A MANUAL FOR MULTISCAT

Boyao Liu

11 December 2020

0. FOREWORD

This manual is intended to be a guide for people who have no experience in using Fortran or Linux. When using the program, they will gradually pick up enough knowledge about them. However, you do need to know a little about helium atom scattering and how to use MATLAB.

This manual does not discuss the underlying theories of Multiscat. If the readers are interested, they can look at the following papers:

1. D.E. Manolopoulos, R.E. Wyatt,
Quantum scattering via the log derivative version of the Kohn variational principle,
Chemical Physics Letters,
Volume 152, Issue 1, 1988, Pages 23-32

2. D.E. Manolopoulos, R.E. Wyatt,
Iterative Solution in Quantum Scattering Theory. The log Derivative Kohn Approach
J. Chem. Soc., Faraday Trans., 1990,86, 1641-1648

3. Yousef Saad
Iterative Methods for Sparse Linear Systems
Second edition, 2000

This manual relied on Fay's guide, other students' work, and discussion with Dr Nadav Avidor.

I. RUNNING MULTISCAT

Multiscat is a program designed to calculate the diffraction intensities from surfaces. The underlying theory is quite complicated and involved. Before figuring out what it actually does, it may be better to run it straight to get a more general feel for it. You can follow the following procedures to run it.

1. Multiscat is a program that runs on Linux. It is recommended that you install an Ubuntu virtual machine if you are using Windows. The first step is to download an iso file that contains Ubuntu from https://ubuntu.com/#download. Just download one of the versions of Ubuntu Desktop.

2. To get your Ubuntu installed on your computer without removing your Windows OS, you can choose to install applications that support virtual machines. VirtualBox is a good choice. It can be downloaded from https://www.virtualbox.org/. The installation of VirtualBox should be relatively straightforward.

3. After VirtualBox is ready, click “New” to create a new virtual machine. In the next window that pops up, enter a name for your virtual machine and select “Linux” type and “Ubuntu (64-bit)” version. In the next window that appears, you can specify the memory size. We need *at least* 8 GB (8192 MB) of memory to run Multiscat successfully. Then continue clicking “Next” before you are asked to give the virtual machine file size. It is recommended that the virtual machine has at least 30 GB of space. Then click “Create”.

4. Double click your virtual machine to start it. You will be asked to supply an iso file that contains Ubuntu. That is what we have downloaded in step 1. After you do that, the installation of Ubuntu should start, and then everything should be intuitive. The installation could take up to 20 minutes.

5. Get Multiscat on your virtual machine. If you download it from GitHub: https://github.com/Cambridge-Atom-Scattering-Centre/Multiscat, open the zip file with Archive Manager. Extract it wherever you like.

6. Navigate to the directory Multiscat/Multiscat, you should see files named diagsub.f, FourierLabels.in, Makefile, multiscat.f90, multiscat.inc, Multiscat.conf, pot10001.in, potsub.f90, scatCond.in, scatsub.f and a folder called Docs.

7. Right-click in the blank space in the file directory, and then click “open in terminal”. You should see the terminal now.

8. Next, we will install the gfortran compiler. Type “sudo apt install gfortran” into the terminal and click “Enter”. You will then be asked to enter your password. After a while, you will be asked whether you want to continue; type “y” and click “Enter”. The download and installation should finish in one minute.

9. Type “sudo apt install make” into the terminal. The installation should finish within one minute as well.

10. Type “make” to configure the program. You will see some warnings, but they do not matter.

11. Type “./multiscat Multiscat.conf” to run the program. Make sure that you get the letter cases right. After it is done, you should see a file named “diffrac10001.out”. That file stores the diffraction intensities. And you have run Multiscat with success.

If you do not want to use VirtualBox, VMware Workstation Player can also run virtual machines. You can use it for non-commercial purposes for free. It can be downloaded from https://www.vmware.com/uk/products/workstation-player/workstation-player-evaluation.html.

Once you have installed VMware Workstation Player. Click “Create a New Virtual Machine”. Choose “Install disc image file (iso)” and browse for the iso file you downloaded in step 1. Then supply your machine’s name, your username, and password. Then just continue to click “Next” several times. Before clicking finish, make sure to customize hardware to let the virtual machine have *at least* 8 GB of memory.

