Medium-energy ions scattering studies of ultrathin metallic films on the face-centred cubic (111) surface

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Additional Information:


Metadata Record: https://dspace.lboro.ac.uk/2134/35637

Publisher: © Christopher J. Howe

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
University Library

Author/Filing Title: Howe, C. J.

Class Mark: T

Please note that fines are charged on ALL overdue items.
Medium Energy Ions Scattering Studies of Ultrathin Metallic Films on the Face Centred Cubic(111) Surface

by

Christopher J Howe

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

May 2008

© by Christopher J Howe 2008
ABSTRACT

An Experimental investigation of ultrathin metallic films on Face Centred Cubic(111) surfaces using Medium Energy Ion Scattering as a primary experimental technique is reported. A suite of custom software written for data-processing, calibration and analysis has also been described. The specific experimental systems investigated are 0.25 ML to 3.89 ML of Palladium on Copper(111) and 0.23 ML to 3.74 ML of Silver on Aluminium(111).

The Pd Cu(111) system was shown to exhibit 42% twinning at 0.25 ML through 4.74 ML. At 0.25 ML coverage it has been shown that the Pd migrates into the top three layers of the substrate. The Pd was shown to be distributed 27% ± 3%, 67% ± 3% and 6% ± 3% in the 1st, 2nd and 3rd layers. The first and second interlayer spacings were also found to contract by 4% and expand by 4% respectively for the 0.25 ML coverage.

The Ag on Al(111) system was shown to exhibit a mixture of fcc and hcp growth from the outset. The initial settling points of the Ag were shown to be 43.6% ± 5.2% in fcc sites and 56.4% ± 5.2% in hcp sites. At 4.74 ML Ag coverage it has been shown that the hcp growth terminates and returns to fcc stacking after a spread of stacking heights ranging from 1 atom high hcp through to 5 atoms high hcp stacking.

The custom software suite includes: A re-tiling package for correcting mistakes made by the current MEIS facility software; a calibration package that compares Vegas simulation to bulk blocking dips to create calibration files; a curve fitting package that extracts surface and substrate integrals; and a program that interfaces with commercial Rutherford Backscattering software to create complex thin film models.
ACKNOWLEDGEMENTS

I would first like to extend my thanks to my supervisor Mike Cropper, formerly Mike Crapper. Your help, wisdom and interesting insights on a variety of subjects has been invaluable. Also, my thanks go out to Tim Noakes, Paul Bailey and Kevin Connell at the UK MEIS facility at Daresbury laboratory.

Mum and Dad, you have been absolute gems. Having to cope with me for the summer taking over the living room, me being generally grumpy as I couldn't figure out my analyses must have required the patience of saints. I couldn't have done it without you two.

Uncle David, as requested, you have your very own paragraph. This isn't quite five minutes of fame, but decades and decades of life on a bookshelf collecting dust is a certainty.

My research could not have been completed without the financial support of Loughborough University. Thank you for making it possible.

You can achieve anything you put your mind to, the only thing that will hold you back is yourself.
# Contents

Table of Contents  iv  
List of Figures  v  

## 1 Introduction  

## 2 Techniques and Analysis  

2.1 Medium Energy Ion Scattering  
2.1.1 Introduction to MEIS  
2.1.2 Channelling  
2.1.3 Rutherford Backscattering (RBS)  
2.1.4 Scattering cross section  
2.1.5 Energy Loss Depth Profiling  
2.1.6 Straggling  

2.2 MEIS simulations  
2.2.1 SIMNRA  
2.2.2 Vegas  

2.3 MEIS instrumentation  

2.4 Complementary Analysis Techniques  
2.4.1 Low Energy Electron Diffraction (LEED)  
2.4.2 Auger Electron Spectroscopy (AES)  

2.5 Crystallography  
2.5.1 Crystal growth classifications  
2.5.2 Face Centered Cubic Structure - Fcc(111) surface  
2.5.3 Twinned Face Centered Cubic Structure - Fcc(111) surface  
2.5.4 Hexagonal Close Packed Structure (hcp)  
2.5.5 Thermal Vibration  

2.6 Data Analysis  
2.6.1 Quantitative Analysis - Creating a scattering cross section  
2.6.2 The R-factor  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Techniques and Analysis</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Medium Energy Ion Scattering</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Introduction to MEIS</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Channelling</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3 Rutherford Backscattering (RBS)</td>
<td>8</td>
</tr>
<tr>
<td>2.1.4 Scattering cross section</td>
<td>9</td>
</tr>
<tr>
<td>2.1.5 Energy Loss Depth Profiling</td>
<td>10</td>
</tr>
<tr>
<td>2.1.6 Straggling</td>
<td>12</td>
</tr>
<tr>
<td>2.2 MEIS simulations</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 SIMNRA</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2 Vegas</td>
<td>15</td>
</tr>
<tr>
<td>2.3 MEIS instrumentation</td>
<td>16</td>
</tr>
<tr>
<td>2.4 Complementary Analysis Techniques</td>
<td>19</td>
</tr>
<tr>
<td>2.4.1 Low Energy Electron Diffraction (LEED)</td>
<td>19</td>
</tr>
<tr>
<td>2.4.2 Auger Electron Spectroscopy (AES)</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Crystallography</td>
<td>22</td>
</tr>
<tr>
<td>2.5.1 Crystal growth classifications</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2 Face Centered Cubic Structure - Fcc(111) surface</td>
<td>23</td>
</tr>
<tr>
<td>2.5.3 Twinned Face Centered Cubic Structure - Fcc(111) surface</td>
<td>26</td>
</tr>
<tr>
<td>2.5.4 Hexagonal Close Packed Structure (hcp)</td>
<td>29</td>
</tr>
<tr>
<td>2.5.5 Thermal Vibration</td>
<td>29</td>
</tr>
<tr>
<td>2.6 Data Analysis</td>
<td>30</td>
</tr>
<tr>
<td>2.6.1 Quantitative Analysis - Creating a scattering cross section</td>
<td>30</td>
</tr>
<tr>
<td>2.6.2 The R-factor</td>
<td>31</td>
</tr>
</tbody>
</table>
# CONTENTS

3 Macros and Programming 33  
3.1 Introduction 33  
3.2 Super Mega Curvefit 34  
3.3 Match Data Macro 39  
3.4 Tiler Macro 41  
3.4.1 The Tiling process 43  
3.5 SIMNRA file creator 45  
3.6 Summary and Comments 49  

4 Pd on Cu(111) 51  
4.1 Introduction 51  
4.2 Experimental Details 54  
4.3 Results and Discussion 55  
4.4 Conclusion 97  

5 Ag on Al(111) 100  
5.1 Introduction 100  
5.2 Experimental Details 102  
5.3 Results and Discussion 103  
5.4 Conclusion 131  

6 Conclusion 134  
6.1 Experimental Conclusions 134  
6.1.1 Pd on Cu(111) Conclusions 135  
6.1.2 Ag on Al(111) Conclusions 135  
6.1.3 General Conclusions and MEIS Best Practices 136  

References 140  
References 146  

A Stereographic Projections 147  

B SuperMega CurveFit key data structures 152  

C SuperMega CurveFit key data structures 155
List of Figures

2.1 A simple illustration of a shadowing cone created by $H^+$ ions incident on a target nucleus ................................................................. 6
2.2 Examples of ions travelling along major crystallographic channels. . . 7
2.3 Low energy ions (left) and high energy ions (right) being scattered. . 8
2.4 An example of how blocking dips are produced .............................. 8
2.5 The variation of the scattering kinematic factor for Hydrogen ions being scattered off Oxygen and Copper. ................................. 9
2.6 Scattering geometry of an ion being back-scattered off a thin film (side view) of thickness $t$ deposited on a substrate. Note that the $k^2$ correction factor depends on the mass of the atom from which the beam ion scatters. ................................................................. 11
2.7 How the beam energy profile is affected by straggling .................. 13
2.8 The Chu correction factor for various beam energies. 100keV MEIS experiments have a correction factor of approximately 0.2 for heavier elements and is therefore very important to take into consideration. . 14
2.9 The UK national MEIS facility located at Daresbury ..................... 17
2.10 A typical LEED setup ................................................................. 20
2.11 Three-stage energy level diagram of the Auger process: Ionisation of a core energy level by a high energy electron beam (1 and 2), and relaxation and Auger emission (3) .................................................. 20
2.12 A differentiated Auger scan of a thin Pd film on a Cu substrate ...... 22
2.13 Epitaxial growth modes: (a) Frank-van der Merwe(FM), (b) Stranski-Krastanov(SK), and (c) Volmer-Weber(VW). ............................... 22
2.14 A Copper face centered cubic crystal. The (111) planes (A,B,C) have been highlighted. ................................................................. 24
2.15 A projection of the fcc unit cell into the fcc(111) surface ................ 24
2.16 A side view of an fcc crystal, showing the ABC structure. The arrow highlights the direction of the major crystal channel. .................... 25
2.17 A top down view on an fcc(111) surface. The left of the diagram shows the normal surface growth mode, with the 3 different deposition sites: A, B and C. The 3 arrows show the direction of the major channel growth in normal "ABC" growth. The right of the diagram shows twinned "ACB" growth. The 3 arrows on the right are a rotation of the 3 arrows on the left by 60°.

2.18 A side view of an fcc(111) crystal (<112> azimuth) coloured in blue, topped by its twinned counterpart in pink (<211> equivalent azimuth). The dashed circle shows where the fcc continuation would have been. The arrows highlight the change in growth direction. Note the ABC to ACB layer change.

2.19 Top and side views of hcp structure. Large circles represent nearer atoms, smaller circles represent atoms in the next layer into the crystal. VEGAS unit cell enclosed in dashed box.

3.1 A cut from a 2D data fit. The different colours indicate the various components of the final curve.

3.2 The user interface panel of the Supermegacurvefit Igor macro.

3.3 First stage of calibration. Raw data with detected peaks (a), and simulated data with detected peaks (b).

3.4 The second stage of calibration. A comparison of the blocking dip locations and the user selection box to apply the comparisons.

3.5 A calibration curve for the data shown in the previous two figures.

3.6 Full MEIS tiles with (b) and without (a) shaving off the top and bottom 5 pixels of each tile.

3.7 An example of the varying energy scales of the tilers produced by MIDAS.

3.8 An example of how the energy scales are affected when tiles are just shaved off (grey areas represent the shaven off areas).

3.9 An example of how the energy scales are rescaled once they have been shaved off.

3.10 The user interface panel of the SIMNRA file maker Igor macro.

3.11 A sample profile designed in SIMNRA-file-maker. This particular graph shows a diffusing Palladium film on a Copper substrate.

3.12 The parameter variation panel of the SIMNRA-file-maker Igor macro.
LIST OF FIGURES

4.1 Run 80842, 0.80 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The first peak at 101.8 keV is the Pd, the second at 100 keV is the Cu substrate. Ideally at a random incidence the second peak should merge into the bulk but due to a crystallographic alignment blocking occurs. The sample profile (bottom) indicates that the Pd and Cu are mixed approximately 50:50 for the first 1.5 ML (3 nm) and then drops to 0:100 Pd:Cu in the second 1.5 ML.

4.2 Run 81003, 1.87 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The Pd concentration appears to steadily drop from 100% through to 0% by travelling 1 nm into the sample, approximately 5 times that of the natural Pd lattice spacing.

4.3 Run 81177, 1.86 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. Even though this sample appears to have the same thickness film as the previous figure, run 81003, the model profile and experimental cut appears to be reasonably different. This could be due to the slightly different “random” alignment or actual sample differences. The Pd material appears to be more towards the surface compared to run 81003, but penetrates deeper into the substrate at a lower concentration.

4.4 Run 81492, 3.89 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The Pd concentration appears to steadily drop from 100% to 0% in the space of 1.6 nm. The experimental results appear to be very well aligned with the bulk signal merging completely with the substrate peak, indicating that beam alignment is at an optimal random alignment.

4.5 Run 81743, 0.25 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. Examining the sample model, it appears that the material is more concentrated near the surface rather than having a steady intermixing into the substrate. The concentration drops rapidly to approximately 30% in the space of 0.04 nm, but then peters off to 0% concentration in the space of 0.8 nm.

4.6 Run 81878, 0.26 ML Pd on Cu(111), annealed at 450°C. Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. It can be seen that the Pd is heavily intermixed into the Cu (bottom), and that the Pd signal is weak.
4.7 The MEIS experimental Pd overlayer blocking curves for the <114> beam incidence direction. An identical beam orientation Vegas simulation of 3.83 ML layer-by-layer growth has been offset plotted on the right hand axis. The fcc Pd simulation assumes the Pd is in registry with the Cu substrate and has Cu lattice spacing. Note the interesting feature at approximately 90° on 3.89 ML Pd experimental curve which is not present in the fcc simulation. ........................................ 67

4.8 The MEIS experimental Pd overlayer blocking curves for the <110> beam incidence direction. An identical beam orientation Vegas simulation of 3.83 ML layer-by-layer growth has been offset plotted on the right hand axis. The fcc Pd simulation assumes the Pd is in registry with the Cu substrate and has Cu lattice spacing. Note the interesting feature at approximately 84° on the 3.89 ML experimental Pd curve that is not present in the Vegas simulation. ............................ 68

4.9 Experimental and Vegas simulation blocking dip comparison for the <110> and <114> beam incidences of 3.89 ML and 3.83 ML respective Pd coverages. Experimental visibility plotted on the left axis, simulation on the right. Both are offset to aid viewing. ................. 70

4.10 Experimental and Vegas simulation blocking dip comparison for the <114> beam incidences of 3.89 ML and 3.83 ML respective Pd coverages. A variety of twinned crystal samples have been plotted. Attention should be drawn to the experimental (233) channel at approximately 85° being offset to the simulation and the compared visibilities of the simulations and experiment. Experimental visibility plotted on the left axis, simulations on the right. Both are offset to aid viewing. 71

4.11 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the <114> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83ML Pd overlayer. The Cu visibility is increasing as Pd overlayer coverage increases. A possible reason for this could be Pd induced Cu structure change. ........................................ 73

4.12 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the <110> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83ML Pd overlayer. There is approximately 2.5 ML extra Cu visible when covered with 3.89 ML of Pd. ................................. 74

4.13 Experimental and simulated Pd overlayer visibility profiles for varying coverages of a Pd film on Cu(111). <110> and <114> beam incidence directions. Various twinning percentages were also simulated. . . . . 75

4.14 Experimental and simulated Cu substrate visibility profiles for varying coverages of a Pd film on Cu(111). <110> and <114> beam incidence directions. Various twinning percentages were also simulated. . . . . 76
4.15 Vegas simulations of clean Cu(111) for the <114> beam incidence direction. A range of the 0-5 layers have had their interplanar lattice spacing increased to that of Pd(111). Note the yield change, and blocking dip offset. ................................................................. 77
4.16 Experimental blocking curves for various coverages of Pd on a Cu(111) substrate for the <233> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. 81
4.17 Experimental blocking curves for various coverages of Pd on a Cu(111) substrate for the <122> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. 82
4.18 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the <233> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. Several experimental curves with Pd coverage have been smoothed to aid the eye. ................................................................. 83
4.19 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the <122> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. Several experimental curves with Pd coverage have been smoothed to aid the eye. ................................................................. 84
4.20 Clean Cu(111) compared to Cu(111) with 0.25ML of Pd coverage: Substrate and surface blocking dips. <114> beam incidence. Notice the angular shifts of the blocking dips at approximately 74°. The Pd dip is shifted left, the same-sample Cu blocking dip is shifted right from the clean Cu blocking dip. Note that the experimental Pd blocking dip at 74° is wider than both of the Cu curves. This suggests that this blocking dip could be formed from two blocking dips that are separated in scattering angle. ................................................................. 91
4.21 A Vegas simulation of clean Cu(111), <114> beam incidence. The overall counts have been separated layer-by-layer to show what layers contribute to each blocking dip. ................................................................. 92
4.22 2D-R-value plot of the angular shift fitting whilst varying the 1st and 2nd interlayer spacing. <114> beam incidence, 70° to 100° scattering angle. The minimum is located at -4% and +4% first and second interlayer expansions. ................................................................. 94
4.23 Experimental and simulated blocking curves for 0.25 ML Pd on Cu(111) for <114> beam incidence, 70° to 100° scattering angle. For the simulation, the first and second interlayer spacings have a 4% contraction and a +4% expansion. ................................................................. 95
4.24 <233> beam incidence bulk (111) crystal simulation and 3.8ML Pd on Cu(111), also <233> incidence. Note the central blocking dip in the simulation only appears in layers 5 and 6, which is weakly visible in the experimental curve. ................................................................. 96
5.1 Hcp Ag on fcc Al, with a $<110>$ beam incidence. It can been seen that for a 6-layer hcp coverage the 1st, 3rd and 5th layers of the hcp structure are illuminated, and the substrate is shadowed (beam A). Only upon addition of a seventh layer the hcp structure starts to repeat itself with respect to the $<110>$ incidence (beam B).  

5.2 Hcp Ag on fcc Al, with a $<233>$ beam incidence. It can been seen that for a 6-layer hcp coverage all hcp layers are illuminated. The top two hcp layers shadow the top two fcc substrate layers. As the $<233>$ incidence is 2-layer, the substrate is fully shadowed by these top two layers. Upon addition of a 7th layer the overlayer begins to shadow itself.

5.3 Ag surface experimental curves of varying Ag film thicknesses on Al(111) with a $<110>$ beam incidence. Bulk fcc Al and hcp Al Vegas simulations have been included to show the location of major fcc blocking dips and suspected hcp-caused blocking dips. Note the hcp simulation is plotted separately on the right-hand axis.

5.4 Ag surface experimental curves of varying Ag film thicknesses on Al(111) with a $<233>$ beam incidence. Bulk fcc Al and hcp Al Vegas simulations have been included to show the location of major fcc blocking dips and suspected hcp-caused blocking dips. Note the hcp simulation is plotted separately on the right-hand axis.

5.5 Evolution of Ag visibility due to Ag film thickness on Al(111) crystal. $<110>$ beam incidence, yields measured at 84.5° (a), 90° (b) and 97.5° (c). Film thicknesses were measured across 7 angular bins, errors were calculated to be the standard deviation of the thicknesses across these channels.

5.6 Evolution of Al substrate visibility due to Ag overlayer film thickness on Al(111) crystal. $<110>$ beam incidence, yields measured at 84.5° (a), 90° (b) and 97.5° (c).

5.7 Vegas simulation of 6 ML of hcp Ag on Al(111), $<110>$ beam incidence, showing the layer by layer Ag yields.

5.8 Experimental comparison of 1.33 ML, 3.42 ML and 4.74 ML Ag coverages.

5.9 Vegas simulation of 6 ML of hcp Ag on Al(111), $<233>$ beam incidence, showing the layer by layer Ag yields.

5.10 Close-up view of 3.42 ML and 4.74 ML experimental signals for the $<233>$ beam incidence. The weak blocking dip at approximately 94° could be formed from a 4-layer hcp structure.
5.11 Left: Vegas simulations of fcc Al(111) with 4.75 ML of fcc Ag and fcc Al(111) with 4 ML hcp Ag and 0.75 ML of fcc Ag on top of that. Right: Combination of the fcc Ag and mostly-hcp Ag simulations. The fcc to hcp ratio was found to be 61% ± 6% total fcc Ag and 39% ± 6% hcp 4-layer + fcc 0.75 ML growth (61:39). The R-factor for this fit was 0.0074. <233> beam incidence. ........................ 120

5.12 <233> beam incidence Vegas simulations of Ag on Al(111). The Ag structure is varied from 4.75 ML 100% fcc-Ag to 4.75 ML 100% hcp-Ag going through varying numbers of ML of hcp-Ag topped with fcc Ag to sum up to 4.75 ML. Note the yield varying on each y-axis. 121

5.13 Side cut diagram of fcc and hcp stacking upon an fcc substrate. Grey is the substrate, blue is hcp stacked atoms and pink is fcc stacked atoms. In certain cases it is arguable whether the atoms are in an fcc or hcp position e.g. fcc-3 hcp-2. These were the models used in the Vegas simulations. 122

5.14 <110> beam incidence Vegas simulations of Ag on Al(111). The Ag structure is varied from 4.75 ML 100% fcc-Ag to 4.75 ML 100% hcp-Ag going through varying numbers of ML of hcp-Ag topped with fcc Ag to sum up to 4.75 ML. Note the yield varying on each y-axis. 124

5.15 Combined Vegas simulation curve fits of <233> and <110> beam incidence laid over their corresponding experimental counterparts. 126

5.16 The hcp/fcc population profile of 4.74 ML of Ag on Al(111). The x-axis represents the number of hcp layers present in a particular stack, with the number of fcc layers covering that particular stack being 5 - hcp layers. Therefore 0 is pure fcc. The y-axis representing the percentage population of that given stack. 127

5.17 Left shows Vegas substrate simulations compared to experimental substrate signal for the <110> beam incidence. The dotted curves are substrate simulations with 0.24 ML of hcp and fcc Ag overlayers. The blue curve is the best fit combination of the fcc and hcp simulations. The yellow curve is the experimental substrate signal. Right shows the corresponding R-factor curve for the combined simulation fitting. The R-factor minimum is found at 47% fcc with a R-factor of 0.00548. 129

5.18 This graph shows the thickness of hcp Ag layers vs. their percentage populations. The Ag hcp layers are then topped with fcc Ag topping the total number of layers to 5. This data is taken from 4.74 ML Ag on Al(111). 130

A.1 Stereographic Projection for the fcc(111) surface ........................ 148
A.2 Azimuthal cut for the fcc(111) surface: (011) azimuth .................. 149
A.3 Azimuthal cut for the fcc(111) surface: (112) azimuth .................. 150
A.4 Azimuthal cut for the fcc(111) surface: (211) azimuth .................. 151
Chapter 1

Introduction

Surface science is a popular and rapidly expanding field of study. It combines several disciplines with its roots firmly placed within Chemistry and Physics. The increasing popularity of the subject is owed to the rapid growth of directly applicable technology such as the continued miniaturisation of electronics and the booming field of nanotechnology as just a fraction of available examples. Cutting-edge technology does not come without a firm grounding in the fundamental science and understanding of that technology, which is the level at which this thesis is based.

This thesis examines the crystallography of ultrathin metallic films upon face centered cubic (111) metallic single crystal surfaces by experimental means. The primary technique used is Medium Energy Ion Scattering (MEIS) [1], with Low Energy Electron Diffraction (LEED) [2] [3] and Auger Electron Spectroscopy (AES) [4] [5] [6] being used as secondary experimental techniques.

The study of thin metallic films upon a metal substrate are of particular interest due to current emerging technologies and the progression of industrial techniques. Spintronics is an active field of research which makes use of thin metallic films upon a metallic substrate. It exploits the intrinsic spin of electrons and its associated
magnetic moment, in addition to its fundamental electronic charge as in the case of standard electronics. This opens possibilities of spintronic devices being able to perform switching operations quicker and more efficiently compared to traditional semiconductor devices.

Catalysis can also make use of thin metallic films upon a metal substrate. As an element, Palladium is of much interest in catalytic processes. This thesis explores the crystallography of Palladium upon a Copper substrate. As the crystallography of these thin films are being investigated, the experiments could determine the lattice parameters and the growth habits under certain conditions. If we know how to replicate certain conditions or how to create certain growth features we could possibly tailor a surface for optimum or novel catalysis settings.

MEIS is a real-space ion scattering technique that, in this case, uses Hydrogen or Helium ions accelerated to an energy of 50 keV to 400 keV which are back-scattered off the sample. This particular energy range allows the accurate observation of energy losses as the ions pass through the material, and channelling effects of the crystal lattice that the beam is passing through. The observation of energy losses allows sample depth profiling to be measured, channelling allows crystallographic structures to be interpreted. Combining the two techniques can lead to a very in-depth analysis. With the relatively non-destructive energy of the ion beam, one could in theory produce a crystallographic structural depth profile revealing information that would prove very difficult using other techniques. The real-space scattering events of the ions can also reveal information that diffractive experiments would find hard to uncover. MEIS can prove to be a very effective analysis tool for a wide variety of experimental systems. All MEIS results were taken from the Daresbury UK MEIS facility [7].

The two experimental systems investigated in this thesis, Palladium on Copper(111) and Silver on Aluminium(111) have also been selected due to their suitability
to be investigated using the experimental technique of MEIS. They are both heavy films deposited upon a noticeably lighter substrate. This mass separation makes it easier to separate the individual signals from each element. The converse system, a light film on a heavier substrate, would also be well suited to a MEIS experiment, but not as well as the previous system due to bulk noise interfering with the film signal. As the masses of the film and the substrate get closer together it will get harder to separate the signals making the MEIS experiment harder to analyse.

The outline of this thesis is as follows. Chapter 2 describes the background of the experimental techniques used in this thesis. This includes MEIS, LEED, AES, crystallography and data analysis techniques. Chapter 3 describes the software suite that was used to process and analyse the MEIS data. Considerable time and effort was put into developing standard programs to mainly process data; calibration and the reduction of large amounts of data is unavoidable and very involved when undertaking a MEIS experiment. This chapter describes the usage and basic underlying structure of these programs. The source codes for these programs are listed in Appendix C. In Chapter 4 the results of a MEIS experiment of ultrathin films of Pd on Cu(111) are presented. Pd film thicknesses vary from 0.22 ML to 3.89 ML. In Chapter 5 experimental system of ultrathin films of Ag on Al(111) is presented, films analysed being of thicknesses 0.23 ML to 4.74 ML. Chapter 6 discusses and summarises conclusions of the experimental and software chapters, and ties these findings in with more general conclusions of MEIS and scientific practice.

An aim of this thesis is to show the reader the thought processes and general investigative techniques that are needed for a MEIS investigation. Within the experimental data results there is a lot of information that can be extracted, some being useful and a lot being not useful. The difficulty in analysing MEIS results is gaining the ability to successfully extract poignant information without wasting too much
time examining red herrings.
Chapter 2

Techniques and Analysis

The experimental method and science behind MEIS has been described in detail in this chapter, along with complementary techniques LEED and AES. Data acquisition and simulation techniques have been detailed. Crystallographic growth modes and structures have also been described, followed by basic data analysis techniques.

2.1 Medium Energy Ion Scattering

2.1.1 Introduction to MEIS

In a typical MEIS experimental setup, light ions (e.g. H⁺, He⁺) are fired with an energy of 50 keV-400 keV [8] at a sample, and the recoiling ions are captured. The DeBroglie wavelength of these ions is of the order of $1 \times 10^{-10}$Å so any scattering that occurs will be a real-space event, i.e. no diffraction. The positively charged ions scatter off the positively charged nuclei and are then detected, with their scattering angle, and energy at that particular angle being measured. With these values being measured we can analyse a sample using two techniques: Rutherford Backscattering (RBS) and channelling. They can be used separately to mainly identify elemental
2.1 Medium Energy Ion Scattering

![Diagram of shadowing cone created by H⁺ ions](image)

**Figure 2.1** A simple illustration of a shadowing cone created by $H^+$ ions incident on a target nucleus.

composition and structure respectively, or can be combined to produce a more detailed analysis for complex samples e.g. specified crystallography of ultrathin films.

### 2.1.2 Channelling

A crystalline sample will have various crystallographic planes, which, if aligned correctly with an incident $H^+$ beam will produce blocking and shadowing (channelling) effects. As the positive ions are scattered from the positive electric field each nucleus will produce a shadowing cone (see figure 2.1) [9], whose width depends on the energy of the beam, the incident ion type, and the sample it is scattering off.

Figure 2.2 illustrates how channelling occurs within a crystal. The left-hand diagram shows the surface layer that has been illuminated. The reflected ions are being measured and they can only be reflected from the surface layer in this particular example as the sub-layers are being shadowed by the surface. The right hand diagram in figure 2.2 shows the top two layers being illuminated. There is a certain distance that the channelled ions can travel into the crystal without being reflected (de-channeled). Amongst other things, this can depend on the quality of the crystal, the width of the shadow cone, the amount of straggling, and how well the beam is aligned in the first
2.1 Medium Energy Ion Scattering

Figure 2.2 Examples of ions travelling along major crystallographic channels.

place. This de-channelling can be seen as a gradual increasing count level as you go further into the sample. i.e. measure a lower and lower scattered ion energy.

If the shadowing cone were too large then correctly aligned ions would still get scattered after only travelling a short distance, making beam/channel alignment difficult. Conversely, if the shadowing cone were very small then there would be room for an unnoticeable angular spread of the beam, meaning that the beam could be misaligned and it would go unnoticed, as shown in figure 2.3. At high enough energies, the shadowing cone would be so small that no blocking dips would be visible. This shadowing cone width \([9]\) can be calculated using equation 2.1, where \(r_c\) is the shadow cone radius, \(Z_1\) and \(Z_2\) are the atomic numbers of the projectile and target, \(e\) is the electron charge, \(E\) is the beam energy and \(d\) is the distance behind the target that the \(r_c\) needs to be calculated.

\[
r_c = 2\sqrt{\frac{Z_1 Z_2 e^2 d}{E}}
\] (2.1)

Once the ions have travelled into the sample and been reflected, they have to travel back out again. Crystallographic planes will again be experienced on the outward bound direction. This part of the ions’ journey produces the characteristic blocking dips known to these experiments.
2.1 Medium Energy Ion Scattering

2.1.3 Rutherford Backscattering (RBS)

When ions are fired at a sample, recoiling ions will lose energy by a factor given in equations 2.2 and 2.3, known as the kinematic factor [10]. This equation takes into account the energy that is transferred to the stationary target when an incident ion recoils.

\[ k^2 = \left( \frac{\sqrt{m_t^2 - m_i^2 \sin^2 \theta + m_i \cos \theta}}{m_i + m_t} \right)^2 \] (2.2)

\[ E_{recoil} = k^2 E_{initial} \] (2.3)

The kinematic factor can be altered by changing scattering angle, the mass of the incident ion \( m_i \), and the mass of the target \( m_t \). After a collision, the recoiling atom will thus have an energy shown in equation 2.3, where \( m_t \) and \( m_i \) are the masses of the target and incident ion, respectively, and \( \theta \) is the scattering angle. Figure 2.5 shows how the kinematic factors vary for Hydrogen ions being scattered off Copper and Oxygen, two masses that are moderately separated on the periodic table. The greater
2.1 Medium Energy Ion Scattering

Figure 2.5 The variation of the scattering kinematic factor for Hydrogen ions being scattered off Oxygen and Copper.

the scattering angle the further the kinematic factors separate. This means that to get more mass distinction using MEIS, one would have to increase the scattering angle, causing masses that are very close together to become more distinguishable.

