ASSIGNMENT 3: SOLUTION

E. ARNOLD, M-S. LIU, R. PRABHU & C.S. RICHARDS
THE UNIVERSITY OF CAMBRIDGE

Written in XeLaTeX

HARD WALL POTENTIALS

PREREQUISITES

The chapter *Surface Scattering* from the *Theory Handbook*

INTRODUCTION

The scattering of helium atoms from a surface is one of the key processes used to probe surface structures. For this reason, it is important to understand how this scattering occurs, and the consequence of atom scattering. To achieve this, one of the tasks we must achieve is to construct a "hard wall potential" for a surface. In this tutorial, we will develop a model for the "hard wall potential" on a close packed plane.

HARD WALL POTENTIALS

In order to analyse the results from scattering experiments, it is useful to have a theoretical model that supports the experiment. This model must predict the scattering dynamics for a given surface. How do we make this model? The first step in constructing this is to generate a potential energy function that describes the potential on the same surface.

The simplest model for this potential energy function that we can use is known as a "hard-wall potential". Covered comprehensively in the *Theory Handbook*, this takes the form:

$$V(\mathbf{R}, z) = \begin{cases} 0 & z > \zeta(\mathbf{R}); \\ \infty & z \leq \zeta(\mathbf{R}), \end{cases}$$

where V is the potential energy function, $\mathbf{R}$ is the position vector of a point on the surface, z is the perpendicular distance of a point from the surface (see Figure 1), and $\zeta(\mathbf{R})$ is the "corrugation function".

Within our model, it can be seen that the points given by $(\mathbf{R}, \zeta(\mathbf{R}))$ define a locus. What does this represent? This locus represents the points at which the incident atom will scatter, and begin to move away from the surface. This is reassuring - it is essentially the definition of

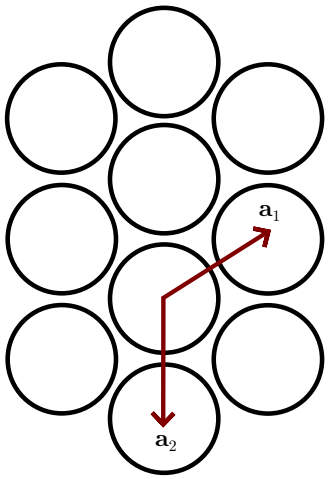

Figure 1 | The coordinate system used to define the potential function. The (001) family is set as the xy plane, and the [001] axis (not labelled) is denoted as the z axis, perpendicular to the surface. The circles are illustrative only - the true lattice has a dirac delta function located at the centre of each circle present.

the corrugation function. What can we say about the corrugation function's behaviour? As a starting point, we would expect the corrugation function to have two key features:

- Local maxima at the position of the surface atoms.
- Periodicity - respecting the symmetry of the lattice.

This information is sufficient to generate a corrugation function for a surface. In this tutorial, we will use Fourier series to construct a corrugation function for a close packed plane, satisfying the properties above. We will make use of the result that a function whose periodicity matches that of a Bravais lattice must take the form:

$$f(\mathbf{r}) = \sum_G A_G e^{i\mathbf{G}\cdot\mathbf{r}}$$

where $\mathbf{G}$ is a reciprocal lattice vector, and A_G is a constant. Where does this equation come from? Although it initially appears to be plucked out of thin air, it does make sense: whenever the argument $\mathbf{r}$ is a multiple of a real space lattice vector, the argument of the exponential is a multiple of $2\pi i$. Evidently, this results in the function being periodic.

How do we approach the problem of constructing a corrugation function? Our strategy to begin is:

1. Determine the lattice that represents the surface.
2. Find the set of basis lattice vectors for the lattice in question, denoted $\mathbf{a_1}$ and $\mathbf{a_2}$.
3. Find the corresponding reciprocal lattice vectors, denoted $\mathbf{b_1}$ and $\mathbf{b_2}$.
4. Use the reciprocal lattice basis vectors to find reciprocal lattice vectors of the form:
$$\mathbf{G} = h\mathbf{b_1} + k\mathbf{b_2},$$
where h and k are integers.
5. Use these reciprocal lattice vectors to construct a Fourier series for the corrugation function. Ensure that the coefficients of the terms in the Fourier series are chosen such that the function has local maxima occuring at the positions of the atoms.