Or Windows Subsystem for Linux can also be a good choice. See Install WSL | Microsoft Learn for more information.

II. A BRIEF INTRODUCTION OF WHAT MULTISCAT DOES

Consider an infinitely large surface. It is perfectly periodic with real space primitive vectors $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$. Here we assume that $\boldsymbol{a_1}$ is along the x-axis. In reciprocal space, the primitive vectors are $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$, which satisfy

$$\boldsymbol{a_1} \cdot \boldsymbol{b_1} = 2\pi, \boldsymbol{a_1} \cdot \boldsymbol{b_2} = 0,$$

and

$$\boldsymbol{a_2} \cdot \boldsymbol{b_2} = 2\pi, \boldsymbol{a_2} \cdot \boldsymbol{b_1} = 0.$$

Then consider a helium atom near the surface. The atom will experience a potential energy V as a function of its position. So the potential looks like $V(\boldsymbol{R}, z)$, where $\boldsymbol{R}$ is the position vector parallel to the surface, which is (x, y), and the z-axis is perpendicular to the surface. The surface potential should be periodic, so for any z, we have

$$V(\boldsymbol{R} + m\boldsymbol{a_1} + n\boldsymbol{a_2}, z) = V(\boldsymbol{R}, z),$$

where m and n are integers. If we know V's values in one unit cell, we have all the information about the potential since it is periodic in the xy plane.

Moreover, when the helium atom is far enough from the surface (which means that z is very large), they should not interact with each other, which means that

$$\lim_{z\to\infty} V(\boldsymbol{R}, z) = 0.$$

Besides, the atoms cannot go very deeply into the surface, which means that the potential will be very strong there. Usually, we have

$$\lim_{z\to-\infty} V(\boldsymbol{R}, z) = \infty.$$

Thus we only need V in a certain range of z to get a reasonably good simulation. The minimum value of z we use is usually called z_{max}, while the maximum is z_{max}.

In summary, to have a fairly good scattering calculation, we only need the potential in a "box" as shown in the figure.

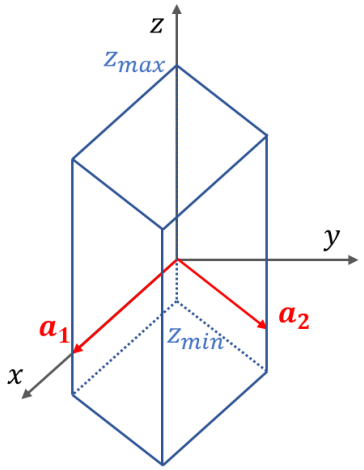

Figure 1. We only need the Helium-surface potential in the blue "box" to do the calculation

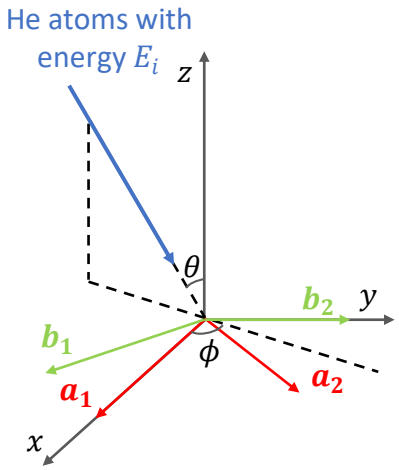

Figure 2. The incident helium beam

Now consider a beam of helium atoms hitting the surface. The energy of any helium atom in the beam is E_i. The polar and azimuthal angles of the beam are θ and ϕ, respectively.

The magnitude of the wave vector, sometimes also called the momentum (which is strictly not right because it does not have the unit of momentum), is $k_i = \frac{\sqrt{2mE_i}}{\hbar}$. The momentum parallel to the surface is

$$\boldsymbol{K} = (k_i \sin\theta \cos\phi, k_i \sin\theta \sin\phi).$$

If the reader is familiar with the theory of surface scattering, they will know that, for the scattered atoms, the momenta parallel to the surface must take the value of $\boldsymbol{K} + \boldsymbol{G}$, where $\boldsymbol{G} = i_1\boldsymbol{b_1} + i_2\boldsymbol{b_2}$ is a reciprocal lattice vector (i_1 and i_2 are integers).

