Ultramicroscopy 158 (2015) 56–64

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

# Benchmark test of accelerated multi-slice simulation by GPGPU

Fumio Hosokawa [a,*], Takao Shinkawa [a], Yoshihiro Arai [b], Takumi Sannomiya [c]

[a] BioNet Ltd, 2-3-28 Nishikityo, Tachikawa, Tokyo, Japan
[b] Terabese Ltd, Hane-nishi 3-5-1-102, Okazaki 444-0838 Japan
[c] Tokyo Institute of Technology, 4259 Nagatsuta, Midoriku, Yokohama 226-8503, Japan

ARTICLE INFO

*Article history:*
Received 18 March 2015
Received in revised form
22 June 2015
Accepted 28 June 2015
Available online 2 July 2015

*Keywords:*
Multi-slice
GPU
Benchmark
Image simulation
TEM
STEM

ABSTRACT

A fast multi-slice image simulation by parallelized computation using a graphics processing unit (GPU) has been developed. The image simulation contains multiple sets of computing steps, such as Fourier transform and pixel-to-pixel operation. The efficiency of GPU varies depending on the type of calculation. In the effective case of utilizing GPU, the calculation speed is conducted hundreds of times faster than a central processing unit (CPU). The benchmark test of parallelized multi-slice was performed, and the results of contents, such as TEM imaging, STEM imaging and CBD calculation are reported. Some features of the simulation software are also introduced.



## 1. Introduction

A general purpose computing on the graphics processing unit (GPGPU) attracts researcher's attention in many scientific fields to its marvelous computational capability. A graphics processing unit (GPU) consists of hundreds of simple cores that calculate the given tasks in parallel. A GPU has originally been developed to calculate the three-dimensional graphics in place of a central processing unit (CPU). Currently, GPU's processing power is getting to be used for a scientific calculation for the purpose of saving computational times. When a huge amount of simple calculation is parallelized and computed by GPU, its calculation speed can be hundreds of times faster than CPU. However depending on the type of calculation, each core may need to access data of CPU via various types of GPU memory multiple times during the calculation. Such memory accessing often becomes the bottle neck and blocks accelerating the calculation speed. Therefore the efficiency of GPGPU depends on each type of scientific calculation case by case. The examples of successful reports about the reduction of computation time were published [1].

The multi-slice method [2,3] is commonly employed in an image simulation technique of electron microscopy. At present, this provides probably the most costless algorithm in regard to the computational time. As to the microscopy hardware performance, a transmission electron microscope (TEM) and scanning transmission electron microscope (STEM) fitted with an aberration correction devices [4] have reached to the resolution of half an angstrom [5], revealing the atomic structure of a specimen. In current microscopy, under the aberration-corrected condition, image simulation has become more important to confirm that the images can be taken with microscope properly adjusted to correct conditions for achieving the specified microscope resolution. In other words, residual aberrations other than third order spherical aberrations are must be taken into account in the calculation. In a multi-slice image simulation, the dynamical diffraction is calculated using two-dimensional (2D) phase gratings and propagation-space (i.e. slice-thickness) which divide the specimen at intervals of several angstroms. The atomic potentials are projected onto slices and assumed to act as a phase grating for an electron wave function. To calculate one cycle of wave transmission from one slice to the next, 2D fast Fourier transform (FFT) is applied twice (forward and inverse) [6] to reduce the calculation time of the convolution of the propagation function. With a recent CPU, image simulation of TEM is reasonably fast to prevent a lot of stress for the user, since usual multi-slice calculation takes several seconds up to several tens of seconds. However for STEM image simulation, calculated image consists of thousands of pixels where each pixel takes the same calculation time of a single dynamical calculation of TEM image, and therefore the entire calculation may take several tens of minutes.

We had used our own multi-slice software, and for the purpose of minimizing the calculation time, we have implemented GPGPU functions to our multi-slice simulation.

* Corresponding author.
*E-mail address:* hosokawa@bio-net.co.jp (F. Hosokawa).

http://dx.doi.org/10.1016/j.ultramic.2015.06.018
0304-3991/

In this paper, the benchmark test relative to the original CPU version and some software features are reported.

## 2. Method

### 2.1. Implementation

Our multi-slice simulation had been written in Visual Studio 2010C++ (Microsoft), and for the development of GPU functions CUDA 6.5 (NVIDIA) [7] was used. CUDA codes, that are almost the same as C++ language, can be compiled on the Visual Studio 2010 in a manner similar to the original codes.

#### 2.1.1. General

A GPGPU works most effectively if it is adopted where a huge number of calculations consume a most of CPU processing time. In an image simulation, this case typically occurs in the calculation of crystal structure factors and transmission cross coefficients (TCC) [8]. For a crystal with a small unit cell, the calculation speed is fast enough regardless of the used core. However, for more complicated crystals or atomic clusters, GPU significantly accelerates the process compared to CPU.

The tasks in GPU cores work in parallel under the same instruction where the index numbers of "thread" and "block" are given [7]. Thus each parallel task is characterized by having unique number(s) in a same instruction. The same instructions which have unique index numbers, such as iteration numbers in "for loop" or "do …while loop" in C program, are processed at one time by many "thread" tasks of GPU. If the iteration count will not be changed while in loop(s) processing in the original code, it can be parallelized by replacing iteration numbers (conventionally written as i, j, etc.) with index numbers of "thread" and "block", and thus we implemented parallelized algorithm in multi-slice simulation. If the iteration count in the loop(s) is a variable, firstly original code should be modified so that iteration count remains constant. Before GPU processes the parallel calculation, the calculation data in CPU memory must be transferred to GPU memory, which is a time consuming process. Therefore frequency of this data transfer should be minimized for the effective usage of GPGPU. In the GPGPU programming, reading and writing data from each "thread" tasks should be designed in some special manner [1,7]. In particular, it should be done in minimum repetitions, and should be in sequential accessing along the local index, and avoid conflict in case of "shared memory", otherwise memory traffic often become a bottleneck of computational time.

#### 2.1.2. FFT

Acceleration of FFT by GPGPU is the main contribution to speed up of the whole multi-slice algorithm. NVIDIA provides a FFT-library named cuFFT in CUDA, which uses the interleaved array format for the numerical complex data in which the real and imaginary parts are stored alternately. Our CPU multi-slice code had been developed as a whole part for the use of the split array where the complex data is stored separately to real array and to imaginary array. For the effective use of cuFFT, a fairly large modification is necessary in whole architecture. Therefore at the first-stage, we developed our own parallelized FFT for GPU to use the split array. We parallelized our FFT of radix 2 by assigning vertical numbers of butterfly computation to indices of "threads", therefore, FFT of data points $N=2^p$ produces $N/2$ "threads" and each "thread" calculate a butterfly computation in a loop of iteration $p$ which corresponds to horizontal numbers of butterfly computation.