In the case of a hexagonal lattice, much of this work has already been done for us! In the Fourier transforms tutorial, you found the reciprocal lattice basis vectors, denoted $\mathbf{b_1}$ and $\mathbf{b_2}$, and you used these to plot the reciprocal lattice. We also need to choose the arguments of the terms in the Fourier series. Any reciprocal lattice vector $\mathbf{G}$ is a suitable choice for the basis of the Fourier series. Figure 2 highlights the basis vectors of a close packed plane, naturally suggesting an elegant basis for the Fourier series.

The simplest possible courrugation function would be one that just uses the two basis reciprocal lattice vectors, $\mathbf{b_1}$ and $\mathbf{b_2}$, to give a

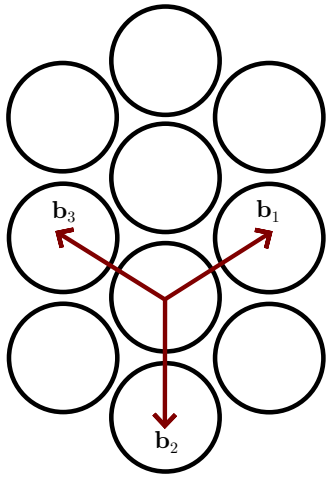

Figure 2 | The reciprocal lattice of a close packed plane. This is also a close packed plane! However, the spacing between the lattice points changes.

Fourier series with just two components. However, we will demonstrate that while this corrugation function would respect the periodicity of the lattice, it would not fully respect the hexagonal symmetry present. To represent this symmetry fully, we would need to use at least three reciprocal lattice vectors, such as $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$, in order to produce a suitable corrugation function.

TASK 1: TWO FOURIER COMPONENTS

Our first task is to construct a corrugation function for a close-packed plane, using just two Fourier components.

1. Find the reciprocal lattice basis vectors for the hexagonal surface.
2. Construct a Fourier series for the corrugation function using just two Fourier components: using one series for each reciprocal lattice basis vector. Ensure that the function has its maxima at the positions of the atoms (*i.e.* the lattice points). Use a lattice constant of $a = 4.52$Å and a corrugation height (*i.e.* the maximum value of the corrugation function) of $h = 1$Å.
3. Produce a suitable plot. Convince yourself that the function does not fully respect the hexagonal symmetry of the system.

Tip 1

The lattice of a hexagonal surface was explored in the tutorial on graph plotting; the lattice vectors used here are the same as from that tutorial. The key result is that the two basis vectors are the same length a, and have an angle of 120° between them.

There is a formula for finding reciprocal lattice basis vectors from real space lattice basis vectors. This can be found in numerous sources. In addition, both the lattice vectors and the reciprocal lattice vectors can be calculated for you by the "auxCls" function, given in the solution.

Once you have calculated the reciprocal lattice vectors, $\mathbf{d_1}$ and $\mathbf{d_2}$, the corrugation function is given by:

$$\zeta(\mathbf{R}) = A(\cos(\mathbf{R} \cdot \mathbf{d_1}) + \cos(\mathbf{R} \cdot \mathbf{d_2}))$$

where A must be chosen such that you get the correct corrugation height.

Note that we have used only cosine functions in the Fourier series so that the corrugation function peaks at the position of the top sites, as required.

Tip 2

When plotting a function on MATLAB, you need to sample the value of the function a set of points in space, and then plot your sampled values as a function of position. The points in space where you sample your function should take the form of a grid that respects the periodicity of the system.

In this case, the corrugation function that you need to plot is a function of two spatial dimensions, and so the function defines a surface in three-dimensional space (rather than a line in two-dimensional space that you would plot for functions of only one spatial dimension). To plot such a function, try the "`surface`" and/or "`pcolor`" functions in Matlab.

TASK 2: THREE FOURIER COMPONENTS

You may have noticed that the function plotted in the previous task did not respect the mirror symmetry of the close-packed surface. We are now going to construct a corrugation function that does fully respect the hexagonal symmetry of the system.

To do this, you will need to find three reciprocal lattice vectors. The three vectors labelled in Figure 2 are a suitable choice. Then you will have to construct a Fourier series as before, but this time with three Fourier components, one for each reciprocal lattice vector. You still need to ensure that the function has its maxima at the positions of the atoms. Use the same lattice constant and corrugation height as before, and produce a suitable plot. This time, you should see that your function does respect the hexagonal symmetry of the system.

Tip

We have already calculated $\mathbf{b_1}$ and $\mathbf{b_2}$ using the "`auxCls`" function. You should be able to see from Figure 2 that the third reciprocal lattice vector is given by:

$$\mathbf{b_3} = \mathbf{b_2} - \mathbf{b_1}.$$

SUMMARY

In this document, we have learned how to create a hard wall potential for a simple system. This skill will remain useful throughout your study of surface physics.