2.1.4 Scattering cross section

The probability of scattering of a positive ion by a positive nucleus is described by the Rutherford scattering cross section [11], equation 2.4

\[
\frac{d\sigma_R}{d\Omega}_{CM} = \left( \frac{\alpha \hbar c}{2mv_0^2} \right)^2 \frac{1}{\sin^4(\theta/2)} \tag{2.4}
\]

At low and high energies experiments have revealed deviations from the Rutherford cross section. At low energy these departures are caused by partial screening of the nuclear charges by the electron shells surrounding both the incident and target nuclei. This screening can be taken into account by a correction factor \( F \), shown in equation 2.5 [12].

\[
F_{L'ecu} = 1 - \frac{0.049Z_1Z_2^{(4/3)}}{E} \tag{2.5}
\]
2.1.5 Energy Loss Depth Profiling

As the ion travels into a material it will lose energy proportional to the distance it has travelled through that material, and at a rate which depends on the ion’s energy. This energy loss can be used to depth profile a material e.g. a thin film. The energy loss due to the type of material is known as its stopping power, and at a given energy can be calculated using a program known as SRIM (Stopping and Range of Ions in Matter) [13]. A typical output of energy loss would be in eV Å⁻¹.

The energy loss per unit length for Hydrogen can be also calculated manually using Anderson-Ziegler stopping data [14] and equations 2.6-2.8. \( A_2 \)-\( A_5 \) are tabulated in [14]. These equations are valid for the energy range 10keV - 1MeV, and thus being applicable to MEIS. These equations are empirical fits as opposed to calculations based on theory. Equations and stopping coefficients are listed for Helium and heavier ions for greater energy ranges in [14]. However, this is beyond the scope of this thesis.

\[
\frac{1}{S_e} = \frac{1}{S_{Low}} + \frac{1}{S_{High}} \tag{2.6}
\]

\[
S_{Low} = A_2 E^{0.45} \tag{2.7}
\]

\[
S_{High} = \frac{A_3}{E} \ln[1 + \frac{A_4}{E} + A_5 E] \tag{2.8}
\]

The stopping power of compounds can be calculated using Bragg’s rule [15]. It is a simple weighted average of the individual stopping contributions of each element in the compound, and is given by equation (2.9), where \( c_i \) is element i’s fractional concentration and \( S_i \) is its stopping power, also assuming \( \sum c_i = 1 \).

\[
S = \sum c_i S_i \tag{2.9}
\]
2.1 Medium Energy Ion Scattering

![Figure 2.6](image)

**Figure 2.6** Scattering geometry of an ion being back-scattered off a thin film (side view) of thickness $t$ deposited on a substrate. Note that the $k^2$ correction factor depends on the mass of the atom from which the beam ion scatters.

The scattering geometry will determine the path length of an ion being scattered. Knowing the energy loss per unit length of a material, it can then be depth profiled. The simplest case would be a heavy thin film placed on a lighter substrate, as shown in figure 2.6. $E_1$, the energy of back-scattered ions from the surface will have an energy given in equation 2.3.

As the ion travels through the thin film it loses energy the further it travels. As it scatters off the substrate (dashed), its energy will drop by a factor equal to the $k^2$ value of the substrate. It then continues to lose more energy as it travels back out of the material. The various stages of this journey can be calculated in equations 2.10 - 2.12. Using these equations we can then form equation 2.13 which can be used to calculate the thickness of a thin film based on the energy loss of the beam scattering off the substrate.

\[ E_2 = E_0 - \frac{t}{\cos \theta_i} \frac{dE}{dx} \]  \hspace{1cm} (2.10)

\[ E_3 = k^2 \left( E_0 - \frac{t}{\cos \theta_i} \frac{dE}{dx} \right) \]  \hspace{1cm} (2.11)
2.1 Medium Energy Ion Scattering

\[ E_4 = k^2 \left( E_0 - \frac{t}{\cos \theta_i} \frac{dE}{dx} \right) - \left( \frac{t}{\cos (180 - \theta_i - \theta_s)} \frac{dE}{dx} \right) \]  \hspace{2cm} (2.12)

\[ t = \frac{E_1 - E_4}{\left( \frac{k^2}{\cos \theta_i} + \frac{1}{\cos (180 - \theta_i - \theta_s)} \right) \frac{dE}{dx}} \] \hspace{2cm} (2.13)

Where \( t \) is the film thickness, \( E_1 \) is the energy of the beam after it has scattered off the top of the film, \( E_4 \) is the energy of the after scattering off the top of the substrate and travelling through the covering film. \( k^2 \) is the kinematic factor of the substrate calculated from the beam incident angle \( \theta_i \) and the beam scattering angle \( \theta_s \). \( \frac{dE}{dx} \) is the energy loss of the beam ions per unit length travelled through the film.

In a real ion scattering experiment there will be large amount of ions being scattered. In experiments that will be detailed later, the total yield of scattered ions amounts to approximately the charge of \( 2 \mu \text{C} \) \( (12.5 \times 10^{12} \text{ ions}) \). When a thin film is being analysed, ions will be scattered off atoms throughout the thin film, not just the surface and substrate interface. This means that a thin film peak would be measured, and the substrate would be a continuum in an energy/counts profile (providing the beam is not crystallographically aligned).

2.1.6 Straggling

As ions pass through a material the beam energy profile spreads. This phenomenon is known as straggling. It is due to statistical fluctuations of the energy transfer in the collision process. This limits both mass and depth resolution in RBS experiments. Straggling could be directly observed in an ion transmission experiment i.e. instead of scattering off a sample, the detector is placed in line with the beam, with a foil sample in the way. Figure 2.7 shows how the beam energy profile is broadened by straggling.
2.2 MEIS simulations

Figure 2.7 How the beam energy profile is affected by straggling

Early work on straggling was first done by Bohr [16] who produced a simple equation to calculate the amount of straggling, equation 2.14, where $\Omega_B$ is the variance of the energy distribution (the full width half maximum (FWHM) being $\sqrt{8\ln 2}\Omega$, $Z_1$ and $Z_2$ are the atomic numbers of the incident ion and target, $e$ is the electronic charge, $N$ is the atomic density and $d$ is the path length.

$$\Omega_B = \frac{Z_1 e^2}{2\varepsilon_0} \sqrt{\frac{Z_2 N d}{\pi}} \quad (2.14)$$

This equation only holds for high energies (MeV) and low atomic numbers. Chu [17] produced a correction factor which corrects for straggling effects at lower energies and is given by equation 2.15. Chu has calculated $H$ by using the Hartree-Fock-Slater charge distribution and produces straggling values far lower than the ones predicted by Bohr's theory. These values have been plotted in figure 2.8 and can also be found in [18].

$$\Omega_{Chu} = H\Omega_{Bohr} \quad (2.15)$$

2.2 MEIS simulations

2.2.1 SIMNRA

SIMNRA [19] is a Microsoft Windows program for the simulation of backscattering spectra for ion beam analysis. It is mainly intended for the simulation of non-
2.2 MEIS simulations

Figure 2.8 The Chu correction factor for various beam energies. 100keV MEIS experiments have a correction factor of approximately 0.2 for heavier elements and is therefore very important to take into consideration.
2.2 MEIS simulations

Rutherford backscattering, nuclear reactions and elastic recoil detection analysis (ERDA), but can be used for the energy ranges found in MEIS. It has a fully graphical user interface and is very useful for performing quick basic depth profiles (to get an idea of approximate coverages) along with straggling calculations. SIMNRA has also been adapted to perform depth profiles on more complex samples that would be very difficult to create using the supplied graphical user interface. This has been achieved using Windows OLE automation in conjunction with Igor Pro and has been described in Section 3.5 starting on page 45.

SIMNRA is used at more than 130 laboratories world wide, and referenced in more than 325 publications (September 2006).

There are several other RBS simulation programs available [18] [20] [21] [22], one being called RUMP [23]. RUMP is notoriously difficult to use, being described in its own manual as “Really Ugly and Mangled Procrastination tool”. This program was tested and decided that the number of simulations required multiplied by the time taken to use the package resulted in a large amount of wasted time. The only benefit of this package was that it allowed the “fuzzing” of interfaces i.e. a concentration gradient from one element to another. This function has been mimicked for use in SIMNRA using the SIMNRA file creator as described in Section 3.5.

2.2.2 Vegas

Vegas [24] is a full ion scattering simulation program which allows the user to create a crystal and perform Monte-Carlo ion scattering experiments upon that crystal. The outputs of a Vegas simulation are: total visibility in ML per detector angular segment, elemental visibility in ML per detector angular segment, layer by layer visibility in ML per detector angular segment, the visibility probability of each atom per detector angular segment, and the hitting probabilities of each atom.
2.3 MEIS instrumentation

Vegas works by comparing the hitting probability of the ingoing beam with a time reversed outgoing beam i.e. taking the reversed outgoing beam for each detection angle and looking at the hitting probabilities.

Vegas calculates the hitting probability by means of a gaussian probability distribution for each atom, equation (2.16). Sub surface atoms' hitting probabilities are calculated by using closer surface atoms' hitting probabilities and how they alter the flux of the beam at various depths, as shown in equation (2.17), where F is the normalised flux and G is the position distribution.

\[ P_{hitting} = \frac{1}{\sqrt{2\pi} \sigma^2} \int_{-\infty}^{+\infty} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]  

(2.16)

\[ H_i = \int F(\text{atom}_i)G(\text{atom}_i) \]  

(2.17)

After the time reversal calculations have been performed the total yield is calculated using equation (2.18).

\[ \text{Yield}_i = \int F_{in}(\text{atom}_i)G(\text{atom}_i)F_{out}(\text{atom}_i) \]  

(2.18)

This is a very basic view of how Vegas works, and is far more complicated than described, but in depth knowledge of Vegas is unnecessary in the scope of this thesis. For a more detailed treatment see [25].

2.3 MEIS instrumentation

The UK national MEIS facility is located at the CCLRC laboratories Daresbury. The experimental set up is shown in figure 2.9, and was used to perform all the experiments described in this thesis. The facility can be split into 3 parts: the beam source, the beamline, and the end station.
2.3 MEIS instrumentation

Figure 2.9 The UK national MEIS facility located at Daresbury

The ion source is a duoplasmatron and generates the ionised gas ($H^+$ or $He^+$ at the time of writing) that is then extracted from the source using a 20 kV potential into a linear accelerator that is able to increase the ions' energy to between 50 keV and 400 keV. Due to the geometry of the beam source, there is an energy spread of the beam of approximately 0.6 keV FWHM. For a more thorough treatment of duoplasmatron beam sources and other beam sources see [26], [27] and [28].

The beamline contains many beam defining, focusing and steering components. The main focusing elements are the two electrostatic quadrupole triplets (known as EQTs). The first is located before the bending magnet, the second after. The magnet has a function of a mass selector, since other ion species are created in the source, such as $H_2^+$ and $H_3^+$. The beam is then steered horizontally and vertically by a series of electrostatic steerers through a series of collimating slits to produce a final beam footprint 0.5 mm high, 1 mm wide with a divergence $< 0.1^\circ$. The beam current is measured by a tungsten mesh just before it enters the scattering chamber and is typically between 30 nA and 100 nA.

The experimental end-station facilities include four interconnected UHV systems
between which samples can be transferred under UHV; the scattering chamber, which houses the toroidal electrostatic ion analyser, sample goniometer and 2 dimensional (energy and angle) position sensitive detector and is where the ion scattering experiments are performed. The preparation chamber, whose facilities include LEED, Auger, sputter cleaning, evaporation sources and gas dosing and is used for sample preparation and characterisation prior to ion scattering experiments. The storage chamber, where six samples can be stored and which forms the junction of the sample transfer system. The loading chamber which is a fast pump-down chamber used for introducing samples into the vacuum systems. A typical operating pressure for the scattering chamber would be in the region of $1 \times 10^{-10} \text{mbar}$.

The TEA (Toroidal Electrostatic Analyser) uses the field created by positive and negative voltages to bend the scattered ions through $90^\circ$ and onto a detector, a pair of channel plates. Their purpose is to amplify the tiny charge pulse generated by each ion. A typical gain figure of $10^6$ means that each ion produces an electron pulse from the plates of $1.6 \times 10^{-13} \text{C}$.

The electron charge cloud produced by the channel plates impinges on an area detector. It is this two dimensional detector which determines the position of the charge cloud and hence the ion. The position of the ion relates to its energy and scattering angle. The area detector produces four charge pulses which are amplified, shaped electronically, passed through an ADC converter and processed digitally by the VME computer to become a 2-D data set.

The data acquisition system is controlled using MIDAS (Multi Instance Data Acquisition System) [29] and can be downloaded at the Daresbury laboratory web site. It is a graphical user interface written in the multi-platform scripting language TCL. The core functions have been written in lower level languages such as C and fortran. MIDAS can be used to control the goniometer, the TEA, data capture, and
basic data analysis.

For more information on the UK MEIS facility refer to the MEIS user manual [30].

2.4 Complementary Analysis Techniques

2.4.1 Low Energy Electron Diffraction (LEED)

LEED is a principal technique used in surface science. Its scope of use stretches from brief qualitative checks (e.g. to see if an evaporator is actually depositing) to in depth quantitative analysis. Qualitative analysis of electron diffraction patterns are concerned with the diffraction spot positions, giving information on the size, symmetry and rotational alignment of the adsorbate unit cell with respect to the substrate unit cell. Quantitative analysis examines the intensities of the various diffracted beams. This generates I-V curves which, when compared to theoretical curves, may provide accurate information on atomic positions.

Experimental Details

As this is a complementary technique, only the qualitative aspects of LEED are used, and thus will only be described in this report. The LEED experiment uses a beam of electrons of a well-defined low energy (typically in the range 20 - 200 eV) incident normally on the sample. The sample itself must be a single crystal with a well-ordered surface structure in order to generate a back-scattered electron diffraction pattern. A typical experimental set-up is shown in figure 2.10.

Only the elastically-scattered electrons contribute to the diffraction pattern; the lower energy (secondary) electrons are removed by energy-filtering grids placed in front of the fluorescent screen that is used to display the pattern.
2.4 Complementary Analysis Techniques

Figure 2.10 A typical LEED setup

Figure 2.11 Three-stage energy level diagram of the Auger process: Ionisation of a core energy level by a high energy electron beam (1 and 2), and relaxation and Auger emission (3)

2.4.2 Auger Electron Spectroscopy (AES)

Auger Electron Spectroscopy is a surface-sensitive spectroscopic technique used for elemental analysis of surfaces. It has a high sensitivity, typically 1% of a monolayer, making it a very useful technique for monitoring the surface cleanliness of a sample. It can also be used to calculate thin film thicknesses, working better on smooth elemental thin films. The Auger process is initiated by exposing a sample to a beam of high energy electrons (typically having an energy in the range 2 keV - 10 keV). These electrons have enough energy to ionise all levels of the lighter elements, and higher core levels of the heavier elements. See figure 2.11.

As shown in figure 2.11, Auger emission occurs when an electron falls from a higher energy level to fill a core hole created by a high energy electron beam. As
the electron falls from the higher level some of its energy is transferred to a second electron enabling it to leave the atom. The kinetic energy of this emitted Auger electron can be estimated using equation (2.19)

\[ KE = E_{IK} - E_{L1} - E_{L23} \]  

(2.19)

As this process occurs with core electrons, which are not involved with chemical bonding, the energy of the emitted Auger electrons will always be the same, thus uniquely identifying that element. Information can also be gained on a sample’s chemical state, but this is not applicable in any of these investigations. The kinetic energy of the emitted electrons are measured using usually a Cylindrical Mirror Analyser (CMA) or a Hemispherical Sector Analyser (HSA). Due to the large number of background counts the Auger scans are usually viewed in a differentiated form. These background counts come from many other electrons that have undergone a multitude of inelastic scattering processes. Figure 2.12 shows a typical differentiated Auger scan that was taken at Daresbury MEIS facility on a basic Auger set up (the main features being the 3 peaks (and dips) on the right and one on the left). This equipment is used to provide information on the cleanliness of a surface and to give an approximate figure of the thickness of any thin film overlayers. Film thickness calculations can be performed by comparing the Auger peak heights of a clean surface to one covered by an overlayer. The number of emitted Auger electrons are reduced by the overlayer by a calculable amount depending on the type of material. No film thickness measurements calculated using AES have been presented in this thesis. This is due to the fact that the system available at Darebury labs where all the experimental work has taken place is relatively basic and has an approximate film thickness measurement error value of ±1 ML. As the MEIS film thickness measurements are much more accurate only these values have ever been used.
2.5 Crystallography

2.5.1 Crystal growth classifications

Nearly 50 years ago the epitaxial growth of crystals was classified into three modes based on thermodynamical considerations [31]: Frank-van der Merwe (FM), Stranski-Krastanov (SK) [32], and Volmer-Weber (VW). See figure 2.13 [33], [34], [35]. In the meantime experimental evidence for growth modes have been collected on a large number of systems and it transpires that real growth processes can be more varied and complex compared to the generic models depicted in figure 2.13.

The surface energy balance described in equation 2.20 decides the proposed growth
modes, where $\sigma_A$ is the surface free energy of the adsorbate, $\sigma_I$ of the interface, $\sigma_S$ of the substrate.

$$\Delta \sigma = \sigma_A + \sigma_I - \sigma_S \quad (2.20)$$

In the case of $\Delta \sigma < 0$ pure two-dimensional growth (FM) is energetically favourable, figure 2.13[a]. When $\Delta \sigma > 0$, pure three dimensional growth (VW) is energetically favored. Between these modes the SK growth mode is situated: the film starts to grow in FM mode, but after several layers growth the energy balance changes such that $\Delta \sigma > 0$ and 3D growth is preferred.

The basic factors that determine these growth modes are of electronic and elastic origin: The strain in the film due to lattice mismatch and the chemical bonding between lattice and substrate. Perfect FM growth can therefore only occur in homoepitaxy.

### 2.5.2 Face Centered Cubic Structure - Fcc(111) surface

The fcc(111) surface [36] [37] is a close-packed, 3-layer repeating structure i.e. an ABC structure. Because MEIS blocking dip data is regularly compared to VEGAS computer simulations, the positions of all atoms in an ideal (or trial) structure needs to be known to a great detail. The following information will describe the geometries for a Copper fcc(111) surface, but can be easily translated to materials with different lattice parameters.

Figure 2.14 shows how the fcc(111) planes fall into relation with the main unit cell. The interplanar spacing of the (111) layers can thus be easily calculated. The (111) surface can be projected from the unit cell, providing us with a more accessible view of this surface, as shown in figure 2.15.

The description of this crystal is rather diagram heavy, but in order to be able to fully visualise the structure, especially in relation to major blocking directions,
2.5 Crystallography

Figure 2.14 A Copper face centered cubic crystal. The (111) planes (A,B,C) have been highlighted.

Figure 2.15 A projection of the fcc unit cell into the fcc(111) surface.
building a strong foundation of understanding is vital.

The final diagram, figure 2.16, used to describe this basic structure is a side view. This highlights the ABC structure, and is particularly relevant when twinning and interfacing with a hexagonal close packed (hcp) structure is being considered.

Using a string of geometric calculations, the individual positions for the Cu fcc(111) structure has been determined, as shown in table 2.1 [38].

A point that has to be taken into consideration when working with fcc(111) crystals is twinning. This is where the order of the ABC structure changes to A-C-B. Figure 2.16 shows how the atoms stack up in a left to right direction (in an upwards direction) e.g. layer A is shifted to the right of layer B, layer C is shifted to the right of Layer A, and so on. The direction of this growth can change to a right-to-left growth mode. Comparing the two growth modes by viewing normal to the surface,
Table 2.1 The coordinates for a Copper fcc unit cell. All units in Å. The \(<110>\) beam incidence direction can then be found at 90° azimuthal rotation and 35.26° polar angle in Vegas.

<table>
<thead>
<tr>
<th>Site</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>1.2781</td>
<td>2.2137</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1.4758</td>
<td>2.087</td>
</tr>
<tr>
<td>B</td>
<td>1.2781</td>
<td>3.6895</td>
<td>2.087</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2.9516</td>
<td>4.174</td>
</tr>
<tr>
<td>C</td>
<td>1.2781</td>
<td>0.7379</td>
<td>4.174</td>
</tr>
<tr>
<td>Repeat Distance</td>
<td>2.5562</td>
<td>4.4274</td>
<td>6.261</td>
</tr>
</tbody>
</table>

one is rotated 180° azimuthally to the other. These two growth modes can often be found intermixed in the same sample: twinning.

2.5.3 Twinned Face Centered Cubic Structure - Fcc(111) surface

Twinned fcc(111) growth occurs when the orientation of fcc crystal changes from its normal growth direction. The normal growth mode is shown in figure 2.16. When an fcc(111) surface is grown there can be three locations for the depositing atoms to settle. Say for example we had a flat fcc(111) surface with the uppermost layer being in position A. Atoms can now settle in sites A, B, or C. This can be seen on the left of figure 2.17. Normal growth would result in the atoms depositing in repeating order ABCABC etc. If the depositing atoms were to break this order and settle on a “C” site after an “A” site and form a full layer in this orientation then the next most energetically favourable site would then be a “B” site, creating a new order
2.5 Crystallography

![Diagram of crystal growth modes](image)

**Figure 2.17** A top down view on an fcc(111) surface. The left of the diagram shows the normal surface growth mode, with the 3 different deposition sites: A, B and C. The 3 arrows show the direction of the major channel growth in normal "ABC" growth. The right of the diagram shows twinned "ACB" growth. The 3 arrows on the right are a rotation of the 3 arrows on the left by 60°.

...of ACBACB. This can be seen on the right of figure 2.17 and on figure 2.18. This change in growth mode basically means that the crystal has been rotated by 60°, with the crystal only being symmetric after rotations of 120°. As MEIS is a real-space scattering technique the 60° rotation will be clearly noticeable given the right beam incidence and outwards directions as different crystallographic features will be visible.

Using this "ACB" model, a twin unit can be formulated and is shown in table 2.2.

If a sample was to be a 50:50 mix of twinned growth and normal growth, MEIS would show a combination of the two directions. If a reciprocal-space technique, LEED for example, was used, then a 6-fold symmetric pattern would be visible as the two 3-fold patterns would combine and the information that the two crystal lattices are offset by 60° in real space would be hard to obtain in the reciprocal transformation. This information can be extracted but would require a very quantitative approach and would not be obvious as it would be in MEIS. An example of extracting twinning
Figure 2.18 A side view of an fcc(111) crystal \(<112>\) azimuth) coloured in blue, topped by its twinned counterpart in pink \(<2\bar{1}1>\) equivalent azimuth). The dashed circle shows where the fcc continuation would have been. The arrows highlight the change in growth direction. Note the ABC to ACB layer change.

Table 2.2 The coordinates for a twinned Copper fcc unit cell. Note the B and C xy positions are the the same as the normally oriented cell, only the z positions have been interchanged. All units in Å

<table>
<thead>
<tr>
<th>Site</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>1.2781</td>
<td>2.2137</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2.9516</td>
<td>2.087</td>
</tr>
<tr>
<td>C</td>
<td>1.2781</td>
<td>0.7379</td>
<td>2.087</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1.4758</td>
<td>4.174</td>
</tr>
<tr>
<td>B</td>
<td>1.2781</td>
<td>3.6895</td>
<td>4.174</td>
</tr>
<tr>
<td>Repeat Distance</td>
<td>2.5562</td>
<td>4.4274</td>
<td>6.261</td>
</tr>
</tbody>
</table>
2.5 Crystallography

![Diagram showing Top and Side views of hcp structure]

**Figure 2.19** Top and side views of hcp structure. Large circles represent nearer atoms, smaller circles represent atoms in the next layer into the crystal. VEGAS unit cell enclosed in dashed box.

Information using LEED is shown in [39].

To further explore the structure of the fcc(111) structure see appendix A.

### 2.5.4 Hexagonal Close Packed Structure (hcp)

Hcp [36] [37] has a similar structure to the fcc(111) structure. The main difference being that hcp has an A-B structure and fcc has an A-B-C structure. Figure 2.19 shows a top and side view of the hcp structure from which atom position calculations were made. The diagrams are not to scale, but using symmetry and ratios, along with atomic spacing, a unit cell has been made up for hcp Cobalt (dashed), shown in table 2.3 [40].

If the top two layers of hcp(0001) and fcc(111) structures were to be examined, they would be indistinguishable, so care has to be taken when describing structure containing only a few layers.

### 2.5.5 Thermal Vibration

As MEIS is a real-space scattering event, the thermal vibrations of the atoms within the target are very important to take into consideration, especially when temperature
2.6 Data Analysis

2.6.1 Quantitative Analysis - Creating a scattering cross section

In order to perform a fully quantitative analysis upon ion scattering experimental results one needs to know the scattering cross-section calibration value. In the case of MEIS results, the cross-section value would take the format: \( \text{counts} Z^{-2} C^{-1} B^{-1} A^{-1} \text{channel}^{-1} \)

where \( Z \) is atomic number, \( C \) is beam dosage in Coulombs, \( B \) is the beam footprint in unit area, \( A \) is per atom, channel is per detector angular channel. From this calibration value we can calculate how much material is present in MEIS scans.

The cross-section calibration value can be obtained from layer-aligned MEIS scans of clean samples. For example: if the experimental system’s substrate is Cu(111)
then we would align the inwards-bound beam to illuminate the top layer i.e. along the $<110>$ direction. The energy-integral of the surface peak along one angular direction would then be extracted. A Vegas simulation then needs to be taken with the exact same system and beam alignment. The number of visible monolayers along the same angular channel as the physical experiment can then be extracted from the simulation. The experimental integral value then needs to be divided by the number of visible monolayers, the aerial density of the sample ($\approx 10^{15} \text{atmm}^{-2}$), the dosage in Coulombs, the beam footprint ($\cos \Theta$ where $\Theta$ is the beam-in angle to the surface normal), the number of channels the energy integral was taken across, and $Z^2$. To improve accuracy this calculation can be taken across a whole sample, but each channel's integral would need to be compared to each channel in the sample.

A more simple way to improve accuracy is to take integrals and simulations across a variety of sample alignments e.g. 1, 2 and 3 layer alignments. In the case of the PdCu(111) experiments, as shown in Chapter 4, the calibration factor varied across inwards alignments by approximately $<1\%$.

Once the scattering cross-section has been calculated the amount of visible material on other scans can be calculated. If the aerial density of an overlayer is known then the number of visible monolayers can be also calculated.

### 2.6.2 The R-factor

Throughout this thesis a "goodness of fit" calculation is needed to basically determine how well a simulation represents experimental data. This quality measure is known as the R-factor.

The R-factor is defined as:

$$R = \frac{1}{N} \sum_{1}^{N} \left[ \frac{(I_{exp} - \lambda I_{sim})^2}{I_{exp}} \right]$$  \hspace{1cm} (2.21)
where \( N \) is the total number of points being compared, \( I_{\text{exp}} \) and \( I_{\text{sim}} \) are the experimental and simulated intensity values (this can also be a monolayer visibility or ion counts at a given point), and \( \lambda \) is a scaling factor used to allow for instrumentation effects such as transmission efficiency and acceptance angle of the ion energy analyzer. In all cases listed here \( \lambda \) is taken to be 1.

This particular R-factor equation has been used as this is the standard fit quality factor used throughout MEIS experiments [41]. As \( I_{\text{exp}} \) equals \( \sigma_{I_{\text{exp}}}^2 \) where \( \sigma_{I_{\text{exp}}} \) is the error of the experimental count rate the definition of the R-factor here is equivalent to the definition of the \( \chi^2 \) value.
Chapter 3

Macros and Programming

Several Igor Pro [42] macros have been written to form a suite of software that is designed to re-tile, calibrate, extract integrals and interact with a commercial Rutherford backscattering package. The first three macros have been used for all experiments presented in this chapter. The final macro has been used in the next chapter detailing the Pd on Cu(111) experiment. A full description of usage and underlying functional structure follows.

3.1 Introduction

MEIS is a relatively new and sparsely used technique, thus the availability of software used to process the large amounts of measured data is slim. The current software used to gather the experimental results at the Daresbury MEIS facility is called MIDAS (Multi Instance Data Acquisition System). This is a multi-platform piece of software and runs through tcl (similar to Java). MIDAS is suitable for gathering data, including hardware interfacing, tiling data, performing energy and angular cuts, and other basic data processing operations. Custom data processing routines
3.2 Super Mega Curvefit

Figure 3.1 A cut from a 2D data fit. The different colours indicate the various components of the final curve.

can be added to MIDAS but would require knowledge of tcl and interfacing to the MIDAS system. Most custom numerical processing tools have therefore been written as Igor Pro macros. Igor Pro is a scientific graphing, data analysis, image processing and programming software tool designed for scientists and engineers. It contains a scripting language that is similar to C and can be learned very quickly. The following sections describe the functioning of custom-made Igor Macros, whose source code can be found in Appendix C.

3.2 Super Mega Curvefit

This Igor Macro essentially fits a series of curves in the energy plane of full 2D MEIS scans. It extracts single energy scans from every angle channel one by one and fits a curve to that. It only fits curves to crystallographically aligned MEIS scans (as opposed to random alignments). A user interface system has been developed for this to significantly speed up the process of data fitting.

The curve that is fit is a sum of a sigmoid-straight-line product, and two Gaussians.
The sigmoid and linear product is for the gradual de-channeling background noise, and the two Gaussians model the surface and substrate peak. See figure 3.1 and equation 3.1.

\[ \text{counts} = \frac{mx + c}{1 + e^{-\frac{x-x_0}{s_0}}} + a_1 e^{\frac{(x-x_1)^2}{k_1}} + a_2 e^{\frac{(x-x_2)^2}{k_2}} \] (3.1)

The curve fitting algorithm needs best guesses for each parameter of the whole curve. These are surface and substrate gaussian centre-points and widths, the sigmoid centre-point and sigmoid width. These are entered manually and the rest of the parameters e.g. gaussian heights, are extracted from the previously entered parameters.

The best guess parameters take the form of polygons on the 2D tile. The shape of these polygons can be changed by the user interface panel on the left side of the macro main screen. These vary the experimental parameters e.g. surface interplanar spacing, altering the output of kinematic and energy-loss calculations.