If you take a look at diffrac10001.out that you have generated previously, you will notice three columns of numbers in it. The first column is the values of i_1, and the second i_2. The third column is the corresponding diffraction intensities. You will notice that all the diffraction intensities should add up to be one.

III. INSTALLING MATLAB ON UBUNTU

In order to proceed to the next step, we need to get MATLAB on Ubuntu. It is okay to work with MATLAB on Windows and transfer the files to Ubuntu. But it would be a lot more time-consuming.

1. People at Cambridge are eligible to use MATLAB for free. Just register with MathWorks using your @cam email and you will get access. To install MATLAB on your machine, go to the official website and download the installer for Linux (https://uk.mathworks.com/downloads/web_downloads/). It should be a zip file.

2. Open the downloaded zip with Archive Manager as instructed and extract it wherever you like.

3. In the extracted folder, you should see a bunch of files and folders. Right-click the blank space to open the terminal. Type “sudo ./install” into the terminal, and you will need your password.

4. You will then be asked to sign in. And the rest of the installation process is straightforward. Towards the end, you need some time to download a lot of stuff.

5. The tricky part about the installation of MATLAB on Ubuntu is that, after you see “Installation Complete”, you still won’t see MATLAB in the list of applications. To solve that, just open the terminal and type in “sudo apt install matlab-support”. Then you will be instructed to install matlab-support, which creates a shortcut for MATLAB in the launcher.

6. You will be asked to provide the location of MATLAB installation(s). By default it should be /usr/local/MATLAB/R2020b, but if you use other versions, that may be different. You can always check that. After that, you still need several steps to complete the installation, but they should be quite simple.

7. After installing matlab-support, you should be able to see the icon of MATLAB in your list of applications.

8. Double click the icon of MATLAB to start it. If it does not run, consider typing “matlab” into the terminal to troubleshoot. If it tells you to reactivate your license, go to the folder where MATLAB is installed. In a folder called “/bin”, open the terminal, and type “./activate_matlab.sh” to reactivate. You need your MathWorks account details to do that.

IV. HOW TO GET THE POTENTIAL FILES

Before getting to the topic, it may be worth noting that the contents in this section and the next section are a little bit involved and abstract. It may be better to reread those when you deal with the exercise.

The information about the potential file is stored in potxxxxx.in (as a starter, we only use one potential file, so that is pot10001.in). But suppose that we now only have V's analytical form in the "box" mentioned in section II. How can we convert the analytical form to pot10001.in so that Multiscat can read it?

That is when the MATLAB file Multiscat.m comes to the rescue. Multiscat.m defines a class called Multiscat and many functions in the class, which generate the file that Multiscat. The details are as follows:

1. suppose that we now have an analytical potential V as a function of $\boldsymbol{R}$ and z, namely $V(\boldsymbol{R}, z)$. Then we need to create a $N \times N \times N_z$ matrix V. The element of matrix V with indices n_i, m_i, and l_i, i.e., V(n_i, m_i, l_i), should be

$$V\left(\frac{n_i}{N}\boldsymbol{a_1} + \frac{m_i}{N}\boldsymbol{a_2}, \frac{l_i}{N_z}(z_{max} - z_{min}) + z_{min}\right).$$

That is not as complicated as it seems. Instead, it is the most intuitive way to sample the potential: just choose points that are evenly distributed in the "box".

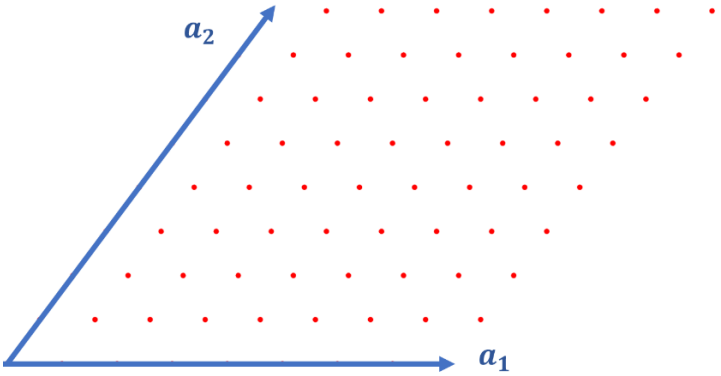

Figure 3. The red dots stand for the points being sampled in one "slice" of z, supposing $N = 8$