SOLUTION

TASK 1: QUESTION

Our first task is to construct a corrugation function for a close-packed plane, using just two Fourier components.

1. Find the reciprocal lattice basis vectors for the hexagonal surface.
2. Construct a Fourier series for the corrugation function using just two Fourier components: using one series for each reciprocal lattice basis vector. Ensure that the function has its maxima at the positions of the atoms (*i.e.* the lattice points). Use a lattice constant of $a = 4.52$Å and a corrugation height (*i.e.* the maximum value of the corrugation function) of $h = 1$Å.
3. Produce a suitable plot. Convince yourself that the function does not fully respect the hexagonal symmetry of the system.

TASK 1: SOLUTION

For this part of the tutorial, we just had to construct a Fourier series using the conventional reciprocal lattice basis vectors of a hexagonal surface. We can find these using the `auxCls` function from Tutorial 1.

```
%-------------------------------------------------------------------
% Initialise variables
%-------------------------------------------------------------------
% corrugation height in Angstroms
corr = 1;

% lattice constant
a = 4.52;
%-------------------------------------------------------------------
%-------------------------------------------------------------------


%-------------------------------------------------------------------
% Finding the lattice vectors using the auxCls function
%-------------------------------------------------------------------
% No rotation of the lattice vectors is required so RotM = 0
RotM = 0;

% a1, a2: lattice vectors;
% b1, b2: reciprocal lattice vectors
[a1, a2, b1, b2] = auxCls.unitCellVecs(1,a,RotM);
%-------------------------------------------------------------------
```

```
23 %--------------------------------------------------------------------
```

We then need create a grid to sample the corrugation function on.

```
24 %--------------------------------------------------------------------
25 % Grid a cell to sample and plot corrugation function (same as
       Tutorial on FT)
26 %--------------------------------------------------------------------
27 % Number of cells per dimension
28 % ie there are numcells of the unit cell along the domain being
       used in each lattice vector
29 numCells = 2;
30
31 % Define domain of x
32 Lx = numCells*[-a/2 a/2];
33
34 % Do the same for y
35 Ly = numCells*[-norm(a1+2*a2)/2 norm(a1+2*a2)/2];
36
37 % Turn the definition of the domain into a useable, uniform, set
       of points
38 lx = linspace(Lx(1), Lx(2), 50 * numCells);
39
40 % Delete the last entry in the domain
41 lx(end) = [];
42
43 % Turn the definition of the domain into a useable, uniform, set
       of points
44 ly = linspace(Ly(1), Ly(2), 30 * numCells);
45
46 % Delete the last entry in the domain
47 ly(end) = [];
48
49 % generate the x and y vectors, to generate all points on the xy
       plane to sample
50 [x,y] = meshgrid(lx, ly);
51
52 % Positions at which potential is sampled
53 R = [x(:) y(:)];
54 %--------------------------------------------------------------------
55 %--------------------------------------------------------------------
```

We can then sample and plot the potential:

```
56 %--------------------------------------------------------------------
57 % Sampling the potential
58 %--------------------------------------------------------------------
59 % Gives value of zeta at each position in grid as column vector
60 % Recall ' denotes transpose
61 zeta_1 = sample_zeta(R, corr, b1', b2');
62
63 %Making zeta the same shape as the grid
64 zeta_1 = reshape( zeta_1, size(x) );
65 %--------------------------------------------------------------------
66 %--------------------------------------------------------------------
67
68
69
70 %--------------------------------------------------------------------
71 % Plotting the potential
72 %--------------------------------------------------------------------
73 % initialise figure
```

```
74  fig1 = figure;
75
76  % setup the subplots: plot first figure
77  subplot(1, 2, 1);
78
79  % hold on to plot both figures
80  hold on
81
82  % plot the surface
83  surface(x, y, zeta_1);
84
85  % specify the colouring
86  shading flat;
87
88  % specify the orientation
89  view(75, 75)
90
91  % setup the subplots: plot second figure
92  subplot(1, 2, 2);
93
94  % hold on
95  hold on;
96
97  % Plot 2D representation of potential
98  % pcolor( C ) creates a pseudocolor plot using the values in
        matrix C
99  pcolor(x, y, zeta_1);
100
101 % specify the colouring
102 shading flat;
103
104 % sets the aspect ratio so that the data units are the same in
        every direction
105 axis equal
106 %-----------------------------------------------------------------
107 %-----------------------------------------------------------------
```