Before this study, we compared the calculation speed in regard to the calculation speed of $512 \times 512$ FFT and found that cuFFT of interleaved array is 1.3 times faster than our FFT of split array. We also tested cuFFT plus data conversion function which is supplied in CUDA (split array to interleaved array), and found that our FFT is 6.4 times faster than the combination of NVIDIA functions. As a consequence, we are using our own FFT without modifying the whole part of our multi-slice code instead of using the cuFFT.

#### 2.1.3. Pixel to pixel operation

A pixel-to-pixel operation can be simply parallelized by replacing the iteration loops (e.g. "for", "do" etc) with indexed threads in GPU function. In the image simulation algorithm, there are many operations similar to pixel-to-pixel operations. For example, multiplication of electron wave function by phase grating or by propagation function can be parallelized in such a way. These calculations are accelerated by GPU although its contribution to the reduction of total multi-slice calculation time is not dominant.

#### 2.1.4. Crystal structure factor

To calculate the crystal structure factor is the first thing to do in image simulation. The crystal structure factor is used to get the projected specimen potential onto the slice. It is necessary to calculate all the structure factors that have a scattering vector which is located in a reciprocal plane of the slice plane. This calculation process is similar to that of the cross section of three-dimensional discrete Fourier transform. For an ordinary crystal cell that has unit length of several angstroms, the calculation time needed for the structure factor is small compared to total computational time. As for large super cell that has unit length of several tens of angstroms, the structure factor calculation takes noticeably long time in CPU processing and sometimes occupies most of the calculation time. Parallelization of the calculation of crystal structure factor tends to be simple like that of pixel to pixel operation. It can be done by replacing the iteration variables with index numbers of thread and block in GPU function. The projected potential is given by Fourier transform of crystal structure factor which is multiplied by a constant.

#### 2.1.5. Phase grating and propagation function

The phase grating represents a phase shift which is a product of an "interaction constant" [2] and a projected potential, and the calculation was done for convenience when the structure factors were calculated by GPU. The propagation function is calculated in the reciprocal plane without using GPU. The calculation of propagation function needs the wavelength of electron, propagation distance between the entrance slice to the next slice and the electron incident angle to the entrance slice. In our program, the propagation distance is kept as the fixed value once one multi-slice calculation starts to run. The phase grating may be varied in one calculation depending on the propagation distance whose value is arbitrarily set at the start of the calculation for a certain crystal data.

#### 2.1.6. Dynamical diffraction

A dynamical diffraction is calculated using the multi-slice method, in which an electron wave function is multiplied alternately by phase grating and propagation function, respectively in the real and reciprocal spaces. These multiplications are repeated with electron's passage through every slice. The parallelization of this multiplication can be done in the same manner as described in Section 2.1.3 (Pixel to pixel operation). The forward and inverse 2D-FFTs are applied to the electron wave function every time before the multiplication by propagation function and phase grating, respectively. Thus 2D-FFT operation is executed twice per slice, in order to perform convolution in the real space. In a dynamical diffraction calculation with CPU, 2D-FFT is the most massive part to consume the calculation time. For example, assuming the

specimen thickness of 10 nm, slice interval of 0.5 nm and scanning 10,000 pixels, multi-slice calculation requires 40 times 2D-FFTs for TEM, and needs 400,000 times 2D-FFTs for STEM imaging. Since 2D-FFT is executed so many times, the parallelization of FFT is the key point of calculation of dynamical diffraction. Once the dynamical diffraction is calculated up to the last slice defined by specimen thickness, the intermediate diffraction at each slice is likewise obtained and can be stored. In the case of memorizing intermediate diffractions, the efficiency of GPGPU is slightly lost because the memory latency in GPU is not very much accelerated unlike numerical ability.

#### 2.1.7. Imaging by using TCC

In TEM imaging, the image is formed by the interferences of all possible combinations of the electron waves that can be assumed to be emitted from virtual incoherent sources in the reciprocal plane, where various aberrations are introduced. If a monochromatic point source was assumed, calculation can be simply and fast done using 2D-FFT. However, realistically with a finite source size and a finite energy width, the image intensities should be integrated over the source size as well as over the energy width, which typically results in degradation of image quality. To calculate the interference in each combination, TCC is used by employing usually only the first order term of Taylor expansions [9]. With this assumption for Taylor series, the integrals become Fourier transforms of source and energy distribution, which can be analytically solved for the Gaussian source [9, 10]. Then the TCC turns to be a product of each interference and individual coherence factor, and with TCC, Fourier transform of image intensity can be calculated [8].

In the calculation of TCC, firstly the number $n$ of diffraction spot in imaging aperture should be defined. Then the number of interference of two electron waves associated with TCC come to be $n$ squared, and each interference involves a calculations of phase shift and coherence due to the optical condition. This means that the calculation of TCC may take considerably long time for a large crystal unit cell, especially with many aberrations included. Here we note that the Fourier transform I($\mathbf{k}$) of image intensity consists of the symmetric real part (Ir($\mathbf{k}$) = Ir(-$\mathbf{k}$)) and asymmetric imaginary part (Ii($\mathbf{k}$) = -Ii(-$\mathbf{k}$)). Using this symmetry, it is practically enough to calculate I($\mathbf{0}$) and one part of symmetric or asymmetric pair. Therefore the necessary number $N$ of calculated TCCs to obtain I($\mathbf{k}$) becomes $n(n+1)/2$. For a parallelization of calculation of TCC, as shown in Appendix B, the number of parallel tasks was set to $N$ by assigning the suitable local index of “thread” and “block”. The number n sometimes exceeds 10,000 in an atomic cluster image calculation, and consequently $n$ squared exceeds 100,000,000, which can be an extremely heavy load for sequentially working CPU and therefore better fits parallelized GPU.

#### 2.1.8. Absorptive form factor

In high angle scattering of electron diffraction, the scattering amplitude is dominated by thermal diffuse scattering (TDS) [11]. The intensity of TDS is incoherently formed by an interaction of beam intensity and absorptive potential on each slice. A Fourier coefficient of absorptive potential is given by the absorptive form factor, in an analogous formula in which a Fourier coefficient of normal potential is given by the atomic scattering factor [11, 12]. For the STEM image simulation of high angle annular dark field (HAADF), intensity of thermal diffuse scattering over the detection area needs to be calculated. Unlike the normal potential, the shape of absorptive potential is varied according to the detection area in the reciprocal plane. An absorptive potential for HAADF is given by using the absorptive form factor that is calculated by limited integration over a detection area [11]. The absorptive form factor was originally derived using the incident plane wave [13,12], and later was expanded its usage for the STEM condition [11, 14]. This absorptive potential given by limited integration can be calculated using 2D-FFT as shown in Appendix A. In our multi-slice program, the absorptive potential thus given is used for HAADF image simulation as the sum of convolution of beam intensity and absorptive potential from each slice [14]. The absorptive potential is used to estimate the total intensity over the detection area, but not gives the intensity distribution itself. As for the CBD calculation involving the TDS effects, Monte Carlo simulation is employed where the CBD intensities are integrated over the different configurations of atomic positions with the thermal displacements given by Gaussian distribution [3].