Figure 3.2 shows the user interface panel. The MEIS data is loaded into the macro as .tile files taken straight from the MIDAS binary outputs. The TEA angle, beam angle in, start angle, angle inc, calib energy, and energy inc are all extracted from the experimental parameters stored in the tile file. The rest of the data has to be manually input. The first group of “interplanar spacing”, “no. of layers”, and “energy loss” are parameters (B in figure 3.2) for the surface Gaussian, the next 3 (C in figure 3.2) are for the substrate Gaussian. Section A describes the initial positioning and shape of the substrate Gaussian. The “ender bender” tick box and max shift selector allows for the slight bending of the results that can be explained as calibration shifts. This is a function of the instrument. The ender-bender makes the polygon follow the contours of the data for better initial guesses. A typical value for the max shift would be 10-15 (pixels). The angle slice point is for checking the
Figure 3.2 The user interface panel of the Supermegacurvefit Igor macro.
3.2 Super Mega Curvefit

parameter-polygon-data intersection. It produces an energy-slice graph and shows where the parameter-polygon cuts through. The "calculate polygon" button updates the shape of the parameter-polygon. The "Speed fit" button fits the data using simple Gaussians for the surface and substrate. This is effectively the "go" button. The "slow fit" button uses a Gaussian summed for the relative isotopic frequencies. All the isotopic data is stored in the root:isotopes: data folder in the Igor Pro environment. The user does not need to worry about this as it is all calculated automatically and all that is needed to do is to enter the relevant atomic number. This calculation takes a lot longer and usually does not make much of a difference for the experiments shown in this thesis.

The "show calculated slice" button shows the fitted curve at a given angular channel. It breaks the whole curve up into its 3 constituent parts, and produces a graph similar to that in figure 3.1. The "load calib file" accepts a calibration file that is a list of angular offsets for each angle channel - in an Igor text (.itx) file format.

The "apply calib" button applies this calibration to the integrals, re-bins the data in equally spaced angular increments, and saves them as an .itx file. All the information that is on the user panel is also stored in the array exp-parameters. The arrangement of this data in the array is listed in the appendix. The best guesses for the curve fitting algorithm are stored in the array coef.matrix, with their specifics listed also in the appendix.

One very important array is n.const. This stores the constraints of the best guesses for the curve fitting algorithm. These have to be carefully selected because if they are too large then the algorithm can go haywire, too small and the algorithm will never reach its best fit.

The "ender-bender" routine works in quite a simple way. It tracks across the uppermost polygon line and measures the number of counts intersecting it. Ideally
3.2 Super Mega Curvefit

these should be constant after Rutherford correction. If they are not then the curve should be shifted so it is closer to being constant. This is done by normalising the number of counts intersecting this line to the “ender-bender max shift”. Each polygon curve is then shifted up by this normalised amount. This method appears to perform quite well, but may not for surface peaks that contain large blocking dips.

The final saved integrals are calibrated (if necessary) and Rutherford corrected. They are stored as an Igor text file as these are easier to load back into Igor pro for further analysis. Igor text files store the data as plain text so is compatible with other data analysis packages.

There is one known bug that is experienced with this macro. Sometimes when the binary tiles are loaded the counts are unusually high. This is because the binary data has not been read in correctly and can be corrected by adding \B to the first GBLoadWave command in the load_wave() function.

There are several improvements that could be made to this macro. Currently the user has to fiddle with the best guesses and the constraints to which the best guesses vary to make the fits work. If these parameters are not suitable then the fitting progress can go haywire and the curves will not get fitted. This may be solvable, but also may be inherent to the in-built curve fitting algorithms essential to Igor Pro.

Another improvement would be to fit more than two gaussians. In the case of two surface elements this would prove to be very useful as the data is very likely a crossover making the separation of the individual integrals very hard using existing data reduction techniques. This would take a reasonable amount of programming but would not be too difficult.
3.3 Match Data Macro

One major problem with the MEIS data is that the angular data would not be correct: the blocking dips of the bulk data were not found where they should be, and sometimes they were completely out. Sometimes the angles would be all systematically out due to the drive rotating the TEA slipping, but sometimes the blocking dips were individually out by different amounts. A solution was suggested during a MEIS users' conference [43], upon which the following macro was written.

This macro was also written using Igor Pro and is used to calibrate angular cuts of data. It functions by comparing a bulk angular cut of the raw data to a Vegas simulation, creating an angular offset file for each angular channel, and then re-bins the data for chi-squared or R-factor comparisons.

As the calibration procedure is a long and arduous process the graphical user interface has been designed with ease and speed of use in mind. Several functions have been designed to eliminate many of the more repetitive tasks.

The calibration process consists of several stages. The first stage is loading and finding the peaks of a bulk-data cut and of the simulation, see figure 3.3. The data may need to be smoothed as otherwise noise would be detected as blocking dips, see figure 3.3b.

Once the blocking dips have been found, the simulation and data needs to be compared. This can be done by using the graph and pull-down menu as shown is figure 3.4. The red and blue peaks show the location and heights of the various raw data and simulation data, with their angular locations on top. The menus have been designed to be as intuitive as possible. From this comparison a calibration curve can then be generated, as shown in figure 3.5. The calibration curve can be a cubic-spline fit but this was not used in this investigation. This would present an
3.3 Match Data Macro

Figure 3.3 First stage of calibration. Raw data with detected peaks (a), and simulated data with detected peaks (b).

Figure 3.4 The second stage of calibration. A comparison of the blocking dip locations and the user selection box to apply the comparisons.

issue with discontinuities in the gradient of the calibration curve but due to the small number of points from which the curve was generated from, there was a risk of “over interpolating” i.e. the spline curve could over estimate the calibration values so the spline could fit better.

After the calibration curve has been created, the angular scale of a selected data set can be added to the offset creating a new more accurate angular scale. As the simulation data has an equally spaced angular scale, the calibrated data needs to have the same scale. This is achieved by creating a duplicate angular scale for the raw data and interpolating the new count values. The inaccuracies created by this
3.4 Tiler Macro

Figure 3.5 A calibration curve for the data shown in the previous two figures

interpolation are negligible.

The array offset values are stored in the array offset_interp and can be imported into the Super Mega CurveFit macro using the in-built function. This process has been used for all data sets in this thesis.

3.4 Tiler Macro

The Tiler macro is a program written to run in Igor Pro V5 and will not function in earlier versions of Igor. It is essentially a MEIS data handling package. It was written to deal with the current inadequacies of the MIDAS data handling package which is the current norm for MEIS users.

The main problem that the Tiler macro has addressed is that on the November 2005 AgAl run reported in Chapter 5, the data “windows” that were being tiled together were too big and overlapped. See figure 3.6a. MIDAS software does not deal with this problem. The Tiler macro allows the user to “shave” rows off the top and bottom of tiles to create a more correct tile, thus converting figure 3.6a into figure 3.6b.

As the MIDAS software only deals with data-formats that have been tailored for
Figure 3.6 Full MEIS tiles with (b) and without (a) shaving off the top and bottom 5 pixels of each tile.
its own usage, a whole environment had to be written in Igor to perform similar functions to what MIDAS can perform.

These include performing energy cuts, angular cuts, and $K^2$ corrections to 2D tiles. There have also been a lot of functions that have been omitted from Tiler that can be found in MIDAS. This is because they are unnecessary for the scope of this thesis and the nature of these investigations, making the user interface a lot more simple and easier to use. This makes the processing of large amounts of data (quite common for a 2 week MEIS experiment) a lot quicker.

3.4.1 The Tiling process

The first and last tiles of a sequence are selected by the user e.g. run65619.2d.data and run65630.2d.data. The function `load_fillers` then loads all the in-between files by extracting the file numbers, in this case 65619 and 65630, and then increments the first number by one, concatenates the original file-name text to the start and end of the number e.g. “run” and “.2d.data”, and loads this new file name. This process is repeated until the incremented file number is equal to the last file number.

The major problem encountered in the tiling process is in the energy scales. Each individual data file has a different pixelar energy increment i.e. the energy range covered by the same number of pixels in each data file will be slightly different, creating a very uneven energy scale, as shown in figure 3.7. Analysis programs however, require that the energy scale is linear, or mappable by a quadratic equation. This means that the tiled sets of data has to be re-binned in its energy scale. Also, with the shaving off of the top and bottoms of each tile, this creates a further problem, both of which have been addressed by the tiler macro.

With the shaving off of the top and bottoms of the data tiles, gaps in the energy scale will also appear, see figure 3.8 left. This has been resolved by shifting the
3.4 Tiler Macro

Figure 3.7 An example of the varying energy scales of the tiles produced by MIDAS.

Figure 3.8 An example of how the energy scales are affected when tiles are just shaved off (grey areas represent the shaven off areas)

maximum energy on each tile to be located where it has been shaved off, and then changing the pixelar energy increment so that the full energy range in contained within a smaller pixel range. This prevents discontinuities in the energy scale. See figure 3.8 and figure 3.9).

The energy scale for each window gets rescaled, and then concatenated to form a non linear energy scale, named first_energies, for the full data tile. A further function, interp_new_full_tile(), then creates a new linear energy scale, named second_energies, whose pixelar energy increment is obtained from the first window
3.5 SIMNRA file creator

The main output of this macro is in the form of a 2-dimensional array which is the retiled data, with one column storing the experimental parameters. This data output can then be easily imported into the previously mentioned Super Mega CurveFit with the majority of beam parameters also imported thus speeding up the curve fitting method. This procedure was followed for all data sets in Chapter 5.

**Figure 3.9** An example of how the energy scales are rescaled once they have been shaven off.

and is then continued throughout the other windows.

The experiments listed in this thesis pushes the programs to their limits as the films involved are ultrathin meaning film roughness and interface concentrations play vital parts.

Two of the more popular programs are RUMP [23] and SIMNRA. RUMP is notoriously difficult to use, being described in its own manual as “Really Ugly and Mangled Procrastination tool”. This program was tested and deemed unsuitable. The only benefit of this package was that it allowed the “fuzzing” of interfaces i.e.
3.5 SIMNRA file creator

Figure 3.10 The user interface panel of the SIMNRA file maker Igor macro.

a concentration gradient from one element to another. SIMNRA does not offer this function, but due to its simple user interface and use of Windows OLE Automation a solution was found.

Using SIMNRA one can create a concentration gradient by creating multiple thin layers of gradually changing elemental concentrations. Doing this manually would take a very long time, especially if trying to fit data. If SIMNRA is controlled by a Visual Basic Script it can be done very quickly. An Igor Pro macro was written that produces Visual Basic scripts to control SIMNRA. This macro allows the user to design thin films and vary one parameter between two fixed points to find a best fit.

Figure 3.10 is the user interface panel for the Igor macro. It allows the user
3.5 SIMNRA file creator

Figure 3.11 A sample profile designed in SIMNRA-file-maker. This particular graph shows a diffusing Palladium film on a Copper substrate.

to input the beam experimental setup e.g. number of incident ions, beam-in angle, scattering angle etc. The user can then set the number of main layers, the start and end concentration of up to three elements in each layer, and the number of roughness steps. This will produce a thickness profile graph as shown in figure 3.11.

Figure 3.12 shows the parameter variation panel used in SIMNRA-file-maker. This allows the user to automate SIMNRA to perform multiple simulations to narrow down a particular parameter. The user can vary the concentration profiles of a particular layer or the thickness of a particular layer. When the concentrations are being varied the user can also select whether to have the concentrations from one layer to the next to be continuous. As the number of simulations can be large (figure 3.12 shows 400) the individual simulation filenames are generated and stored so the loading of the simulation results can be done automatically in the same Igor macro. A simple Chi-squared test is then employed to find the best fit parameter.

One important pre-simulation check that has to be made is that the default beam type of SIMNRA is changed to $H^+$ ions as opposed to $He^{++}$ which is the default after initial installation of the program. This can be easily done by going into the /Default directory contained in the SIMNRA main directory and altering the setup.nra file.

This macro has been used for the PdCu(111) system for depth profiling.
Figure 3.12 The parameter variation panel of the SIMNRA-file-maker Igor macro.
3.6 Summary and Comments

Several custom made programs that deal with MEIS data processing and analysis have been presented. They stretch from the very first stages of data processing through to some of the final stages of data reduction and analysis that do not require human intelligence.

The Tiler macro is involved in the first stage of data processing and its main function is to tile the data sets and deal with tiling errors that cannot be resolved in the current MEIS standard data processing package, MIDAS. The output of this macro is a 2-dimensional data tile which can be easily imported into SuperMega CurveFit.

The matchdata macro deals with the calibration of scattering angle vs. counts data. It requires an angular cut of the bulk data which are taken as the reference blocking dips, which it then compares to Vegas simulations to create calibration files. This can then be used to correct further data sets or to output a calibration file which can then be imported into Super Mega CurveFit.

The Super Mega CurveFit macro is used to extract surface, substrate and bulk integrals which are calibrated and Rutherford corrected. This output data can then be used for direct analysis. The macro uses Igor's in-built curve fitting routines so as to extract the integrals by using user input parameters.

Super Mega CurveFit comes into its element when the surface, substrate and bulk integrals overlap which would make standard polygon-based extraction techniques difficult and erroneous. Due to the macro fitting individual curves for each of these elements, the separate integrals can be easily extracted.

There are cases where the use of Super Mega CurveFit is unnecessary. The first case is where the surface integral is separated enough from the substrate integral so
that a polygon-based or simple projection method can be used more quickly. The other similar case is where the substrate and bulk integrals do not overlap where again, a simpler extraction method can be used. In the majority of cases shown here this was not the case however.

A significant effort has been made to make sure that these three previous macros can interface with each other, meaning that they form a more complete suite of software that can process large sets of data with greater efficiency. This means that anyone undertaking an experiment can focus less on the arduous but very important process of extracting useful information and focus more on the analysis stage. Another benefit of using these macros is that the use of a robust data processing suite prevents mistakes being made early on in the data reduction stage which would then to great dismay follow through to the final analysis stage.

The final macro, SIMNRA file creator, is used to interface with the popular Rutherford Backscattering simulation program, SIMNRA. The main function of this macro is to generate crystallographic depth profiles that could not normally be created in the current interface of SIMNRA; specifically, samples that have an interfacial concentration gradient. A curve fitting algorithm has also been produced that sweeps through a range of values for a given sample's geometric parameter, such as film thickness, and outputs an R-factor curve. This is also an improvement on the current curve fitting algorithm in SIMNRA that only produces a best fit curve with its associated parameters. This macro was used only for the PdCu(111) system and results are outputted in Chapter 4.
Chapter 4

Pd on Cu(111)

Medium Energy Ion Scattering has been used to crystallographically depth profile ultrathin films of Pd on Cu(111). The analysis of the Pd on Cu(111) system has been approached through various MEIS analysis techniques using a variety of custom made software. Twinning has been found and quantified for this system, along with the Pd absorption percentages for the first three substrate layers for 0.25 ML coverage. The interplanar surface relaxation has also been quantified for the top three layers for 0.25 ML coverage.

4.1 Introduction

Ultrathin films of Pd on Cu(111) have been investigated using a number of theoretical and experimental techniques in previous literature. Previous experimental techniques include LEED, X-ray photoelectron diffraction (XPD) combined with LEED and Scanning Tunnelling Microscopy (STM).

A number of studies have also been performed on the converse system, Cu on Pd(111) using reflection high energy electron diffraction (RHEED) combined with
4.1 Introduction

X-ray photoelectron spectroscopy (XPS).

Panagio et al. [44] have used RHEED to investigate the growth of Pd on Cu(111) at room temperature. They show that at lower coverage the Pd grows with the in-plane Cu(111) lattice parameter. With increasing coverage the lattice parameter rapidly changes to that of Pd(111). They propose the mechanism for this increase is by alloying and Vegard’s law to release the stress of the 7.6% lattice misfit i.e. Pd has a fcc structure with lattice parameter 7.6% greater than that of Cu. They found that with a coverage > 2ML pure Pd layers are formed based on the fact that the lattice spacing at this coverage is that of pure Pd. They conclude that in early stages of growth, up to one ML, a random alloy is formed. They also raise the question of whether at some thickness the Pd-Cu transition is “de-alloyed” to form a sharp transition as observed in the Ag-Cu(100) system.

Bach Aaen et al. [45] have studied sub-monolayer Pd on Cu(111) growth using STM, RBS, and AES. At deposition temperatures relevant to this investigation (< 100°C) they found that the Pd essentially nucleated at ascending steps in fingered brims, and on large terraces, in fingered islands. The lack of order suggests that the brims and islands are a disordered alloy formed by an exchange between the Pd and Cu from the layer underneath. They have explained this by using a calculation of the surface energy. Once the population of nearest-neighbour sites is excluded there is practically no energy gain by ordering. The main subject of discussion of this paper is the area and shape of the brim, which is not relevant to the MEIS technique.

A. de Siervo et al. [46] have studied approximately 1 ML of Pd on Cu(111) using X-ray photoelectron diffraction (XPD) in combination with LEED. They found that a random surface alloy was formed in at least the first three layers, with the first interlayer distance expanding by 5%, whilst the second contracts by 2%. The Pd concentration was found to be higher in the second layer than the first and third
layers, with approximately 20%, 70%, and 20% of the deposited Pd found in the first, second, and third layers.

A Canzian et al. [47] have produced a theoretical model for the above listed experimental system. They confirmed experimental observations of surface alloying, the presence of sub-surface Pd, the formation of Pd bands off terrace steps, and weak short range ordering. The depth of the Pd-Cu substitution was found to be temperature dependent. The BFS method used disagrees with the interlayer spacing found in de Siervo et al [46]. They are calculated here to be $-1.83\%$ and $+2.62\%$ for the first and second layers. Calculations also showed that a $(\sqrt{3} \times \sqrt{3})R30^\circ$ structure could be formed as surface energetics preferred for Pd not to be nearest neighbours, but this may not always be formed due to the diffusion of Pd into the bulk.

Bozzolo et al. [48] have theoretically studied the structure of Cu$_{1-x}$Pd$_x$ alloys. The two alloys formed are CuPd and Cu$_3$Pd. These have equivalent bulk structures of Cu$_3$Au (L1$_2$) and CsCl (B2), face-centred cubic and body-centred cubic respectively. The binary alloy phase diagrams [49] indicate that at room temperature these structures could possibly form anywhere from 10% Pd through to 67% Pd. However, the phase diagram indicates that the ordered phases need a temperature of 500 $^\circ$C to 600 $^\circ$C to form. Below 500 $^\circ$C the ordering process takes a considerable amount of time such that metastable disordered alloy can be realised. Based on the results of the phase diagrams one expects a random substitutional alloy, with Pd and Cu settling in fcc positions [45]. This agrees with experimental findings in the present literature [45].

The converse system of Cu on Pd(111) has been very sparsely studied [50], [51], [52], [53]. The most recent paper by Siervo et al. indicate FV growth with a discontinuous lattice spacing change from the natural Pd(111) to Cu(111) with a 4 ML coverage of Cu.
So in summary it can be seen that there is very strong evidence for the Pd to form a random substitutional alloy \[44\] \[45\] \[46\] \[47\] which, at coverages up to 1 ML can absorb up to 3 ML into the Cu. The concentrations \[46\] measured were found to be 20%, 70%, and 20% in the first, second, and third layers. Experimental measurements showed that there was a relaxing of the first and second layers of 5% and \(-2\%\) \[46\], whilst theoretical studies showed \(-1.83\%\) and \(+2.62\%\) \[47\]. After 2 ML coverage it has been shown that the Pd returns to its natural lattice spacing \[44\].

### 4.2 Experimental Details

A standard fcc(111) Cu crystal stock sample from the Daresbury MEIS facility was used in this experiment.

The sample was initially prepared by cleaning using 1 keV to 1.4 keV Ar\(^+\) ion sputtering at normal and grazing incidences, followed by flash annealing to 450°C (i.e. raising up to the annealing temperature for a few seconds and then allow to cool to room temperature for approximately 2 hours). Before annealing, the crystal was checked using Auger for any metal contamination as annealing with a contaminant would run the risk of alloying and deep intermixing. The sputtering and annealing cycles were repeated until no contaminants were visible using Auger, and a \((1 \times 1)\) fcc(111) pattern was visible using LEED.

The Pd was then deposited onto the Cu(111) crystal at room temperature using an in-house made electron-beam evaporator. An ultrasonic cleaned piece of 1 mm thick Pd wire was used as the feedstock for the evaporator. Previous deposition rates for other materials used in the same evaporator (calibrated by MEIS) were used as a starting point for the Pd evaporation.

The conditions used for the evaporator were as follows: Filament 5 A, Anode 2 kV,
4.3 Results and Discussion

discharge current 1.75 mA, base pressure $6 - 8 \times 10^{-8} mb$. The base pressure tended to vary due to the slow response of moving the feedstock to control the pressure. Due to the number of variables to control, and the general unpredictable nature of e-beam evaporation, the same conditions and same deposition time would not necessarily produce the same thickness coverages.

After Pd thin films had been produced, the amount of Pd was quantified using Auger and through approximate MEIS calculations. The MEIS calculations take the ratio of the $Z^2$ corrected integrals of a 1 ML aligned sample. This would generally give a coverage amount to an accuracy of $\pm 0.5$ ML. Channelling effects would make this value less accurate, but this method was very quick for the accuracy obtained.

The attempted coverages based on the continual calibration of the evaporator were $1/3$ ML, 1 ML, 2 ML, 3 ML, and 9 ML. All MEIS measurements were taken at room temperature.

4.3 Results and Discussion

As the size of the data set is large i.e. a multitude of layer thicknesses and beam incidence angles, the depth of analysis goes from very qualitative to more in-depth quantitative treatments.

The MEIS thickness measurements shown in table 4.1 were calculated for each channel across the tiles and then averaged. In several cases the average was only taken across selected chunks of the tile as the substrate signal tended to overlap the surface signal, or shadowing effects were observed. The error was taken as the standard deviation of the thicknesses calculations. The MEIS calculations were performed using calibration values taken from a clean Cu sample. This method is described in Chapter 2.
4.3 Results and Discussion

Table 4.1 MEIS random alignment layer thickness measurements. Measurements were made for each angular bin across the whole tile and then averaged, errors were calculated from the standard deviation of these thickness measurements.

<table>
<thead>
<tr>
<th>Run number</th>
<th>MEIS(ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80842</td>
<td>0.80 ± 0.02</td>
</tr>
<tr>
<td>81003</td>
<td>1.87 ± 0.04</td>
</tr>
<tr>
<td>81177</td>
<td>1.86 ± 0.06</td>
</tr>
<tr>
<td>81368 (81177 anneal)</td>
<td>1.48 ± 0.15</td>
</tr>
<tr>
<td>81492</td>
<td>3.89 ± 0.12</td>
</tr>
<tr>
<td>81743</td>
<td>0.25 ± 0.02</td>
</tr>
<tr>
<td>81878 (81743 anneal)</td>
<td>0.26 ± 0.06</td>
</tr>
</tbody>
</table>

The samples were then analysed using SIMNRA and the SIMNRA file-creator Igor Pro macro (see Chapter 3.5 for details). This then allows a depth profiling of the sample to be made. Table 4.2 shows the comparison of the film thickness measurements from a basic MEIS calculation and a variety of SIMNRA film thickness measurements. The first MEIS SIMNRA integral is essentially an integral of a Pd concentration vs. depth graph i.e. the total thickness of Pd if it were to be taken as layer-by-layer growth. The MEIS 66% initial concentration column is a measurement of how deep one has to go into the sample before the Pd concentration reaches 66% of the concentration at the surface. The last two columns in table 4.2 show the equivalent number of monolayers of Pd present if the same amount of Pd found in the SIMNRA integral (column 3) was to grow in a layer-by-layer fashion. Two lattice spacings were used to calculate these numbers, that of Cu(111) interplanar spacing (2.087 Å) and that of Pd(111) interplanar spacing (2.255 Å).

It can be seen in table 4.2 that the SIMNRA results (last two columns) are slightly
### Table 4.2 Various MEIS results - Surface integrals compared to SIMNRA integrals and corresponding number of monolayer coverage based on Pd and Cu lattice spacing.

<table>
<thead>
<tr>
<th>Run number</th>
<th>MEIS (surface integrals ML)</th>
<th>MEIS SIMNRA integrals&lt;sup&gt;1&lt;/sup&gt;</th>
<th>MEIS 66% initial conc.&lt;sup&gt;2&lt;/sup&gt; in SIMNRA</th>
<th>SIMNRA thickness spacing&lt;sup&gt;3&lt;/sup&gt;</th>
<th>SIMNRA thickness spacing&lt;sup&gt;4&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>80842</td>
<td>0.80 ± 0.02</td>
<td>0.23 nm</td>
<td>0.36 nm</td>
<td>1.10 ML</td>
<td>1.02 ML</td>
</tr>
<tr>
<td>81003</td>
<td>1.87 ± 0.04</td>
<td>0.46 nm</td>
<td>0.27 nm</td>
<td>2.20 ML</td>
<td>2.04 ML</td>
</tr>
<tr>
<td>81177</td>
<td>1.86 ± 0.06</td>
<td>0.39 nm</td>
<td>0.30 nm</td>
<td>1.88 ML</td>
<td>1.75 ML</td>
</tr>
<tr>
<td>81368&lt;sup&gt;5&lt;/sup&gt;</td>
<td>1.48 ± 0.15</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>81492</td>
<td>3.89 ± 0.12</td>
<td>0.81 nm</td>
<td>0.60 nm</td>
<td>3.88 ML</td>
<td>3.60 ML</td>
</tr>
<tr>
<td>81743</td>
<td>0.25 ± 0.02</td>
<td>0.06 nm</td>
<td>0.025 nm</td>
<td>0.29 ML</td>
<td>0.27 ML</td>
</tr>
<tr>
<td>81878&lt;sup&gt;5&lt;/sup&gt;</td>
<td>0.26 ± 0.06</td>
<td>0.08 nm</td>
<td>N/A</td>
<td>0.38 ML</td>
<td>0.36 ML</td>
</tr>
</tbody>
</table>

<sup>1</sup>Integrated from fitted profiles.  
<sup>2</sup>Taken from fitted profiles.  
<sup>3</sup>Assuming film has Cu lattice spacing.  
<sup>4</sup>Assuming film has Pd lattice spacing.  
<sup>5</sup>Annealed sample.

Higher than the basic integrals. As this new technique of interfacing another program with SIMNRA really pushes SIMNRA to its boundaries of accuracy, as described in the SIMNRA manual [19], the results using this technique should be taken with caution.

A point to be made from the results in table 4.2, especially the last two columns, that they do not reflect the nature of the PdCu(111) system. Simple monolayer coverage values are appropriate for systems that clearly exhibit layer-by-layer (or thereabouts) growth and will start to lose meaning when a sample is intermixed. The following results further illustrate this point.

Figures 4.1 to 4.6 show the SIMNRA simulation results compared to the exper-
4.3 Results and Discussion

Experimental energy slices and the corresponding sample profile. It should be noted that even though the beam was put in at a random alignment there was still some blocking and shadowing effects observed. This is particularly noticeable on the full 2D tiles for runs 80842, 81743, and 81878 (figures 4.1, 4.5 and 4.6). Because of this, the chi-squared values used to fit the samples were calculated from the substrate peak upwards. This slight blocking and shadowing effect will produce uncertainty in the results.

It can be seen that some of the simulations do not fit the data as well as others. The main reason for this is due to the fact that systems being simulated are unknown and the model will not entirely reflect the exact nature of the system. Also, there may be more complex phenomena occurring such as channelling and non-linearity of the energy scale.

A possible source of inaccuracy is due to the fact that the sample models are treated as having a continuously changing concentration gradient whereas the true depth profile realistically should have a much more discrete depth profile. i.e. for the 0.25 ML coverage the top 3 layers in the sample, for example, could contain all the Pd making the concentration drop in steps related to how many layers one has travelled into the sample. The main point drawn from this is that the method of analysing the sample could be ascribing too much detail and presenting it in a way that is unsuitable for the physical system. Using SIMNRA to generate models can be good for gathering a quick picture of the system, attaining such details as the film thickness and general trends like the presence of material intermixing.

Examining figures 4.1 to 4.6 does reveal general trends. All samples appear to have an intermixed interface, with this intermixing becoming very extreme after annealing the sample to 450°C. The effects of annealing are consistent with the bulk phase diagrams. The intermixing signals and increased rate of atomic diffusion however the
annealing process was not sufficient to yield ordered alloys. It would be of interest to explore this possibility further with a more systematic study of annealing conditions. The initial concentration of the Pd surfaces on the samples appears to vary from 60% to 100% and drops steadily to 0% the deeper one goes into the sample. For the coverages 0.25 ML to 1.8 ML the intermixing region appears to be approximately 3-6 times the interlayer spacing times the equivalent monolayer coverage. So 0.8 ML coverage has an intermixed region of approximately 5 ML deep, where the monolayer thickness is that of Pd. A more specified trend does not seem apparent as there is a large variation in the models.
4.3 Results and Discussion

Figure 4.1 Run 80842, 0.80 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The first peak at 101.8 keV is the Pd, the second at 100 keV is the Cu substrate. Ideally at a random incidence the second peak should merge into the bulk but due to a crystallographic alignment blocking occurs. The sample profile (bottom) indicates that the Pd and Cu are mixed approximately 50:50 for the first 1.5 ML (3 nm) and then drops to 0:100 Pd:Cu in the second 1.5 ML.
4.3 Results and Discussion

Figure 4.2 Run 81003, 1.87 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The Pd concentration appears to steadily drop from 100% through to 0% by travelling 1 nm into the sample, approximately 5 times that of the natural Pd lattice spacing.
Figure 4.3 Run 81177, 1.86 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. Even though this sample appears to have the same thickness film as the previous figure, run 81003, the model profile and experimental cut appears to be reasonably different. This could be due to the slightly different “random” alignment or actual sample differences. The Pd material appears to be more towards the surface compared to run 81003, but penetrates deeper into the substrate at a lower concentration.
4.3 Results and Discussion

Figure 4.4 Run 81492, 3.89 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. The Pd concentration appears to steadily drop from 100% to 0% in the space of 1.6 nm. The experimental results appear to be very well aligned with the bulk signal merging completely with the substrate peak, indicating that beam alignment is at an optimal random alignment.
Figure 4.5 Run 81743, 0.25 ML Pd on Cu(111). Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. Examining the sample model, it appears that the material is more concentrated near the surface rather than having a steady intermixing into the substrate. The concentration drops rapidly to approximately 30% in the space of 0.04 nm, but then peters off to 0% concentration in the space of 0.8 nm.
4.3 Results and Discussion

Figure 4.6 Run 81878, 0.26 ML Pd on Cu(111), annealed at 450°C. Experimental energy cut (dotted) and SIMNRA simulation (line) vs counts (top) corresponding sample model used in the SIMNRA simulation shown on top. It can be seen that the Pd is heavily intermixed into the Cu (bottom), and that the Pd signal is weak (top).
4.3 Results and Discussion

The main information that can be taken from the depth profiles shown in figures 4.1 to 4.6 is that the Pd appears to intermix into the Cu substrate.