Our task is essentially complete. To see why this potential doesn't respect the symmetry of the system, we will also plot the high-symmetry sites, using the functions provided in the first tutorial.

```
108 %-----------------------------------------------------------------
109 % Plot the top sites, bridge sites and hollow sites
110 %-----------------------------------------------------------------
111 % find the top sites
112 top_sites = auxCls.spanVecs(a1, a2, -1:1, -1:1);
113
114 % find the bridge sites and hollow sites
115 [bridge_sites,hollow_sites] = construct_bridge_hollow(top_sites,
        a1, a2);
116
117 % plot all features
118 plot_unit_cell_sites(top_sites, bridge_sites, hollow_sites);
119
120 % set the limits on the axes
121 % axis([xmin xmax ymin ymax zmin zmax])
122 axis([-5 5 -5 5])
123 %-----------------------------------------------------------------
124 %-----------------------------------------------------------------
```

To generate the corrugation function, we can use the function below.

In this function, `R` is the set of points at which the corrugation function is sampled, `h` is the corrugation height, and `G1` and `G2` are the two reciprocal lattice vectors (which will be used as the frequencies of the two Fourier components)

```
125 %-----------------------------------------------------------------
126 % Make a function that produces a corrugation function
127 %-----------------------------------------------------------------
128 % zeta is the symbol used for the corrugation function
129 function [zeta] = sample_zeta_2(R, h, G1, G2, G3)
130
131 % Function that produces corrugation function with three Fourier
        components
132 % Inputs:
133 %   - R: positions to sample (Nx2 matrix, where N is number of
        positions)
134 %   - h: corrugation height (scalar)
135 %   - G1, G2, G3, : recpiprocal lattice vectors (column vectors)
136
137 % Output:
138 %   - zeta: corrugation function at each position (Nx1 matrix)
139
140
141 % Unscaled Fourier series
142 zeta_0 = cos(R * G1) + cos(R * G2) + cos(R * G3);
143
144 % So that zeta is always greater than 0
145 zeta_0 = zeta_0 - min( min(zeta_0) );
146
147 % Giving that zeta has max value of 1
148 zeta_0 = zeta_0 / max( max(zeta_0) );
149
150 %Giving zeta a max value of h
151 zeta = h * zeta_0;
152 end
153 %-----------------------------------------------------------------
154 %-----------------------------------------------------------------
```

The other functions used in the script are:

```
 1 %-----------------------------------------------------------------
 2 % plot_unit_cell_sites function
 3 %-----------------------------------------------------------------
 4
 5 % plots the unit cell onto the figure
 6 function plot_unit_cell_sites(top_sites, bridge_sites,
       hollow_sites)
 7
 8     hold on
 9     plot(     top_sites(:,1),     top_sites(:,2),'o','
           MarkerFaceColor','y',"MarkerSize",15);
10     plot(bridge_sites{1}(:,1),bridge_sites{1}(:,2),'o','
           MarkerFaceColor','b',"MarkerSize",15);
11     plot(bridge_sites{2}(:,1),bridge_sites{2}(:,2),'s','
           MarkerFaceColor','b',"MarkerSize",15);
12     plot(bridge_sites{3}(:,1),bridge_sites{3}(:,2),'d','
           MarkerFaceColor','b',"MarkerSize",15);
13     plot(hollow_sites{1}(:,1),hollow_sites{1}(:,2),'v','
           MarkerFaceColor','g',"MarkerSize",15);
14     plot(hollow_sites{2}(:,1),hollow_sites{2}(:,2),'^','
           MarkerFaceColor','g',"MarkerSize",15);
```

```
15
16      % Name the sites
17      [hleg1, ~] = legend(...
18          {'Top Sites','Bridge Sites','Bridge Sites',...
19          'Bridge Sites','Hollow Sites','Hollow Sites'});
20      hleg1.Position(2) = 0.2; % [Left Bottom Width Height]
21      hleg1.Position(4) = 0.6;
22  end
23  %--------------------------------------------------------------------
24  %--------------------------------------------------------------------
25
26
27
28
29  %--------------------------------------------------------------------
30  % construct_bridge_hollow function
31  %--------------------------------------------------------------------
32  function [bridge_sites, hollow_sites] = construct_bridge_hollow(
        top_sites, a1, a2)
33      % If required, fix the dimensions
34      if iscolumn(a1), a1 = a1'; end
35      if iscolumn(a2), a2 = a2'; end
36      if size(top_sites,2) ~= 2, top_sites = top_sites'; end
37
38      bridge_sites(1) = {top_sites + a1/2};
39      bridge_sites(2) = {top_sites + a1+a2/2};
40      bridge_sites(3) = {top_sites + a1/2+a2/2};
41      hollow_sites(1) = {top_sites + (2/3)*(2*a1+a2)/2};
42      hollow_sites(2) = {top_sites + (2/3)*(a2+0.5*a1)};
43  end
44  %--------------------------------------------------------------------
45  %--------------------------------------------------------------------
```

Examining the plots in the margin makes it evident that the hexagonal symmetry does not appear in our output.