2. After getting the matrix V in the workspace in MATLAB (you just need to create it, and it will be there), you should then create a struct with a field called "V", and this field is just the matrix V in your workspace. The code should look like:

```
potStructArray.V=V; %this creates a struct called "potStructArray with a field V, and this field is defined by the matrix V in your MATLAB's Workspace.
```

3. Then, feed the struct into the function PreparePotentialFiles. This function creates pot10001.in. Besides, we also need to feed the matrix V into the function prepareFourierLabels. This creates a file called FourierLabels.in, which is also needed for Multiscat to get the information about the potential. Code:

```
Multiscat.PreparePotentialFiles(potStructArray); %this function creates the pot10001.in file, when executing this line, make sure that Multiscat.m is in your current folder.
Multiscat.prepareFourierLabels(V); %this function creates the FourierLabels.in file, which is also essential for Multiscat to get the information about potential.
```

4. If you take a look at pot10001.in and FourierLabels.in, you will only find some numbers that may not make any sense to you now. They do not contain the information about the base vectors $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$. Nor do they contain information about the z range. The information about them should be in Multiscat.conf. So we need to create this file using Multiscat.m as well. To do that, you still need to append more fields to the struct created in Step 2. You can do that with the code:

```
potStructArray.a1=a1; potStructArray.a2=a2; %make sure you have the base vectors a1 and a2 in your Workspace. The orientation of a1 or a2 does not matter, you just need to get the length and angle between them right. As an example, for LiF (001) surface, a1=[2.84, 0] and a2=[0, 2.84]. Note that the length unit is angstrom.
potStructArray.zmin=z(1);
potStructArray.zmax=z(end);
potStructArray.zPoints=length(z);
%here we have an array called z that stores the values of z being considered. So zmin and zmax are just the first and the last element of z, respectively. The length of z is the number of z points used.
confStruct=Multiscat.createConfigStruct(potStructArray);
Multiscat.prepareConfigFile(confStruct);
%the above two lines create Multiscat.conf
```

5. You can move pot10001.in, FourierLabels.in, and Multiscat.conf into the folder which contains all the other files (you may need to replace the old ones). Now the potential files are ready.

V. (ACTUALLY) RUNNING MULTISCAT

Now we have everything you need, but you may still wonder: how to let Multiscat know the atoms' energy and the direction of the beam? Actually, they are contained in the file called scatCond.in.

If you open it, you can see many lines of numbers. Each line has three numbers separated by commas and each line stands for a scattering event. The first number is the energy of the beam in meV. The second and the third are θ and ϕ in °. The definition of the angles is in Figure 2. Note that $\boldsymbol{a_1}$ is along the x-axis.

Suppose you want to calculate a single configuration. You only need to provide three numbers in scatCond.in. Open the terminal, type in “make” and “./multiscat Multiscat.conf” as you did in section I. Then you should be able to see the file diffrac10001.out afterwards.

VI. AN EXERCISE

This exercise is taken from the paper *Iterative solution in quantum scattering theory-The log derivative Kohn approach* by David E. Manolopoulos, Robert E. Wyatt, and David C. Clary.

Suppose that we are now considering ^{4}He atoms hitting a LiF (001) surface. It is a square lattice with a lattice constant of 2.84 Å. The surface potential V can be approximated as

$$V(x,y,z) = V_0(z) + V_1(z)Q(x,y)$$

with

$$V_0(z) = De^{2\alpha(z_0-z)} - 2De^{\alpha(z_0-z)},$$

and

$$V_1(z) = -2\beta De^{2\alpha(z_0-z)},$$

and

$$Q(x,y) = \cos\left(\frac{2\pi x}{a}\right) + \cos\left(\frac{2\pi y}{a}\right).$$

The constants are $D = 7.63$ meV, $\alpha = 1.1$ Å^{-1}, $z_0 = 1.0$ Å, and $\beta = 0.10$. In this problem, taking the range of z from -2 Å to 6 Å should be enough, and we need only about 100 z points.

