Efficient Calculation of the Two-Dimensional Wigner Potential

P. Ellinghaus, M. Nedjalkov, and S. Selberherr
Institute for Microelectronics, TU Wien, Gußhausstraße 27–29/E360, 1040 Wien, Austria
E-mail: {ellinghaus | nedjalkov | selberherr}@iue.tuwien.ac.at

I. Introduction

The solution of the two-dimensional (2D) Wigner equation has become numerically feasible in recent times, using the Monte Carlo method [1] fortified with the notion of signed particles [2]. The calculation of the Wigner potential (WP) in these 2D simulations consumes a considerable part of the computation time. A reduction of the latter is therefore very desirable, in particular, if self-consistent solutions are pursued, where the WP must be recalculated many times. An algorithm is introduced here – named box discrete Fourier transform (BDFT) – that reduces the computational effort roughly by a factor of five.

The semi-discrete WP is defined as

$$V_W\left(\mathbf{r}, \mathbf{P}\Delta\mathbf{k}\right) \equiv \frac{1}{i\hbar\mathbf{L}} \int_{-\mathbf{L}/2}^{\mathbf{L}/2} d\mathbf{s}\, e^{-i2\mathbf{P}\Delta\mathbf{k}\cdot\mathbf{s}} \delta V \tag{1}$$

$$\delta V\left(\mathbf{s}; \mathbf{r}\right) \equiv V\left(\mathbf{r}+\mathbf{s}\right) - V\left(\mathbf{r}-\mathbf{s}\right), \tag{2}$$

where $\mathbf{s}$ is bounded by a finite coherence length, $\mathbf{L}$. The momentum vector $\mathbf{P}\Delta\mathbf{k}$ is discretized in steps of $\Delta\mathbf{k} = \frac{\pi}{\mathbf{L}}$. The length and position vectors are discretized and defined as

$$\begin{aligned} \mathbf{r} &\equiv (x, y) \\ \mathbf{s} &\equiv (m\Delta s, n\Delta s) \\ \mathbf{L} &\equiv (M\Delta s, N\Delta s) \\ \mathbf{P}\Delta\mathbf{k} &\equiv \left(p\frac{\pi}{M\Delta s}, q\frac{\pi}{N\Delta s}\right), \end{aligned} \tag{3}$$

in the following. This yields the 2D computational domain, as depicted in Fig. 1, for which the fully discretized WP,

$$\begin{aligned} V_W\left(x, y, p, q\right) = \frac{1}{i\hbar MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e^{-i2\left(pm\frac{\pi}{M} + qn\frac{\pi}{N}\right)} \\ \delta V\left(x \pm \left(m - \frac{M}{2}\right)\Delta s, y \pm \left(n - \frac{N}{2}\right)\Delta s\right), \end{aligned} \tag{4}$$

must be calculated at each node of the mesh.

Eq. (4) is akin to a 2D DFT of (2), conventionally calculated using a row-column decomposition scheme, which entails the successive application of a one-dimensional (1D) DFT algorithm: With reference to Fig. 2, consider a $M' \times N'$ matrix of values representing the calculated potential differences, as per (2). First the 1D DFT of each row of values is calculated, which yields a $M' \times N'$ matrix of Fourier coefficients. Thereafter, the 1D DFT of each column of the latter matrix is calculated, the result of which corresponds to the 2D DFT.

The fast Fourier transform (FFT) algorithm has a computational complexity, for a problem size N, of $\mathcal{O}(N \log_2 N)$ and presents the *de facto* standard algorithm for calculating 1D DFTs, thanks to flexible, highly optimized implementations, which are freely available in libraries, like FFTW3 [3]. Algorithms which directly calculate multi-dimensional DFTs exist, e.g. [4], [5], and have a reduced computational complexity, which should theoretically result in superior computational performance. However, these advantages are most often completely eroded in practical implementations, which can be attributed to the fact that the cache complexity of algorithms implemented on modern hardware architectures plays an equally important role as the computational complexity. Moreover, these multi-dimensional algorithms require extensive 'tailoring', e.g. to the dimension or transform size, making general purpose implementations difficult, thereby hampering a wide-spread adoption of such algorithms.