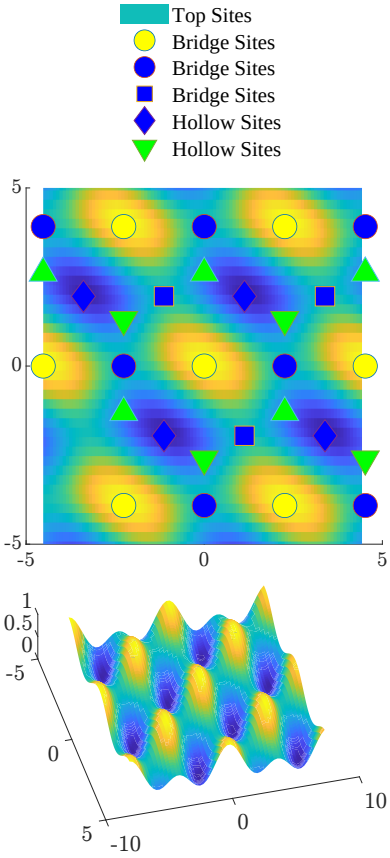

Figure 3 | The plots produced for Task 1. These are crude representations of the corrugation function.

TASK 2: QUESTION

You may have noticed that the function plotted in the previous task did not respect the mirror symmetry of the close-packed surface. We are now going to construct a corrugation function that does fully respect the hexagonal symmetry of the system.

To do this, you will need to find three reciprocal lattice vectors. The three vectors labelled in Figure 2 are a suitable choice. Then you will have to construct a Fourier series as before, but this time with three Fourier components, one for each reciprocal lattice vector. You still need to ensure that the function has its maxima at the positions of the atoms. Use the same lattice constant and corrugation height as before, and produce a suitable plot. This time, you should see that your function does respect the hexagonal symmetry of the system.

TASK 2: SOLUTION

For this part of the tutorial, we can construct a Fourier series using the two reciprocal lattice vectors found in Task 1, plus a third reciprocal lattice vector, which had to be chosen in such a way as to respect the hexagonal symmetry of the system. The third vector that we have used here is drawn in Figure 2 of the tutorial, and is given by $\mathbf{b_3} = \mathbf{b_2} - \mathbf{b_1}$.

