduced to $\mathcal{O}(D^3\log D)$.

As an example we consider the case $D = 21 \times 23 = 3 \times 7 \times 23$ and calculated the Weyl function of a random vector (produced by qiskit [19]) with the normal Fourier transform and with the fast method given above. We found numerically that the ratio of the corresponding computational times is $T/T_f = 14.7$ (with the $D = 21 \times 23$ factorisation), and $T/T_f = 17.6$ (with the $D = 3 \times 7 \times 23$ factorisation).

7 Discussion

We have presented a fast implementation of the Fourier transform F in a large quantum system. This replaces the large Fourier transform with many small Fourier transforms. The small Fourier transforms can be performed classically or (if available) in a quantum computer in which case we have the well known additional advantages of quantum Fourier transforms. We used two methods.

The first method is for the case $D = d^n$ with d an odd integer. This is based on the bijective map between the sets $\mathbb{Z}(D)$ and $[\mathbb{Z}(d)]^n$ in Eq. (2). The algorithm is described in Eqs. (31)–(34) and the relevant quantum circuit is shown in Fig. 1.

The complexity (based on the number of multiplications) of the normal Fourier transform is $\mathcal{O}(D^2)$ and of the fast Fourier transform $\mathcal{O}(D \log D)$. This has been supported with numerical work shown in Figs 2 and 3. As expected the fast Fourier transform is much faster than the normal Fourier transform (Fig. 3). A limitation of the method is the fact that the ring $\mathbb{Z}(D)$ (with $D = d^n$) is not isomorphic to the ring $[\mathbb{Z}(d)]^n$. Consequently, this method cannot be used with Eq. (63) for the fast calculation of the Wigner and Weyl functions.

The second method is for the case $D = d_0 ... d_{n-1}$ with d_0, ..., d_{n-1} odd integers coprime to each other. This is based on the bijective map between the rings $\mathbb{Z}(D)$ and $\mathbb{Z}(d_0)$, ..., $\mathbb{Z}(d_{n-1})$ in Eq. (13). These two rings are isomorphic. The algorithm is described in Eqs. (58)–(60) and the relevant quantum circuit is shown in Fig. 5. Numerical work shown in Figs. 6 and 7 confirm that the complexity of the normal Fourier transform is $\mathcal{O}(D^2)$ and of the fast Fourier transform $\mathcal{O}(D \log D)$. Figure 8 shows that the fast Fourier transform requires much less computational time than the Normal Fourier Transform.

This second method can be used with Eq. (63) for the fast calculation of the Wigner and Weyl functions. The algorithm for the Weyl function is given in Eqs. (67)–(69) and the relevant quantum circuit is shown in Fig. 9.

Data Availability Statement No data were used in this paper.

Declarations

Conflict of interest We have no Conflict of interest to disclose.

References

1. M.A. Nielsen, I.L. Chuang, *Quantum computation and quantum information* (Cambridge University Press, Cambridge, 2000)
2. A. Pavlides, E. Floratos, Quantum Fourier transform based quantum arithmetic with qudits. Phys. Rev. A **103**, 032417 (2021)
3. J.W. Cooley, J.W. Turkey, An algorithm for the machine computation of complex Fourier series. Math. Comput. **19**, 297 (1965)
4. J.W. Cooley, P.A.W. Lewis, P.D. Welch, Historical notes on the fast Fourier transform. IEEE Trans. Audio Electroacoust. **15**, 76 (1967)
5. I.J. Good, The interaction algorithm and practical Fourier analysis. J. R. Stat. Soc. B **20**, 361 (1958)
6. L.H. Thomas, *Using a computer to solve problems in physics in Applications of digital computers* (Gina, Boston, 1963)
7. I.J. Good, The relationship between two fast Fourier transforms. IEEE Trans. Comput. **100**, 310 (1971)
8. J.H. McClellan, C.M. Rader, *Number theory in digital signal processing* (Prentice Hall, New Jersey, 1979)
9. R.E. Blahut, *Fast algorithms for digital signal processing* (Addison-Wesley, Reading Mass, 1985)
10. D.F. Elliott, K.R. Rao, *Fast transforms* (Academic Press, London, 1982)
11. A. Vourdas, *Finite and profinite quantum systems* (Springer, Berlin, 2017)
12. M. Horibe, A. Takami, T. Hashimoto, A. Hayashi, Existence of the Wigner function with correct marginal distributions along tilted lines on a lattice. Phys. Rev. A **65**, 032105 (2002)
13. T. Durt, About mutually unbiased bases in even and odd prime power dimensions. J. Phys. A **38**, 5267 (2005)
14. J. Zak, Doubling feature of the Wigner function: finite phase space. J. Phys. A **44**, 345305 (2011)
15. C. Lei, A. Vourdas, Unitarily inequivalent local and global Fourier transforms in multipartite quantum systems. Quantum Inf. Proc. **22**, 78 (2023)
16. A. Vourdas, Factorisation in finite quantum systems. J. Phys. A **36**, 5645 (2003)
17. A. Terras, *Fourier analysis on finite groups and applications* (Cambridge University Press, Cambridge, 1999)
18. W.M. Gentleman, G. Sande, Fast Fourier transforms for fun and profit. AFIPS Proc. **29**, 563 (1966)
19. M.D. Sajid Anis et al. Qiskit: An open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505https://qiskit.org/ (2021)
20. A. Vourdas, Quantum systems with finite Hilbert space. Rep. Prog. Phys. **67**, 267 (2004)