#### 2.1.9. CBD

Calculation of convergent beam diffraction (CBD) needs sufficiently large super cell which is an extended form of original crystal unit cell. The large super cell is necessary to avoid an overlapping of the adjacent beams which causes an artifact based on a finite Fourier transform. Keeping a sufficient Nyquist frequency, the calculation array size consequently becomes large compared to calculation of conventional plane wave diffraction. When higher order Laue zones are of interest, the $z$ dimension of super cell should be several hundreds of angstroms, and propagation distance (slice thickness) should be suitably smaller than the lateral crystal repeat length. The calculation of CBD is in principle the same as the calculation of plane wave diffraction: The incident beam comes into the slice, passes the phase grating, propagates, repeats these processes until the bottom of the specimen, and finally is Fourier transformed. The CBD differs from the plane wave diffraction in the point that the entrance beam is focused with a large convergent angle. The parallelization of CBD calculation was done in the same manner as described in Section 2.1.6.

For CBD calculation, the 2D memory arrays should be prepared one by one for each slice since the projected potential may differ according to the slice. Therefore, as shown later in the benchmark result of CBD, in the first time to calculate CBD under certain condition, the calculation time of all of the phase gratings amounts to most of calculation time of CBD. After the first calculation, if in the same condition except the entrance beam position, aberration and/or convergent angle, the calculation will be done in reduced time which is several percent of the required time in the first calculation, by reusing the pre-calculated phase gratings. The beam amplitude and CBD intensity at each slice can be stored if the user has activated the software switch.

#### 2.1.10. STEM imaging

STEM image simulation is the most time consuming task in our program. This task requires a set of CBD intensities successively calculated at each point with the scanning entrance beam, which is covering the minimum area in translation symmetry in the super cell. The number of scanning beams corresponds to the number of substantive image pixels of STEM image and also the number of repetitions of multi-slice calculations. A STEM image consists of elastic and TDS components that dominate the contrasts in bright field imaging and HAADF imaging, respectively. In the repetitions of multi-slice calculation, the phase grating and absorptive potential on every slice is calculated at the first beam position, which are then reused in successive repetitions. The parallelization of STEM image calculation is basically the same as that of CBD. The iteration itself according to the beam positioning is done as CPU task.

### 2.2. Benchmark environment

The multi-slice software accelerated by GPGPU was tested in

**Table 1**
Hardware configurations for benchmark test of GPU multi-slice.

| |
|---|
| Computer: Fujitsu Celcius M720 16 GB RAM |
| CPU: Xenon E5 1650 v2 3.5 GHz |
| GPU: Quadra K5000 / NVIDIA |
| CUDA Cores 1536 |
| Memory size total 4 GB |
| Memory interface 256 bit |
| Memory bandwidth 173 GB/sec |
| Compute capability |
| 3 C++ compiler: Visual studio 2010 /Microsoft |
| GPU compiler: CUDA 6.5/NVIDIA |

various calculations in comparison with the original CPU code. The time needed in each calculation was measured by the function "timeGetTime()" and accuracy was set to one millisecond by "timeBeginTime()", which are supplied in "Visual Studio (Microsoft)". Some of the calculations are done within a few milliseconds. In such a case, the calculation was repeated, e.g. $n$ times, and the measured time was divided by $n$ to obtain the accuracy. The hardware configuration for benchmark test is shown in Table 1.

## 3. Result and discussion

### 3.1. FFT

Table 2 shows the compute time of 2D-FFT of complex data in milliseconds. The measurement was performed about three different array sizes. Before the FFT calculation, preliminary tables were prepared for each array size. These previous calculations are not included in the times listed in Table 2, since these are executed only once. The calculation speed ratios of GPU to CPU are also shown as CPU/GPU. In this result, 2D-FFT is accelerated by GPU as almost 20 times faster than CPU.

### 3.2. Crystal structure factor

The calculation of crystal structure factors for the zero order Laue zone is necessary to get the phase grating for the incident electron wave. The compute times for $SrTiO_3$ along the incident direction [001] are shown in Table 3. The efficiency of GPU increases as the array size becomes larger. For the array size of $1024 \times 1024$, GPU works 190 times faster than CPU. This can be understood that the proportion of calculation duty to memory latency become bigger in CPU, by contrast, the proportion becomes smaller in GPU.

### 3.3. Dynamical diffraction

The calculation of dynamical diffraction is performed along the multi-slice method. Basically the efficiency of GPGPU for multi-slice calculation is almost the same degree as FFT, however, memory latency due to the recording of the complex amplitude for later use at every slice causes a loss of efficiency. The results of compute time are shown in Table 4. The accelerating voltage and specimen information are shown in the note. As already described in Section 2.1.6, the number of FFT calculations in the calculation of dynamical diffraction is twice the number of slices. The values listed in Table 4 exceed slightly the values estimated from the FFT results because of the additional time for memory latency.

**Table 2**
Result of 2D-FFT of complex data.

| Array size | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|
| GPU (ms) | 0.104 | 0.442 | 2.077 |
| CPU (ms) | 1.994 | 9.116 | 40.036 |
| CPU/GPU | 19.17 | 20.62 | 19.27 |

**Table 3**
Result of crystal structure factors for phase grating.

| Array size | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|
| GPU (ms) | 1.31 | 2.61 | 6.91 |
| CPU (ms) | 65.23 | 293.40 | 1320.93 |
| CPU/GPU | 49.79 | 112.60 | 191.16 |

*Note*: $SrTiO_3$ cubic ($a=b=c=0.3905$ nm ) [001].

**Table 4**
Result of Dynamical diffraction.

| Array size | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|
| GPU (ms) | 16 | 63 | 277 |
| CPU (ms) | 226 | 1037 | 4499 |
| CPU/GPU | 14.1 | 16.5 | 16.2 |

*Note*: 300 kV, $SrTiO_3$ cubic ($a=b=c=0.3905$ nm ) [001], slice thickness 0.3905 nm. 52 slices as specimen thickness.

**Table 5**
Result of TCC imaging.

| TCC with $\chi_0$ (rotational symmetric aberrations) | | | |
|---|---|---|---|
| $n$ | 149 | 4397 | 15193 |
| GPU (ms) | 7 | 13 | 67 |
| CPU (ms) | 11 | 1217 | 10380 |
| CPU/GPU | 1.6 | 93.6 | 154.9 |

| TCC with $\chi_1$ (all possible axial aberrations for aberration corrector) | | | |
|---|---|---|---|
| $n$ | 149 | 4397 | 15193 |
| GPU (ms) | 7 | 46 | 471 |
| CPU (ms) | 33 | 20642 | 244059 |
| CPU/GPU | 4.7 | 448.7 | 518.2 |

*Note*: $SrTiO_3$ ($a=b=c=0.3905$ nm ) [001], $n=149$; h-$Nb_2O_5$ monoclinic ($a=2.115$ nm, $b=0.3823$ nm, $c=1.936$ nm, $\alpha=90.0°$, $\beta=119.8°$, $\gamma=90°$) [010], $n=4397$; Hemoglobin super cell ($a=b=c=7$ nm), $n=$ 15,195; n: number of diffraction waves in objective aperture; $\chi_0$: defocus, third order spherical aberration and fifth order spherical aberration and chromatic aberration; $\chi_1$: $\chi_0$ plus astigmatism (2,3,4,5 and 6-fold), coma (2nd and 4th order), star and three lobe aberration.