Therefore, the code for this task is almost unchanged, except that the corrugation function now has to include this third Fourier component. The remainder of the complete code is included below:

```
%-------------------------------------------------------------------
% Initialise variables
%-------------------------------------------------------------------
% corrugation height in Angstroms
corr = 1;

% lattice constant
a = 4.52;
%-------------------------------------------------------------------
%-------------------------------------------------------------------




%-------------------------------------------------------------------
% Finding the lattice vectors using the auxCls function
%-------------------------------------------------------------------
% No rotation of the lattice vectors is required so RotM = 0
RotM = 0;

% a1, a2: lattice vectors;
% b1, b2: reciprocal lattice vectors
[a1, a2, b1, b2] = auxCls.unitCellVecs(1,a,RotM);

%Finding b3
b3 = b2 - b1;
%-------------------------------------------------------------------
%-------------------------------------------------------------------




%-------------------------------------------------------------------
% Grid a cell to sample and plot potential (same as Tutorial on
    FT)
%-------------------------------------------------------------------
% Number of cells per dimension
% ie there are numcells of the unit cell along the domain being
    used in each lattice vector
numCells = 2;

% Define domain of x
Lx = numCells * [-a/2 a/2];

% Do the same for y
Ly = numCells * [-norm(a1+2 * a2)/2 norm(a1+2 * a2)/2];
```

```
45
46  % Turn the definition of the domain into a useable, uniform, set
        of points
47  lx = linspace(Lx(1), Lx(2), 50 * numCells);
48
49  % Delete the last entry in the domain
50  lx(end) = [];
51
52  % Turn the definition of the domain into a useable, uniform, set
        of points
53  ly = linspace(Ly(1), Ly(2), 30 * numCells);
54
55  % Delete the last entry in the domain
56  ly(end) = [];
57
58  % generate the x and y vectors, to generate all points on the xy
        plane to sample
59  [x,y] = meshgrid(lx, ly);
60  %------------------------------------------------------------------
61  %------------------------------------------------------------------
62
63
64
65
66  %------------------------------------------------------------------
67  % Sampling the potential
68  %------------------------------------------------------------------
69  % Positions at which potential is sampled taken from grid
70  R = [x(:) y(:)];
71
72  % Gives value of zeta at each position in grid as column vector
73  zeta_1 = sample_zeta_2(R, corr, b1', b2', b3');
74
75  % Making zeta the same shape as the grid
76  zeta_1 = reshape(zeta_1, size(x));
77  %------------------------------------------------------------------
78  %------------------------------------------------------------------
79
80
81
82
83  %------------------------------------------------------------------
84  % Plotting the potential
85  %------------------------------------------------------------------
86  % initialise figure
87  fig1 = figure;
88
89  % setup the subplots: plot first figure
90  subplot(1, 2, 1);
91
92  % hold on to plot both figures
93  hold on;
94
95  % plot the surface
96  surface(x, y, zeta_1);
97
98  % specify the colouring
99  shading flat;
100
101 % specify the orientation
102 view(75, 75)
```

```
103
104 % setup the subplots: plot second figure
105 subplot(1, 2, 2);
106
107 % hold on
108 hold on;
109
110 % Plot 2D representation of potential
111 % pcolor( C ) creates a pseudocolor plot using the values in
        matrix C
112 pcolor(x, y, zeta_1);
113
114 % specify the colouring
115 shading flat;
116
117 % sets the aspect ratio so that the data units are the same in
        every direction
118 axis equal
119 %-----------------------------------------------------------------
120 %-----------------------------------------------------------------
121
122
123
124
125 %-----------------------------------------------------------------
126 % Plot the top sites, bridge sites and hollow sites
127 %-----------------------------------------------------------------
128 % find the top sites
129 top_sites = auxCls.spanVecs(a1, a2, -1:1, -1:1);
130
131 % find the bridge sites and hollow sites
132 [bridge_sites, hollow_sites] = construct_bridge_hollow(top_sites
        , a1, a2);
133
134 % plot all features
135 plot_unit_cell_sites(top_sites, bridge_sites, hollow_sites);
136
137 % set the limits on the axes
138 % axis([xmin xmax ymin ymax zmin zmax])
139 axis-([5 5 -5 5])
140 %-----------------------------------------------------------------
141 %-----------------------------------------------------------------
```

The new function is:

```
142 %-----------------------------------------------------------------
143 % Define the function to generate the corrugation function
144 %-----------------------------------------------------------------
145 % zeta is the symbol for the corrugation function
146 function [zeta] = sample_zeta_2(R,h,G1,G2,G3)
147
148 % Function that produces corrugation function with three Fourier
        components
149 % Inputs:
150 %   - R: positions to sample (Nx2 matrix, where N is number of
        positions)
151 %   - h: corrugation height (scalar)
152 %   - G1, G2, G3, : recpiprocal lattice vectors (column vectors)
153
154 % Output:
155 %   - zeta: corrugation function at each position (Nx1 matrix)
156
```

```
157
158 % Unscaled Fourier series
159 zeta_0 = cos(R * G1) + cos(R * G2) + cos (R * G3);
160
161 % So that zeta is always greater than 0
162 zeta_0 = zeta_0 - min( min(zeta_0) );
163
164 % Giving that zeta has max value of 1
165 zeta_0 = zeta_0/max( max(zeta_0) );
166
167 % Giving zeta a max value of h
168 zeta = h * zeta_0;
169 end
170 %----------------------------------------------------------------
171 %----------------------------------------------------------------
```