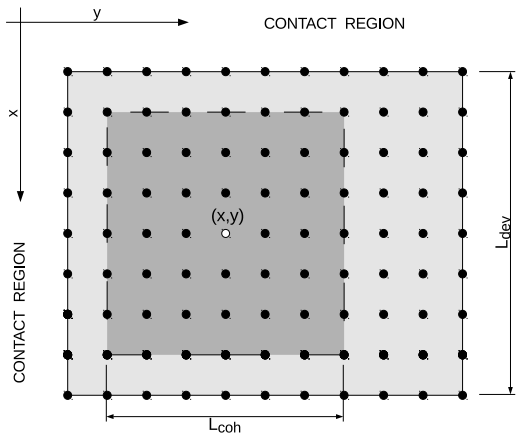

Fig. 1. Coherence box of size $\mathbf{L}_{coh} = (M'\Delta x, N'\Delta y)$, centred at node (x, y) in the discretized domain, of size $\mathbf{L}_{dev} = (M\Delta x, N\Delta y)$, surrounded by semi-infinite contact regions.

II. Algorithm

To derive our BDFT algorithm, we adopt the idea of the 1D sliding DFT [6], [7]: The algorithm calculates the Fourier coefficients of a sequence $\{x_c..x_{c+N-1}\}$ using the coefficients calculated for $\{x_{c-1}..x_{c+N-2}\}$:

$$X_c(p) = e^{i\frac{2\pi p}{N}} \left(X_{c-1}(p) + x_{c+N-1} - x_{c-1}\right). \tag{5}$$

Each application of (5) consists of two real additions and two complex multiplications, which have to be repeated for

978-1-4799-5433-9/14/$31.00 © 2014 IEEE

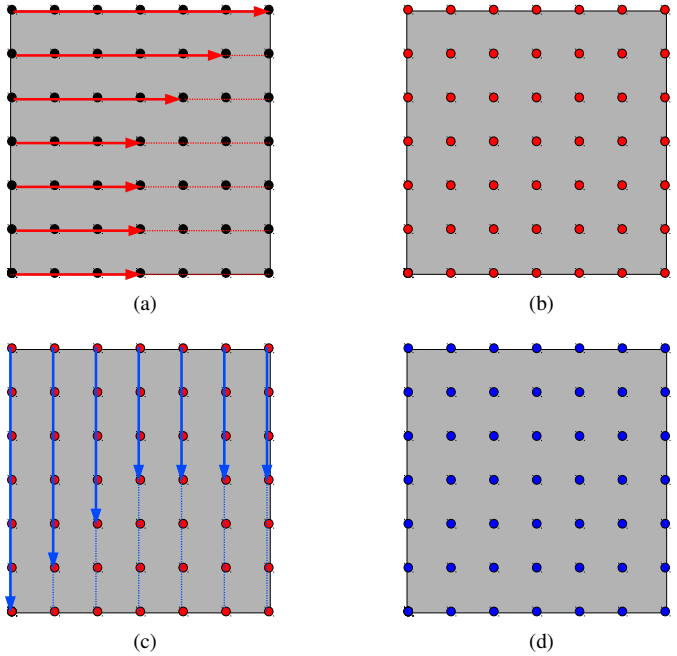

Fig. 2. Illustration of the calculation of a 2D DFT using the successive application of a 1D DFT: (a) the 1D DFT is calculated for each row, yielding (b) a matrix of Fourier coefficients (red). The (c) 1D DFT of each column of the prior result yields (d) the final result (blue).

each value of p. Therefore, the sliding DFT has a computational complexity of $\mathcal{O}(N)$. To apply the sliding DFT, as in (5), the two sequences under consideration must differ by only a single value; the potential values of each row (column) of the coherence boxes associated with two horizontally (vertically) adjacent nodes in the domain also differ only by a single value. This observation is exploited to calculate the 2D WP in an efficient manner.

Unlike the potential values, all the values of the potential difference (2) change between adjacent nodes. To allow a direct application of (5) to calculate (4), (1) is reformulated using a substitution of variables (Fourier shift theorem), such that

$$V_W\left(\mathbf{r}, \mathbf{P\Delta k}\right) = \frac{2}{\hbar \mathbf{L}} \mathbb{I}\mathrm{m} \int_{-\mathbf{L}/2}^{\mathbf{L}/2} d\mathbf{s} e^{-i2\mathbf{P\Delta k}\cdot\mathbf{s}} V\left(\mathbf{r}+\mathbf{s}\right).$$

This formulation has the further advantage that it avoids the calculation of the potential difference, saving further computation time. We also note that (5) allows, unlike the FFT, to easily compute only selected momentum (p, q) values, which do not have to be uniformly spaced. This can be of interest under certain physical considerations, e.g. uniformly spaced energy grid, and offers a further possibility to reduce computational costs.