The next stage in analysing the Pd Cu(111) system is to compare the layer-by-layer aligned results. This will reveal structural information such as twinning, and layer-by-layer Pd and Cu populations. The visible amounts of Pd and Cu for each alignment were extracted from the whole tiles using the SuperMegaCurvefit Igor macro.

Figure 4.7 shows the full experimental Pd signals for the <114> beam incidence direction for various Pd coverages, with a counts-offset Vegas simulation of maximum coverage (3.89 ML). This simulation is to show where the blocking dips should be for a perfect layer-by-layer coverage of Pd on the Cu, with the Pd being in registry with, and having the same lattice spacings as the Cu crystal.

The <114> incidence direction has been chosen for several reasons. It is a 3-layer illumination (see appendix A, (112) azimuthal cut, beam E), it provides plenty of blocking dips on which to perform calculations upon, and it is the twinned direction of the <110> beam incidence.

Firstly it can be seen that several of the blocking dips present in the simulation can be seen in the experimental curves. It can be also seen that there are blocking dips clearly present in the experimental 3.89 ML coverage which are not present in the simulation. **This graph has been repeated for the <110> beam incidence direction, figure 4.8. Due to the single layer illumination, fewer blocking dips are expected, and in figure 4.7 there are blocking dips present in the maximum Pd coverage that are not present in the layer by layer simulated coverage.**
Figure 4.7 The MEIS experimental Pd overlayer blocking curves for the \( <114> \) beam incidence direction. An identical beam orientation Vegas simulation of 3.83 ML layer-by-layer growth has been offset plotted on the right hand axis. The fcc Pd simulation assumes the Pd is in registry with the Cu substrate and has Cu lattice spacing. Note the interesting feature at approximately 90° on 3.89 ML Pd experimental curve which is not present in the fcc simulation.
Figure 4.8 The MEIS experimental Pd overlayer blocking curves for the <110> beam incidence direction. An identical beam orientation Vegas simulation of 3.83 ML layer-by-layer growth has been offset plotted on the right hand axis. The fcc Pd simulation assumes the Pd is in registry with the Cu substrate and has Cu lattice spacing. Note the interesting feature at approximately 84° on the 3.89 ML experimental Pd curve that is not present in the Vegas simulation.
Figure 4.9 shows a comparison of the blocking curves for the 3.89 ML experimental coverage in the <110> and <114> incidence directions, complemented by their Vegas simulations of 3.83 ML* coverage. It can be seen that the simulations have very distinct features, namely the (100) channel in the <110> curve, and the (111), (233) and the (110) channels in the <114> curve. The experimental curves show a crossover of features i.e. blocking dips found exclusively in the <114> simulation are found in the <110> experimental curve along with the expected <110> features, and vice versa.

The mixing of features between the <110> and <114> beam incidences for the Pd signal indicate twinning of the Pd overlayer. Twinning of an fcc(111) surface involves the rotation of the 3-fold symmetry planes by 180° around the direction of the (111) plane for a given percentage of the material i.e. the ABC growth turn into ACB growth for a given percentage. For a more detailed explanation of twinning see section 2.5.3. In the case of the <110> beam incidence, the twinned beam incidence would be <114>, and vice versa.

To illustrate this point further, twinned simulations have been added to the same graph as the <114> experimental curve and plotted in figure 4.10. Points noted on this graph are the yield for the experimental signal is approximately equal to that of the 50% twinning simulation. Also the (233) blocking channel in particular is offset by approximately 2°. This angular offset is an indication of surface relaxation.

*The reason for simulating 3.83 ML is that the fractional part (0.83ML) of the Pd coverage can be simulated by 10 atoms out of 12 in a crystal layer, whereas 0.89 ML would require 89 out of 100 atoms in a layer. This greatly reduces simulation time.
4.3 Results and Discussion

![Graph showing experimental and simulated visibility for different scattering angles and Pd coverages.](image)

**Figure 4.9** Experimental and Vegas simulation blocking dip comparison for the <110> and <114> beam incidences of 3.89 ML and 3.83 ML respective Pd coverages. Experimental visibility plotted on the left axis, simulation on the right. Both are offset to aid viewing.

The blocking curves for the Cu substrate, <114> beam incidence, are presented in figure 4.11.

The models used for the twinned crystals were generated using a custom made Igor macro that essentially takes a regular non-twinned crystal unit cell and based on the desired crystal properties input by the user, a crystal model ready to be input into Vegas would be produced. Crystal properties would include substrate thickness, substrate vibration, film thickness, film vibration, percentage twinning and first and second layer percentage expansion (relaxation). The twinned crystals essentially were constructed by having a grid of unit cells with the desired percentage of twins being rotated around the fcc(111) direction by 180°. This means that the percentage twinning is that of surface area.

A problem arises when investigating twinning due to different orientations of unit cells on the surface. One way of dealing with this problem is to simulate full crystals...
Figure 4.10 Experimental and Vegas simulation blocking dip comparison for the \(<114>\) beam incidences of 3.89 ML and 3.83 ML respective Pd coverages. A variety of twinned crystal samples have been plotted. Attention should be drawn to the experimental (233) channel at approximately 85° being offset to the simulation and the compared visibilities of the simulations and experiment. Experimental visibility plotted on the left axis, simulations on the right. Both are offset to aid viewing.
with a desired twinning percentage. However, for 1% twinning i.e. 99 cells oriented in a normal direction and 1 in the twinned direction, a large number of unit cells are required. Simulations that have been carried out with large numbers of unit cells indicate that the results of the scattering processes are obtained with an incoherent superposition of intensities. This implies that on the level of present sensitivity, no interference occurs between unit cells of different orientation. It is therefore entirely appropriate to the obtain the intensities for normal and twinned unit cells and perform a weighted average calculation to find the intensity for a given twinning percentage.

It must be noted that when simulating Pd on top of Cu(111) the Pd lattice spacing taken was that of the lateral Cu(111) surface and Cu interplanar spacing. This implies that the Pd is essentially compressed compared to its natural bulk structure. It is unlikely that such a situation is realised experimentally.

One feature to notice is the visibility of the Cu curves in figure 4.11. It can be seen that the visibility decreases from clean Cu to 0.25 ML Pd overlayer coverage then increases from 0.25 ML to 0.8 ML to 1.83 ML, but does not increase any further. One would expect that with an overlayer an undisturbed substrate's visibility would decrease. Also, the visibility of the Cu is a lot greater in the experiment than in the simulations. Similar curves have been plotted for the <110> beam incidence direction in figure 4.12. The key difference to the previous figure is that the substrate visibility consistently increases with coverage. Again, the visibility is far greater than that of the simulations. It can also be noted that for the same Pd overlayer coverage, no simulated combination of twinning and beam incidence direction can produce the visibility of the substrate measured in the experiments for the two beam alignments dealt with so far. There are also no new blocking dips present in the substrate blocking curves, indicating that the substrate does not twin, even if it is intermixed with a twinned overlayer.
Figure 4.11 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the \( <114> \) beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83ML Pd overlayer. The Cu visibility is increasing as Pd overlayer coverage increases. A possible reason for this could be Pd induced Cu structure change.
Figure 4.12 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayers, for the <110> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayers. There is approximately 2.5 ML extra Cu visible when covered with 3.89 ML of Pd.
4.3 Results and Discussion

Figure 4.13 Experimental and simulated Pd overlayer visibility profiles for varying coverages of a Pd film on Cu(111). $<110>$ and $<114>$ beam incidence directions. Various twinning percentages were also simulated.

For further investigation of the twinning effect, yields of both the Pd overlayer and Cu substrate have been plotted for a narrow angle scattering window in figures 4.13 and 4.14. Simulations for the varied Pd coverages, with twinning amounts from 0% to 100%, have also been plotted. For the Pd coverage, figure 4.13, the visibilities of various different simulated twinning amounts appear to separate when the Pd coverage exceeds 1 ML. For the $<110>$ incidence the experimental visibility appears to be between the visibilities of the 50% and 75% twinned Pd simulations, with this becoming apparent at the 1.8 ML coverage. This pattern is similar in the $<114>$ plot, but experimental error seems to be more stated. At the 3.89 ML coverage the $<110>$ and $<114>$ Pd visibilities are very similar, suggesting a twinning value of 60%. This means that twinning is likely from at least 1.8 ML coverage, and could therefore be present in thinner films.
4.3 Results and Discussion

Figure 4.14 Experimental and simulated Cu substrate visibility profiles for varying coverages of a Pd film on Cu(111). <110> and <114> beam incidence directions. Various twinning percentages were also simulated.

Figure 4.14 shows the visibility of the Cu substrate for varying Pd film thicknesses. The visibility is shown as the twinning percentage of the Pd overlayer is varied. It can again be clearly seen that amount of Pd twinning is not linked to the Cu visibility and some other surface effect must be involved. It can been seen however that the Cu visibility stays the same even with a 0.25 ML coverage of Pd, and the simulation suggests it decreases. This means that the Pd cannot be blocking any beam to the Cu substrate. Twinning of the Pd even reduces the Cu visibility so the Pd cannot be settling in registry on the Cu surface. This could be explained by surface relaxation, non-fcc absorption sites, or sub-surface migration.

At this point of the investigation the results have tended towards the following points:

- Extra blocking dips are present in the Pd overlayer that indicate twinning in the Pd.
- There are no extra blocking dips present in the Cu substrate signal which indicates that no twinning is present in the substrate.
4.3 Results and Discussion

Figure 4.15 Vegas simulations of clean Cu(111) for the <114> beam incidence direction. A range of the 0-5 layers have had their interplanar lattice spacing increased to that of Pd(111). Note the yield change, and blocking dip offset.

- Pd overlayer visibility is greater than simulations with Cu lattice spacing throughout indicate, and can be explained by twinning.

- Cu substrate visibility is greater than simulations with Cu lattice spacing throughout indicate, which cannot be explained by twinning. The increased visibility must be due to other structural differences to that of bulk Cu(111), lattice relaxation being one example.

- Cu visibility is unchanged even with a 0.25 ML Pd coverage, the Pd cannot be in registry on top of the Cu substrate.

- As twinning is not obviously present in the Cu, other factors affecting the Cu visibility could be lattice parameter changing, and intermixing/interdiffusion/-substitution being the most likely candidates suggested by previous literature.
As a preliminary investigation of lattice expansion, a simulation was carried out on clean Cu(111) to see what happened to the visibility of the Cu as the top layers' lattice spacing were increased to that of Pd. Clean Cu was simulated, with the top 0 to 5 layers being expanded and the resulting blocking curves being plotted in figure 4.15. It can be seen that the 3.89 ML experimental visibility matches the 3-4 layer expanded simulations shown in figure 4.15.

The other two experimental incidence directions taken were the <233> and <122> incidence directions. These produce 2 and 3 layer visibilities along the same azimuth as the <110> incidence.

Figures 4.16 - 4.19 show the extracted surface and substrate blocking curves for the <233> and <122> beam incidence directions.

On first inspection it seems that the major blocking dip in figure 4.16 appears to be the same as the 100% twinned simulation. This blocking dip is the (111) blocking channel, perpendicular to the surface, and is present at any azimuthal rotation and beam incidence. As the (111) channel is made from 4 full fcc(111) layers and the <233> beam incidence only illuminates 2 layers, the (111) blocking dip would not be expected to be very large, if visible at all, as the 0% twinned simulation suggests. The twinned incidence beam geometry would produce alignment with the <1411>, which would illuminate enough layers for the (111) channel to shadow the outgoing beam. The apparent yield from the 3.89 ML experimental coverage also appears to be far greater than the 0% twinned simulation. This increase in yield and (111) channel visibility could also appear from a surface relaxation as lower layers will be illuminated.

The curves displayed in figure 4.17, Pd <122>, appear to be rather uneventful. Some blocking dips are present in the 3.89 ML experimental coverage as predicted by the 0% twinned simulation. The two major blocking dips predicted by the non-
4.3 Results and Discussion

twinned simulation are the (112) channel at 104°, and the (111) channel at 123°. The 
(112) channel is made from 3 full layers and is not present in the twinned direction i.e. 
it is situated in the (211) azimuth. This is not plainly visible in figure 4.17. The (111) 
channel is made from 4 layers and is visible in both the twinned and non-twinned 
crystals. The fact that a 4-layer feature is more visible than a 3-layer feature can 
be explained by twinning and a lattice relaxation e.g. twinning to remove the (112) 
feature and a lattice relaxation to highlight the (111) feature.

When we examine the visibilities of the <233> and <122> Pd blocking curves 
we have an interesting result. The <233> and <122> beam directions respectively 
illuminate 2 and 3 layers, and the clean simulations suggest so. The experimental 
results show a different story with the <122> yielding less than the <233> curves.

Figures 4.18 and 4.19 show the Cu substrate blocking curves for the <233> and 
<122> beam incidence directions.

Upon inspection of the substrate curves, no new blocking dips have appeared. 
This suggests that the Cu structure does not twin even though it may intermix with 
twinned Pd, as suggested by the <110> and <114> substrate blocking curves. The 
main information provided by these curves are that the substrate yield again increases 
as Pd deposition increases for both the <233> and <122> curves. This increase in 
visibility suggests lattice relaxation.

So as a quick recap, the important points to note about all the curves are as 
follows:

- The Pd visibility for the <110> is more than the in-registry simulations, the 
  <114> Pd visibility is less, with both meeting in the middle.

- Blocking dips in the <110> and <114> Pd curves combined with the Pd visi-
  bility suggest approximately 50% twinning is present.
• Cu substrate visibility increases and levels off as Pd coverage increases for the \textless 110 \textgreater beam incidence direction, and continually increases for the \textless 114 \textgreater curves. Simulations indicate that when Pd twinning is present, Cu coverage decreases Cu visibility. This means another surface effect aside from Pd twinning is present e.g. Cu displacement, other Pd growth modes, surface relaxation.

• No new blocking dips are present in any of the Cu substrate blocking curves, indicating that the Cu growth mode must stay the same.
4.3 Results and Discussion

Figure 4.16 Experimental blocking curves for various coverages of Pd on a Cu(111) substrate for the $<233>$ beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer.
Figure 4.17 Experimental blocking curves for various coverages of Pd on a Cu(111) substrate for the <122> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer.
Figure 4.18 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the <233> beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. Several experimental curves with Pd coverage have been smoothed to aid the eye.
4.3 Results and Discussion

Figure 4.19 Experimental blocking curves for the Cu substrate with various coverages of Pd overlayer, for the \( <122> \) beam incidence direction. Vegas simulations have been added to an offset right hand axis for 3.83 ML Pd overlayer. Several experimental curves with Pd coverage have been smoothed to aid the eye.
Upon detailed inspection of the unannealed 0.25 ML results it can be seen that the Pd yield at different incidence angles changes. From Vegas simulations the individual layer visibilities for each beam incidence can be extracted. A set of simultaneous equations can be then built up from these visibilities, equations 4.1 - 4.4 highlight this. This approach has been seen in [54] and [55].

\[ X_1 A_{<110>} + X_2 B_{<110>} + X_3 C_{<110>} = V_{<110>} \] (4.1)
\[ X_1 A_{<233>} + X_2 B_{<233>} + X_3 C_{<233>} = V_{<233>} \] (4.2)
\[ X_1 A_{<122>} + X_2 B_{<122>} + X_3 C_{<122>} = V_{<122>} \] (4.3)
\[ X_1 A_{<114>} + X_2 B_{<114>} + X_3 C_{<114>} = V_{<114>} \] (4.4)

Where \( A_{<110>} \) is the simulated visibility of the first layer at the \(<110>\) beam incidence. \( B_{<110>} \) is the simulation visibility for the second layer, \( C_{<110>} \) being the 3rd. \( V_{<110>} \) is the experimental visibility at the \(<110>\) incidence. These experimental values are shown in table 4.3. \( X_{1,2,3} \) are the fractional concentrations of Pd in the 1st, 2nd and 3rd layers.

The simulation that has been tabulated in table 4.3 was of a clean Copper crystal, with a surface relaxation based on Gustafsson’s findings [56]. These values were a 1% expansion in the first interlayer spacing and a 0.2% expansion in the second inter-layer spacing compared to a (111) oriented bulk Cu crystal. The surface vibrational amplitudes were also reported to have changed. That being 0.10 Å and 0.09 Å for the perpendicular and in-plane top layer vibrational amplitudes, compared to 0.078 Å for the bulk.

As there is only a fractional Pd coverage and the surface effects are this far an unknown it can be argued that using Gustafsson’s clean Cu model for this case is the closest approximation. The other option would be to use the bulk lattice spacing and
4.3 Results and Discussion

Table 4.3 Experimental visible Cu and Pd for 0.25 ML Pd coverage on a Cu(111) crystal for various beam incidences. These values are used in equations 4.5 to 4.8

<table>
<thead>
<tr>
<th>Beam Incidence</th>
<th>Visible Pd in ML ($V_{&lt;xxz&gt;}$)</th>
<th>Visible Cu in ML ($V_{&lt;xxx&gt;}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;110&gt;</td>
<td>0.178 ± 0.009</td>
<td>1.269 ± 0.019</td>
</tr>
<tr>
<td>&lt;233&gt;</td>
<td>0.271 ± 0.011</td>
<td>1.941 ± 0.020</td>
</tr>
<tr>
<td>&lt;122&gt;</td>
<td>0.221 ± 0.008</td>
<td>3.058 ± 0.013</td>
</tr>
<tr>
<td>&lt;114&gt;</td>
<td>0.241 ± 0.013</td>
<td>3.151 ± 0.023</td>
</tr>
</tbody>
</table>

atomic vibrations all the way to the surface but this would pose the question of why the surface would suddenly revert to bulk properties when fractionally covered.

As twinning is suspected, the visibilities for each incidence angle could be altered. Table 4.4 highlights this, with A, B, and C listed for the normal and twinned directions.

It can be seen in table 4.4 that the visibility for layer 3+ (layers 3, 4, 5) in the twinned direction is 5.391. This may be an unfair visibility weighting. For example, if we took a poorly intermixed 0.25 ML coverage, say 70%, 25% and 5% of the Pd going into the first, second and third layers, the 5% would be taken to be spread across the 3rd, 4th and 5th layers. As the Concentration of the Pd rapidly decreases from the 1st to the 3rd layers, the absolute concentration in the 4th and 5th layers could be negligible, but the simulation weight would suggest otherwise. For this reason the 3+ values that are above 1 have been changed to 1. This then essentially assumes the Pd will always be situated entirely in the first, second and third layers for sub-monolayer coverages.

With these values being corrected a new set of simultaneous equations can be generated from weighted normal and twinned visibilities. i.e. for 20% twinning you would take the overall layer visibilities as 80% of the normal visibilities and 20% of
### Table 4.4 Simulated layer-by-layer visibilities in ML for various beam incidence directions for Cu(111).

$A_{1,2,3,<xxx>}$ are layers 1, 2, and 3+ i.e. layer 3 onwards taken at the arbitrary $<xxx>$ beam incidence direction.

<table>
<thead>
<tr>
<th>Normal</th>
<th>$A_{1,&lt;xxx&gt;}$</th>
<th>$A_{2,&lt;xxx&gt;}$</th>
<th>$A_{3,&lt;xxx&gt;}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;110&gt;$</td>
<td>1.001</td>
<td>0.150</td>
<td>0.003</td>
<td>1.154</td>
</tr>
<tr>
<td>$&lt;114&gt;$</td>
<td>1.000</td>
<td>1.032</td>
<td>1.028</td>
<td>3.060</td>
</tr>
<tr>
<td>$&lt;122&gt;$</td>
<td>0.999</td>
<td>1.009</td>
<td>1.327</td>
<td>3.335</td>
</tr>
<tr>
<td>$&lt;233&gt;$</td>
<td>1.001</td>
<td>1.003</td>
<td>0.235</td>
<td>2.239</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Twinned</th>
<th>$A_{&lt;xxx&gt;}$</th>
<th>$A_{2,&lt;xxx&gt;}$</th>
<th>$A_{3,&lt;xxx&gt;}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;110&gt;$</td>
<td>1.000</td>
<td>1.006</td>
<td>1.040</td>
<td>3.046</td>
</tr>
<tr>
<td>$&lt;114&gt;$</td>
<td>1.001</td>
<td>0.150</td>
<td>0.003</td>
<td>1.154</td>
</tr>
<tr>
<td>$&lt;122&gt;$</td>
<td>1.002</td>
<td>0.336</td>
<td>0.059</td>
<td>1.397</td>
</tr>
<tr>
<td>$&lt;233&gt;$</td>
<td>1.002</td>
<td>1.054</td>
<td>5.391</td>
<td>7.447</td>
</tr>
</tbody>
</table>

the twinned. This forms equations 4.5 - 4.8.

\[
\sum_{i=1}^{3} X_i ( (1 - T) A_{i,N,<110>} + TA_{i,t,<110>} ) = V_{<110>} \tag{4.5}
\]

\[
\sum_{i=1}^{3} X_i ( (1 - T) A_{i,N,<233>} + TA_{i,t,<233>} ) = V_{<233>} \tag{4.6}
\]

\[
\sum_{i=1}^{3} X_i ( (1 - T) A_{i,N,<122>} + TA_{i,t,<122>} ) = V_{<122>} \tag{4.7}
\]

\[
\sum_{i=1}^{3} X_i ( (1 - T) A_{i,N,<114>} + TA_{i,t,<114>} ) = V_{<114>} \tag{4.8}
\]

Where $X_i$ is the fractional concentration of material in the i'th layer. $A_{i,N,<xxx>}$ is the normal aligned visibility of the i'th layer. $A_{i,t,<xxx>}$ is the twinned aligned visibility of the i'th layer. $T$ is the fractional amount of twinning. Equations 4.5 - 4.8 essentially have the same structure as equations 4.1 - 4.4 but with the visibility
Table 4.5 A sweep of twinning values were trialed when solving equations 4.5-4.8 for the 0.25ML of Pd on Cu(111) system. The only fully non-negative values are listed below. The absolute Pd concentrations (fractional concentration of the whole layer) shown on top, the percentage ratios of Pd concentration (total percentage of the deposited Pd) shown underneath.

<table>
<thead>
<tr>
<th>Twinning (%)</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>0.076</td>
<td>0.186</td>
<td>0.016</td>
<td>0.278</td>
</tr>
<tr>
<td>43</td>
<td>0.065</td>
<td>0.185</td>
<td>0.039</td>
<td>0.289</td>
</tr>
<tr>
<td>44</td>
<td>0.049</td>
<td>0.182</td>
<td>0.071</td>
<td>0.303</td>
</tr>
<tr>
<td>45</td>
<td>0.028</td>
<td>0.177</td>
<td>0.118</td>
<td>0.323</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% ratio</th>
<th>% ratio</th>
<th>% ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>27 ±3%</td>
<td>67 ±3%</td>
</tr>
<tr>
<td>43</td>
<td>22 ±3%</td>
<td>64 ±3%</td>
</tr>
<tr>
<td>44</td>
<td>16 ±3%</td>
<td>60 ±3%</td>
</tr>
<tr>
<td>45</td>
<td>9 ±3%</td>
<td>55 ±3%</td>
</tr>
</tbody>
</table>

being weighted between the normal and twinned visibilities.

The next stage is to solve the simultaneous equation to find $X_1$, $X_2$ and $X_3$ ($X_i$) through a range of twinning values of 0% to 100% for Pd, as shown in table 4.5. This produced a variety of results, with the majority of twinning values causing the concentration of Pd in one of the layers being a negative value, which is un-physical. The only range of twinning values with all positive solutions are shown in table 4.5.

It can be seen that the only range of twinning values that produce non-negative layer concentrations is between 42% and 45%. The visibility of Pd in random incidence is 0.27 ML, which essentially is the true amount, the closest total amount is resulting from the 42% twinning calculations. For all cases it is apparent that the majority of Pd is in the second layer, seconded by the top layer, and a small fraction is in the third layer. When the experimental visibility errors are taken into account,
4.3 Results and Discussion

Table 4.6 Equations 4.5-4.8 were solved for twinning values 0% to 100% for Cu visibilities in the 0.25ML Pd on Cu(111). The only fully non-negative results are listed below. The absolute Cu concentrations for each layer are shown.

<table>
<thead>
<tr>
<th>Twinning (%)</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.172</td>
<td>0.284</td>
<td>1.168</td>
<td>2.624</td>
</tr>
<tr>
<td>1</td>
<td>1.177</td>
<td>0.181</td>
<td>1.252</td>
<td>2.609</td>
</tr>
<tr>
<td>2</td>
<td>1.184</td>
<td>0.062</td>
<td>1.346</td>
<td>2.592</td>
</tr>
</tbody>
</table>

The twinning window expands from 40% to 47% when the errors are either all added or subtracted. The ratio of the layer-by-layer Pd populations vary by approximately 3%. When the errors are added and subtracted to and from the visibilities in the same calculation the twinning always went down, with a minimum limit of 35% twinning. The ratios of the layer-by-layer Pd populations remained within the previously mention percentage error. The percentage error values are not the result of calculations but from observation of applying the three visibility errors in different combinations and solving the resulting simultaneous equations. The stated errors are what was observed at the fringe points. There were no inconsistent results from these error applications so the results are deemed sound.

The above process has then been repeated for the Cu visibilities, with the only non-negative results shown in table 4.6. As the Cu is the substrate the original simulated layer visibilities shown in table 4.4 have been used e.g. the <233> visibility for the 3+ layer has been taken as 5.391.

It can be seen in table 4.6 that the Cu twinning is practically zero, and the concentration of the second layer is the minimum, showing the concentration of Pd must be at a maximum. The absolute concentrations in the first and third layers are greater than one, suggesting that the simultaneous equations are possibly missing an
unknown component. This could be a number of things, the most likely being due to the noise in the substrate signal, but could also be contributed to by effects such as channelling affecting the overall yield values.

As the approximate twinning and concentration values have been calculated, Vegas simulations can now be performed to attempt to measure the effect of the blocking dip shift.

Figure 4.20 shows the key features in the main blocking dips for both the Pd and Cu experimental curves for the $<114>$ incidence. It can be seen that the Pd is shifted left (indicating a surface contraction), whilst the equivalent Cu blocking channel is right shifted (indicating a surface expansion). On first inspection one could argue rumpling being present i.e. the Pd in the same layer is shifted down into the substrate whilst the Cu is shifted up.

A problem experienced using Vegas is the crystal size in the simulation. To simulate a crystal with 40% twinning the minimum crystal layer size would have to be 1x5 unit cells. If we want a layer to have 5% concentration then the minimum layer size would have to be 1x20 unit cells. So with the smallest size for each layer having 20 unit cells (40 atoms) and the standard crystal usually consisting of 10 layers, totals in 400 atoms. This produces a very long simulation time, approximately one hour.

A work-around to the above listed problem was to simulate a clean Cu lattice and fractionally add individual layers to build up full spectra for both the Pd and Cu. As a clean Cu lattice is being simulated, the minimum lattice size is just 1x1, meaning the simulation time is very small, approximately 5 minutes. The major benefit from this method however is that we can effectively generate our spectra from two small simulations, and vary the amount of simulated twinning and layer concentrations immediately instead of having to produce a new simulation for every one of the four variables listed (twinning, 1st 2nd, and 3rd layer concentrations). As these variable
Figure 4.20 Clean Cu(111) compared to Cu(111) with 0.25ML of Pd coverage: Substrate and surface blocking dips. <114> beam incidence. Notice the angular shifts of the blocking dips at approximately 74°. The Pd dip is shifted left, the same-sample Cu blocking dip is shifted right from the clean Cu blocking dip. Note that the experimental Pd blocking dip at 74° is wider than both of the Cu curves. This suggests that this blocking dip could be formed from two blocking dips that are separated in scattering angle.
4.3 Results and Discussion

Figure 4.21 A Vegas simulation of clean Cu(111), <114> beam incidence. The overall counts have been separated layer-by-layer to show what layers contribute to each blocking dip.

Alterations can be generated from a single data file, a fitting algorithm can also be used if desired.

Figure 4.21 shows a clean Cu unexpanded-lattice simulation, split into layer-by-layer visibilities. The main blocking channel that is of interest is the (111) channel, at approximately 74°. The main components of this channel are from ions being scattered from the 2nd and 3rd layers. A result of this fact combined with the layer by layer concentrations revealed by the simultaneous could suggest full plane contraction and expansion rather than rumpling.

As the Pd(111) channel is left shifted i.e. the scattering angle is lower, and the majority of this Pd being situated in the top 2 layers, this suggests a contraction of the 1-2 crystal planes. As the Cu(111) blocking dip is right shifted, and also equally contributed to by the 2-3 surface plane, an expansion of this plane is suggested.

Clean Cu lattices have been simulated, with a matrix of first interlayer spacing
variation of $-8\%$ to $0\%$ of its original lattice spacing (a contraction), and second interlayer lattice spacing of $0\%$ to $8\%$ (an expansion). The original lattice spacing was taken to be the Gustafsson clean Cu model.

At this stage of analysis the key values that we want to measure and take into account are the angular shifts of the blocking dips, rather than the overall counts and curves as a whole. For this reason an R-factor has been generated from the angular shifts rather than the more standard R-factor calculation based on yields. The R-factor is defined as:

$$ R = \frac{1}{N} \sum_{1}^{N} \left[ \frac{(I_{exp} - \lambda I_{sim})^2}{I_{exp}} \right] $$

(4.9)

Where $N$ is the total number of points being compared, $I_{exp}$ and $I_{sim}$ are the experimental and simulated intensity values (can be visibility or scattering angle at a given point), and $\lambda$ is a scaling factor used to allow for instrumentation effects such as transmission efficiency and acceptance angle of the ion energy analyzer. In all cases listed here $\lambda$ is taken to be 1.

Another reason for avoiding the traditional R-factor calculation is that the bulk Debye-Waller factor may not be the same for both the Pd and Cu once intermixed. Various factors would affect the atomic vibrations, such as Pd/Cu concentration, local neighbour distances, twinning and many other variables. This would most definitely affect the overall counts as much as a layer expansion would, and with the blocking dip shift being a more subtle effect, would drown out any effect the angular shifts would have on the overall, or average, counts across the sample angular range.

Figure 4.22 shows the R-value plot from the angular measurements. It can be seen that the minimum is located at a first interlayer contraction of $4\% \pm 3.3\%$ and a second interlayer expansion of $4\% \pm 4.5\%$, the errors being calculated using equation 4.10, which has been taken from [41] and [57]. Figure 4.23 shows the corresponding
Figure 4.22 2D-R-value plot of the angular shift fitting whilst varying the 1st and 2nd interlayer spacing. <114> beam incidence, 70° to 100° scattering angle. The minimum is located at -4% and +4% first and second interlayer expansions.
4.3 Results and Discussion

Figure 4.23 Experimental and simulated blocking curves for 0.25 ML Pd on Cu(111) for <114> beam incidence, 70° to 100° scattering angle. For the simulation, the first and second interlayer spacings have a 4% contraction and a +4% expansion.