### 3.4. TEM imaging with TCC

Computational time in imaging with TCC depends on the calculation condition. It depends firstly on the number of diffraction wave $n$ in the objective aperture, and secondly on the number of

**Table 6**
Result of CBD.

| | 1st | 2nd | 3rd |
|---|---|---|---|
| GPU (ms) | 21820 | 590 | 2480 |
| CPU (ms) | 520175 | 12284 | 12528 |
| CPU/GPU | 23.8 | 20.8 | 5.05 |

*Note*: Si [111] super cell ($a=8.868$ nm, $b=9.215$ nm, $c=50$ nm, $\alpha=\beta=\gamma=90.0°$) 30 kV, $512 \times 512$, $\alpha=18$ mrad, slice thickness 0.0783 nm, 639 slices.
1st (calculate phase grating at every slice).
2nd (reuse pre-calculated phase gratings).
3rd (reuse pre-calculated phase gratings, and record CBD and.beam amplitude at each slice).

**Table 7**
Result of STEM imaging of $SrTiO_3$.

| FFT array size | 256 × 256 | 512 × 512 |
|---|---|---|
| GPU (ms) | 32705 | 125945 |
| CPU (ms) | 463721 | 1930224 |
| CPU/GPU | 14.2 | 15.3 |

*Note*: $SrTiO_3$ [001] super cell ($a=b=3.16$ nm, $c=7.5$ nm, $\alpha=\beta=\gamma=90.0°$); Slice thickness 0.5 nm; Scanning 80 × 80 pixels on 0.79 nm × 0.79 nm; 300 kV with no aberration and $\alpha$ of 30 mrad; Detector 30 mrad/20 mrad (outer/inner).

aberrations considered in the imaging. The compute times of TCC imaging for several conditions are shown in Table 5. The most effective case of GPGPU, GPU works 500 times faster than CPU.

### 3.5. CBD

Calculation of CBD is processed as the calculation of dynamical diffraction using the focused beam at the entrance of the specimen. The benchmark results of CBD with calculation conditions such as convergent half-angle $\alpha$ are shown in Table 6. The calculation of CBD is applied only to the super cell data. The super cell data which was created by this software from Si [111] orientation are shown in the note in the table. At the first time in calculation of CBD, the phase gratings are needed to be calculated for large super cell, and this calculation is the main task in the first calculation, as described in 2.1.4 and 2.1.9. After the first calculation has been done, by reusing the phase gratings, the multi-slice method performs the fast calculation as shown in the 2nd and 3rd calculation, especially with GPU. Thus recalculation of CBD will be done very fast, with different beam focusing with some aberrations and different convergent angle. The program has flag switches which activate the recording of CBD intensity and/or incident beam amplitude at every slice, shown as the 3 rd calculation in Table 6.

### 3.6. STEM imaging

The benchmark results of STEM imaging are shown in Tables 7 and8. The calculation of STEM imaging can be applied only to the super cell data like CBD calculation. In Tables 7 and 8, some calculation conditions are shown in note, including scanning area with equivalent repeat length 0.79 nm in Table 7, and 0.3818 nm and 1.1668 nm in Table 8. The detector size of 30 mrad/20 mrad (outer/inner) means the anular bright field (ABF) method [15]. The horizontal cell size is not as large as that for CBD. The definition of suitable cell size is not straightforward, but experimental calculations with several sizes can suggest them as a guide. The calculation of STEM imaging with GPU is 14 times and 15 times faster than CPU, respectively for array size of 256 × 256 and 512 × 512. The efficiency of GPU is 75% of that in the 2nd case of CBD calculation. This is mainly due to the extra time needed for the recording of STEM image at every slice.

**Table 8**
Result of STEM imaging of $YBa_2Cu_3O_7$.

| FFT array size | 256 × 256 | 512 × 512 |
|---|---|---|
| GPU (ms) | 20,184 | 78,106 |
| CPU (ms) | 286,910 | 1,205,913 |
| CPU/GPU | 14.2 | 15.4 |

*Note*: $YBa_2Cu_3O_7$ [010] super cell ($a=1.909$ nm, $b=2.3336$ nm, $c=7$ nm, $\alpha=\beta=\gamma=90.0°$); Slice thickness 0.5 nm; Scanning:38 × 111 pixels on 0.3818 nm × 1.1668 nm; 300 kV with no aberration and $\alpha$ of 30 mrad; Detector: 30 mrad/20 mrad (outer/inner).

### 3.7. The numerical comparison between GPU and CPU

GPU has been developed originally for a single precision floating point number to exert its best graphical performance. In this paper, all the results obtained by GPU were given by using the single precision floating point number. In case of the current 64 bit CPU, its calculation speed is not affected whether to use the double or single precision floating point, and all the results in this paper obtained by CPU were given by using the double precision floating point number. Therefore, apart from the calculation speed, calculated results of GPU and CPU were numerically investigated to check how much difference exists between them. For the numerical comparison, Si [110] crystal was used for calculation under the condition of accelerating voltage 200 kV, reciprocal aperture for the propagator two thirds of Nyquist frequency, slice thickness 0.7679 nm, mean square displacement of Si atom 0.0045 $Å^2$, the fitting coefficients for the atomic scattering factors suggested by Weickenmeier and Kohl [16]. The multi-slice calculations were performed in terms of crystal thickness variation of 5 nm, 10 nm, 20 nm, 30 nm, 40 nm and 50 nm. In each calculation of crystal thickness, the intensity of the diffracted waves of (000), (002), (004), (006), (008) was calculated by GPU and CPU for the quantitative comparison. As to calculation array size, array size of 256 × 256, 512 × 512 and 1024 × 1024 had been tested in a preliminary check with CPU and confirmed that the calculated values by using the each array size had shown the agreement up to significant digits of eight. The comparison results, which were calculated using the 512 × 512 array size, are shown in significant digits of eight in Table 9. The differences between GPU and CPU are shown as Δ in percent figures. The averaged value, maximum value and minimum value of the obtained Δ are, respectively, 0.000166%, 0.000442% and 0.000000%. We can see that the forbidden reflections such as (002) and (006) tend to have larger Δ values than normal Bragg reflections. In total, the difference in the calculated values by GPU and CPU is supposed to be acceptable in most case of TEM and STEM simulations.

The multi-slice calculation of diffracted intensity along Si [110] with 200 kV; reciprocal aperture two thirds of Nyquist frequency; slice thickness 0.7679 nm; mean square displacement of Si atom 0.0045 $Å^2$; the fitting coefficients for the atomic scattering by Weickenmeier and Kohl [16] and crystal thickness variation of 5 nm, 10 nm, 20 nm, 30 nm, 40 nm and 50 nm. The diffracted waves of (000), (002), (004), (006), (008) were calculated by GPU and CPU for the quantitative comparison.