The other functions used in the script are:

```
172 %----------------------------------------------------------------
173 % plot_unit_cell_sites function
174 %----------------------------------------------------------------
175
176 % plots the unit cell onto the figure
177 function plot_unit_cell_sites(top_sites, bridge_sites,
        hollow_sites)
178
179     hold on
180     plot(     top_sites(:,1),     top_sites(:,2),'o','
            MarkerFaceColor','y',"MarkerSize",15);
181     plot(bridge_sites{1}(:,1),bridge_sites{1}(:,2),'o','
            MarkerFaceColor','b',"MarkerSize",15);
182     plot(bridge_sites{2}(:,1),bridge_sites{2}(:,2),'s','
            MarkerFaceColor','b',"MarkerSize",15);
183     plot(bridge_sites{3}(:,1),bridge_sites{3}(:,2),'d','
            MarkerFaceColor','b',"MarkerSize",15);
184     plot(hollow_sites{1}(:,1),hollow_sites{1}(:,2),'v','
            MarkerFaceColor','g',"MarkerSize",15);
185     plot(hollow_sites{2}(:,1),hollow_sites{2}(:,2),'^','
            MarkerFaceColor','g',"MarkerSize",15);
186
187     % Name the sites
188     [hleg1, ~] = legend(...
189         {'Top Sites','Bridge Sites','Bridge Sites',...
190         'Bridge Sites','Hollow Sites','Hollow Sites'});
191     hleg1.Position(2) = 0.2; % [Left Bottom Width Height]
192     hleg1.Position(4) = 0.6;
193 end
194 %----------------------------------------------------------------
195 %----------------------------------------------------------------
196
197
198
199
200 %----------------------------------------------------------------
201 % construct_bridge_hollow function
202 %----------------------------------------------------------------
203 function [bridge_sites, hollow_sites] = construct_bridge_hollow(
        top_sites, a1, a2)
204     % If required, fix the dimensions
205     if iscolumn(a1), a1 = a1'; end
206     if iscolumn(a2), a2 = a2'; end
207     if size(top_sites,2) ~= 2, top_sites = top_sites'; end
```

```
208
209     bridge_sites(1) = {top_sites + a1/2};
210     bridge_sites(2) = {top_sites + a1+a2/2};
211     bridge_sites(3) = {top_sites + a1/2+a2/2};
212     hollow_sites(1) = {top_sites + (2/3)*(2*a1+a2)/2};
213     hollow_sites(2) = {top_sites + (2/3)*(a2+0.5*a1)};
214 end
215 %-------------------------------------------------------------------------
216 %-------------------------------------------------------------------------
```

You should note from these plots that using three Fourier components does indeed produce a corrugation function that respects the symmetry of the system. However, you can see that there is still information in our system that cannot be represented by our potential. For example, the function treats all three of the different types of bridge sites as being equivalent to each other, and similarly for the two different types of hollow site. A function that is sensitive to these features of the surface would require even more Fourier components.