Simulated curve for the minimum found in figure 4.22.

Figure 4.22 appears to be reasonably circular, but at the edges the R-values increase sharply. This means that within the lower gradient area of the contour plot there is more uncertainty and the best guess could more easily vary to somewhere in this region. This is where the error boundary calculations were performed and may produce greater error values than desired. As the gradient of the R-factor plot increases i.e. near the edges of the plot, it becomes less and less likely that the error could spill out into the higher relaxation percentages. This means that we can narrow down the best estimate to the centre area and outside that region it rapidly becomes less probable that the true values could be located here. One could assume a smaller error boundary, but this would be more judgement related.
4.3 Results and Discussion

Figure 4.24 <233> beam incidence bulk (111) crystal simulation and 3.8ML Pd on Cu(111), also <233> incidence. Note the central blocking dip in the simulation only appears in layers 5 and 6, which is weakly visible in the experimental curve.

\[
\frac{2}{\sigma^2} = \left( \frac{d^2R_k}{dZ^2} \right)_{R=R_{\text{min}}}
\]

It can be seen that the second interlayer relaxation R-values do not vary as much as the first interlayer. This means that there is more uncertainty in the second interlayer relaxation, which could be explained due to the fact that the measurement will not be as sensitive to the second interlayer as the blocking dips will be smaller, and is obscured by the first interlayer. This is reflected in the final error values of 4% ± 3.3% and 4% ± 4.5%.

As the coverage increases no new blocking dips appear in the Cu blocking curves, and the twinned effect continues. Figure 4.24 shows that with a 3.89 ML coverage, the Pd is located at least to a depth of 5 or 6 ML. Figure 4.24(b) shows a blocking dip at approximately 110°, which is only produced by layers 5 and 6 in a bulk sample, figure 4.24(a). This suggests that the interface width does not change with increasing Pd coverage.

We have shown using various beam incidences and yield measurements that from initial coverage twinning is present, at a percentage of approximately 42%. The Pd
tends to mainly migrate to the second layer, with a Pd ratio of approximately 27:67:6 for the first, second, and third layers for 0.25 ML coverage. The intermixed Copper shows no signs of twinning. As the coverage continues the twinning continues, along with a gradually decreasing intermixing of the Cu and Pd.

4.4 Conclusion

Making MEIS measurements from random, 1-layer, 2-layer and 3-layer beam incidence alignments at room temperature, we have shown that the initial stages of growth of Pd on Cu(111) do not strictly match the current thin film growth models e.g. Frank-van der Merwe (FM), Stranski-Krastanov (SK) or Volmer-Weber (VW). The Pd tended to intermix with the Cu up to at least (and appreciably) 3 ML deep even from the very initial stages of growth. This intermixed Pd and Cu has been shown to retain an fcc structure, the Pd exhibiting twinning and the Cu staying in registry. This twinning has been shown to continue up to at least 3.89 ML coverage, with an intermixing still being apparent at this coverage.

SIMNRA has been used to depth profile the Pd on Cu(111) system. It was found to give a reasonable approximation to the system, that being to show that the Pd was intermixed into the Cu. It appeared that this method unsuitably ascribed too much detail to the depth profiling and would be more suited to thicker film measurements e.g. > 10ML.

The percentage of twinning has been measured to be 42% from the very initial stages of growth (0.25 ML) and evolves to approximately 60% in coverages up to 3.89 ML i.e. 60% of Pd has been rotated by 180°. The intermixed Copper does not show twinning at any coverage.

For initial coverages (0.25 ML) we have measured the Pd concentrations for the
first three layers of the Pd Cu(111) sample. These were found to be 27% ± 3%, 67% ± 3% and 6% ± 3% of Pd in the 1st, 2nd and 3rd layers. This was calculated by solving simultaneous equations that incorporated the experimental Pd visibility at 1, 2 and 3 layer beam alignments, and Vegas simulations of twinned and normal Cu. The simultaneous equations were solved using the Vegas simulations from 0% through to 100% twinning, with non-negative concentrations being seen for all three layers in only a few twinning values. The absolute concentrations for each layer were summed and this value was compared to the true total amount of Pd coating. This gave the closest twinning value of 42%.

The surface relaxation has also been measured for the first and second interlayer spacing for 0.25 ML coverage on Pd Cu(111). The first interlayer was measured to have a 4% ± 3.3% contraction and the second interlayer being an expansion of 4%±4.5% relative to bulk Cu. These values were measured by comparing the angular shifts of the major blocking dips of a 3-layer aligned sample to the blocking dip shifts in a variety of Vegas simulations which also had their first and second interlayer spacings adjusted.

Tying these results in with current literature we find that our results are in approximate agreement with [46] with respect to the layer by layer Pd concentration distribution, but disagrees with first and second layer relaxations. The layer expansion results we have found agree with [47] with respect to the first interlayer being a contraction and second being an expansion, but the values found here are larger.

The question posed by [44], whether continued Pd coverage would de-alloy the Pd and Cu interface, reducing its size has been answered by energy depth profiling and by observing blocking dips. The interfacial width remains approximately the same.

Twinning in the Pd coverage has been clearly observed which has not been reported anywhere else before.
4.4 Conclusion

Linking the results back to the frequently quoted growth modes e.g. layer-by-layer etc. one can see that the PdCu(111) system does not strictly follow any of these theories. The above system grows in an intermixed state with two structures forming. A possible mechanism for growth is that the Pd that is absorbed into the Cu substrate will most likely keep the in-registry fcc structure whilst islands form on top of the Cu which will contain the twinned Pd. This formation would be most energetically favourable and supports the results that indicate lack of twinning of the Cu substrate.
Chapter 5

Ag on Al(111)

Medium Energy Ion Scattering has been used to crystallographically depth profile ultrathin films of Ag on Al(111). The system was shown to exhibit a mixture of fcc and hcp growth from the outset. The initial settling points of the Ag were shown to be 43% in fcc sites and 57% in hcp sites. At 4.74 ML Ag coverage it has been shown that the hcp growth terminates and returns to fcc stacking after a spread of stacking heights ranging from 1 atom high hcp through to 5 atoms high hcp stacking. At intermediate coverages 1.33 ML and 1.83 ML, stacked island growth is suggested.

5.1 Introduction

Ultrathin films of Ag upon Al(111) has seen only experimental papers, no theoretical papers.

The earliest paper [58] looked at Ag upon Al(111) grown at 300 K using AES, LEED, and angle-resolved UPS. They use Auger Electron Spectroscopy to show that the Ag initially grows in two-dimensional islands until the surface is 87%±7% covered when the second layer starts growing i.e. Stranski-Krastanov(SK) growth. This
5.1 Introduction

claim is further substantiated by using oxygen titration and measuring the adsorbed oxygen using ARUPS. This initial growth mode is explained by the diffusion length of the incoming Ag atoms not being long enough to reach the border of the islands to fill in the remaining 13% of the monolayer. It is shown that the first Ag layer matches perfectly with the Al substrate. This is not surprising as there is only a \(\sim 0.9\%\) lattice mismatch. Before the first layer is filled completely filled clusters 3-4 layers thick start growing and after a transition of more than 10 ML a recrystallisation into bulk Ag takes place. These thicker films were seen to have patches of different stacking sequences due to the six-fold LEED pattern.

Losch and Niehus [61] use low energy ion scattering (LEIS) to investigate the structure and growth mode of Ag evaporated on Al(111) in a fairly brief report. They find up to two layers coverage produces an fcc growth continuation, with a strong indication of 3D growth. Comparison with data also suggests an inward surface relaxation of about 1% but this falls into the experimental uncertainty.

Kim et al. [62] have studied the initial growth mode of Ag on Al(111) using AES, LEED, and XPS. They observed an exponentially decreasing LEED pattern that completely disappeared between 2 - 4 ML and reappeared at 4 ML. Between 4 and 9 ML the LEED pattern was seen to be 6-fold symmetric. They propose a growth model that includes stacking faults induced by an interface alloy formation: domains of \(Ag_2Al\) grow in hcp [63] stacking whilst other parts of the surface grow in fcc Ag islands, with the fcc continuation follows where the previous Ag had landed. These islands have a large lateral contraction which is used to explain the loss of LEED pattern between 2-4 ML. It was suggested that the fcc and hcp stacking find the same adsorption sites from the fifth layer onwards, with large Ag terraces, also showing a large density of “kink” sites where the stacking domain boundaries lie. This essentially means that the hcp growth ends after 4 ML of coverage.
5.2 Experimental Details

From these papers it appears that the general consensus is that the growth mode of Ag upon Al(111) is a 3D continuation of the fcc substrate up to 2 ML. [58] [61] [62] After that the growth appears to be rather untidy, with a combination of stacking modes being simultaneously found on the same surface up to 4 ML coverage [62]. Bulk Ag is then formed after either 5 ML [62] or 10 ML [58] Ag coverage.

5.2 Experimental Details

A standard fcc(111) Al crystal stock sample from Daresbury MEIS facility was used in this experiment.

The sample was initially prepared by cleaning using 0.8 keV to 1.8 keV Ar$^+$ ion sputtering at normal and grazing incidences, followed by flash annealing to 400°C. Before annealing, the crystal was checked using Auger Electron Spectroscopy for any metal contamination as annealing with a contaminant would run the risk of alloying and deep intermixing. The sputtering and annealing cycles were repeated until no contaminants were visible using Auger Electron Spectroscopy, and a (1 x 1) fcc(111) pattern was visible using LEED.

The Ag was deposited using a K-cell that heated up the Ag to approximately 935°C. A K-cell is easier to use than an E-beam evaporation method, but at higher melting temperatures contaminants were observed. The K-cell was calibrated starting with critical temperature values provided at the MEIS facility [64] [65]. It was found that at the above temperature the deposition rate was approximately 1 ML per 10 minutes.

The desired coverages were 0.5, 1, 2, 4 and 6 ML of Ag. Auger Electron Spectroscopy and LEED measurements were taken after each deposition. MEIS measurements were taken at beam incidences <110>, <233> and at a random incidence.
5.3 Results and Discussion

The data was found to have major overlapping issues which prompted the writing of the Tiler Igor macro, see Chapter 3.4. Every tile had to be corrected by essentially shaving off several data points from the top and bottom of each data tile using this macro, and then calibrated using the Match Data macro, Chapter 3.3. All the Ag information could be extracted directly from the tile (i.e. a simple projection) rather than having to fit a gaussian to it, and in many cases the Ag curves were asymmetric so gaussians could not be fitted anyway. This asymmetry was due to either straggling and/or intermixing.

The Al substrate curves were still fitted and extracted using the SuperMegaCurve-fit macro as the background counts in several cases were quite high. Due to the stopping power of the Ag overlayers, the Al substrate signal could not be extracted for some of the 4 ML and 6 ML attempted coverages as the signal from the Al was too weak and in some cases not even visible.
5.3 Results and Discussion

Figure 5.1 Hcp Ag on fcc Al, with a <110> beam incidence. It can be seen that for a 6-layer hcp coverage the 1st, 3rd and 5th layers of the hcp structure are illuminated, and the substrate is shadowed (beam A). Only upon addition of a seventh layer the hcp structure starts to repeat itself with respect to the <110> incidence (beam B).

Figures 5.1 and 5.2 show crystal side cuts of the various beam incidences. They show a silver hcp structure upon an Aluminium substrate fcc structure.

The first azimuthal cut, figure 5.1, shows the <110> beam incidence. It can be seen that for a 6-layer hcp structure, the fcc <110> beam incidence will illuminate the 1st, 3rd and 5th layers of the hcp structure, with the fcc substrate being shadowed by the 1st and 2nd layers, beam A on the diagram. Once a 7th layer is added then the hcp structure starts to repeat itself with respect to the <110> beam, shadowing the underlying hcp structure, as shown with beam B. These facts suggest that the visibility of a hcp overlayer will steadily increase up to and including six monolayers. Upon the addition of a seventh layer the visibility will only increase due to thermal vibration. The visibility of the fcc substrate should not dramatically increase due to the addition of hcp overlayer until a five monolayer thickness is reached, where the
Figure 5.2 Hcp Ag on fcc Al, with a <233> beam incidence. It can be seen that for a 6-layer hcp coverage all hcp layers are illuminated. The top two hcp layers shadow the top two fcc substrate layers. As the <233> incidence is 2-layer, the substrate is fully shadowed by these top two layers. Upon addition of a 7th layer the overlayer begins to shadow itself.

Substrate is directly shadowed. Due to the slight closeness of the <110> planes in the hcp structure (indicated by the three beams in beam A in figure 5.1) some indirect shadowing may occur. Vegas simulations of a hcp-fcc structure will reveal the extent of this indirect shadowing and is tackled further on.

It should also be noted that there appears to be two settling points for hcp growth, this is not true however. Examining figure 5.1 one could suggest that the hcp growth instead of following an “ABC→BC” pattern (where the second B is the starting of the hcp growth), the growth would follow an “ABC→AC” growth. This however is actually an fcc continuation for 1 ML and then going into hcp growth.

The second azimuthal cut, figure 5.2, shows an fcc <233> beam incidence into a six-layer hcp structure on an fcc substrate. It shows that for a six layer hcp coverage
the hcp structure is fully illuminated, but the top two hcp layers will shadow the top
two fcc layers. As the \(<233>\) incidence is two-layer aligned these top two hcp layers
will fully shadow the underlying fcc structure. Upon addition of a seventh layer the
hcp structure begins to repeat itself with respect to the \(<233>\) beam, shadowing
itself. Translating this information to scattered ion counts one would expect a hcp
overlayer yield to steadily increase as overlayer thickness increases up to and including
6 ML. Upon addition of a seventh layer of hcp the yield would not increase anywhere
near as much. With respect to the substrate, the yield should not change drastically
until a hcp overlayer reaches thickness of \(>4\) ML. Indirect shadowing may reduce the
fcc substrate yield however, which will become clear through Vegas simulations.

Figure 5.3 shows the experimental curves for the Ag signal for the \(<110>\) beam
incidence for Ag film thicknesses 0.23 ML to 4.74 ML. Bulk fcc [59] and hcp Vegas
simulations have also been added to the plot to show where the main fcc blocking dips
are located, and to discover whether there are any correlations between the features
found in the hcp bulk simulation and the experimental signals.

Features to note in figure 5.3 are as follows: The difference in yield between the
1.33 ML Ag coverage and the 1.84 ML coverage is very small. This indicates that the
Ag growth is continuing in fcc epitaxial growth as the \(<110>\) beam is single layer
aligned, so continuing the fcc growth would have minimal effect on the yield.

The major blocking dip located at 90° is visible with just 1.33 ML Ag coverage
indicating 3D island growth. If layer-by-layer growth (Franke-van der Merwe growth)
were present then the blocking dip present would only be formed by essentially a two
layer stacked feature and would be weaker. This has also been observed in Vegas
by simulating a 1.33 ML Ag overlayer and measuring the depth of the blocking dips
which were found to have a lot smaller depth.

At approximately 97° in figure 5.3 there is a new feature present in the 3.42 ML
5.3 Results and Discussion

Figure 5.3 Ag surface experimental curves of varying Ag film thicknesses on Al(111) with a \( <110> \) beam incidence. Bulk fcc Al and hcp Al Vegas simulations have been included to show the location of major fcc blocking dips and suspected hcp-caused blocking dips. Note the hcp simulation is plotted separately on the right-hand axis.
and the 4.74 ML coverage. This feature is not present in the bulk fcc simulation, but is present in the bulk hcp simulation. There is also a slight shift to a smaller scattering angle of these new features, this could be due to experimental noise, or an interlayer contraction.

The yield of the curves in figure 5.3 increase gradually due to increasing coverage, and appear to exceed that of the bulk simulated fcc values but not the bulk hcp values, being between the two simulated curves' yields.

Figure 5.4 shows the experimental curves for the Ag signal for the <233> beam incidence for Ag film thicknesses 0.23 ML to 4.74 ML. There were two scattering windows taken for the <233> beam incidence. There are a number of visible blocking dips present in the experimental data, but are present in both the hcp and fcc simulations. The three blocking blocking dips visible are the fcc (114) channel at 84.24° scattering angle, (112) at 100.03° and (111) at 119.50°. These blocking dips are formed from 3, 2 and 3 layers respectively. With this knowledge one would expect the (112) channel to be the strongest channel in fcc growth as it is formed from layers that are closer together. In the experimental curves in figure 5.4 the two 3-layer blocking dips appear to be much bigger than the 2-layer blocking dip for the 3.42 ML and 4.74 ML curves. The two large blocking dips present in the experimental data are found in both the fcc and the hcp simulations, and the blocking dip that would be expected to be the biggest is not, but is only found in the fcc simulation. This evidence points towards the thicker films containing some hcp structure.

Substrate curves have been extracted for all incidences and were found to be very uneventful having only the same blocking dips as the bulk fcc simulations, and were very noisy due to the signal being straggled and shadowed by the Ag overlayer. The yields however appear to be more interesting as they appear to reduce as the overlayer thickness increases: by how much could lead to structural information.
5.3 Results and Discussion

Figure 5.4 Ag surface Experimental curves of varying Ag film thicknesses on Al(111) with a <233> beam incidence. Bulk fcc Al and hcp Al Vegas simulations have been included to show the location of major fcc blocking dips and suspected hcp-caused blocking dips. Note the hcp simulation is plotted separately on the right-hand axis.
Figures 5.5 and 5.6 show how the yields of the Ag overlayer and Al substrate evolve with increasing Ag overlayer coverage for the <110> beam incidence. Three 2° windows to measure the yields have been taken. Point (a) taken at 84.5°, point (b) at 90° and point (c) taken at 97.5°. Point (a) has no blocking dips in the fcc bulk simulation and minimal blocking dips in the hcp bulk simulation. Point (b) is located at the major blocking dip found in the fcc bulk simulation, and point (c) is located at the largest blocking dip found in the bulk hcp simulation. Added to figures 5.5 and 5.6 are the Vegas simulated yields of an entirely fcc and entirely hcp Ag overlayer for thicknesses matching that of the experiment, with the yield measurements being taken at the same scattering angles and of the same 2° window size. Figure 5.5 shows the Ag overlayer yield, figure 5.6 shows the Al substrate yield.

The Ag yield at 0.23 ML Ag coverage appears to be appreciably the same for experiment, simulated fcc Ag and simulated hcp Ag. The Al experimental yield at this coverage appears to be higher than the fcc simulation. For both the non-blocked scattering angle (a) and the fcc-blocked scattering angle (b) the yield appears to be the same as a clean bulk Al simulation (not shown). For the hcp-blocked scattering angle (c) the Al yield appears to be less than for simulated clean bulk Al, also slightly less than for entirely hcp-Ag covered Al, but greater than entirely fcc-Ag covered Al. This information suggests that the covering Ag tends to prefer hcp sites as opposed to fcc sites, but not 100% of Ag atoms taking the hcp sites. Taking a weighted average of the fcc and hcp simulations at the 0.23 ML coverage figures 5.6(a) and 5.6(b) suggested 100% ± 20% of Ag atoms sat on hcp sites. Figure 5.6(c) suggested 50% ± 20% of Ag atoms sat on hcp sites.

When the Ag coverage is increased to 1.33 ML and 1.84 ML the experimental Ag yield appears to be less than both simulations for the scattering angles that contain hcp or fcc blocking dips. For the scattering window that contains no blocking dips
5.3 Results and Discussion

Figure 5.5 Evolution of Ag visibility due to Ag film thickness on Al(111) crystal. <110> beam incidence, yields measured at 84.5° (a), 90° (b) and 97.5° (c). Film thicknesses were measured across 7 angular bins, errors were calculated to be the standard deviation of the thicknesses across these channels.
Figure 5.6 Evolution of Al substrate visibility due to Ag overlayer film thickness on Al(111) crystal. <110> beam incidence, yields measured at 84.5° (a), 90° (b) and 97.5° (c).
the yield is the same for experiment and both simulations for 1.33 ML coverage, but then less than the simulations for the 1.84 ML coverage. This reduced visibility suggests stacked islands as the Ag overlayer would shadow more of itself on the beam incidence, and block more of the outgoing beam causing the yield to be reduced.

For an fcc overlayer, less Ag would be visible at 90° than at any other scattering angle. For hcp coverage, less Ag would be visible at 97.5° than at 90°, which would be less than at 84.5°. For the 1.33 ML coverage it is apparent that in the non-blocked scattering angle the yield matches that of the fcc and hcp simulation. Comparing the fcc and hcp-blocked scattering angle (90° and 97.5°) to the Vegas simulations it can be seen that the difference between the simulation and the hcp-blocked yield is greater than the simulation and the fcc-blocked yield difference. This difference is even more exaggerated at the 1.84 ML coverage. This evidence suggests the presence of hcp stacked islands because there must be enough Ag to form a hcp blocking dip, and as atoms are added the yield will not increase along this blocking dip as the atoms will be blocking themselves.

If we now look at the Al substrate yield when covered by 1.33 ML and 1.84 ML of Ag overlayer it can be seen that the experimental yields are larger than the fcc Vegas simulation, but less than the hcp Vegas simulation. There is also a notable drop in yield from 1.33 ML coverage to 1.84 ML coverage, with the largest drop being in the fcc-blocked scattering angle. The most obvious solution to this drop is an increase in percentage of fcc sites being filled as the overall visibility of Al appears to decrease across all scattering angles.

At this point we have established that there is evidence for 50% to 100% of Ag atoms settling in hcp sites starting from the initial growth upon Al(111). We have also established that this coverage forms stacked island growth at least up to 1.84 ML coverage. Quantifying the islanding is a difficult task, especially using MEIS. To start
5.3 Results and Discussion

Figure 5.7 Vegas simulation of 6 ML of hcp Ag on Al(111), <110> beam incidence, showing the layer by layer Ag yields.

with, what properties of the islanding should be quantified? As MEIS takes a sample across a macroscopic scale, the microscopic features will be difficult to measure, the nanoscopic features however being easier to measure as the statistical sampling size is much greater.

Figure 5.7 shows a Vegas simulation of the breakdown of the layer by layer yields from 6 ML of hcp Ag on Al(111) for the <110> beam incidence. This figure is used to show how many layers of Ag are needed to form individual blocking dips.

As described earlier, for the <110> beam incidence the 1st, 3rd and 5th layers are strongly illuminated by the incoming beam and thus are the major contributors to the overall yield. The first two major blocking dips are produced from ions scattering off the 3rd Ag layer and are located at approximately 80° and 97°. The next major blocking dips are produced from ions scattering off the 5th Ag layer and are located at approximately 86° and 102°. A slight blocking dip can also be seen from ions
scattered off the second layer, but this is located at 90° scattering angle which is at the same location as the major fcc blocking dip.

Comparing these blocking dips to the experimental Ag signals shown in figure 5.3 it can be seen that for the 1.33 ML and 1.83 ML coverages there is a slight appearance of a blocking dip at 97° and becomes more apparent at 3.42 ML and 4.74 ML coverage. Figure 5.8 shows a more close comparison of the 1.33 ML, 3.42 ML and 4.74 ML Ag coverage by overlaying the curves to highlight any differences. It can be seen that the 3-layer feature is very slight at 1.83 ML coverage and more clear at 3.42 ML and 4.74 ML Ag coverage. For 3.42 ML coverage there appears to be a blocking dip at the 5-layer feature point at 102° scattering, but not at the stronger 5-layer feature located at 86°. This situation is reversed for 4.74 ML coverage where a slight blocking dip is apparent at 86° scattering, but not at 102° scattering. An explanation for the blocking dip at 102° could be due to the known unreliability of data at the edges of the scattering window.

Using the information provided by the presence of these blocking dips, the presence of a 3-layer hcp structure is strongly indicated at 3.43 ML and 4.75 ML Ag coverage. A slight presence of 3-layer hcp structure is visible at 1.33 ML and 1.84 ML Ag coverage. Weak evidence suggests 5-layer hcp structure in 3.42 ML and 4.74 ML Ag coverage. This points to stacked hcp islands at 1.33 ML and 1.83 ML, and the possibility of stacked islands at 3.42 ML. There is no evidence for stacked islands at 4.74 ML Ag coverage.

Figure 5.9 shows a Vegas simulation of 6 ML Ag upon Al(111) for the <233> beam incidence split up into layer-by-layer yields. This again shows what features are produced by what layers, and how many layers are needed to produce certain blocking dips. Upon initial inspection it can be seen that the first features are produced by ions scattering off the 3rd layer of Ag. These blocking dips are located at approximately
5.3 Results and Discussion

Figure 5.8 Experimental comparison of 1.33 ML, 3.42 ML and 4.74 ML Ag coverages.

Figure 5.9 Vegas simulation of 6 ML of hcp Ag on Al(111), \(<233>\) beam incidence, showing the layer by layer Ag yields.
85° and 120° scattering angle. The 4th Ag layer produces a feature at 95° but is close to a 5-layer feature located at approximately 91°. The other 5-layer feature is located at approximately 108°. Unfortunately for the <233> incidence, the 3-layer hcp blocking dips are in the same location as fcc blocking dips. This will make it more difficult to establish the presence of hcp stacking.

Figure 5.10 shows a more focussed view of the 3.42 ML and 4.74 ML Ag coverages for a smaller range of scattering angles. The major blocking dip at approximately 85° is a 3-layer fcc feature and is very prominent. The weaker fcc 2-layer feature is also visible at 100°, but is a lot weaker than purely fcc simulations. The final point of interest is a weak blocking dip visible approximately 94° on the 4.74 ML Ag coverage. This feature is an hcp blocking dip created from 4 layers. There is slight evidence for this blocking dip in the 3.42 ML coverage but its existence is debateable. Evidence
of any 5-layer structure is not present, even for the 4.74 ML coverage. The main conclusion that can be drawn from this is that there is no hcp islanding present at 4.74 ML coverage i.e. the hcp stacking does not reach higher than 4 ML. This suggests there is no hcp stacking in the top layer. Following on from that it is suggested that the natural fcc stacking is continued from the 4th layer onwards. This fits in with the growth model because if the fcc growth were to follow on from an odd layer number the new fcc growth would be rotated by 180° i.e. in the twinned direction.

Tying in the findings so far with current literature a number of agreements and a number of disagreements have been found. Stacked islands have been found at 1.33 ML and 1.84 ML Ag coverages, but it is not seen at coverages >= 3.42 ML. Frick and Jacobi [58] state that islands 3 - 4 ML thick start growing before the first layer is completely filled, it has been found here that for coverages of 1.83 ML hcp islands of up to <= 3 ML thick are formed. Hcp islands do not appear to form higher than 4 ML even at coverages > 4 ML suggesting that fcc growth is returned to at 4.74 ML coverage, which is in disagreement with [58], who suggest that after 10 ML coverage a recrystallization to bulk Ag occurs. The claim of a mixture of stacking sequences is agreed with. These measurements have been mainly qualitative however.

Losch and Niehus [61] claim there is an fcc continuation at up to 2 ML Ag coverage with a strong indication of 3D growth. An fcc/hcp mixed growth has been found from the outset with 3D growth, so are in half agreement, half disagreement with their results.

Kim et al. [62] propose a growth model as follows: at the Ag/Al interface alloying results in $Ag_2Al$ domains of hcp stacking in parts, and fcc Ag stacked islands in other parts. On top of these islands the Ag growth follows what is underneath. For the fifth and sixth layers, they discovered that the fcc and hcp found same adsorption sites resulting in a more natural fcc stacking of Ag. These findings are in agreement
5.3 Results and Discussion

with the findings in this report. Results shown in [62] have also found a large lateral contraction of the islands from initial growth. This would be hard to show in MEIS as lateral expansion would result in the broadening of blocking dips: Poor epitaxial growth, straggling of various kinds and excess thermal vibration at the immediate surface would also contribute to the broadening of blocking dips.

To investigate the hcp growth further, Vegas simulations were produced to more closely match the real physical system. This first involved producing a simulation of fcc Al(111) with 4.75 ML of fcc Ag and another simulation of fcc Al(111) with 4 ML hcp Ag and 0.75 ML of fcc Ag on top of that, all in the <233> beam incidence. These two simulations were then added in varying ratios to sum up to 100% and R-factors were calculated for each ratio. The ratio with the lowest R-factor was found to be 61% ± 6% total fcc Ag and 39% ± 6% hcp 4-layer + fcc 0.75 ML growth. The final fit is shown in figure 5.11. The error value was calculated from the corresponding mixing ratio for a R-factor that was 68% greater than the R-factor minimum.

Upon inspection of figure 5.11 it can be seen that the simulation best fit has the blocking dips in the same place as the experiment but they are a lot bigger suggesting that either the epitaxial growth is poor, or there are other features being observed. An entirely hcp Ag overlayer was then simulated to see what features would actually be available from a hcp structure, and was seen to also match several of the features. This led to the additional simulation of 1 layer hcp + 3.75 ML fcc Ag, 2 layer hcp + 2.75 ML fcc Ag and 3 layer hcp + 1.75 ML fcc Ag. The resulting Ag blocking curves are shown in figure 5.12. The physical layout of each model used is shown in figure 5.13.

The variety of curves shown in figure 5.12 reveal that for several stacking sequences the blocking curves are identical in shape and yield. Namely the first and third curves, and the second and fifth curves (from the bottom upwards). To explain this we have
5.3 Results and Discussion

Figure 5.11 Left: Vegas simulations of fcc Al(111) with 4.75 ML of fcc Ag and fcc Al(111) with 4 ML hcp Ag and 0.75 ML of fcc Ag on top of that. Right: Combination of the fcc Ag and mostly-hcp Ag simulations. The fcc to hcp ratio was found to be 61% ± 6% total fcc Ag and 39% ± 6% hcp 4-layer + fcc 0.75 ML growth (61:39). The R-factor for this fit was 0.0074. <233> beam incidence.

to examine the physical layout of the stacking sequences.

Figure 5.13 shows a side view of the fcc/hcp stacking sequences ranging from an entirely fcc overlayer through fcc/hcp stacked coverage, to entirely hcp coverage. If we look at the fcc-3 hcp-2 stack it can be seen that the overlayer is only stacking in an hcp fashion with respect to the Al(111) substrate, taken alone, the Ag overlayer is in fact entirely fcc and thus the scattering curve is expected to be the same as the fcc-5 hcp-0 stacked curve.

The important point to note from this is that the initial starting point of the hcp growth upon the Al i.e. hcp or fcc, would be undetectable from the Ag signal. Also, as the substrate signal is noisy from this initial coverage, it is very difficult to calculate the percentage of initial growth points upon the Al.