### 3.8. Figures

The calculated images of TEM, CBD, STEM ABF and STEM HAADF are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4, respectively. They are presented as the visual examples that are typically extracted from the results described above.

A TEM image was calculated from Hemoglobin super cell ($a=b=c=7$ nm) in the condition of slice thickness 0.5 nm, total thickness 14 slices, accelerating voltage 300 kV, spherical aberration 0.0, defocus −6 nm, chromatic aberration 1.0 mm, electron energy spread 0.2 eV, objective aperture 10 $nm^{-1}$ and calculation array size 512 × 512. The image is displayed in one second after activating the calculation button. The projection can be changed to arbitrary direction by software immediately. The calculation parameter such as defocus, aberrations, imaging aperture size, and thickness up to calculated values can be altered and applied to the calculated image, which reflects the change at once.

Two diffraction patterns were calculated from Si [111] super cell ($a=8.868$ nm, $b=9.215$ nm, $c=50$ nm, $\alpha=\beta=\gamma=90.0°$) in the condition of slice thickness 0.0783 nm and 2D array size 512 × 512, the accelerating voltage 30 kV and convergent half angle 18 mrad.

**Table 9**
Quantitative comparison of calculated results by GPU and CPU.

| 5 nm | (000) | (002) | (004) | (006) | (008) |
|---|---|---|---|---|---|
| CPU | 0.81617856 | 0.011343629 | 0.096469596 | 0.0021949543 | 0.013187863 |
| GPU | 0.81617844 | 0.011343652 | 0.096469618 | 0.0021949557 | 0.013187863 |
| Δ (%) | 0.00001470 | 0.000246835 | 0.00002281 | 0.00006378 | 0.00000000 |
| 10 nm | (000) | (002) | (004) | (006) | (008) |
| CPU | 0.36573529 | 0.044073224 | 0.10707059 | 0.0054303189 | 0.012857249 |
| GPU | 0.36573517 | 0.044073340 | 0.10707057 | 0.0054303268 | 0.012857242 |
| Δ (%) | 0.00003281 | 0.00026320 | 0.00001868 | 0.00014548 | 0.00005444 |
| 20 nm | (000) | (002) | (004) | (006) | (008) |
| CPU | 0.69977629 | 0.15477523 | 0.066462465 | 0.0067163068 | 0.0056719920 |
| GPU | 0.69977713 | 0.15477556 | 0.066462539 | 0.0067163329 | 0.0056719887 |
| Δ (%) | 0.00012004 | 0.00021321 | 0.00011134 | 0.00038861 | 0.00034379 |
| 30 nm | (000) | (002) | (004) | (006) | (008) |
| CPU | 0.85532290 | 0.20834675 | 0.055604573 | 0.0062355399 | 0.0057564359 |
| GPU | 0.85532379 | 0.20834737 | 0.055604696 | 0.0062355576 | 0.0057564471 |
| Δ (%) | 0.00010405 | 0.00029758 | 0.00022120 | 0.00028386 | 0.00019456 |
| 40 nm | (000) | (002) | (004) | (006) | (008) |
| CPU | 0.094851598 | 0.28272310 | 0.12633735 | 0.0075162319 | 0.012581608 |
| GPU | 0.094851412 | 0.28272396 | 0.12633747 | 0.0075162444 | 0.012581615 |
| Δ (%) | 0.00019610 | 0.00033602 | 0.00009498 | 0.00016631 | 0.00005564 |
| 50 nm | (000) | (002) | (004) | (006) | (008) |
| CPU | 0.74029970 | 0.33945933 | 0.042041071 | 0.013782486 | 0.0054923743 |
| GPU | 0.74030095 | 0.33946022 | 0.042041108 | 0.013782547 | 0.0054923734 |
| Δ (%) | 0.00016885 | 0.00026218 | 0.00008801 | 0.00044259 | 0.00001639 |

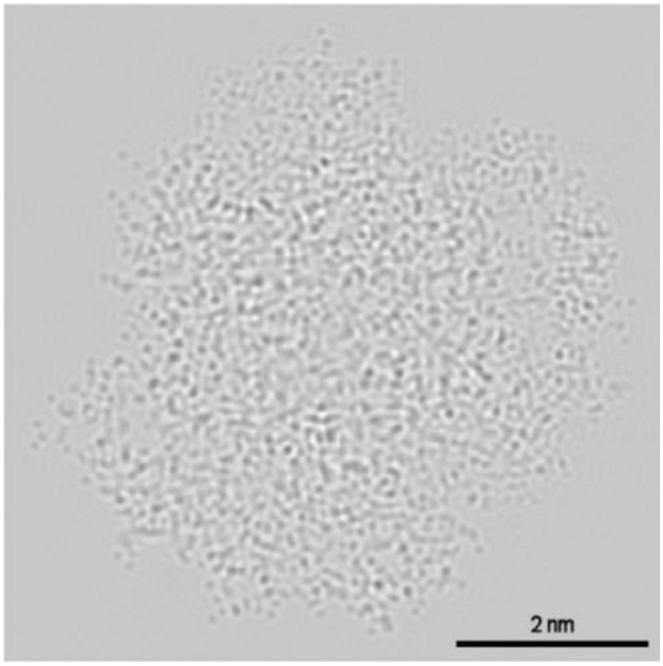


**Fig.1.** Simulated TEM image of hemoglobin.

Fig. 2a shows the elastic CBD pattern which does not involve the TDS. The image of Fig. 2a was displayed after activating the calculation button, in 23 s and one second, respectively in the first and the second calculations. Fig. 2b shows the CBD pattern which involves the TDS calculated by Monte Carlo simulation with mean square displacement of Si atom $u^2$ of 0.0045 Å$^2$ and iteration of calculation 12 times. The image of Fig. 2b was displayed after activating the calculation button in 152 s. For the better visibility of Kikuchi bands, the display contrast is adjusted to see the weak intensity in higher scattering region in Fig. 2b.

A STEM ABF image was calculated from $YBa_2Cu_3O_7$ [010] super cell ($a$=1.909 nm, $b$=2.3336 nm, c=7 nm, $\alpha=\beta=\gamma=90.0^\circ$) in the condition of slice thickness 0.5 nm, 2D array size 512 × 512, scanning pixels 38 × 111, scanning area 0.3818 nm × 1.1668 nm, accelerating voltage, 300 kV, no aberration, $\alpha$ 30 mrad, detector 30 mrad/20 mrad (outer/inner). The ABF image of Fig. 3a is displayed in 79 s after activating the calculation button. Fig. 3b shows the calculated projected potential of $YBa_2Cu_3O_7$ along the [010] direction.