The BDFT algorithm is applied to calculate the WP at each node in the domain, using the following procedure (as visualized in Fig. 3): First, the 1D DFT of the first N' potential values of all M rows in the domain are calculated, using an FFT algorithm (Fig. 3(b)); the resulting Fourier coefficients are retained in an array of size $N' \times M$. Thereafter, the 1D DFTs of the first M' Fourier coefficients of each of the N' columns are calculated (Fig. 3(c)), which yields an $M' \times N'$ matrix of Fourier coefficients representing the WP for the top-left node, $V_W\left(0, 0, p, q\right)$. After this initialization, the coherence box is moved downwards to the next node for which the WP is calculated by simply applying (5) to calculate the DFTs of the columns (Fig. 3(d),(e)). Once the WP has been calculated for each node in the first column of the domain, the $N' \times M$ array is developed to the right (Fig. 3(f)), again using (5), and the same procedure is repeated for the second column of nodes (Fig. 3(h)) etc. until the entire domain has been covered. Variations of the initialization approach can be envisioned, but the presented procedure shows favorable serial performance and cache complexity; a parallelized implementation would require multiple (modified) initializations.

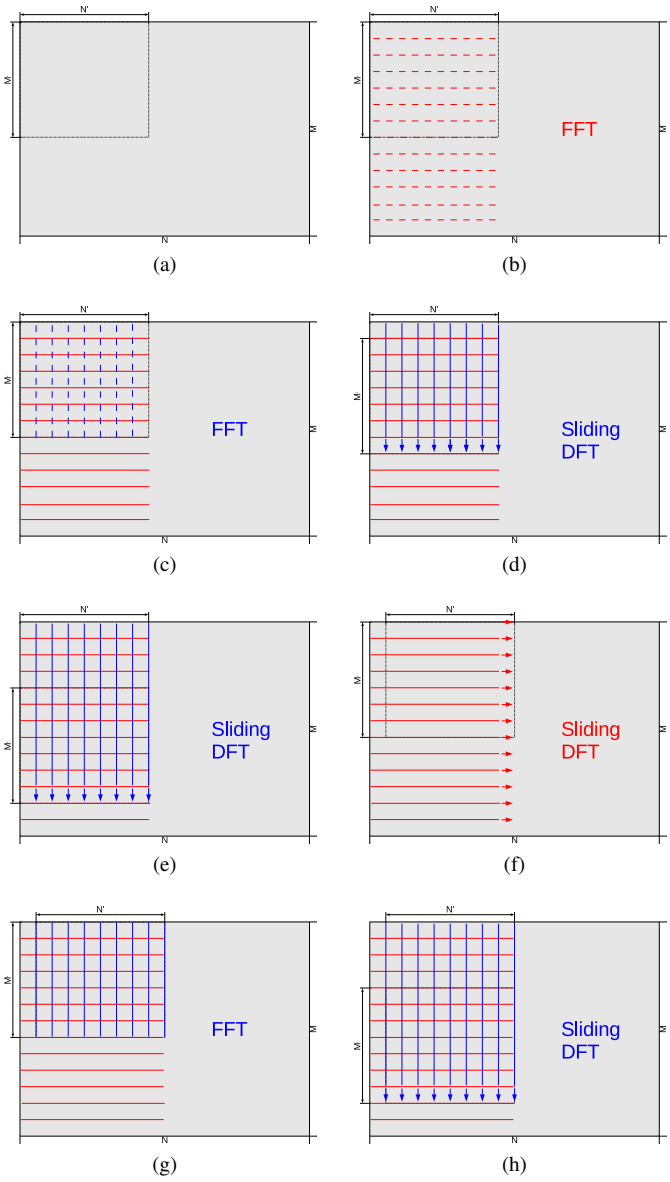

Fig. 3. Initialization and application of the BDFT algorithm, used to sequentially compute the WP at every node in the computational domain by successively applying the FFT and sliding DFT algorithms.