The simulations were combined with the in-built curve fitting routines in Igor Pro following equation 5.1. Igor pro uses the Levenberg-Marquardt algorithm [66] [67] to search for the coefficient values that minimize chi-square, defined in equation 5.2.
Figure 5.12 <233> beam incidence Vegas simulations of Ag on Al(111). The Ag structure is varied from 4.75 ML 100% fcc-Ag to 4.75 ML 100% hcp-Ag going through varying numbers of ML of hcp-Ag topped with fcc Ag to sum up to 4.75 ML. Note the yield varying on each y-axis.
5.3 Results and Discussion

Figure 5.13 Side cut diagram of fcc and hcp stacking upon an fcc substrate. Grey is the substrate, blue is hcp stacked atoms and pink is fcc stacked atoms. In certain cases it is arguable whether the atoms are in an fcc or hcp position e.g. fcc-3 hcp-2. These were the models used in the Vegas simulations.
5.3 Results and Discussion

\[ f(p) = \sum_{i=0}^{5} Z_{(f=1-i, h=i)} A_{(f=1-i, h=i)}(p) \]  
\text{(5.1)}

Where \( f(p) \) is the final combined simulation, \( Z_{(f=1-i, h=i)} \) is the percentage ratio of each of the simulated curves, and \( A_{(f=1-i, h=i)}(p) \) are the individual simulated curves.

\[ \chi^2 = \sum_i \left( \frac{y - y_i}{\sigma_i} \right)^2 \]  
\text{(5.2)}

Where \( y \) is a fitted value for a given point, \( y_i \) is the measured data value for the point and \( \sigma \) is an estimate of the standard deviation for \( y_i \).

Chi-square defines a surface in a multidimensional error space. The search process involves starting with an initial guess at the coefficient values, supplied by the user. Starting from the initial guesses, Igor searches for the minimum value by travelling down hill from the starting point on the chi-square surface.

One drawback in using this method is that finding a best fit curve involves finding the bottom of a valley on a multi-dimensional surface, concluding that it is the minimum. Igor does not take into account that this surface may have multiple valleys and will stop at the first valley it encounters. To reduce this problem various starting points as initial guesses were used for each fitting coefficient to start off the curve fitting routine. In all curve fitting operations in this report only one minimum was ever found.

It was found that for \( i = 2 \), where the curve shape was the same as for \( i = 0 \) in the \(<233>\) beam incidence direction, that including both curves in the curve fitting routing produced large errors as essentially both curves produced an equal contribution to the final curve. In reality this may not be the case however. It was decided not to use both the \( i = 0 \) and \( i = 2 \) and just use one instead. The results for this curve fitting will be shown further on.
5.3 Results and Discussion

Figure 5.14 $<110>$ beam incidence Vegas simulations of Ag on Al(111). The Ag structure is varied from 4.75 ML 100% fcc-Ag to 4.75 ML 100% hcp-Ag going through varying numbers of ML of hcp-Ag topped with fcc Ag to sum up to 4.75 ML. Note the yield varying on each y-axis.
5.3 Results and Discussion

Figure 5.14 shows the varying Vegas simulated curves of Ag upon Al(111) for the <110> beam incidence. The Ag has been varied from an fcc structure to an hcp structure, including structure mixes. The physical layout is again shown in figure 5.13. It can be seen in figure 5.14 that the entirely fcc Ag structure has the same blocking curve shape and yield as 2 ML hcp + 2.75 ML fcc, which is the same as figure 5.12. This again presents the same issue as mentioned before that the fcc percentage estimate will be combined with 2-layer hcp + 2.75-layer fcc percentage estimate. The results from the curve fitting for this beam incidence is also shown further on.

Figure 5.15 shows the results of the combined Vegas simulation curve fitting. It can be seen that both simulated curves have very similar shape and yield as their experimental counterparts. The calculated R-factors for the final fits are 0.00335 for the <233> beam incidence and 0.00158 for the <110> beam incidence. This is a much better improvement on the first pass fit of the <233> beam incidence where the R-factor was calculated to be 0.0074.

The R-factor is defined as:

$$R = \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{(I_{\text{exp}} - \lambda I_{\text{sim}})^2}{I_{\text{exp}}} \right]$$  \hspace{1cm} (5.3)

Where N is the total number of points being compared, $I_{\text{exp}}$ and $I_{\text{sim}}$ are the experimental and simulated intensity values (can be monolayer visibility or ion counts at a given point), and $\lambda$ is a scaling factor used to allow for instrumentation effects such as transmission efficiency and acceptance angle of the ion energy analyzer. In all cases listed here $\lambda$ is taken to be 1.

Figure 5.16 shows the initial fitted parameters for the best fit curves shown in figure 5.15. The x-axis essentially is the number of hcp Ag layers underneath fcc layers to sum up to 4.75 ML in total of Ag. So 0, for example, would equate to an entirely fcc structure, 4.75 would be an entirely hcp structure. It can be seen that both
5.3 Results and Discussion

Figure 5.15 Combined Vegas simulation curve fits of <233> and <110> beam incidence laid over their corresponding experimental counterparts.

curves mostly fall into each others’ error boundaries. The sum of all the percentages for the <233> curve was 98.4% ± 21.3% and <110> curve was 87.5% ± 27.4%. So for the <233> incidence, 100% of the total yield can be found within the error boundaries of the Vegas simulations i.e. no material is unaccounted for.

Figure 5.16 does however ignore the fact that the hcp-0 and hcp-2 structures have the same curves and during the fitting process they were counted as one. Figure 5.16 counts hcp-0 as having a 100% contribution and hcp-2 as a 0% contribution. This is most probably not the case. Figure 5.16 is an example of an edge calculation, the other being hcp-0 being 0% and hcp-2 being approx 57%. This example would appear to be unrealistic as it would not follow the apparent trend of the other varied-thickness hcp islands. The first immediately obvious solution would be to follow the line that joins the points on figure 5.16 and place a percentage to where the joining line intersects the hcp-2 axis point, and then subtract this percentage from the hcp-0 value.

The other method allow for the combining of the hcp-0 and hcp-2 curves would be
Figure 5.16 The hcp/fcc population profile of 4.74 ML of Ag on Al(111). The x-axis represents the number of hcp layers present in a particular stack, with the number of fcc layers covering that particular stack being 5 - hcp layers. Therefore 0 is pure fcc. The y-axis representing the percentage population of that given stack.
to take into account the percentage hcp/fcc growth taken from the 0.23 ML coverage and incorporating this value into the layer structure distribution. This method is based on calculations rather than assumption and has therefore been deemed better.

At the beginning of this chapter we found that the percentage of Ag settled in hcp sites was found to be between 50% and 100% by using basic calculations upon 3 scattering angles. This calculation has been expanded to cover all points across the experimental angle sample window. The substrate signal with 0.23 ML Ag coverage was compared to Vegas simulations of the substrate with 0.25 ML of Ag in the hcp site and in the fcc site. The two simulations were added in varying ratios according to equation 5.4 and an R-factors were calculated for each ratio (similar to the construction of 6 curves described in figure 5.12).

\[ f_{\text{combined}}(x) = i f_{\text{fcc}}(x) + (1 - i) f_{\text{hcp}}(x) \] (5.4)

Where \( x \) is scattering angle, \( f_{\text{combined}}(x) \) is the combined average simulation, \( f_{\text{fcc}}(x) \) is the fcc simulation, \( f_{\text{hcp}}(x) \) is the hcp simulation, and \( i \) is the decimal percentage of fcc structure present.

Figure 5.17 shows the best substrate fit simulation compared to the experimental substrate curve, left, and the R-factor curve on the right for the \(<110>\) beam incidence. The R-factor minimum is found at 43% fcc with a R-factor of 0.00548. A point to note upon the fitted curve is that the experimental noise is comparable to the difference in yields of the fcc and the hcp Vegas simulations. This has implications on the accuracy of the best fit and is reflected in depth of the R-factor curve.

A further method was employed to find the best fit percentage of fcc and hcp for the 0.23 ML coverage. It essentially used equation 5.4 in combination with the in-built curve fitting algorithm contained in Igor Pro. The reason for this was an error value could be calculated from the fitting procedure. Using this method, the
5.3 Results and Discussion

Figure 5.17 Left shows Vegas substrate simulations compared to experimental substrate signal for the <110> beam incidence. The dotted curves are substrate simulations with 0.24 ML of hcp and fcc Ag overlayers. The blue curve is the best fit combination of the fcc and hcp simulations. The yellow curve is the experimental substrate signal. Right shows the corresponding R-factor curve for the combined simulation fitting. The R-factor minimum is found at 47% fcc with a R-factor of 0.00548.

percentage fcc was found to be 43.6% ± 5.2%. This value encompasses the previously calculated value of 47% but is preferred due to the more advanced fitting techniques and statistical methods used. For further details on Igor’s fitting mechanisms and mathematics, refer to [68], [69] and [70].

A similar curve fitting procedure was attempted for the <233> incidence. The simulation curves have a lot of narrow and deep blocking features very close together. This meant that features between the fcc and the hcp curves were almost overlapping. Combined with the noise of the experimental signal, any fcc percentage measurement would be very inaccurate.

Using the 47% measured percentage of fcc structure at 0.23 ML, the difference
between this percentage measurement and the fcc percentage calculated for the 4.74 ML Ag coverage was computed. This difference was then taken to be the percentage of hcp-2. The two separate layer populations shown in figure 5.16 were then averaged and the new data points at hcp-0 and hcp-2 were then inserted to produce figure 5.18. It can be seen that the data point at hcp-2 seems to follow the trend of all the other hcp layer thicknesses. The absolute values of this curve are shown in table 5.1.

The percentage fcc growth measurement was extracted from only the 0.23 ML coverage for three reasons. Firstly, as mentioned before, the fcc and hcp-2 structures cannot be individually distinguished from the overlayer structure, leaving the measurements to be taken from the substrate. Secondly, the Ag overlayer quite heavily straggles the incoming ion beam meaning the thicker the overlayer the noisier and harder it is to extract the substrate signal. Thirdly, the Ag overlayer has been shown to stack in multiple monolayer thicknesses before forming a complete initial single overlayer. The aim of the above calculation was to measure the amount of Ag atoms

Figure 5.18 This graph shows the thickness of hcp Ag layers vs. their percentage populations. The Ag hcp layers are then topped with fcc Ag topping the total number of layers to 5. This data is taken from 4.74 ML Ag on Al(111).
Table 5.1 A profile of the percentage population of hcp stacking. The number of hcp layers is then topped with fcc Ag topping the thickness to 5 layers. This data is taken from 4.74 ML Ag on Al(111).

<table>
<thead>
<tr>
<th>No. of hcp Ag layers</th>
<th>Percentage population of Ag in given layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>43.6 ± 5</td>
</tr>
<tr>
<td>1</td>
<td>8.4 ± 8</td>
</tr>
<tr>
<td>2</td>
<td>15.2 ± 5</td>
</tr>
<tr>
<td>3</td>
<td>8.2 ± 2</td>
</tr>
<tr>
<td>4</td>
<td>7.9 ± 6</td>
</tr>
<tr>
<td>5</td>
<td>10.6 ± 2</td>
</tr>
</tbody>
</table>

that settle in the fcc and hcp sites from the initial growth. The greater the coverage the greater the stacking effect thus adding more unknown variables into the calculation. With 0.23 ML coverage the stacking effect will be at a minimum. At 4.74 ML coverage substrate measurements could not be made as the substrate was not appreciably visible.

Comparing these results to Kim’s findings it can be seen that more than just fcc and 4-layer hcp structures are present. The relative percentage growth has not been measured anywhere else.

5.4 Conclusion

The epitaxial structure of ultrathin films of Ag upon Al(111) has been studied using MEIS. Findings have disagreed with some literature, partially agreed with other literature, and uncovered new details measurements on the growth modes present.

The initial settling points of the Ag atoms at 0.23 ML coverage were found to be 43.6% ± 5.2% fcc, 56.4% ± 5.2% hcp, with the Ag rapidly forming stacked islands.
The stacking is indicated by deeper blocking dips being present at lower coverages than is found in flat layer growth Vegas simulations of the same overlayer coverage. The presence of stacked islands is also indicated at 1.33 ML and 1.83 ML coverage due to the weak appearance of blocking dips that are formed from 3 layer structures.

Evidence for the presence of stacked islands at 3.42 ML coverage and 4.74 ML coverage became weaker suggesting that instead of forming higher stacked islands, continued deposition contributed more to the lateral growth of these islands i.e. outwards not upwards growth. The continued growth was shown to be a mixture of fcc and hcp. This evidence was obtained from qualitative analysis of the Ag overlayer signals.

After performing a qualitative analysis of the 4.74 ML Ag coverage and combining this with the initial fcc/hcp site settling percentages, it was found that the Ag structure consisted of 43.6% ± 5% fcc structure and 56.4% ± 5% hcp structure. The hcp structure was divided into substructures that initially grew as hcp and then reverted back to its native fcc structure. The thicknesses of the hcp structures varied from 1 ML hcp topped with 3.75 ML fcc through to 4.74 ML hcp and no fcc structure. The percentage population of these hcp structures are shown in table 5.1 and figure 5.18. At 3.75 ML and 4.74 ML coverage Stranski-Krastanov or Volmer-Weber growth is indicated, but the evidence is weak at 4.74 ML coverage.

Comparing these findings to current literature we see that we are again in disagreement with some literature, partial agreement with others, and have reported a phenomenon that has not been previously reported. In particular, our agreements are that stacked hcp and fcc islands are present. Our disagreements are as to how thick the hcp grows.

The earliest paper [58] shows that before the first layer is completely filled clusters 3-4 layers thick are formed. We have shown that clusters 3 layers thick are found
at coverages as low as 1.33 ML. A recrystallization into bulk Ag occurs after 10 ML coverage according to [58], we have found that bulk Ag is approached at 4.74 ML thick Ag as the percentage of full 5-layer hcp structure at this coverage is 10.5%±2.2%. This percentage could however carry on up to 10 ML Ag coverage. Losch and Niehus [61] show that up to 2 ML Ag coverage there is an fcc continuation. This work disagrees with these findings.

Kim et al. [62] propose a growth model of a mixture of fcc and hcp islands growing in individual domains. The hcp resulting from an interfacial alloy of Ag2Al forming. They propose that the hcp growth continues up to a thickness of 4 ML where the hcp stacking order can resume with the surrounding fcc order, resulting in the coalescing of the fcc and hcp islands. The findings in this report suggest that the hcp reverting into fcc growth occurs continually through the continued deposition resulting in hcp structures of thicknesses 1 ML to 5 ML, as previously mentioned.

The presence of twinning was investigated and it was found to not be present. There would be several clearly visible unique blocking dips if twinning were present and these are not seen in the experiment. These results are consistent with information available in the current literature.

The presence of varying thickness hcp structures has not been reported before, neither has the percentage population of the fcc and hcp structures.

The size of the fcc/hcp domains has not been measured as it is not a suitable MEIS experiment. Scanning tunnelling microscopy would prove to be a likely candidate if further investigations were to proceed in this direction.
Chapter 6

Conclusion

This chapter summarises and draws conclusions from the work and experimental findings in this thesis. General tips and best practices for MEIS have been discussed, followed by suggestions for improvements for the MEIS experimental equipment and currently available software.

6.1 Experimental Conclusions

MEIS has been used to show the crystallography of ultrathin films of Pd on Cu(111) and Ag on Al(111). To achieve this, a custom software suite has been developed to process, calibrate, analyse the data output by the MEIS experimental equipment, and to produce more complex simulations of the depth profiling of this data.

The use of the custom built software proved to be a very time consuming affair, but in the long run saved a lot of time and made the processed data very consistent. They were very effective tools. If the use of this software is continued then the time spent developing it will have been well spent.
6.1 Experimental Conclusions

6.1.1 Pd on Cu(111) Conclusions

Making MEIS measurements upon ultrathin films of Pd on Cu (111) we have shown that the Pd tended to intermix with the Cu up to at least (and appreciably) 3 ML deep even from the very initial stages of growth. This intermixed Pd and Cu has been shown to retain an fcc structure, the Pd exhibiting twinning and the Cu staying in registry with the bulk. This twinning has been shown to continue up to at least 3.89 ML coverage, with an intermixing still being apparent at this coverage.

The percentage of twinning has been measured to be 42% from the very initial stages of growth (0.22 ML) and evolves to approximately 60% in coverages up to 3.9 ML. The intermixed Copper does not show twinning at any coverage.

For initial coverages (0.22 ML) we have measured the Pd concentrations for the first three layers of the Pd Cu(111) sample. These were found to be 27% ± 3%, 67% ± 3% and 6% ± 3% of Pd in the 1st, 2nd and 3rd layers.

The surface relaxation has also been measured for the first and second interlayer spacing for 0.22 ML coverage on Pd Cu(111). The first interlayer was measured to have a 4% ± 3.3% contraction and the second interlayer having an expansion of 4% ± 4.5% relative to bulk Cu.

These results have both confirmed findings in current literature, disagreed with other literature and also covered new ground.

6.1.2 Ag on Al(111) Conclusions

We have used MEIS to make structural measurement upon ultrathin films of Ag upon Al(111). We have shown that the initial settling points of the Ag upon the Al(111) surface at a coverage of 0.23 ML is 43.6% ± 5.2% fcc and 56.4% ± 5.2% hcp. This growth evolves into stacked islands as Ag is further deposited. The stacked hcp Ag
6.1 Experimental Conclusions

Table 6.1 A profile of the percentage population of hcp stacking. The number of hcp layers is then topped with fcc Ag topping the thickness to 5 layers. This data is taken from 4.74 ML Ag on Al(111).

<table>
<thead>
<tr>
<th>No. of hcp Ag layers</th>
<th>Percentage population</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>43.6 ± 5.2</td>
</tr>
<tr>
<td>1</td>
<td>8.4 ± 8.3</td>
</tr>
<tr>
<td>2</td>
<td>15.2 ± 5.2</td>
</tr>
<tr>
<td>3</td>
<td>8.2 ± 2.0</td>
</tr>
<tr>
<td>4</td>
<td>7.9 ± 6.2</td>
</tr>
<tr>
<td>5</td>
<td>10.6 ± 2.2</td>
</tr>
</tbody>
</table>

reverts to the natural fcc growth after a range of thicknesses, as shown in table 6.1. The thickness of the film used to make these measurements was 4.74 ML.

These findings agree with certain current literature, disagree with others and covers previously unreported information.

6.1.3 General Conclusions and MEIS Best Practices

The Holy Grail of fitting MEIS curves is to have a fitted curve that matches both the blocking dip locations and the final yield. An attempt at this was made for the Ag Al(111) chapter, but was not for the Pd Cu(111) chapter. The Ag Al curve is a good match with small deviations, particularly at high and low scattering angles. This will be due to unknown structural effects that have not been simulated e.g. domain boundaries, noise and structural details created by crystallographic imperfections. The structure domain boundaries is a relatively macroscopic effect and to implement this into Vegas simulations would be unfeasible. As the number of possibilities of simulated structures is effectively infinite, the presence of an unknown structure would be akin to searching for a needle in a haystack. One has to use complimentary
techniques to try and reveal information that MEIS analysis would find difficult.

When performing MEIS experiments it is essential that one has thoroughly investigated the crystalline structure before performing experiments. This would cover having a plan to take measurements at more beam angles than could be fit into a working day (and the lifetime of a sample before being contaminated at UHV pressures). A prime reason for this was experienced during the Ag Al(111) experiment. The plan was to have 1, 2 and 3 layer alignment beam incidences. Unfortunately due to the high stopping power, straggling and shadow cone size of the Ag overlayer, aligning the beam in a 3 layer alignment was not achieved. The planned experiment of trying to reveal the first, second and third Ag and Al populations had to be changed. Fortunately the other aim was to investigate the quantity of hcp and fcc structure present in the Ag Al system.

Curve fitting is a procedure that comes automated in a lot of analysis packages. The main problem found with using an automated procedure to perform this process is that the algorithm used is not intelligent. The best fit curve that models a real system could turn out to be completely unphysical. This was experienced when using the simulation fitting algorithm in SIMNRA. A PdCu surface was simulated with an initial surface roughness instead of interfacial roughness. The resulting model fits almost perfectly, the problem being the physical model would have been similar to the dimensions of a pin cushion, which is highly unlikely. The main point here is that having a physically realistic model should take a higher precedence than the aesthetics of a fitted curve.

The process of fitting curves essentially involves designing a model, seeing how well that model fits to the real system by varying parameters and thus refining the model. Then more detail is added to the model and the process is repeated. This increasing of detail could essentially go on until the model is the real system. As the real system
6.1 Experimental Conclusions

is not perfect i.e. made in the real world, the only 100% accurate representation of a physical system is the physical system itself. We can only approximate. The key point from this is knowing where to stop refining a model of a system. The level of detail needs to be decided as the level of specificity of that system varies in which the system is made. For example, in this thesis we found the amount of twinning exhibited in 0.24 ML of Pd on Cu created by electron beam deposition at a given UHV pressure. There may be many factors that could affect this twinning. If we were to narrow down the model to a very high level of accuracy then we would be tying this accuracy to the growth conditions in which the sample was made. If another group were to repeat the experiment they may get different results due to the fact that the growth conditions were different. The point being made is that general trends are a lot more useful than increasingly exacting specifics. As a technology moves from fundamental science through to engineered and high-technology manufactured products, the knowledge of the physical development of a system for a less than optimal growth conditions (e.g. a manufacturing environment) is more beneficial than in the perfect case (e.g. a strictly controlled lab). This is not an excuse for lack of detail when investigating the fundamental properties of a novel system.

Blocking curves in MEIS are present from the blocking of the outward ion beam. The features can reveal a lot about the structure that is causing the blocking dips. Unfortunately these blocking dips are not always present. If trying to fit a Vegas simulation to a featureless MEIS angular projection one can only measure the yield and hence only the inward beam’s alterations. If blocking dips are present then one can take advantage of blocking and shadowing effects.

The amount of data output by MEIS can be a double edged sword. The sheer amount of data can be incredibly useful in the fact that if a particular beam incident and scattering angle proves to be fruitless, there will always be more data to look at.
The downside to this however is that it takes time to analyse and prepare the data. Vegas simulations take time to set up and to run. If one has an awful lot of interesting data, which sets are the most useful? This idea also extends to Vegas simulations. One could perform thousands of simulations. The key thought that comes with this is that working intelligently is far more fruitful than throwing everything at a system. One only needs to get experimental results from a couple of scattering angles, and only perform a couple of Vegas simulations for comparison. So long as they are the right ones.

A method to approach a large data set is to skim across lots of data, performing quick analyses and basic simulations. This can result in the poorer data sets being discarded earlier on, but also being checked to see if they are valid.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

Stereographic Projections
Figure A.1 Stereographic Projection for the fcc(111) surface
Figure A.2 Azimuthal cut for the fcc(111) surface: (011) azimuth
Figure A.3 Azimuthal cut for the fcc(111) surface: (112) azimuth
Figure A.4 Azimuthal cut for the fcc(111) surface: (211) azimuth
Appendix B

SuperMega CurveFit key data structures

****** Exp_parameters array ******
[5] Time (doesn’t exist in older files)
[par_nums-6] N/A
[par_nums-5] Start Angle
[par_nums-4] Angle increment
[par_nums-3] N/A
[par_nums-2] Start energy
[par_nums-1] Energy increment
[20] BEAM energy (manual input)
[21] MASS OF SURFACE
[22] MASS OF SUBSTRATE
[23] INTERPLANAR spacing
[24] no of LAYERS
[25] energy loss in surface ev/A
[26] constant substr width
[27] energy loss in SUBSTRATE ev/A
[28] Substrate interlayer spacing
[29] no. of LAYERS in Substrate
[30] Substrate fudge factor
[31] Bulk counts finish
[32] Ender Bender Max shift

[182] number of parameters in original data file (usually 11 or 12)

$\text{par_nums}$

****** Coef_matrix array ******

coef_matrix[0] = $m$, gradient of line combined with sigmoid
coef_matrix[1] = $c$, intercept of line combined with sigmoid
coef_matrix[2] = $x_0$, centre of sigmoid
coef_matrix[3] = $b$, width of sigmoid
coef_matrix[4] = $a_1$, height of substrate gaussian
coef_matrix[5] = $x_1$, position of gaussian
coef_matrix[6] = $b_1$, width of gaussian
coef_matrix[7] = $a_2$, height of surface gaussian
coef_matrix[8] = $x_2$, position of gaussian
coef_matrix[9] = $b_2$, width of gaussian

****** N_Const array ******

N_Const[0] = gradient of line BG (lower)
N_Const[1] = gradient of line BG (upper)
N_Const[2] = intercept of line BG (lower)
N_Const[3] = intercept of line BG (upper)
N_Const[4] = centre of sigmoid (lower)
N_Const[5] = centre of sigmoid (upper)
\textbf{N\_Const}[6] = width of sigmoid (lower)
\textbf{N\_Const}[7] = width of sigmoid (upper)
\textbf{N\_Const}[8] = height of substrate gaussian (lower)
\textbf{N\_Const}[9] = height of substrate gaussian (upper)
\textbf{N\_Const}[10] = position of sub gaussian (lower)
\textbf{N\_Const}[11] = position of sub gaussian (upper)
\textbf{N\_Const}[12] = width of sub gaussian (lower)
\textbf{N\_Const}[13] = width of sub gaussian (upper)
\textbf{N\_Const}[14] = height of surface gaussian (lower)
\textbf{N\_Const}[15] = height of surface gaussian (upper)
\textbf{N\_Const}[16] = position of gaussian (lower)
\textbf{N\_Const}[17] = position of gaussian (upper)
\textbf{N\_Const}[18] = width of gaussian (lower)
\textbf{N\_Const}[19] = width of gaussian (upper)
Appendix C

SuperMega CurveFit key data structures

#pragma rtGlobals=1 // Use modern global access method.

// Super Mega Curvefit Igor Macro v1.22
// Written By Chris Howe at Loughborough University
// chrishowe2@hotmail.com

// Version 1.0 - first fully functioning version
// Version 1.1 - Tile fixer added
// Version 1.2 - Calibration added
// Version 1.21 - Rutherford correction added after calibration
// Version 1.22 - Can now load waves from tile function using load_igor_wave() function

// This function returns a sum of the gaussians from the isotopes of the surface

function isotope_gauss_surf(x, height, centre, width)
  variable x
  variable height
  variable centre
  variable width
  WAVE W_freq_surf
  WAVE W_k_fact_surf
  NVAR current_angle
  variable i = 0
  variable output = 0
  variable no_of_isotopes = dimsize(W_mass_surf,0)
  do
    output += W_freq_surf[i] * exp(-.5*(x-((centre/W_k_fact_surf[i][current_angle])*W_k_fact_surf[i][current_angle])/width)^2)
    i += 1
  while(i < no_of_isotopes)
  return height*output
End

// This function returns a sum of the gaussians from the isotopes of the substrate
function isotop_gauss_sub(x, height, centre, width)
    variable x
    variable height
    variable centre
    variable width
    WAVE W_k_fact_sub
    WAVE W_freq_sub
    NVAR current_angle
    variable i = 0
    variable output = 0
    variable no_of_isotopes = dimsize(W_mass_surf, 0)
    do
        output += W_freq_sub[i] * exp(-.5 * (((x - (centre/W_k_fact_sub[0][current_angle]) * W_k_fact_sub[i][current_angle]) / width)^2)
        i += 1
    while (i < no_of_isotopes)
    return height - output
End

function fit_BG_line(angle)
    variable angle
    WAVE surf_poly_points
    WAVE final_tile
    Make/N=(dimsize(final_tile, 1))/0 counts_data
    counts_data = final_tile[angle][p] // take a slice of the final data to perform fit
    CurveFit/Q line kvWave=line_coef, counts_data[0,(surf_poly_points[angle][6])] /X=energy_scale // fit straight line BG
End

function copy_fit_parameters(angle)
    variable angle
    WAVE line_coef
    WAVE surf_poly
    WAVE surf_poly_points
    WAVE final_tile
    Make/N=10/0/0 coef_matrix
    coef_matrix[0] = line_coef[1] //m
    coef_matrix[1] = line_coef[0] //c
    coef_matrix[2] = surf_poly[angle][5] //x0 centre of sigmoid
    coef_matrix[4] = final_tile[angle][surf_poly_points[angle][4]] //a1 height of substrate gaussian
    coef_matrix[7] = final_tile[angle][surf_poly_points[angle][1]] //a2 height of surface gaussian
    coef_matrix[8] = surf_poly[angle][1] //x02 position of gaussian
    coef_matrix[9] = surf_poly[angle][0] - surf_poly[angle][1] //b2 width of gaussian
End

function produce_constraints(angle)
    variable angle
    variable i = 0
function produce_constraints_without_lin(angle)

variable angle
variable i = 0
WAVE surf_poly
WAVE final_data
WAVE coef_matrix

Make/O/T/N=20 T_Constraints
Make/N=16/D/O N_Const

N_Const[0] = surf_poly[angle][5] - (surf_poly[angle][4] - surf_poly[angle][5]) // centre of sigmoid

do
    // convert this to a text wave
    T_Constraints[i] = "K" + num2str((i/2)) + " > " + num2str(N_Const[i])
    T_Constraints[i+1] = "K" + num2str((i/2)) + " < " + num2str(N_Const[i+1])

end

while(i<dimsize(N_Const,0))

End

This function produces the upper and lower limits for the cure fitting routine - without fitting the linear BG

Function produce_constraints without_lin(angle)
do // convert this to a text wave
    T_Constraints[i] = "K" + num2str((i/2+2)) + " > " + num2str(N_Const[i])
    T_Constraints[i+1] = "K" + num2str((i/2+2)) + " < " + num2str(N_Const[i+1])
    i += 2
while(i < dimsize(N_const, 0))