A STEM HAADF image was calculated from Si [211] super cell (a=3.762 nm, b=3.840 nm, c=8 nm, $\alpha=\beta=\gamma=90.0$ deg) in the condition of slice thickness 0.5 nm, 2D array size 512 × 512, scanning pixels 32 × 77, scanning area 0.313 nm × 0.768 nm, accelerating voltage, 300 kV, no aberration, α 25 mrad, Gauss image size 0.02 nm, detector 130 mrad/100 mrad (outer/inner). The image is displayed in 53 s after activating the calculation button. The image intensity is consists of elastic and inelastic components of TDS, which is the dominant for the HAADF intensity. The TDS intensity is given by the interaction of the beam intensity and absorptive potential. Fig. 4b shows the calculated absorptive potential in the condition of mean square displacement of Si atom $u^2$ 0.0045 Å$^2$ and detector 130 mrad/100 mrad (outer/inner).

## 4. Conclusion

The GPGPU has been implemented to an image simulation of TEM and STEM using the multi-slice method. Several benchmark

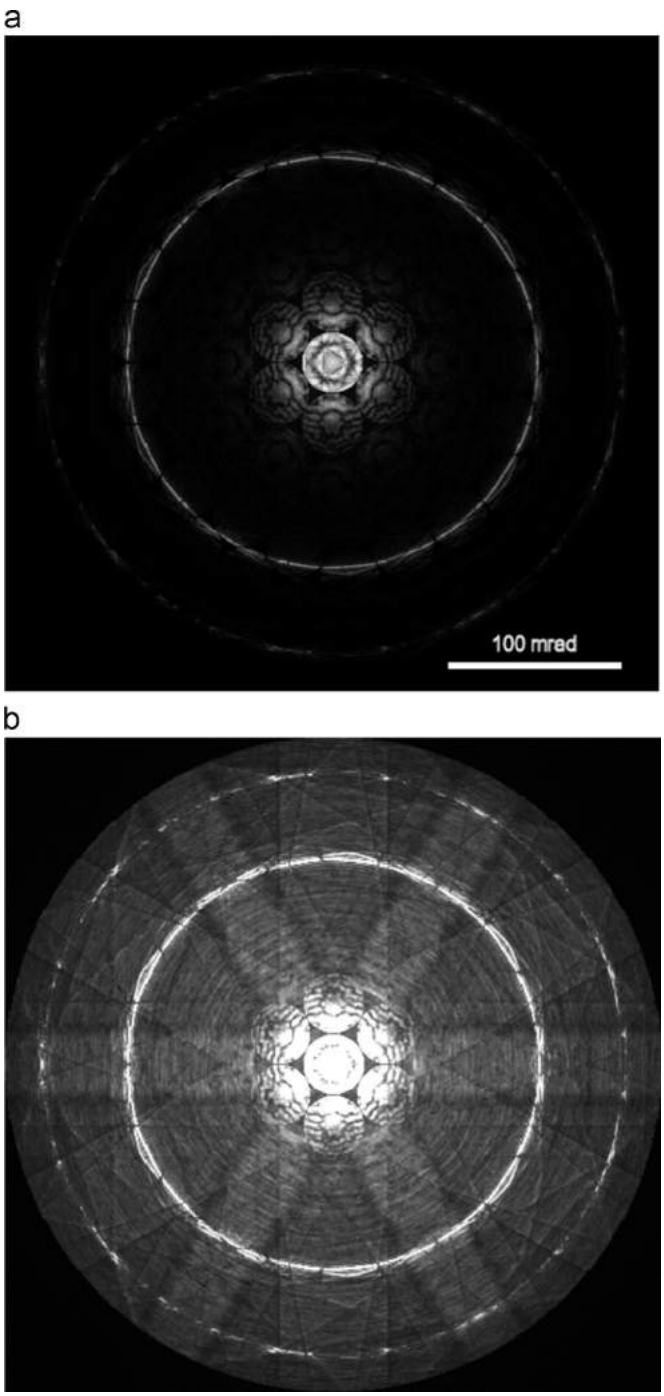


**Fig. 2.** Simulated CBD of Si [111].

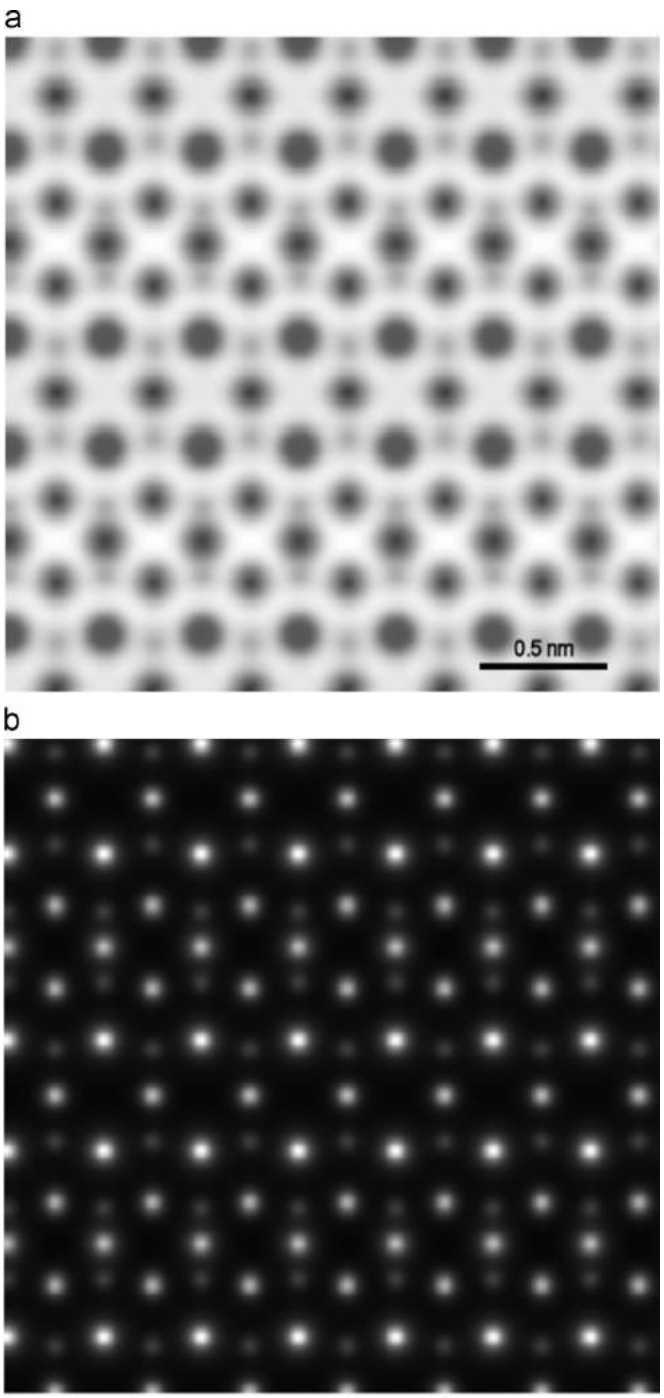


**Fig. 3.** Simulated STEM (ABF) of $YB_2Cu_3O_7$.

tests have been performed and efficiency of GPU varies, as already reported in many other articles, depending on the type of the calculation. For example, the calculation of complex 2D-FFT was accelerated about twenty times, STEM imaging was accelerated about fifteen times, and most effectively, TCC was accelerated about five hundred times faster than CPU. The quantitative comparison of the calculated results of GPU and CPU show that the difference between them is less than 0.005%. Overall, large benefit of using GPU in the simulation of electron microscopy was clearly demonstrated. In the near future, online image fitting with multi-slice simulation will be realized during the measurement.

