Quantitative Simulation of Transmission Electron Microscope Micrographs using the Multislice Technique

William Collie
University of Warwick, Undergraduate Physics Department (GRP Bursary Scheme)
(Dated: September 1, 2017)

Transmission Electron Microscope simulation software was developed. This software makes use of the multislice algorithm and fast Fourier transforms to create simulated TEM micrographs from CASTEP generated pseudo potential files. The simulation allows for adjustments to spherical aberration, electron dose and microscope voltage.

I. INTRODUCTION

Transmission Electron Microscope (TEM) image simulation plays a vital role in helping to both interpret experimental images and in optimising the performance of such instruments. A high-powered computer these days will cost significantly less than the cost of an electron microscope, but when running sophisticated simulation software, can achieve comparable results in seconds. Simulations aim to produce images from first principle calculations, a detailed description of the specimen and good knowledge of the instrument. Our software uses a description of the projected potential throughout the specimen (produced using CASTEP software) and from this uses the multislice algorithm to produce an image of the specimen. The software allows for parameters such as spherical aberration, defocus and electron dose to be adjusted.

The multislice algorithm is highly accurate, assuming Δz is sufficiently small. Although the theory for multislice was developed in the late 1950s it was not until the early 1970s when it was used seriously to compute high resolution TEM images. The delay was due to limitations in both computing speed and microscope capabilities. Today computation speed is not an issue and image resolution less than 0.2nm is achieved routinely. These two advances have transformed high-resolution TEM with image simulations into a widely-used technique.

II. THE TRANSMISSION ELECTRON MICROSCOPE

Transmission Electron Microscopy is a microscopy technique which utilizes the small De Broglie wavelength of fast moving electrons to produce high resolution images. A beam of electrons is generated and accelerated down a vacuum column where it passes through a thin specimen. TEMs have significantly higher resolution than optical microscopes and can even achieve atomic resolution. Resolution is primarily limited by aberration from the lenses and multiple scattering events in the specimen. When considered as a whole the TEM is a complicated instrument but for image simulation purposes it can be simplified to that shown in Figure 1.

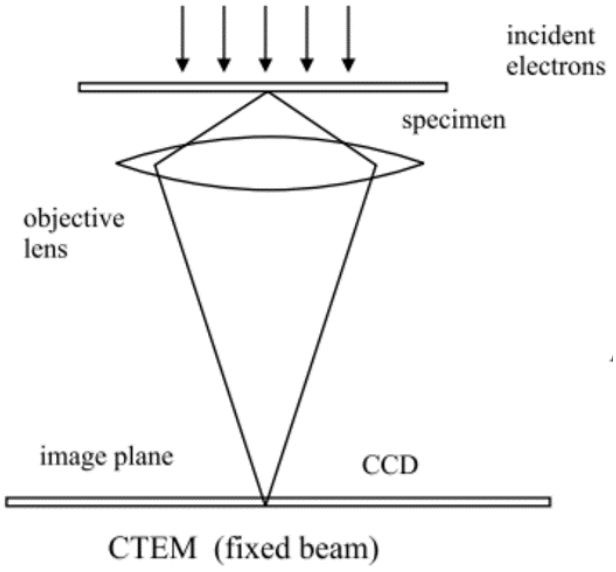

FIG. 1. A simplified schematic of a conventional transmission electron microscope (CTEM). Only the objective lens is considered as other lenses just act to magnify the image.

A. Relativistic Electrons

The electrons used in TEM usually have an energy in the range of 100keV up to 1MeV, and therefore move at relativistic speeds. Typical potentials in the specimen are usually much less, in the range of 10eV to 1000eV, and so the incident electrons travel through the specimen relatively unaffected. In the multislice algorithm both the relativistic De Broglie wavelength and electron interaction parameter are required. These can be calculated using the following equations respectively:

$$\lambda = \frac{hc}{\sqrt{eV(2m_0c^2 + eV)}} \tag{1}$$

$$\sigma = \frac{2\pi}{\lambda V}\frac{m_0c^2 + eV}{2m_0c^2 + eV} \tag{2}$$

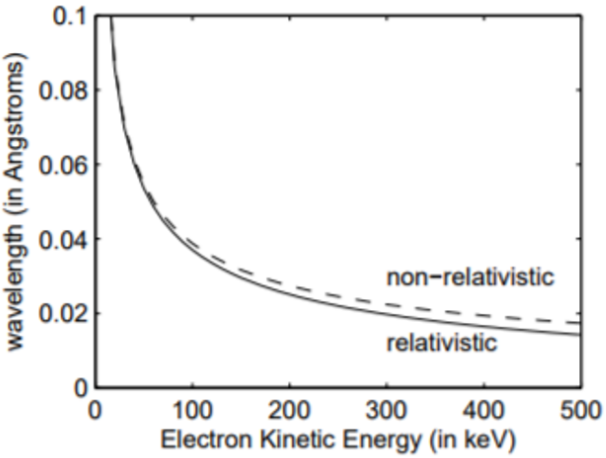

FIG. 2. A comparison between the relativistic (solid line) and non relativistic (dashed line) De Broglie wavelength of electrons for various microscope voltages.