TABLE I. BENCHMARK AND SETUP SPECIFICS

Hardware OS	Intel Core 3110M; 8 GB (dual channel) Ubuntu 13.10 (64 bit)
Compiler flags	gcc 4.8.1 -O3 -fastmath -march=native
FFT library interface \ flags	FFTW 3.3 (SIMD enabled) dft_r2c_2d \ FFTW_MEASURE

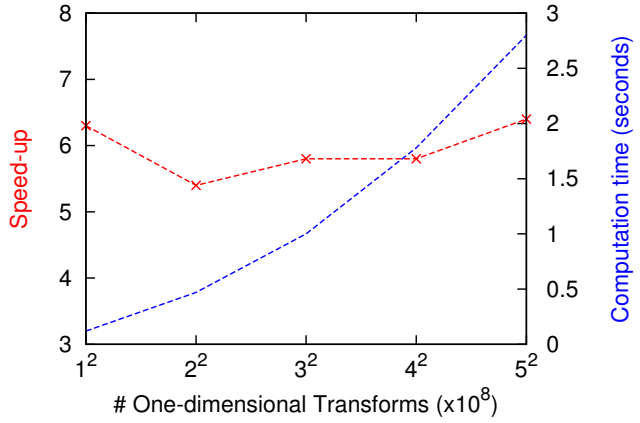

Fig. 4. Speed-up (as per Table II) of the BDFT algorithm versus a (pure) FFT implementation, to calculate the 2D WP for devices of various sizes, using a coherence box containing 100×100 potential values.

III. RESULTS

The BDFT algorithm was benchmarked against an FFT implementation using the FFTW library [3], with a setup detailed in Table I. The FFT implementation was optimized by exploiting the fact that (2) is real-valued and anti-symmetric and therefore must yield a purely imaginary output with conjugate symmetry.

Table II makes it evident that the BDFT reduces the computation time by at least a factor of five over a range of (plausible) domain sizes, as visualized in Fig. 4. Table II also reveals that the performance of the FFT implementation strongly depends on the transform size (coherence length), because the algorithms selected by the FFTW library perform best with transform sizes that are products of small prime numbers. The BDFT algorithm, on the other hand, is insensitive to the transform size and scales at a constant rate with size.

TABLE II. COMPUTATION TIME OF 2D WIGNER POTENTIAL

L_{dev} [a.u.]	L_{coh} [a.u.]	BDFT [s]	FFT [s]	Speed-up [1]
100	100	0.12	0.75	6.3
200	100	0.47	2.53	5.4
300	99	0.96	10.66	11.1
300	100	1.00	5.77	5.8
300	101	1.04	61.25	55.9
400	100	1.78	10.27	5.8
500	100	2.80	17.84	6.4

IV. CONCLUSION

The presented box discrete Fourier transform (BDFT) algorithm is shown to be an efficient approach to compute the WPs in a two-dimensional domain, with a significant reduction in computation time, thereby making self-consistent simulations of the Wigner equation more feasible. Moreover, this algorithm can easily be extended to three dimensions.

ACKNOWLEDGEMENT

This work has been supported by the Austrian Science Fund, project FWF-P21685-N22.

REFERENCES

[1] I. T. Dimov, *Monte Carlo Methods for Applied Scientists*. World Scientific, 2008.

[2] M. Nedjalkov and D. Vasileska, "Semi-discrete 2D Wigner-particle approach," *Journal of Computational Electronics*, vol. 7, no. 3, pp. 222–225, 2008.

[3] M. Frigo and S. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, Feb 2005.

[4] H.-Y. Huang, Y.-Y. Lee, and P.-C. Lo, "A novel algorithm for computing the 2D split-vector-radix FFT," *Signal Processing*, vol. 84, no. 3, pp. 561 – 570, 2004.

[5] Z. Chen and L. Zhang, "Vector coding algorithms for multidimensional discrete Fourier transform," *Journal of Computational and Applied Mathematics*, vol. 212, no. 1, pp. 63 – 74, 2008.

[6] E. Jacobsen and R. Lyons, "The sliding DFT," *Signal Processing Magazine, IEEE*, vol. 20, no. 2, pp. 74–80, Mar 2003.

[7] ——, "An update to the sliding DFT," *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 110–111, Jan 2004.