## Appendix A: Absorptive form factor

For the monatomic specimen, the Fourier components of absorptive potential in volt unit are given by, (For the compound, sum of Eq. (A.1) ($\sum V_{\mathbf{g}}'$) gives the total potential).

$$V_{\mathbf{g}}' = -h^2/(2\pi me\Omega) \sum exp(-irg) f'_{\mathbf{g}} \quad \text{(A.1)}$$

where $h$ is the Planck constant, m the electron rest mass, $e$ the elementary charge, $\Omega$ the volume of scattering object, $\mathbf{r}$ the atomic position, $g$ ($g=2\pi\mathbf{k}$) the reciprocal vector, $\mathbf{k}$ the spatial frequency vector, and $f'_{\mathbf{g}}$ the absorptive form factor [12].

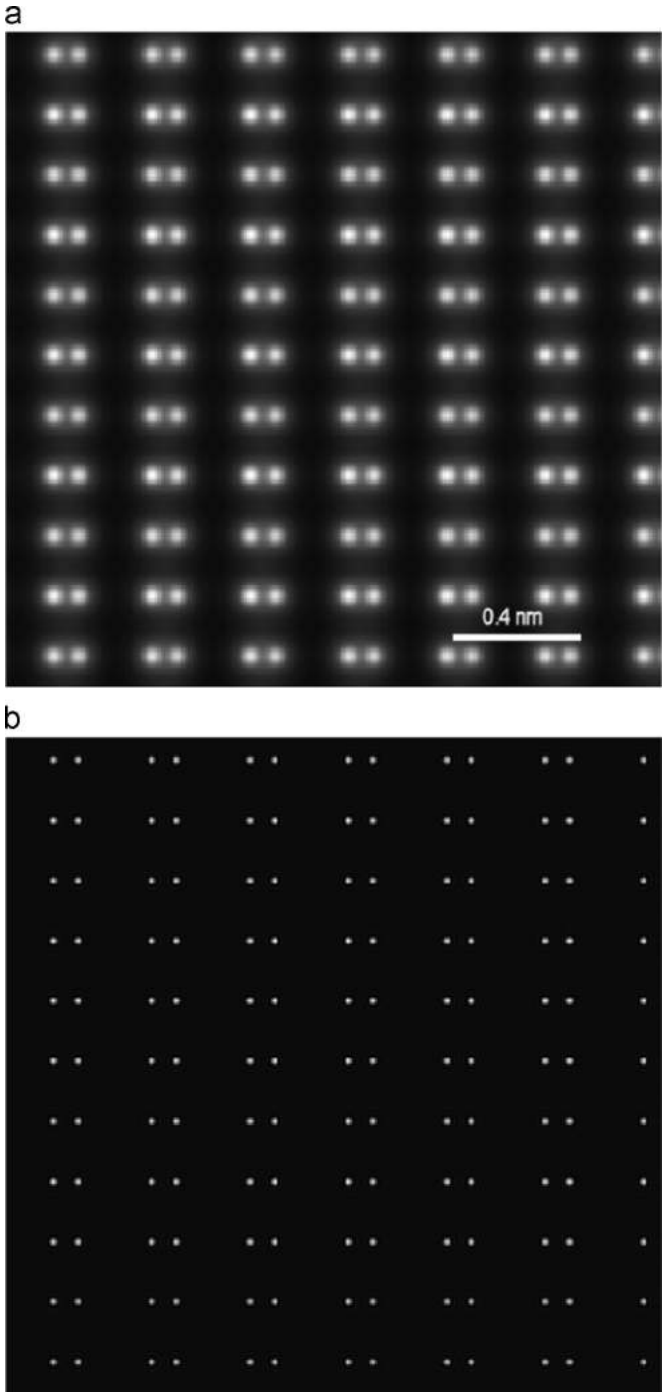


**Fig. 4.** Simulated STEM (HAADF) Si[211].

$$f'_{\mathbf{g}} = (2h/mv)\int f_{\mathbf{s}} f_{\mathbf{s}-\mathbf{s}'}\left\{\exp\left(-M\mathbf{s}^2\right) - \exp\left(-M\mathbf{s}'^2\right)\exp\left(-M(\mathbf{s}-\mathbf{s}')^2\right)\right\}d^2\mathbf{s}' \tag{A.2}$$

Here $v$ is the electron velocity, $\mathbf{s}$ the reciprocal vector ($\mathbf{s}=\mathbf{g}/4\pi=\mathbf{k}/2$), $M=8\pi^2\mathbf{u}^2$, $\mathbf{u}^2$ the mean-squared thermal vibration amplitude, $f$ the non-relativistic form factor. As for the STEM HAADF imaging, the interval of integration should correspond to the detection area and Fourier component $V_{\mathbf{g}}^{\mathrm{HA}}$ for HAADF becomes [11]

$$V_{\mathbf{g}}^{HA} = -h^2/(2\pi me\Omega)\sum exp(-i\mathbf{rg}) f'^{HA}_{g} \tag{A.3}$$

where

$$f'^{HA}_{g} = (2h/mv)\int_{detector} f_{s'} f_{s-s'}\left\{\exp\left(-M\mathbf{s}^2\right) - \exp\left(-M\mathbf{s}'^2\right)\exp\left(-M(\mathbf{s}-\mathbf{s}')^2\right)\right\}d^2\mathbf{s}' \tag{A.4}$$

In our calculation, defining the detector function as,
$D(\mathbf{s}')=1$ if $\mathbf{s}' \in$ detector,
0 if $\mathbf{s}' \notin$ detector,
then Eq. (A.4) becomes

$$f'^{HA}_{g} = (2h/mv)\{A(\mathbf{s}) - B(\mathbf{s})\} \tag{A.5}$$

$$A(\mathbf{s}) = exp\left(-M\mathbf{s}^2\right)\{(D(\mathbf{s})f_{\mathbf{s}})*f_{\mathbf{s}}\} \tag{A.6}$$

$$B(\mathbf{s}) = (D(\mathbf{s})f_{\mathbf{s}}exp\left(-M\mathbf{s}^2\right))*\left(f_{\mathbf{s}}exp\left(-M\mathbf{s}^2\right)\right) \tag{A.7}$$

where mark $*$ means convolution as

$$(D(\mathbf{s})f_{\mathbf{s}})*f_{\mathbf{s}} = \int D(s')f_{\mathbf{s}}f_{\mathbf{s}-\mathbf{s}'}d^2\mathbf{s}' \tag{A.8}$$

and

$$(D(\mathbf{s})f_{\mathbf{s}}exp\left(-M\mathbf{s}^2\right))*\left(f_{\mathbf{s}}exp\left(-M\mathbf{s}^2\right)\right) = \int D(\mathbf{s}')f_{\mathbf{s}'}exp\left(-M\mathbf{s}'^2\right)f_{\mathbf{s}-\mathbf{s}'}exp\left(-M(\mathbf{s}-\mathbf{s}')^2\right)d^2\mathbf{s}' \tag{A.9}$$

By using Eqs. (A.5)–(A.7), $V_{\mathbf{g}}^{\mathrm{HA}}$ can be calculated fast enough because of the acceleration of convolution with GPGPU.