Consider a beam of Helium-4 atoms with $E_i = 20$ meV, $\theta = 30°$, and $\phi = 0°$. Feed the potential and the scattering conditions into Multiscat and get the diffrac10001.out file. The diffraction intensities should look like:

diffraction spot	intensity	diffraction spot	intensity
(0,0)	0.025	(-2,±1)	0.083
(-1,0)	0.011	(-1,±2)	0.054
(1,0)	0.028	(-3,0)	0.02
(0,±1)	0.02	(-2,±2)	0.048
(-2,0)	0.029	(-3,±1)	0.029
(0,±2)	0.022	(-4,0)	0.002
(1,±1)	0.101	(-3,±2)	0.008
(-1,±1)	0.077	(-4,±1)	0.001

Solution

The following MATLAB code is used for creating pot10001.in, FourierLabels.in, and Multiscat.conf. Make sure that Multiscat.m is in the same directory when running it.

```
%this function is used to create the potential files.
clear all; close all;
z=linspace(-2,6,100);
%z points, note that the points we take should be evenly spaced
a=2.84;%lattice constant
a1=[a, 0]; a2=[0, a];%base vectors in real space
[b1,b2]=Reciprocal(a1,a2);%reciprocal base lattice vectors
gridp=32;%grid points parallel to the surface in one dimension
i1=1:gridp; i2=1:gridp;%point grids in a1 and a2 directions
```

```
V=zeros(length(i1),length(i2),length(z));%potential matrix

D=7.63; alpha=1.1; z0=1.0;%the constants used for generating potential

V0=zeros(size(z));
%now assign values to V0
V0=D*exp(2*alpha*(z0-z))-2*D*exp(alpha*(z0-z));
V0=repmat(reshape(V0,1,1,[]),size(V,1),size(V,2),1);
%repeat matrix V0 to prepare to add it to V
V=V+V0;

for ia1=1:length(i1)
   for ia2=1:length(i2)
      X(ia1,ia2)=(a1(1)*ia1+a2(1)*ia2)./gridp;%the x grid points
      Y(ia1,ia2)=(a1(2)*ia1+a2(2)*ia2)./gridp;%the y grid points
   end
end

beta=0.10;%the constant used to create V1
V1=-2*beta*D*exp(2*alpha*(z0-z));
Q=zeros(size(X));
for ia1=1:length(i1)
   for ia2=1:length(i2)
      Q(ia1,ia2)=cos(2*pi*X(ia1,ia2)/a)+cos(2*pi*Y(ia1,ia2)/a);
      for indz=1:length(z)
         V(ia1,ia2,indz)=V(ia1,ia2,indz)+V1(indz)*Q(ia1,ia2);
         %add V1*Q to V
      end
   end
end

%now we have got the matrix V, it's time to create the files used by
%Multiscat.
potStructArray.V=V;
%this creates a struct called "potStructArray with a field V,
%and this field is defined by the matrix V in your MATLAB's Workspace.

Multiscat.PreparePotentialFiles(potStructArray);
%this function creates the pot10001.in file, when executing this line,
%make sure that Multiscat.m is in your current folder.

Multiscat.prepareFourierLabels(V);
%this function creates the FourierLabels.in file, which is also essential
```

```
%for Multiscat to get the information about potential.

potStructArray.a1=a1; potStructArray.a2=a2;
%make sure you have the base vectors a1 and a2 in your Workspace.
%The orientation of a1 or a2 does not matter, you just need to get
%the length and angle between them right. As an example, for
%LiF (001) surface, a1=[2.84, 0] and a2=[0, 2.84].
%Note that the length unit is angstrom.

potStructArray.zmin=z(1);
potStructArray.zmax=z(end);
potStructArray.zPoints=length(z);
%here we have an array called z that stores the values of z being
%considered. So zmin and zmax are just the first and the last element
%of z, respectively. The length of z is the number of z points used.

confStruct=Multiscat.createConfigStruct(potStructArray);
Multiscat.prepareConfigFile(confStruct);
%the above two lines create Multiscat.conf

%this function calculates the reciprocal lattice vectors b1 and b2
%from the real space basis vectors a1 and a2
function [b1,b2]=Reciprocal(a1,a2)
   a3=[0 0 1];
   crsprod=a3*(cross([a1 0],[a2 0]))';
   %the volume spanned by a1, a2, and a3
   b1=2*pi*cross([a2 0],a3)/crsprod;
   b1=b1(1:2);
   b2=2*pi*cross(a3,[a1 0])/crsprod;
   b2=b2(1:2);
end
```

After these files are created, move them to the folder that contains other files of Multiscat. You can then open scatCond.in and write “20, 30, 0” into the second line (you should not delete the comments in the first line as Multiscat will ignore the first line and read input parameters from the second line).