End

// This function stores all the fits produced by the fitting routine

function store_data(angle)
    variable angle
    variable i = 0
    WAVE final_fit
    WAVE fit_counts_data
    WAVE final_fit_BG
    WAVE final_fit_sub_gauss
    WAVE final_fit_surf_gauss
    WAVE coef_matrix
    WAVE energy_scale
    WAVE W_k_fact_surf
    WAVE W_k_fact_sub
    WAVE W_freq_surf
    WAVE W_freq_sub
    NVAR kwik
    if(kwik == 1)
        do
            final_fit(angle)[i] = round(fit_counts_data[i])
            final_fit_BG(angle)[i] = round((coef_matrix[0] + energy_scale[i] + coef_matrix[1]) / (1 + exp((energy_scale[i] - coef_matrix[2]) / (coef_matrix[3]))))
            final_fit_sub_gauss(angle)[i] = round(coef_matrix[4] * exp(-0.5 * (energy_scale[i] - coef_matrix[5]) / coef_matrix[6])^2)
            final_fit_surf_gauss(angle)[i] = round(coef_matrix[7] * exp(-0.5 * (energy_scale[i] - coef_matrix[8]) / coef_matrix[9])^2)
        i += 1
        while(i < dimsize(fit_counts_data, 0))
    else
        do
            final_fit(angle)[i] = fit_counts_data[i]
            final_fit_BG(angle)[i] = ((coef_matrix[0] + energy_scale[i] + coef_matrix[1]) / (1 + exp((energy_scale[i] - coef_matrix[2]) / (coef_matrix[3]))))
            final_fit_sub_gauss(angle)[i] = isotope_gauss_sub(energy_scale[i], coef_matrix[4], coef_matrix[5], coef_matrix[6])
            final_fit_surf_gauss(angle)[i] = isotope_gauss_surf(energy_scale[i], coef_matrix[7], coef_matrix[8], coef_matrix[9])
        i += 1
        while(i < dimsize(fit_counts_data, 0))
    endif
End

// This function integrates all the 2D plots that have been fit - produces substrate and surface blocking curves

function integrate_data(angle)
    variable angle
    WAVE fit_counts_data
    WAVE final_fit_BG
    WAVE final_fit_sub_gauss
    WAVE final_fit_surf_gauss
    WAVE surf_integral
    WAVE substr_integral
    WAVE bg_integral
    1 += 2
    do
        final_fit(angle)[i] = fit_counts_data[i]
        final_fit_BG(angle)[i] = (coef_matrix[0] + energy_scale[i] + coef_matrix[1]) / (1 + exp((energy_scale[i] - coef_matrix[2]) / (coef_matrix[3])))
        final_fit_sub_gauss(angle)[i] = isotope_gauss_sub(energy_scale[i], coef_matrix[4], coef_matrix[5], coef_matrix[6])
        final_fit_surf_gauss(angle)[i] = isotope_gauss_surf(energy_scale[i], coef_matrix[7], coef_matrix[8], coef_matrix[9])
    i += 1
    while(i < dimsize(fit_counts_data, 0))
End
WAVE angle_scale
Make/N=(dimsize(final_fit_BG,1))/D/O temp
temp[*] = final_fit_BG[angle][p]
bg_integral[angle] = sum(temp)
temp[*] = final_fit_sub_gauss[angle][p]
substr_integral[angle] = sum(temp)
temp[*] = final_fit_surf_gauss[angle][p]
surf_integral[angle] = sum(temp)
End
/***********************************************************************/
// This function stores all the coefficients from all the waves
/***********************************************************************/
function store_coefs(angle)
    variable angle
    variable i = 0
    WAVE coef_matrix_total
    WAVE coef_matrix
do
        coef_matrix_total[i][angle] = coef_matrix[i]
i += 1
    while(i < 10)
End
/***********************************************************************/
// Coordinates the whole lot
/***********************************************************************/
function total_fit()
    variable i = 0
    variable j = 0
    variable k = 0
    variable/G current_angle
    variable/V_fitoptions = 4
    WAVE counts_data
    WAVE angle_scale
    WAVE W_mass_surf
    WAVE W_freq_surf
    WAVE W_mass_sub
    WAVE W_freq_sub
    NVAR kwik
dowindow/K progress_graph
execute("progress_graph();")
surf_polygon("button0")
Make/N=183/D/O chi_sq_wave,surf_integral,substr_integral,bg_integral // waves for the integrals, and of the chi squared fits
Make/N=(10,183)/D/O coef_matrix_total
duplicate/O final_tile final_fit //stores the waves for every total fit
final_fit = 0
chi_sq_wave = 0
duplicate/O final_fit final_fit_BG //stores the fitted background
duplicate/O final_fit final_fit_sub_gauss //stores the fitted substrate gaussian
duplicate/O final_fit final_fit_surf_gauss //stores the fitted surface gaussian
Make/N=(dimsize(W_mass_surf,0),183)/D/O W_k_fact_surf
Make/N=(dimsize(W_mass_sub,0),183)/D/O W_k_fact_sub
final_fit_BG = 0
final_fit_sub_gauss = 0
final_fit_surf_gauss = 0
do // calculate all the k-sq values for all the isotopes for all angles - used in the isotope gaussian functions
do
    W_k_fact_surf[j][i] = ksq(1,(W_mass_surf[j]),angle_scale[i])
j += 1
while(j < dimsize(W_mass_surf,0))
do
  \( W_k\text{\_fact\_sub}[k][i] = \text{ksq}(1, (W\text{\_mass\_sub}[k]), \text{angle\_scale}[i]) \)
  \( k = 1 \)
  while (k < \text{dimsize}(W\text{\_mass\_sub}, 0))
    \( j = 0 \)
    \( k = 0 \)
    \( i = 1 \)
    while (i < 183)
      \( i = 0 \)
      do //main loop
        current\_angle = i
        fit\_BG\_line(i)
        copy\_fit\_parameters(i)
        produce\_constraints(i)
        //produce\_constraints\_without\_lin(i)
        Smooth 1, counts\_data
        if (kwik == 1) //quick fit or not
          FuncFit/Q/N/L=(\text{dimsize(counts\_data, 0)})/H="0000000000" KWIK\_Spec\_fit coef\_matrix
counts\_data /X=energy\_scale /D /C=T\_Constraints
        else
          FuncFit/Q/N/L=(\text{dimsize(counts\_data, 0)})/H="0000000000" Spec\_fit coef\_matrix
counts\_data /X=energy\_scale /D /C=T\_Constraints
        endif
        chi\_sq\_wave(i) = V\_chisq
        store\_data(i)
        integrate\_data(i)
        store\_coeffs(i)
      \( i = 1 \)
    while (i<183)
  print "Total Fit = " + num2str(sum(chi\_sq\_wave))
End

/*****************************
// This is the function for the complete fit
/*****************************/

Function Spec\_fit(w,x) : FitFunc
Wave w
Variable x
//CurveFitDialog/ These comments were created by the Curve Fitting dialog. Altering them will
//CurveFitDialog/ make the function less convenient to work with in the Curve Fitting dialog.
//CurveFitDialog/ Equation:
//CurveFitDialog/ f(x) = (m*x+c)/(1+exp(-(x-xO)/b)) + a1*exp(-.5*((x-x01)/b1)^2) + a2*exp
//CurveFitDialog/ (-.5*((x-x02)/b2)^2)
//CurveFitDialog/ End of Equation
//CurveFitDialog/ Independent Variables 1
//CurveFitDialog/ x
//CurveFitDialog/ Coefficients 10
//CurveFitDialog/ w[0] = m line\_coef[1]
//CurveFitDialog/ w[1] = c line\_coef[0]
//CurveFitDialog/ w[4] = a1 counts[pts\_conv[1]]
//CurveFitDialog/ w[5] = x01 pts\_conv[1]
//CurveFitDialog/ w[7] = a2 counts[pts\_conv[2]]
//CurveFitDialog/ w[8] = x02 pts\_conv[2]
return (w[0]*x+w[1])/(1+exp(-(x-w[2])/w[3])) + isotopo\_gauss\_sub(x, w[4], w[5], w[6]) +
isotope\_gauss\_surf(x, w[7], w[8], w[9])
End
/*****************************/
// This is the quick function for the complete fit

//**********************************************************************
Function KWIK_Spec_fit(w,x) : FitFunc

Wave w
Variable x
//CurveFitDialog/ These comments were created by the Curve Fitting dialog. Altering them will
//CurveFitDialog/ make the function less convenient to work with in the Curve Fitting dialog.
//CurveFitDialog/ Equation:
//CurveFitDialog/ f(x) = (m*x+c)/(1+exp((x-x0)/b)) + a1*exp(-.5*((x-x01)/b1)^2) + a2*exp
//CurveFitDialog/ (-.5*((x-x02)/b2)^2)
//CurveFitDialog/ End of Equation
//CurveFitDialog/ Independent Variables 1
//CurveFitDialog/ x
//CurveFitDialog/ Coefficients 10
//CurveFitDialog/ w[0] = m line_coef[1]
//CurveFitDialog/ w[1] = c line_coef[0]
//CurveFitDialog/ w[4] = a1 counts[pts_conv[1]]
//CurveFitDialog/ w[7] = a2 counts[pts_conv[2]]
//CurveFitDialog/ w[8] = x02 pts_conv[2]
return (w[0]*x+w[1])/(1+exp((x-w[2])/w[3])) + w[4]*exp(-.5*((x-w[5])/w[6])^2) + w[7]*exp
(-.5*((x-w[8])/w[9])^2)

End

/**********************************************************************
// Returns the K squared coefficient for a binary collision
//**********************************************************************

function ksq(m1,m2,angle)
variable m1 //beam mass
variable m2 //target mass
variable angle
return ((m2^2-(m1*sin(angle/57.29578))^2)*0.5+m1*cos(angle/57.29578))/(m1+m2)^2

End

**********************************************************************
// Takes an energy cut at a given angle (selected by side panel)
**********************************************************************

/**********************************************************************
// Creates all the polygons and puts them in a arrays of energy, point number, and as a mask over
final tile
**********************************************************************

function surf_polygon(ctrlName)
String ctrlName
WAVE exp_parameters
WAVE angle_scale
WAVE final_tile
NVAR mass_surf
NVAR mass_sub
variable angle
variable i = 0
variable j = 0
variable par_nums = exp_parameters[182]
variable thickness = exp_parameters[23] * exp_parameters[24]
variable pl_in = thickness/(cos(exp_parameters[3]/57.29578))
variable pl_in_substr = (exp_parameters[29] * exp_parameters[28])/(cos(exp_parameters[3]/57.29578)) //the path length in the Sub only
variable pl_out
variable pl_out_substr //the path length out the Sub only
calc_masses() // work out masses of substrate and surface
```c
Make/N=(183,2)/D/O k_factors
Make/N=(183,7)/D/O surf_poly // holds the energy values of the polygons
Make/N=(dimsize(final_tile,1))/D/O energy_scale
duplicate/O final_tile polygons //final wave to store polygons in polygons = 0
surf_poly = 0
energy_scale = exp_parameters[exp_parameters[182]-2] + (exp_parameters[exp_parameters[182]-1] + p)
angle_scale = ( p * exp_parameters[par_nums-4]) + exp_parameters[par_nums-5]
do // this loop goes through each angular slice
   angle = i * exp_parameters[par_nums-4] + exp_parameters[par_nums-5]
   pl_out = thickness/ (cos((180 - exp_parameters[3] - angle)/57.29578))
   k_factors[i][0] = ksq(1,mass_surf,angle)
   k_factors[i][1] = ksq(1,mass_sub,angle)
surf_poly[i][0] = k_factors[i][0] * exp_parameters[20] //upper surface values
surf_poly[i][2] = ( k_factors[i][0] * (exp_parameters[20] - (pl_in * (exp_parameters[28] /1000)))) //Lower surf values
surf_poly[i][2] = surf_poly[i][2] - (pl_out * (exp_parameters[28] /1000))
surf_poly[i][1] = (surf_poly[i][0]+surf_poly[i][2])/2 //halfway point
if(K0 == 1) // use energy loss through surface for substrate
else
   surf_poly[i][3] = ( k_factors[i][1] * exp_parameters[20]) + exp_parameters[30]
   //take substr to be at surface as well
endif
if(K1 == 1) // use constant width of substrate
   surf_poly[i][5] = surf_poly[i][3] - exp_parameters[26]
else // take into account energy losses of surface and substrate layer for bottom
   substrate
endif
surf_poly[i][4] = (surf_poly[i][3]+surf_poly[i][5])/2 //halfway point
surf_poly[i][6] = (exp_parameters[31]/k_factors[0][1]) + k_factors[0][1] //bulk
i++1
while(i < 183)
```
duplicate/O sur polishing
do // turn these data points into a 2D plot (a mask to go over final data)
do FindValue/V.(surf_poly[i][j])/T.=((energy_scale[i]-energy_scale[0])/2)
energy_scale
if(K2 == 1)
polygons[i][V.value + ender_bender[i]] = 1
surf_poly_points[i][j] = V.value + ender_bender[i]
else
polygons[i][V.value] = 1
surf_poly_points[i][j] = V.value
endif
j += 1
while(j<7)
j = 0
i += 1
while(i<183)
polygons[i][V.value + ender_bender[i]] = 1
surf_poly_points[0][j] = V.value
killwaves ender_bender k_factors // Tidy up!
End
/*********************************
 // Loads a fill tile that has been exported from the Tiler macro
/*********************************
function load_igor_wave()
// exp_parameters wave info
// [5] Time (doesnt exist in older files - good one!!!
// [20] BEAM energy (manual input)
// [21] MASS OF SURFACE
// [22] MASS OF SUBSTRATE
// [23] INTERPLANAR spacing
// [24] no of LAYERS
// [26] energy loss in surface ev/A
// [27] energy loss in SUBSTRATE ev/A
// [28] Substrate interlayer spacing
// [29] no of LAYERS in Substrate
// [30] Substrate fudge factor
// [31] Bulk counts finish
// [32] Ender Bender Max shift
// [182] number of parameters in original data file (usually 11 or 12) par_nums
LoadWave/O/M/D/A=annoying
exp_parameters = 0
continue_to_load() // INCREDIBLY ANNOYING BUG - have to split into 2 functions.
End
function continue_to_load()
VAVE annoyingO
VAVE exp_parameters
variable par_nums
variable i = 0
do exp_parameters[i] = annoyingO[i][0]
i += 1
while(i < 183)
par_nums = exp_parameters[182]
exp_parameters[par_nums-1] = exp_parameters[par_nums+1]
Redimension/I annoyingO
Make/N=(dimsize(final_tile,1))/D/O energy_scale
energy_scale = exp_parameters[exp_parameters[182]-2] + (exp_parameters[exp_parameters[182]-1] * p)
Make/N=(183)/D/O angle_scale
angle_scale = exp_parameters[exp_parameters[182]-5] + (exp_parameters[exp_parameters[182]-4] * p)
DeletePoints/H=1 0,1, annoying0
duplicate/0 annoying0 final_tile
killwaves annoying0

End

//*******************************************************************************
// Loads a .tile wave into a big long stream
//*******************************************************************************

function load_wave()
  WAVE shaggy0
  WAVE exp_parameters
  string file_name
  string file_location
  GBLoadWave/O/A=shaggy/T={32,32}/S=512
  file_name = _filename
  file_location = _path
  GBLoadWave/O/A=param_wave/T={8,4}/S=0/W=1 (file_location + file_name)
  stream_to_array() //stupid IGOR has troubles loading files that are long
  extract_exp_params()
  Make/N=(dimsize(final_tile,1))/D/O energy_scale
  energy_scale = exp_parameters[exp_parameters[182]-2] + (exp_parameters[exp_parameters[182]-1] * p)
  Make/N=(183)/D/O angle_scale
  angle_scale = exp_parameters[exp_parameters[182]-5] + (exp_parameters[exp_parameters[182]-4] * p)
  killwaves shaggy0 //... and is a big stream
  killwaves param_wave0
dowindow/K colour_plot
  execute("colour_plot()")
end

//*******************************************************************************
// Changes the stream to a proper array
//*******************************************************************************

function stream_to_array()
  WAVE shaggy0
  variable i = 0
  variable j = 0
  variable no_of_energies = 0
  no_of_energies = round((dimsize(shaggy0,0)-304)/183) //calculate the no. of energy channels
  print no_of_energies
  Make/N=(183,(no_of_energies))/1/O final_tile
  final_tile = 0
  do
    final_tile[j][i] = shaggy0[i + (j*no_of_energies)]
    i += 1
    while (i < no_of_energies)
    i = 0
    j += 1
    while (j < 183)
  end

  //******************************************************************************
  // this function extracts all the parameters for the experiment from the last data file
  //******************************************************************************

function extract_exp_params()
  // exp_parameters wave info
// [5] Time (doesnt exist in older files - good one!!!
// [20] BEAM energy (manual input)
// [21] MASS OF SURFACE
// [22] MASS OF SUBSTRATE
// [23] INTERPLANAR spacing
// [24] no of LAYERS
// [25] energy loss in surface ev/A
// [26] constant substrate width
// [27] energy loss in SUBSTRATE ev/A
// [28] Substrate interlayer spacing
// [29] no of LAYERS in Substrate
// [30] Substrate fudge factor
// [31] Bulk counts finish
// [32] Ender Bender Max shift
// [182] number of parameters in original data file (usually 11 or 12) par_nums

WAVE param_wave0
  variable i = dimsize(param_wave0,0)-1000
  variable number, fraction, decimal
  variable j = 0
  variable par_nums = 12

Make/N=183/D/0 exp_parameters //make a wave containing all the parameters
exp_parameters = 0
  do //go to TIA angle in text
    if(param_wave0[i] == 65 && param_wave0[i+1] == 110 && param_wave0[i+2] == 103 ) //
      search for "Ang" in angle
        number = i
        i = (dimsize(param_wave0,0)) // once "Ang" is found quit loop
      endif
    i = i + 1
  while(i < (dimsize(param_wave0,0)))

 i = number
  do // this outer loop goes through each parameter
    number = 0
    fraction = 0
    decimal = 0
    do // this loop goes up to where the number begins
      if ((param_wave0[i]<48 || param_wave0[i] >57))
        i = i + 1
      endif
    while ((param_wave0[i]<48 || param_wave0[i] >57))

    do // this loop converts the stream of single numbers into an actual number
      if (param_wave0[i] == char2num(".")) //handles the decimal point
        decimal = 1
        i = i + 1
      endif
      if (decimal == 0)
        number = number*10 + str2num(num2char(param_wave0[i])) //do the main number
      elseif (decimal > 0)
        fraction = fraction + ((str2num(num2char(param_wave0[i])))/(10^decimal)) //do the fraction
        decimal = 1
      endif
      i = i + 1
    while((param_wave0[i]>47 && param_wave0[i] < 58) || (param_wave0[i] == char2num(".")))

    number = number + fraction
    exp_parameters[j] = number //store number in wave
if (j == 7 && exp_parameters[j] < 1) // check to see if this value is energy increment 
or energy value
    par_nums = 11 // as older files are offset by -1 due to the addition of a
time variable
else if (j == 7 && exp_parameters[j] > 1)
    par_nums = 12
endif
j = j + 1
while (j < par_nums)
    exp_parameters[182] = par_nums
End

// Calculates the masses from the atomic numbers selected in the Global variables shown below

//**********************************************************
function calc_masses()
    WAVE exp_parameters
    variable/G mass_surf
    variable/G mass_sub
    variable i = 0
    string n_mass_surf = num2str(exp_parameters[21]) + "mass" // copy out isotope info
    string n_mass_sub = num2str(exp_parameters[22]) + "mass"
    string n_freq_surf = num2str(exp_parameters[21]) + "freq"
    string n_freq_sub = num2str(exp_parameters[22]) + "freq"
    duplicate/0 root:isotopes:$(n_mass_surf) W_mass_surf
    duplicate/0 root:isotopes:$(n_mass_sub) W_mass_sub
    duplicate/0 root:isotopes:$(n_freq_surf) W_freq_surf
    duplicate/0 root:isotopes:$(n_freq_sub) W_freq_sub
    mass_sub = 0
    mass_surf = 0
    do
        mass_surf += W_mass_surf[i] * W_freq_surf[i]
        i += 1
    while (i < dimsize(W_mass_surf, 0))
    i = 0
    do
        mass_sub += W_mass_sub[i] * W_freq_sub[i]
        i += 1
    while (i < dimsize(W_mass_sub, 0))
    i = 0
End

//**********************************************************
// Button to decide whether you do a quick or slow fit
//**********************************************************
Function ButtonProc(ctrlName) : ButtonControl
    String ctrlName
    variable/G kwik
    if(cmpstr(ctrlName[6], "1") == 0)
        kwik = 1
    else
        kwik = 0
    endif
    total_fit()
End

//**********************************************************
// Shows a particular calculated slice
//**********************************************************
Function SetVarProc(ctrlName, varNum, varStr, varName) : SetVariableControl
    String ctrlName
    Variable varNum
    String varStr
    String varName
WAVE final_fit
WAVE final_fit_BG
WAVE final_fit_sub_gauss
WAVE final_fit_surf_gauss
WAVE final_tile
variable i = 0
Make/N=(dimsz(final_fit_BG,1),5)/D/O
final_cuts holds the energy values of the polygons
do
final_cuts[i][0] = final_fit[K4][i]
final_cuts[i][1] = final_fit_BG[K4][i]
final_cuts[i][2] = final_fit_sub_gauss[K4][i]
final_cuts[i][3] = final_fit_surf_gauss[K4][i]
final_cuts[i][4] = final_tile[K4][i]
i += 1
while(i < dimsiz(final_fit,1))
dovindov/K
final_cuts_graph
execute("final_cuts_graph()")
End

//*************************************************************************/
// ALL BUTTONS OR GRAPHS FROM THIS POINT ONWARDS
//*************************************************************************/
Window final_cuts_graph() : Graph
PauseUpdate; Silent 1 // building window...
Display/W=(159,173.75,537,352.25) final_cuts[*][4],final_cuts[*][1],final_cuts[*][2]
AppendToGraph final_cuts[*][0],final_cuts[*][3]
ModifyGraph mode(final_cuts) = 4
ModifyGraph marker(final_cuts) = 19,marker(final_cuts#4) = 19
ModifyGraph lSize(final_cuts#1) = 1.2,lSize(final_cuts#2) = 1.2,lSize(final_cuts#3) = 1.2
ModifyGraph lSize(final_cuts#4) = 1.2
ModifyGraph rgb(final_cuts) = (47872,47872,47872),rgb(final_cuts#1) = (0,45224,0),rgb(final_cuts #2) = (0,15872,65280)
ModifyGraph rgb(final_cuts#3) = (65280,0,0),rgb(final_cuts#4) = (65280,43520,0)
ModifyGraph meze(final_cuts) = 1,meze(final_cuts#4) = 1
ModifyGraph useMrkStrokeRGB(final_cuts) = 1,useMrkStrokeRGB(final_cuts#4) = 1
ModifyGraph fSize = 7
ModifyGraph axOffset(left) = -2.71429,axOffset(bottom) = -0.833333
EndMacro

Window progress_graph() : Graph
PauseUpdate; Silent 1 // building window...
Display/W=(174,45.5,493.5,194.75) counts_data vs energy_scale
AppendToGraph fit_counts_data
ModifyGraph rgb(fit_counts_data) = (0,0,0)
ModifyGraph tick(left) = 2,tick(bottom) = 1
ModifyGraph fSize = 7
ModifyGraph axOffset(left) = -5.2,axOffset(bottom) = -1.75
Label bottom "\\038"
ModifyGraph rgb = (0,15872,65280)
netaxis right 0, (chi_sq_wave[0]/4)
ModifyGraph fSize = 6
ModifyGraph axOffset(right) = -4.75,axOffset(bottom) = -1.4
ModifyGraph lbLatPos(right) = 71
RenameWindow #,0
SetActiveSubwindow ##
EndMacro

Window integral_graph() : Graph
PauseUpdate; Silent 1 // building window...
Display/W=(5.25,42.5,377.25,243.5) substr_integral
AppendToGraph/R surf_integral
ModifyGraph mode(surf_integral) = 4
ModifyGraph fSize=7
ModifyGraph lbMargin(bottom)=3
ModifyGraph offset(left)=-2, axOffset(bottom)=1.25, axOffset(right)=-1
ModifyGraph lbLatPos(bottom)=21
Label left "Substrate Counts (line)"
Label bottom "Angle Channel"
Label right "Surface Counts (line + marker)"

EndMacro

Window show_chisq() : Graph
  
  PauseUpdate; Silent 1  // building window...
  Display /w=(378,64,25,772,5,272,75) chi_sq_wave
  ModifyGraph lbMargin(bottom)=5
  ModifyGraph axOffset(left)=-4, axOffset(bottom)=-1.125
  ModifyGraph lbLatPos(bottom)=-30
  Label left "Chi Squared Fit Value"
  Label bottom "Angle Channel"

EndMacro

Window show_final_fit() : Graph
  
  PauseUpdate; Silent 1  // building window...
  Display /w=(375,75,55,25,594,570,5)
  AppendImage final_fit
  ModifyImage final_fit ctab=1,*;Spectrum,0
  ModifyImage final_fit minRGB=(65535,65535,65535)
  ModifyGraph mirror=2
  ModifyGraph axOffset(left)=-4.42857, axOffset(bottom)=-0.9375

EndMacro

Window colour_plot() : Graph
  
  PauseUpdate; Silent 1  // building window...
  Display /w=(628.5,53.75,870.75,570.5)
  AppendMatrixContour polygons
  ModifyContour polygons rgbLines=(0,0,0), labels=0
  AppendImage final_tile
  ModifyImage final_tile ctab=1,*;Spectrum,0
  ModifyImage final_tile minRGB=(65535,65535,65535)
  ModifyGraph mirror=2
  ModifyGraph axOffset(left)=-3.85714

EndMacro

Function ButtonProc_1(ctrlName) : ButtonControl
  
  String ctrlName
  load_wave()

End

 /////////////////////////////////////////////////////////////////////////////
 // The main control panel
 /////////////////////////////////////////////////////////////////////////////

Window exp_info() : Panel
  
  PauseUpdate; Silent 1  // building window...
  NewPanel /w=(5,57,214,759)
  SetVariable tea_set, pos={11,31}, size={150,16}, title="TEA angle"
  SetVariable tea_set, value= exp_parameters[0]
  SetVariable Setvar0, pos={11,51}, size={150,16}, title="Beam angle in"
  SetVariable Setvar0, value= exp parameters[3]
  ValDisplay valdisp0, pos=(8,130), size={150,15}, title="Start Angle"
  ValDisplay valdisp0, limits=(0,0,0), barmisc(0,1000)
  ValDisplay valdisp0, value= exp Parameters[exp_parameters[182]-5]
  ValDisplay valdisp1, pos=(9,151), size={150,15}, title="Angle inc."
  ValDisplay valdisp1, limits=(0,0,0), barmisc(0,1000)
  ValDisplay valdisp1, value= exp Parameters[exp_parameters[182]-4]
  ValDisplay valdisp2, pos=(10,171), size={150,15}, title="Calib Energy"
  ValDisplay valdisp2, limits=(0,0,0), barmisc(0,1000)
  ValDisplay valdisp2, value= exp Parameters[exp_parameters[182]-2]
String ctrlName
Variable varNum
String varStr
String varName
WAVE final_tile
WAVE polygons
variable slice_at = X3
Make/N=(dsize(final_tile,i))/D/O energy_cut
Make/N=(dsize(final_tile,i))/D/O parm_points
energy_cut[*] = final_tile[slice_at][p]
parm_points[*] = polygons[slice_at][p]
dowindow/k slice_graph
execute("slice_graph()")
end

// ********** New Additions (V1.1) onwards
//**************************************************
// This function controls the fixing of the tiles - it takes away the jaggies
//**************************************************************************
function fix_final_fit(ctrlName)
String ctrlName
prop_fix_final_fit()
WAVE fixed_fit
WAVE final_fit
WAVE fixed_fit_trace
WAVE angle_scale
WAVE exp_parameters
variable i = 0
variable j = 0
variable k = 0
variable m = 0
Make/N=(dsize(surf_poly_points,0))/D/O/O shift_amount, shift_amount_scaled, fixed_slice
Make/N=30/D/O stats_wave
Make/N=(183.30)/O/O fixed_slice_img
Make/N=(dsize(fixed_fit,1))/D/O old_counts, rescaled_counts, new_channel_scale
do // extract counts from the polygon top line
  shift_amount[i] = final_fit[i][fixed_fit_trace[i]] / sin(angle_scale[i] * 3.14159265 / 180)"^4
  i = i + 1
while(i < dsize(surf_poly_points,0))
i = 0
wavestats/Q shift_amount
shift_amount = V_max-shift_amount[p] //invert..
wavestats/Q shift_amount
shift_amount = shift_amount[p] / V_max // normalize
do
  shift_amount_scaled = shift_amount[p] * m // Alter the shifting
wavestats/Q shift_wave()
do // extract counts from the polygon top line for fit
  fixed_slice[i] = fixed_fit[i][fixed_fit_trace[i]] / sin(angle_scale[i] * 3.14159265 / 180)"^4
  fixed_slice_img[i][m] = fixed_slice[i]
wavestats/Q fixed_slice_img
wavestats/Q fixed_slice
stats_wave[m] = V_sdev
i = i + 1
while(i < dsize(surf_poly_points,0))
i = 0
m = m + 1
while(m < 30)
wavestats/Q stats_wave
\[ n = V_{\text{minloc}} \quad // \text{find the minimum standard deviation - assumed to be the flattest cut} \]
\[ \text{shift\_amount\_scaled} = \text{shift\_amount}[p] \times m \]
\[ \text{shift\_wave()} \quad // \text{recalculate the tile with the smoothest correction} \]
\[ \exp\_\text{parameters}[32] = m \]

End

//********************************************
// This function shifts the \text{final\_fit} tile to the amount given by \text{"shift\_amount\_scaled"} \n//********************************************
function \text{shift\_wave()}

variable \text{j} = 0
variable \text{i} = 0
variable \text{k} = 0
WAVE \text{old\_counts}
WAVE \text{new\_channel\_scale}
WAVE \text{rescaled\_counts}
WAVE \text{shift\_amount\_scaled}
WAVE \text{final\_fit}
WAVE \text{fixed\_fit}

variable \text{max\_e\_channel} = \text{dimsize}(_{\text{final\_fit}},1)
variable \text{max\_A\_channel} = \text{dimsize}(_{\text{final\_fit}},0)
do // shift energy cuts up by value shown in \text{"shift\_amount"} and interpolate correct values
\text{old\_counts} = \text{final\_fit}[\text{j}][\text{p}]
\text{new\_channel\_scale} = \text{p} + \text{shift\_amount\_scaled}[\text{j}]
do
\text{rescaled\_counts}[\text{i}] = \text{interp}(\text{i}, \text{new\_channel\_scale}, \text{old\_counts})
i += 1
while(\text{i} < \text{max\_e\_channel})
do // put rescaled wave into final image
\text{fixed\_fit}[\text{j}][\text{k}] = \text{rescaled\_counts}[\text{k}]
k += 1
while(\text{k} < \text{max\_o\_channel})
\text{k} = 0
\text{i} = 0
\text{j} += 1
while(\text{j} < \text{max\_A\_channel})
do
\end