## Appendix B. Parallelization of TCC imaging

Let $n$ be the number of waves in imaging aperture, then the number $N$ of required TCCs to calculate the image spectrum $I(\boldsymbol{k})$ (using the Friedel's) law are

$$N = n(n+1)/2 \tag{B.1}$$

The image spectrum $I(\boldsymbol{k})$ is expressed as [8],

$$I(\boldsymbol{k}) = \sum_{n=0}^{N}\mathrm{TCC}(\boldsymbol{k}+\boldsymbol{k}'_n, \boldsymbol{k}'_n)\Phi(\boldsymbol{k}+\boldsymbol{k}'_n)\Phi^*(\boldsymbol{k}_{n'}) \tag{B.2}$$

In case of that $\mathbf{k}$ is chosen as its $x$ (or $y$) component $\mathbf{k}x$ (or $\mathbf{k}y$) is always not negative (or positive).
then I($-\mathbf{k}$) is given by

$$\boldsymbol{I}(-\boldsymbol{k}) = \boldsymbol{I}^*(\boldsymbol{k})\ (*\text{complex conjugate}) \tag{B.3}$$

Introducing the numbers

$$x_1 = [(n+1)/2] \tag{B.4}$$

$$y_1 = x_1 \tag{B.5}$$

$$x_2 = n - x_1 + 1 \tag{B.6}$$

$$y_2 = n - x_1, \tag{B.7}$$

here bracket [ ] means Gauss' notation that gives the integer part.
Then $N$ is expressed as

$$N = x_1y_1 + x_2y_2. \tag{B.8}$$

This means that the calculation of image spectrum $I(\mathbf{k})$ can be parallelized in two parts as $I_1(\mathbf{k})$ and $I_2(\mathbf{k})$ respectively associated with $x_1y_1$ and $x_2y_2$.
The image spectrum I($\mathbf{k}$) is given by

$$I(\mathbf{k}) = I_1(\mathbf{k}) + I_2(\mathbf{k}) \tag{B.9}$$

$I_1$ and $I_2$ are given by

$$I_1 = \sum_{i=0}^{x1}\sum_{j=0}^{y1}\ \mathrm{TCC}\left(\boldsymbol{k}_i, \boldsymbol{k}_{i+j}\right)\Phi(\boldsymbol{k}_i)\Phi^*\left(\boldsymbol{k}_{i+j}\right) \tag{B.10}$$

$$I_2 = \sum_{i=0}^{x2} \sum_{j=0}^{y2} \mathrm{TCC}\left(\boldsymbol{k}_{[m/n](n-1-2j)+j2}, \boldsymbol{k}_{-[m/n](1+j)+m}\right) \Phi\left(\boldsymbol{k}_{[m/n](n-1-2j)+j2}\right) \Phi^*\left(\boldsymbol{k}_{-[m/n](1+j)+m}\right) \quad \text{(B.11)}$$

where

$$m = i + j + x_1 \quad \text{(B.12)}$$

$I_1(\mathbf{k})$ can be calculated in parallel task by assigning the $i_1$ ($i_1$=0,1,2.. $x_1-1$) and $j_1$ ($j_1$=0,1,2.. $y_1-1$) to "thread" index and "block" index. Likewise, by assigning the $i_2$ ($i_2$=0,1,2.. $x_2-1$) and $j_2$ ($j_2$=0,1,2.. $y_2-1$) to "thread" index and "block" index, $I_2(\mathbf{k})$ can be calculated. The variable $\mathbf{k}$ in equation B.10 and B(11) are given by $\boldsymbol{k}_i - \boldsymbol{k}_{i+j}$ and $\mathbf{k}_{[m/n](n\text{-}1\text{-}2j)\,+\,j2}$ - $\mathbf{k}_{-[m/n](1+j)\,+\,m}$, respectively.

In practice, there exists the maximum number of the "thread" index (e.g. 512 or 1024). However the "block" is able to form the layered structure where a huge amount of threads can be involved so that multiple tasks exceeding the thread limit can be realized with proper coding techniques.

## References


[1] W.W. Hwu, GPU Computing Gems Emerald Edition, Elsevier Inc., a Delaware corporation, New York, 2011.
[2] J.M. Cowley, Diffraction Physics, North-Holland Publishing Company, Amsterdam, 1975.
[3] E.J. Kirkland, Advanced Computing in Electron Microscopy, Plenum Press, New York, 1998.
[4] F. Hosokawa, H. Sawada, Y. Kondo, K. Takayanagi, K. Suenaga, Development of Cs and Cc correctors for transmission electron microscopy, Microscopy 62 (2013) 23–41.
[5] H. Sawada, Y. Tanishiro, N. Ohashi, T. Tomita, F. Hosokawa, T. Kaneyama, Y. Kondo, K. Takayanagi, STEM imaging of 47 pm-separated atomic columns by a spherical aberration-corrected electron microscope with a 300 kV cold field emission gun, J. Electron Microsc. 58 (2009) 357–361.
[6] K. Ishizuka, N. Uyeda, A new theoretical and practical approach to the Multislice method, Acta Crystallogr. A333 (1977) 740–749.
[7] ⟨https://developer.nvidia.com/cuda-zone⟩.
[8] K. Ishizuka, Contrast transfer of crystal imaging in TEM, Ultramicroscopy 5 (1980) 55–65.
[9] J. Frank, The envelope of electron microscopic transfer functions for partially coherent illumination, Optik 38 (1973) 519–536.
[10] R.H. Wade, J. Frank, Electron microscopic transfer functions for partially coherent axial illumination and chromatic defocus spread, Optik 49 (1977) 81–92.
[11] S.J. Pennycook, D.E. Jesson, High-resolution Z-contrast imaging of crystals, Ultramicroscopy 37 (1991) 14–38.
[12] D.M. Bird, Q.A. King, Absorptive form factors for high-energy electron diffraction, Acta Crystallogr. A46 (1990) 202–208.
[13] C.R. Hall, P.B. Hirch, Effect of Thermal diffuse scattering on propagation of high energy electrons through crystals, Proc. R. Soc. London Ser. A 286 (1965) 158–177.
[14] K. Ishizuka, A practical approach for STEM image simulation based on the FFT multislice method, Ultramicroscopy 90 (2002) 71–83.
[15] S.D. Findlay, N. Shibata, H. Sawada, E. Okunishi, Y. Kondo, Y. Ikuhara, Dynamics of annular bright field imaging in scanning transmission electron microscopy, Ultramicroscopy 110 (2010) 903–923.
[16] A. Weickenmeier, H. Kohl, Computation of absorptive form factors for high-energy electron diffraction, Acta Crystallogr. A47 (1991) 590–597.