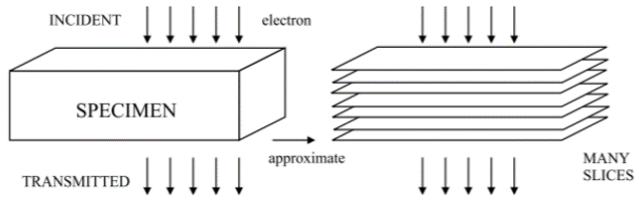

FIG. 3. The multislice approximation converts the specimen (left) into many thin slices (right)

III. MULTISLICE

If the initial wave function is given, then the wave function at any point inside the specimen can be calculated, given a description of potential inside the specimen. For CTEM the initial wave function is unity and for STEM it is given by the probe wave function.

The multislice approximation splits the specimen into a number of thin slices, each of which is thin enough to be approximated as a simple phase shift of the electron beam. The electron beam propagates between slices, as a small angle outgoing wave. The wave is transmitted through a slice of thickness Δz and then propagates a distance Δz to the next layer.

With some mathematical manipulation of the slowly varying portion of the electron wave function in the Schrodinger equation one can derive Equation 3,

$$\phi_{n+1}(x,y) = p(x,y,\Delta z) \circledast [t(x,y,z)\phi(x,y,z)] \qquad (3)$$

where $\circledast$ represents convolution. This form is extremely well suited to the fast Fourier transform (FFT) - a highly efficient algorithm. Since a convolution in real space is the equivalent to multiplication in reciprocal space, we can rewrite this as shown in equation 4.

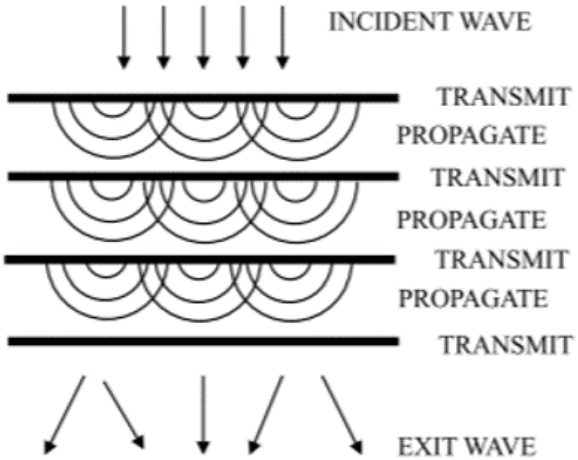

FIG. 4. The wave function as it propagates through the specimen.

$$\phi_{n+1}(x,y) = FT^{-1}[P_n(k_x.k_y,\Delta Z)FT(t_n(x,y)\phi_n(x,y)] \qquad (4)$$

The term FT refers to the Fourier transform, $t_n(x,y)$ the transmission function in real space, and $P_n(k,\Delta Z)$ the propagation function in reciprocal space. Note that k_x and k_y are the reciprocal lattice vectors.

$$t(x,y) = exp[i\sigma \int_z^{z+\Delta z} V(x,y,z')dz' = exp(-i\sigma V dz) \qquad (5)$$

Here V is the projected potential on a 2D plane and σ the interaction parameter.

$$P(k,\Delta Z) = exp(-i\pi\lambda(k_x^2 + k_y^2)dz) \qquad (6)$$

The multislice method reduces to a succession of transmit and propagate operations with a Fourier transform in between each. This is summarized in Figure 4.

A. Instrumental Effects (Lens Aberrations

The second part of the image simulation procedure takes the electron wave function from the multislice calculation, and modifies it by the effects of the microscope. Only the objective lens is considered in these calculations, because the subsequent intermediate and projector lenses simply serve to magnify the image without serious alterations of the image intensities.

The lens aberrations can be accounted for by calculating the transfer function h(x) of the objective lens, and

then convolving this with the exit wave function in reciprocal space. As before the convolution is more easily calculated using Fourier transforms.

$$\begin{aligned} g(x,y) &= |\phi(x) \circledast h(x)|^2 \\ &= |FT^{-1}[\phi(k)H(k)]|^2 \end{aligned} \tag{7}$$

$$H(k) = exp(-i\chi(k)) \tag{8}$$

$$\chi(k) = \frac{2\pi}{\lambda}(\frac{1}{4}C_s\lambda^4k^4 - \frac{1}{2}\Delta f\lambda^2k^2) \tag{9}$$

In these equations χ is the aberration function, C_s the coefficient of spherical aberration (third order) and Δf the defocus. This equation can be defined with Δf both positive or negative. With the definition in Equation 9 - a positive Δf represents an under focus of the objective lens. Generally speaking the optimum aperture and defocus are given by the Scherzer conditions.

$$\Delta f_{opt} = \sqrt{1.5C_s\lambda} \tag{10}$$

$$\alpha_{max} = (\frac{6\lambda}{C_s})^{\frac{1}{4}} \tag{11}$$

Algorithm for CTEM Multislice:

- Divide the specimen into thin slices
- Calculate the projected atomic potential for each slice (Castep)
- Calculate the transmission function for each slice and symmetrically limit each to $\frac{2}{3}$ of its maximum to prevent aliasing.
- Initialise the incident wave function to unity.
- Recursively transmit and propagate the wave function through each slice using Fast Fourier Transforms (FFTs). Repeat until the wave function is all the way through the specimen.
- Fourier transform the wave function at the exit surface and multiply this by the transfer function for the objective lens.
- Invert this back to real space and take the square modulus to give the final image intensity.