Run Multiscat as you did in section I and V. You should get the correct results.

VII. OTHER THINGS TO BE NOTED

1. Segmentation fault

If you run Multiscat but the terminal returns “Segmentation fault”, you probably should enlarge the memory of your virtual machine. When the memory is less than 8 GB, such problems happen.

2. Recompiling

If you want to change something in the program, make sure you type “make” into the terminal before rerunning it. This command recompiles the program.

3. Units used in Multiscat

The input parameters of Multiscat are in angstrom, meV, and degree. And we use the atomic mass unit, which makes helium-4 have a mass of 4 and helium-3 3.

4. How to change the helium atom’s mass?

The mass of the helium atoms can either be 3 or 4. Multiscat gets the piece of information from the last line of Multiscat.conf. You can modify Multiscat.conf directly or change Multiscat.m. In Multiscat.m the struct confStruct has a file called incidentParticleMass. You just need to change that.

5. Convergence

Multiscat uses the close-coupling method and generalized minimal residual method (GMRES) to calculate. That means that the scattering calculation's accuracy depends on the number of iterations and how complicated the potential is.

The potential used in the exercise is very simple, containing only a few Fourier components. So it is easier to converge, and it won’t take too long to calculate. But for more complicated potentials, it might be harder to converge.

To see whether a calculation converges, you can rely on two standards. First, the sum of all the diffraction intensities should be 1 or 0.99999… The sum of all the diffraction intensities should be displayed in the terminal, so you won’t need to calculate yourself.

Second, try increasing the sampling points in the “box”, namely, increasing N or N_z, and see whether the results vary significantly. If not, then you should have reached convergence. Do the same to the range of z, i.e., increasing z_{max} and decreasing z_{min}. Also, try increasing dmax in Multiscat.conf (same as the mass of helium atoms, you can either modify Multiscat.conf straight away or modify Multiscat.m and generate a new Multiscat.conf file). If you have done all of them and the diffraction intensities do not change much, you will likely have converging results.

6.Hard limits

You may notice the file multiscat.inc. It puts hard limits on the parameters Multiscat uses. You should find there that the number of z grid points should not exceed 550. And the

number of potential Fourier components should not be over 4096. This just means that N should satisfy $N \leq \sqrt{4096} = 64$. If you break these hard limits, you can get problematic results or may not be able to run the program normally.

Of course, you can always change the numbers in multiscat.inc to raise the hard limits if you have enough time and computing power.

7. Common blocks

There may be a problem with Multiscat if you use an older version of it. You need to be cautious that, in Multiscat.f90, potsub.f90, and scatsub.f, there is a common block called const. You will see code like “common /const/ hemass” or “common /const/ rmlmda”. If you see them, delete them all, and replace them with “common /const/ hemass, rmlmda”. Ensure that in each subroutine, if hemass or rmlmda is used, they should be de called by “common /const/ hemass, rmlmda”.

The reason is that, according to the current version of Fortran, if we define hemass and rmlmda in the common block const, (the code should look like:)

```
common /const/ hemass
common /const/ rmlmda
hemass=4.0 !the mass of helium-4
rmlmda=2.0*hemass/hbarsq !hbarsq stand for the square of hbar. In this case it
should be 1.9
```

there will be only two numbers, 4.0 and 1.9, stored in const. The name of the two variables will NOT be recorded. If another file uses it by “common /const/ rmlmda” without referring to hemass, rmlmda will take the the first value in the common block const, which is 4.0.

Note that in scatsub.f the common block const is used twice in different subroutines, and in potsub.f90 it is used once. You need to make sure in all three places they are used correctly.

If you do not modify the code accordingly, Multiscat can still give quite reasonable results, but they will not be correct.