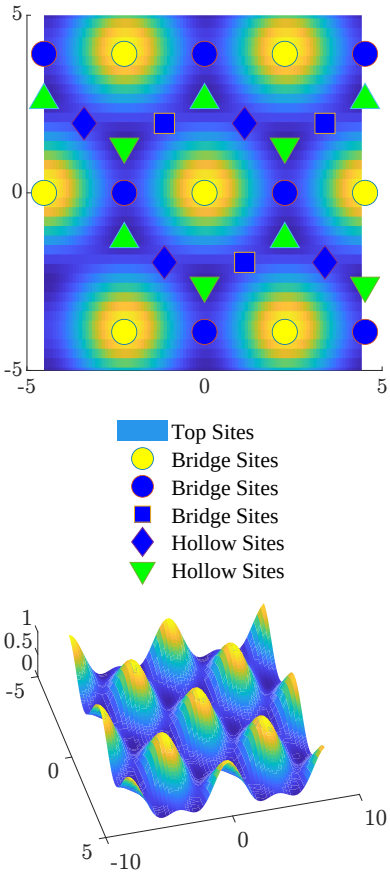

Figure 4 | The plots produced for Task 2

APPENDICES

The listing for the omnipresent `auxCls` function is given below.

```
1  %-------------------------------------------------------------------------
2  % The auxCls class listing
3  % Contains:
4  % - unitCellVecs
5  % - real2recip
6  % - sampleUnitCell
7  % - spanVecs
8  %-------------------------------------------------------------------------
9  classdef auxCls
10     properties(Constant)
11     % no special properties to define
12     end
13
14     % define the functions, one by one
15     methods(Static)
16
17         %-----------------------------------------------------------
18         % first function: unitCellVecs
19         %-----------------------------------------------------------
20         function [a1, a2, b1, b2] = unitCellVecs(surfaceType,
               lttcCnst, rotation)
21             % UNITCELLVECS creates the vectors of a unit cell (
                   Real & Reciprocal space)
22             % Currently, two types of symmetries are supported,
                   hexagonal and fcc. Also. tt is possible to
                   define rotation of the unit cell.
23             % Input:   surfaceType - 1-hexagonal 2-fcc
24             %         lttcCnst - Lattice constant
25             %         rotation - either a 2x2 rotational Matrix,
                    or the rotation in degrees.
26             % Output: {a1,a2}  - real space vectors
27             %         {b1,b2}  - reciprocal vectors
28
29             % This section determines what the surface type is.
                   This is used to structure the real space vectors
```

```
                accordingly
30
31              % Substrate vector
32              if length(surfaceType) == 3
33                  angle=surfaceType(3);
34                  Rmat = [cosd(angle) sind(angle); -sind(angle)
                        cosd(angle)];
35                  a1s = [1 0]*surfaceType(1);
36                  a2s = [Rmat*[1 0]']'*surfaceType(2);
37
38              % if surface type is 1, then the surface is
                    hexagonal. Basis vectors are (orienting one
                    along x): x + 0y and -0.5x + root3 / 2 y
39              elseif surfaceType == 1
40
41                  % first vector: x + 0y
42                  a1s=[1 0]*lttcCnst;
43
44                  %second vector: -0.5x + root3 / 2 y
45                  a2s=[-0.5 sqrt(3)/2]*lttcCnst;
46
47              % if surface type is 2, then the surface is fcc(100)
                    . This is a square lattice! so vectors are x + 0
                    y and 0x + y
48              elseif surfaceType == 2
49
50                  % first vector is x + 0y, magnitude lttcCnst
51                  a1s=[1 0]*lttcCnst;
52
53                  % second vector is 0x + y, magnitude lttcCnst
54                  a2s=[0 1]*lttcCnst;
55
56              % end set of if statements
57              end
58
59
60
61              % This section rotates the unit cell vectors
62
63              % assume rotation angle in degrees
64              if length(rotation)==1,
65                  Rmat = [cosd(rotation) sind(rotation); -sind(
                        rotation) cosd(rotation)];
66                  a1 = [Rmat*a1s']';
67                  a2 = [Rmat*a2s']';
68              else
69                  % Adsorbate vectors
70                  a1 = rotation(1,1)*a1s + rotation(1,2)*a2s;
71                  a2 = rotation(2,1)*a1s + rotation(2,2)*a2s;
72              end
73
74
75              %find reciprocal vectors
76              b1=2*pi*cross([a2 0],[0 0 1])/([a1 0]*[cross([a2
                    0],[0 0 1])]');
77              b1=b1(1:2);
78
79
80              b2=2*pi*cross([0 0 1],[a1 0])/([a2 0]*[cross([0 0
                    1],[a1 0])]');
81              b2=b2(1:2);
```

```
82          end
83          %-----------------------------------------------------------
84          %-----------------------------------------------------------
85
86
87
88
89
90          %-----------------------------------------------------------
91          % real2recip function
92          %-----------------------------------------------------------
93          % finds the reciprocal vectors
94          function [d1,d2]=real2recip(c1, c2, toRecip)
95              if toRecip
96                  d2=2*pi*cross([0 0 1],[c1 0])/norm(cross([c1
                        0],[c2 0]));
97                  d2=d2(1:2);
98                  d1=2*pi*cross([c2 0],[0 0 1])/norm(cross([c1
                        0],[c2 0]));
99                  d1=d1(1:2);
100             else
101                 disp('real2recip: not implemented yet ...')
102             end
103         end
104         %-----------------------------------------------------------
105         %-----------------------------------------------------------
106
107
108
109         %-----------------------------------------------------------
110         % sampleUnitCell function
111         %-----------------------------------------------------------
112         function  R = sampleUnitCell(len, a1, a2)
113             % Used to be called uniSpacedPointsInUnitCell
114             % Uniformly distributed over the unit cell
115             vec=0:1/len:1-1/len;
116             len=length(vec);
117             [N,M]=meshgrid(vec);
118             for n=1:len
119                 for m=1:len
120                     R((n-1)*len+m,1:2)=N(n,m)*a1+M(n,m)*a2;
121                 end
122             end
123         end
124         %-----------------------------------------------------------
125         %-----------------------------------------------------------
126
127
128         %-----------------------------------------------------------
129         % spanVecs function
130         %-----------------------------------------------------------
131         function G = spanVecs(b1,b2,nVec1,nVec2)
132
133             [N_g,M_g]=meshgrid(nVec2,nVec1);
134             Gx = N_g * b1(1) + M_g * b2(1);
135             Gy = N_g * b1(2) + M_g * b2(2);
136             G = [Gx(:) Gy(:)];
137         end
138         %-----------------------------------------------------------
139         %-----------------------------------------------------------
140
```

```
141     % end of functions
142     end
143 
144 end
145 %-----------------------------------------------------------------------
146 %-----------------------------------------------------------------------
```