Number of Data Points	DFT	FFT	Ratio
32	1024	160	6.4
64	4096	384	10.7
128	16,364	896	18.3
256	65,536	2048	32
512	262,144	4608	56.9

TABLE I. Comparison of the relative CPU time required for a simple discrete Fourier Transform (DFT) and a fast Fourier transform (FFT) for different array lengths.

B. Multislice and Computational Speed

When implemented on a computer the convolutions can be efficiently calculated using Fast Fourier Transforms (FFTs). If the wave function is sampled with N_x points in the x direction and N_y points in the y direction then there are $N = N_xN_y$ Fourier coeffients. FFTs scale with $Nlog_2N$ instead of N^2 as in a direct matrix solution. When the multislice solution is implemented using the FFT, far less computing time is required than other methods.

C. Convergence Tests

There are a number of approximations that go into the derivation of the multislice algorithm, and so it is not always guaranteed to give the right answer for every possible set of input parameters. There are many more reasons for the simulation to fail than there are for it to succeed and so it is important to be skeptical and try to test if the program is in fact working correctly. The electron wave function only interacts via elastic scattering and this means that the total number of electrons should be conserved. A good test is to watch the total integrated intensity of the wave function as it progresses through the specimen. The total integrated intensity is normalised at the start.

The total integrated intensity can become less than one if the sampling is inadequate. When the specimen scatters electrons to high angles, some may be scattered outside the maximum allowed angle. Once they are scattered out of this limit they are effectively lost to calculation. A value of $I_n < 0.9$ is probably wrong and so is a value > 1 (typically this means the slice thickness is too large). Our program will output the total integrated intensity to the terminal, and so this should be used as a indicator to the correctness of the simulation.

IV. DOCUMENTATION FOR MULTISLICE PROGRAM

The Python script takes a CASTEP generated pseudo potential file as input, passes it to a C program where

Parameter	Typical Value
Electron Energy	400000 keV
Electron Dose	200000 $\AA m^{-2}$
Spherical Abberation	0.00001 m

TABLE II. Typical values for input parameters.

it undergoes the multislice calculations, before being passed back to the Python script for image processing. A greyscale image of the specimen is then created. The program allows for changes in certain experimental parameters such as microscope voltage, spherical aberration and electron dose. Changing these parameters creates subtle changes to the image.

1. Running the Python Script

Once the FFTW library and required python modules have been downloaded, the program should run with just the following shell command:

```
$ python multislice.py
```

Upon execution of the script a simple graphical user interface should pop up prompting the user for certain inputs. Typical values for these are as follows:

Once these have been inputted, click OK and the multislice image should appear in your current directory after a few seconds (Note: Very large potential files may take a few minutes to process).

2. FFTW Libary

It is essential that the FFTW library is downloaded in order for the program to run. FFTW is a software library for computing fast Fourier transforms efficiently. It is well known for being the fastest free software implementation of the FFT algorithm. It does this by supporting a variety of algorithms and choosing the one it estimates to be preferable in a particular circumstance. FFTW functions used in the code are fully documented on the FFTW website. FFTW3 first needs to be downloaded from the FFTW website and can then be installed via the following shell commands:

```
$ ./configure
$ make
$ make install
```

3. Non-rectangular Unit cells

The FFT algorithm requires a rectangular array as input. This is certain with a rectangular unit cell, but for non rectangular unit cells the data must be mapped and potential interpolated onto a rectangular domain. This has been implemented for the case of a Hexagonal unit cell. For non rectangular unit cell shapes, this manipulation must be done by the user preferably pre-simulation.

4. Python Modules

The following is a list of Python modules used in multislice.py - these will be required to run the script.

- Python Package Index
- Tkinter
- Python Image Library
- numpy

The respective shell commands to install such modules are:

pip:

```
$ sudo apt-get install python-pip
```

Tkinter:

```
$ sudo apt-get install python-tk
```

PIL:

```
$ sudo apt-get build-dep python-imaging
$ sudo apt-get install libjpeg8 libjpeg62-dev
libfreetype6 libfreetype6-dev
$ sudo pip install Pillow
```

numpy:

```
sudo pip install numpy
```

5. Linux Sort

The Castep pseudo potential files will not always be given in the row major order required. They must be reordered which can cause a significant increase in computation time for particularly large files. The most efficient way to sort data back into the correct order is by using Linux sort with a command such as (for a front on view):

```
$ sort -n -k3,3 -k2,2 k1,1 < inputfile
< outputfile
```

Fortunately the Python script automatically executes this command for the user, and even changes the Linux sort command for a side on view so no manipulation of this should be required.