End

//********************************************
// This function does a few of the preliminary wave creations before the shifting occurs
//********************************************
function \text{prep\_fix\_final\_fit()}

surf\_polygon("")
WAVE \text{final\_fit}
WAVE ender\_bender
WAVE surf\_poly\_points
variable \text{i} = 0
Make/N=(\text{dimsize}(\text{surf\_poly\_points},0))/D/O \text{fixed\_fit\_trace} \n\text{fixed\_fit\_trace} = \text{surf\_poly\_points}[\text{p}][0]
duplicate/O \text{final\_fit} \text{fix\_fit\_poly} \text{fixed\_fit} \n\text{fix\_fit\_poly} = 0 \n\text{fixed\_fit} = 0
\do // make a contour plot to display on graph
\text{fix\_fit\_poly}[\text{i}][\text{fixed\_fit\_trace}[\text{i}]] = 1
\text{i} += 1
\while(\text{i} < \text{dimsize}(\text{surf\_poly\_points},0))
execute("show\_fixed\_fit()")
\end

End

//********************************************
// Displays the final fixed fit

End
//*******************************************************************************
Window show_fixed_fit() : Graph
dowindow= k show_fixed_fit
PauseUpdate; Silent i // building window...
Display /= (275.75, 55.26, 494, 570.5)
AppendImage fixed_fit
ModifyImage fixed_fit ctab = {1, *, Spectrum, 0}
ModifyImage fixed_fit minRGB = (65535, 65535, 65535)
ModifyGraph mirror = 2
ModifyGraph axOffset(left) = -4.42857, axOffset(bottom) = -0.9315
AppendMatrixContour fix_fit_poly; DelayUpdate
ModifyContour fix_fit_poly labels = 0, rgbLines = (0, 0, 0)
EndMacro

// calculates and individual shift given in exp_parameters[33] (fix_shift on control panel)

Function individual_fix_show(ctrlName) : ButtonControl
String ctrlName
variable i = 0
WAVE final_fit
WAVE fixed_fit trace
WAVE angle_scale
WAVE exp_parameters
Make/N=(dimsize(surf_poly_points, 0))/0/0 shift_amount, shift_amount_scaled, fixed_slice
Make/N=(dimsize(fixed_fit, 1))/0/0 old_counts, rescaled_counts, new_channel_scale
prep_fix_final_fit()
shift_amount = 0
do // extract counts from the polygon top line
shift_amount[i] = final_fit[i][fixed_fit_trace[i]] / min(angle_scale[i] + 3.14159265 / 180) * 4
i += 1
while i < dimsize(surf_poly_points, 0)
wave stats/Q shift_amount // Creates a normalised wave that shows how much the final image
should be shifted
shift_amount = V_max-shift_amount[p] // invert...
wave stats/Q shift_amount
shift_amount = shift_amount[p] / V_max // normalize
shift_amount_scaled = shift_amount * exp_parameters[33]
shift_wave()
execute("show_fixed_fit()")

End
Function load_calib_file(ctrlName) : ButtonControl
String ctrlName
LoadWave/T/O // Wave loaded should be called calibcurve
End
Function correct_data(ctrlName) : ButtonControl
String ctrlName
variable i = 0
variable j = 0
WAVE angle_scale
WAVE substr_integral
WAVE surf_integral
WAVE bg_integral
WAVE offset_interp
Make/N=183/D/O temp_angular_scale, final_corrected_angle, final_corr_bg_counts
Make/N=183/D/O final_corr_surf_counts, final_corr_substr_counts
temp_angular_scale = 0
final_corrected_angle = 0
temp_angular_scale = offset_interp[p] + angle_scale[p]
do // Add the offsets to the angular scale

//*******************************************************************************
```c
final_corrected_angle[i] = temp_angular_scale[0] + (i*(((temp_angular_scale[182] -
    temp_angular_scale[0]) / 182))
i = i + 1 // Create new FINAL angular scale with equal increments
while (i < dmsize(bg_integral, 0))
i = 0
    do
        final_corr_bg_counts[i] = interp(final_corrected_angle[i], temp_angular_scale,
            bg_integral)
        final_corr_bg_counts[i] = final_corr_bg_counts[i] * ((sin((angle_scale[i]*3.1415927)
            /180/2)*4))
        final_corr_surf_counts[i] = interp(final_corrected_angle[i], temp_angular_scale,
            surf_integral)
        final_corr_surf_counts[i] = final_corr_surf_counts[i] * ((sin((angle_scale[i]
            +3.1415927)/180/2)*4))
        final_corr_substr_counts[i] = interp(final_corrected_angle[i], temp_angular_scale,
            substr_integral)
        final_corr_substr_counts[i] = final_corr_substr_counts[i] * ((sin((angle_scale[i]
            +3.1415927)/180/2)*4))
        i += 1
    while(i<=dmsize(bg_integral, 0))

Save/T/M="\r\n" final_corr_bg_counts,final_corrected_angle,final_corr_substr_counts,
    final_corr_surf_counts

End
```
#pragma rtCglobals=1 // Use modern global access method.

//*******************************************
// Tiler Macro v0.852
// Written By Chris Howe at Loughborough University
// chrishowe2 @ hotmail. com
//*******************************************

// version 0.852 A lot of bugs fixed
// version 0.75 K-square corrections
// version 0.72 Does cuts
// version 0.70 Exports tiles in custom format
// version 0.68 deals with differing energy scales on each tile!!!
// version 0.67 loads up experimental parameters for all tiles and creates .info files
// version 0.65 loads binary files completely
// version 0.64 loads up experimental parameters from the binary file
// version 0.63 Tile loader couldn't cope with Os - fixed
// version 0.5 updated colour scales for IQR 5

//Global variable reference
// k2 = beam mass
// k3 = TEA angle
// k4 = target mass
// k6 = angle per pixel
// k7 = energy per pixel
// k8 = calibration energy

// Info wave data...
// [5] Time (doesn't exist in older files - good one!!!
// (par_nums-6) N/A, (par_nums-5) Start Angle, (par_nums-4) Angle increment
// (par_nums-3) N/A, (par_nums-2) Start energy, (par_nums-1) Energy increment
// (par_nums) End Energy
// (par_nums+1) new energy increment
// [182] number of parameters in original data file (usually 11 or 12) par_nums

//variable/G calilb_energy = 10
//variable/G energy_inc = 11
//variable/G end_energy = 12
//variable/G load_binary = 0
variable/G par_nums = 12
//Make/N=1/T/D list_of_tiles

function refresh_panel()
donwindow /K main_panel
execute "main_panel()"
end

//***************************************************************
// This loads the first tile
//***************************************************************
Function select_first_data(ctrlName) : ButtonControl
String ctrlName
WAVE/T list_of_tiles
NVAR load_binary = root:load_binary
string/G first_file_name
string/G file_location
string temp_string
variable i = 0
do
temp_string = list_of_tiles[i] + ".info"
kilwaves $list_of_tiles[i] $temp_string
i = i + 1
while (i < DimSize(list_of_tiles, 0))
GBLoadWave/O/A = binary_load_wave/T={32,32}/S=512/W=1
binary_redimension()
first_file_name = S_filename
file_location = S_path
GBLoadWave/O/A = param_wave/T={8,4}/S=68611/W=1 (file_location + first_file_name) //load end of previous file for parameters
extract_exp_params() //load all the experimental parameters
Duplicate/O exp_parameters $first_file_name + ".info" //create an information wave for that tile
if (waveExists($first_file_name))
   KillWaves $first_file_name
endif
Rename wave 0, $first_file_name
Redimension/D $first_file_name
refresh_panel()

End

//***************************************************************************************
// This loads the last tile
//******************************************************************************************
Function select_last_data(ctrlName) : ButtonControl
String ctrlName
string/G last_file_name
NVAR load_binary
string file_location
GBLoadWave/O/A = binary_load_wave/T={32,32}/S=512/W=1
binary_redimension()
last_file_name = S_filename
file_location = S_path
GBLoadWave/O/A = param_wave/T={8,4}/S=68611/W=1 (file_location + last_file_name) //load end of previous file for parameters
extract_exp_params() //load all the experimental parameters
Duplicate/O exp_parameters $last_file_name + ".info" //create an information wave for that tile
if (waveExists($last_file_name))
   KillWaves $last_file_name
endif
Rename wave 0, $last_file_name
Redimension/D $last_file_name
refresh_panel()

End

#SBATCH---------------------------------------------------------
//this loads the main panel
#SBATCH---------------------------------------------------------
Window main_panel() : Panel
PauseUpdate; Silent 1 // building window...
NewPanel /W=(14,67,194,506)
SetDrawLayer UserBack
DrawRect 162,66,4,40
DrawText 27,59,first_file_name
DrawRect 5,94,163,124
DrawText 27,117,last_file_name
Button first_data, pos={7,14}, size={100,20}, proc=select_first_data, title="First data block"
Button last_data, pos={9,69}, size={100,20}, proc=select_last_data, title="Last data Block"
Button load_fillers, pos={25,135}, size={120,20}, proc=load_fillers, title="Load Sequential Files"
Button tile_files, pos={26,165}, size={120,20}, proc=tile_files_button, title="Tile files"
SetVariable start_pixel_select, pos={7,189}, size={150,16}, proc=start_pixel_sel, title="Tile start pixel"
SetVariable start_pixel, limits={0,93}, value= X0
SetVariable setvar0, pos={7,211}, size={150,16}, title="Tile end pixel (93 max)"
SetVariable setvar0, limits={0,93}, value= X1
Button button0,pos={21,241},size={130,20},proc=loacCqs_panel,title="K-Squared Correct ------->"

PopupMenu popup0,pos={8,271},size={55,21},proc=set_active_wave
PopupMenu popup0,model=1,popvalue="select active wave",value=list_of_new_tiles_string
Button button1,pos={26,302},size={120,20},proc=kill_tile,title="KHL Active Wave"
Button button2,pos={27,326},size={120,20},proc=load_saved_wave,title="Load Saved Wave"
Button button3,pos={25,358},size={120,20},proc=save_active_wave,title="Save Active Wave"
Button button4,pos={28,390},size={120,20},proc=display_active_wave,title="Display Active Wave"

EndMacro
/****************************/
// this loads all the in between files
/****************************/

Function load_fillers(ctrlName) : ButtonControl
String ctrlName
string tempstring
variable counting_filenum
variable tempnum
variable first_file_num
variable last_file_num
variable counter
variable i = 0
variable j = 0
SVAR file_location
SVAR first_file_name
SVAR last_file_name
SVAR par_num
WAVE exp_parameters
do //loop to extract first filenumber
tempnum = str2num(first_file_name[i]) //character extraction from name
if(tempnum || tempnum == 0) //if it is a number then multiply the previous number by
ten and add new one on
first_file_num = (first_file_num * 10)+ tempnum
endif
i = i + 1
while(cmpstr (first_file_name[i], ".") !=0) //keep on going until a . is reached
do //loop to extract last fileoumber
tempnum = str2num(last_file_name[j])
if(tempnum || tempnum == 0)
last_file_num = (last_file_num * 10)+ tempnum
endif
j = j + 1
while(cmpstr (last_file_name[j], ".") !=0)
Make/N=(last_file_num-first_file_num+1)/T/O list_of_tiles //store filenames in this list
list_of_tiles = ""
counting_filenum = first_file_num
i = 0
list_of_tiles[0] = first_file_name

Do counting_filenum = counting_filenum +1 //increment the filename number.
tempstring = num2str(counting_filenum)
tempstring[0] = "run" //add necessary text on the start and end of the file
tempstring[8] = ".2d.data"
GBLoadWave/O/A=binary_load_wave/T={32,32}/S=512/W=1 (file_location + tempstring)
binary_redimension()
GBLoadWave/O/A=param_wave/T={8,4}/S=68611/W=1 (S_path + S_filename) //load end of
previous file for parameters
extract_exp_parameters() //load all the experimental parameters
Duplicate/O exp_parameters $S_filename + ".info" //create an information wave for that

tile
if(waveExists($tempstring)) //kill the wave if it already exists
KillWaves $tempstring
endif
Rename waveO, $tempstring
Redimension/D $tempstring
i = i + i

list_of_tiles[i] = tempstring  //make a note in the list of waves
while(i<(last_filenum - first_filenum))
tempstring = list_of_tiles[0] + ".info"
duplicate/D $tempstring temp_info_wave
  // k2 = beam mass
  k3 = temp_info_wave[0]  // TEA
  // k4 = target mass
  k5 = temp_info_wave[par_nums - 4]  // angle per pixel
  k7 = temp_info_wave[par_nums - 1]  // energy per pixel
  tempstring = list_of_tiles[dimsize(list_of_tiles,0)-1] + ".info"  // load up last wave (bottom ) to extract calib energy
duplicate/D $tempstring temp_info_wave
  k8 = temp_info_wave[par_nums - 2]  // calib energy
  killwaves param_waveO  // tidy up
  killwaves binary_load_waveO temp_info_wave
End

function tile_files_button(ctrlName) : ButtonControl
  String ctrlName
  execute ("tile_files()")
End

Exports ctrlName :
  String ctrlName
  execute ("tile_files()")
End

//************************************************
// this tiles all the files together
//************************************************

function tile_files() :
  variable i=0
  variable j=0
  variable m=0
  SVAR first_file_name
  SVAR list_of_new_tiles_string
  NVAR par_nums
  WAVE/T list_of_tiles
  WAVE/T list_of_new_tiles
  string new_file_name = list_of_tiles[0] + ".tile"
  string temp_file_name
  variable number_of_rows = (k1-kO)*(dimsize(list_of_tiles,0)-1)
  Make/D/O/W(183,number_of_rows) tt // make full wave
  tt[0]=0
do
    temp_file_name = list_of_tiles[dimsize(list_of_tiles,0)-1-i]  // start at last wave
    duplicate/D/O $temp_file_name temp_wave // copy to temp wave due to pointer issues
    do
      do
        do
          tt[m][j+(i*(k1-kO))]=temp_wave[m][j]  // copy single datapoint from temp wave to full wave point
          m = m+1
          while(m<183) // row
            m=0
            j = j+1
            while(j<k1) // column
              j=kO // reset back to start pixel
              i = i + 1
              while(i<(dimsize(list_of_tiles,0))) // tile
                new_energy_scale_b4_tile()  // big function
                interp_new_full_tile()  // big function
                duplicate/D/O tt $new_file_name // copy the temp full wave to a proper named wave + .tile
                redimension/D O $new_file_name
              endwhile
              i = i + 1
            endwhile
          endwhile
        endwhile
      endwhile
    endwhile
  endwhile

tile_files_button("this")
dowindow/k Panel0 //display it
NewPanel(/=(screen_dim("width")-0.78,10,screen_dim("width")-10,screen_dim("height")-115) as "Graph0"
Display/HOST=/V(0,0,1,1)
AppendImage/V=# $new_file_name
ModifyImage/V= $new_file_name minRGB=(65535,65535,65535),maxRGB=0
ModifyImage/V= $new_file_name cmap= (1,*,Spectrum,0)
ModifyGraph/W= mirror=2
ModifyGraph/W= axOffset(left)=-4.42857
Make/N 183/D/O

angle temp_file_name = list_of_tiles[dimsize(list_of_tiles,0)-1]+".info" //extract calib energy from last tile
duplicate/0 $temp_file_name temp_info_wave
i = temp_info_wave[par_nums -2]

if((check_wave_in_table(list_of_tiles[0]+".tile")))
InsertPoints dimsize(list_of_tiles,0),1,list_of_new_tiles
list_of_new_tiles[dimsize(list_of_new_tiles,0)] = list_of_tiles[0]+".tile"
list_of_new_tiles_string = list_of_new_tiles_string + list_of_tiles[0]+".tile;";
endif
killwaves temp_info_wave
End
End

Function start_pixel_set(ctrlName,varNum,varStr,varName) : SetVariableControl
String ctrlName
Variable varNum
String varStr
String varName
End

Function set_max_pixel(ctrlName,varNum,varStr,varName) : SetVariableControl
String ctrlName
Variable varNum
String varStr
String varName
End

Function load_ksq_panel(ctrlName) : ButtonControl
String ctrlName
dowindow/k k_sq_panel
execute "k_sq_panel()"

End

/***************************************************
//this loads the k-squared control panel
/***************************************************
Window k_sq_panel() : Panel
  PauseUpdate; Silent 1 // building window...
  NewPanel /W=(49,469,383,585)
  SetVariable Set_beam_mass,pos=(8,7),size=(130,16),title="Beam mass"
  SetVariable Set_scattering_angle,pos=(5,50),size=(130,16),title="Scattering Angle"
  SetVariable Set_tgt_mass,pos=(8,64),size=(130,16),title="Target mass"
  SetVariable Set_energy_pixel,pos=(140,30),size=(180,16),title="Energy per pixel"
  SetVariable Set_angle_increment,pos=(171,7),size=(150,16),title="Angle per pixel"
  SetVariable Set_calib_energy,pos=(153,76),size=(167,16),title="Calib Energy (keV)"
  Button k_sq,pos=(8,76),size=(80,20),proc=k_sq_button,title="K correct"
  Button revert_button,pos=(58,76),size=(100,20),proc=revert_to_original,title="Revert to original"
EndMacro

/***************************************************
//this massive function k-squares a full 3d tile, yes, it is a pain in the *neck*
/***************************************************
Function ksq_button(ctrlName) : ButtonControl
  String ctrlName
  String tempstring
  string tempstring2
  variable i = 0
  variable j = 0
  variable beam_mass = K2
  variable tgt_mass = K4
  variable k_sq_max_y_scale = dimsize(tt,1)
  variable max_y_value
  variable min_new_energy
  variable max_new_energy
  variable energy_per_pixel = K7
  variable calib_energy = K8
  variable new_energy_array_point
  //this stores the array point for the k-squared energy value
  variable new_energy_max_point
  WAVE tt
  WAVE second_energies
  WAVE angle
  WAVE/1 list_of_new_tiles
  SVAR active_wave
  SVAR list_of_new_tiles_string
  Make/A=183/D/O k_factors
duplicate/O $active_wave temp_tile
tempstring = active_wave + ".info"
duplicate/O $temp_tile temp_info_wave
angle[p] = p*k6 + temp_info_wave[ temp_info_wave[182] - 5 ] // k6 = angle per pixel
  do // make an array of K2 factors for each angle
    k_factors[i] = ((tgt_mass^2-(beam_mass*sin((angle[i]/57.4))2)/beam_mass*cos((angle[i]/57.4))/(beam_mass*tgt_mass))2
  i = i + 1
  while (i<193)
Make/N=(dimsize(tt,1))/D/O new_energies
Make/N=(dimsize(tt,1))/D/O old_energy_counts
Wavestat/Q k_factors
min_new_energy = calib_energy/V~ax
max_new_energy = second_energies[1]/D/O old_energy_counts
max_y_value = (max_new_energy - min_new_energy)/energy_per_pixel //find num of points in new k2 wave
max_y_value = (((dimsize(tt,1))*energy_per_pixel)+(calib_energy))/V~in - calib_energy)/energy_per_pixel //find num of points in new k2 wave
Make/N=(max_y_value+1)/D/O new_energy_scale // make a wave for the new energy scale
k_corr_tile = new k-sq tile
k_corr_tile[j][i] = 0
new_energy_max_point = max_y_value

//this next loop creates a new energy scale with increased size to fit all k2 values in
do
new_energy_scale[i] = min_new_energy + (i*energy_per_pixel)
i = i + 1
while(i<new_energy_max_point)
i = 0
//this loop copies a whole single column from the uncorrected tile so that the interp function
//and creates its corresponding new k2 corr'd energy at that particular angle
:i=0
do
new_energies[i] = (calib_energy + (i*energy_per_pixel))/k_factors[j] //generate new k-sq-energy scale
old_energy_counts[i] = temp_tile[j][i] //copy single column energy projection to new temp wave
i = i + 1
while(i<k_sq_max_y_scale)
i=0
while(j<183)
tempstring = active_wave + ".k2." + num2str(K4) //create new wave (k^2 wave)
i = 0
// remove the .2d.data. from the file name as it is too long otherwise
if (tempstr(tempstring[1, i+8], ".2d.data.")
tempstring[i, i+8] = "."
Endif
i += 1
while(i<strlen(tempstring))
duplicate/O k_corr_tile tempstring
if(!(check_wav9_in_table(tempstring)) ) // make a note of it
    InsertPoints dimsize(list_of_new_tiles,0),1, list_of_new_tiles
    list_of_new_tiles[dimsize(list_of_new_tiles,0)] = tempstring
    list_of_new_tiles_string = list_of_new_tiles_string + tempstring + "];
endif

i = 0

tempstring2 = active_wave + ".info" // this section creates a new .info wave for the k2 wave
duplicate/O $tempstring2 temp_info_wave

temp_info_wave[ temp_info_wave[182] -2 ] = new_energy_scale[0] // calib energy
temp_info_wave[ temp_info_wave[182] ] = new_energy_scale[new_energy_max_point] // max energy
tempstring = tempstring + ".info"
duplicate/O temp_info_wave $tempstring
execute("display_kcorr("")
SetScale/Y new_energy_scale[0], new_energy_scale[new_energy_max_point], "k_corr_tile"
killwaves old_energy_counts temp_tile temp_info_wave // tidy up

End

//===FUNCTIONS===
// this function displays the k-corrected data
//===FUNCTIONS===

Window display_kcorr(): Graph
dowindow k display_kcorr
PressUpdate; Silent 1 // building window...
NewPanel /K=W(screen_dim("width")*0.65,10,screen_dim("width")*0.64+330,screen_dim("height") -130)
Display /HOST=display_kcorr /W=(0, 0, 1, 0.9) As "k_corr_graph"
AppendImage/W=x_k_corr_tile
ModifyImage/W=x_k_corr_tile minRGB=(65535,65535,65535), maxRGB=0
ModifyImage/W=x_k_corr_tile ctab={1,1,Spectrum,0}
ModifyGraph/W=mirror=2
ModifyGraph/W=stick(1)=20
ModifyGraph/W=sfsize=0
ModifyGraph/W=olfsize=0
ModifyGraph/W=axOffset(100)=2.9, axOffset(bottom)=3.0625
Label/W=left "kz80 Energy (keV)"
Button kcorr_project_energy, pos=(25,screen_dim("height")-200), size=(80,20), proc=proj_energy_button, title="Project energy"
    SetVariable kcorr_project_energy_low,pos=(24,screen_dim("height")-170), size=(68,16), proc=SetVarProc, title="From"
    SetVariable kcorr_project_energy_low,limits={0,183,1}, value= K10
    SetVariable kcorr_project_energy_hi,pos=(96,screen_dim("height")-170), size=(56,16), proc=SetVarProc, title="to"
    SetVariable kcorr_project_energy_hi,limits={0,183,1}, value= K11
Button kcorr_project_angle, pos=(227,screen_dim("height")-200), size=(80,20), proc=project_angle, title="Project Angle"
    SetVariable kcorr_project_angle_low,pos=(175,screen_dim("height")-170), size=(66,16), proc=SetVarProc, title="from"
    SetVariable kcorr_project_angle_low,limits={0,1400,1}, value= K12
    SetVariable kcorr_project_angle_hi,pos=(251,screen_dim("height")-170), size=(55,16), proc=SetVarProc, title="to"
    SetVariable kcorr_project_angle_hi,limits={0,1400,1}, value= K13
CheckBox check0, pos=(110,screen_dim("height")-165), size=(43,14), proc=energy_display_contour, title="show"
CheckBox check0, value= 0
CheckBox check1, pos=(180,screen_dim("height")-165), size=(45,14), proc=angle_display_contour, title="Show"
CheckBox check1, value= 0
EndMacro
//===FUNCTIONS===

// this is a button controller

Function proj_energy_button(ctrlName) : ButtonControl
    String ctrlName
    project_energy()
End

// This function puts the energy projection boundaries on/off the k-corrected graph

Function energy_display_contour(ctrlName, checked) : CheckBoxControl
    String ctrlName
    Variable checked
    if (checked)
        // set active subwindow
        GO
        AppendMatrixContour /=display_kcorr#GO energy_boundary_markers; DelayUpdate
        ModifyContour /=display_kcorr#GO energy_boundary_markers rgb-lines=(0,0,0), autoLevels = (1, *, 1), labels = 0
    endif
    if (!checked)
        RemoveContour /=display_kcorr#GO energy_boundary_markers
    endif
End

// This function creates/updates an array for the energy and angle projection contour maps

Function SetVarProc(ctrlName.varNum, varStr, varName) : SetVariableControl
    String ctrlName
    Variable varNum
    String varStr
    String varName
    WAVE k_corr_tile
    variable points_of_kcorr = 0
    variable points_of_boundary = 0
    if (!cmpstr(ctrlName[11], "e"))
        // work out whether size of kcorr has changed (for speed of changing boundaries)
        if (!waveexists(energy_boundary_markers))
            duplicate/O k_corr_tile energy_boundary_markers
        endif
        Wavestats/Q k_corr_tile
        points_of_kcorr = V_npnts
        Wavestats/Q energy_boundary_markers
        points_of_boundary = V_npnts
        if (points_of_kcorr = points_of_boundary)
            duplicate/O k_corr_tile energy_boundary_markers
        endif
        energy_boundary_markers = 0 // redraw lines
        energy_boundary_markers[X10][] = 1
        energy_boundary_markers[X11][] = 1
    endif
    // same as above, but for angular boundaries
    if (!cmpstr(ctrlName[11], "a"))
        if (!waveexists(angle_boundary_markers))
            duplicate/O k_corr_tile angle_boundary_markers
        endif
        Wavestats/Q k_corr_tile
points_of_kcorr = V_npts
Wavestats/Q angle_boundary_markers
points_of_boundary = V_npts
points_of_kcorr = points_of_kcorr - points_of_boundary
endif
angle_boundary.markers
points.ct.boundary = points.of.kcorr
points.of.kcorr = points.of.kcorr - points.ef_boundary
if«!waveexists(angle_boundary_markers»)
I1 (points.of.kcorr != 0)
endif
angleboundary.markers ~ 0 /1
redraw
line
angle_boundary~arkers[K12]
m 1
angle.boundary.markers[K13] = 1
#endif
End
//******************************************
//This function puts the angular projection boundaries on/off the k-
corrected graph
//******************************************
Function angle_display_contour(ctrlName,checked) : CheckBoxControl
String ctrlName
Variable checked
if (checked)
ModifyContour /W=display_kcorr#GO angle_boundary_marker$;DelayUpdate
labels = 0
endif
if(!checked)
RemoveContour /W=display_kcorr#GO angle_boundary_markers
endif
End
//******************************************
//This projects the energy between the points selected on the kcorr graph
function project_energy()
WAVE k_corr_tile
Make/OJN=(dimsize(k_corr_tile,l»/r energy_projection
energy_projection = 0
variable i = k0, j = 0
do
j=0
do
energy_projection[j] = energy_projection[j] + k_corr_tile[i][j]
j = j + 1
while(j<dimsize(energy_projection,0))
i = i + 1
while(i<kll)
execute("display_energy_cut")
End
//******************************************
//This function projects the angular information from the limits set on the graph
function project_angle(ctrlName): ButtonControl
String ctrlName
string tempstring
SVAR active_wave
WAVE k_corr_tile
WAVE angle

tempstring = active_wave + ".info"
duplicate/O $tempstring temp_info_wave
angle[p] = p*x5 + temp_info_wave[temp_info_wave[182] - S]
Make/O/N=(dimsize(k_corr_tile,0))/1 angle_projection
angle_projection = 0
variable i = k12, j = 0
do
do
    angle_projection[j] = angle_projection[j] + k_corr_tile[j][i]
    j = j + 1
while(j < dimsize(angle_projection,0))
    i = i + 1
while(i < k13)
SetScale/I x angle[0], angle[dimsize(angle,0)], "", angle_projection
execute("display_angular_cut()")
End

Function revert_to_original(ctrlName) : ButtonControl
String ctrlName
WAVE new_energy_scale
WAVE k_corr_tile
WAVE second_energies
SVAR active_wave
duplicate/O $active_wave k_corr_tile
SetScale/I y second_energies[0], second_energies[dimsize(second_energies,0)-1], ",", k_corr_tile
execute("display_kcorr()")
End

Function save_angular_cut(ctrlName) : ButtonControl
String ctrlName
SVAR first_file_name
string file_save_name = first_file_name + ",angle-cut. +"+(num2str(k12)) +"-"+(num2str(k13))
Save/G/M="
" angle, angle_projection as (file_save_name)
End

Function save_energy_cut(ctrlName) : ButtonControl
String ctrlName
SVAR first_file_name

EndMacro
string file_sav9_name = first_file_name + ".energy-cut." +(num2str(kl0)) +"-(num2str(kll))
Save/G/M"
"new_energies, energy_projection as (file_sav9_Dame)
End
//****************************************************
// this function displays the energy cut in a specific format
//****************************************************
Window display_energy_cut() : Graph
dowindow /k display_energy_cut
PauseUpdate; Silent 1 // building window...
Display /=148.5,84.5,643,293) energy_projection
ModifyGraph grid=1
ModifyGraph nsticks(bottom)=10
ModifyGraph fsize=8
ModifyGraph bMargin(bottom)=6
ModifyGraph axOffset(left)=1, axOffset(bottom)=0.615385
ModifyGraph gridStyle=1
ModifyGraph b1LatPos(bottom)=-8
Label left "Counts"
Label bottom "Energy channel"
Button save_energy_cut_button, pos={45,254}, size={100,20}, proc=save_energy_cut, title="Save cut"
EndMacro
//****************************************************
// when the binary file is loaded it needs to be turned from a stream into a table
//****************************************************
function binary_redimension()
WAVE binary_load_waveO
variable i =0
variable j =0
Make/O=N'(183,93)/1 wavey0
wavey0 = 0
do
do
wavey0[i][j] = binary_load_wave0[j+(93*i)]
i = i + 1
while(i<183)
i=0
j = j + 1
while(j<93)
End
//****************************************************
// this function extracts all the parameters for the experiment from the last data file
//****************************************************
function extract_exp_params()
variable i =0
variable number, fraction, decimal
variable j =0
NVAR par_nums
WAVE param_waveO
Make/N=183/D/0 exp_parameters //make a wave containing all the parameters
exp_parameters = 0
do //go to TEA angle in text
    number = i
    i = (dimsize(param_waveO,0))
endif
i = i + 1
while(i < (dimsize(param_waveO,0)))
    i = number
    do // this outer loop goes through each parameter
        number = 0
fraction = 0
declal = 0

// this loop goes up to where the number begins
if ((param_wave0[i]<48 || param_wave0[i] >57))
i = i + 1
endif

while ((param_wave0[i]<48 || param_wave0[i] >57))
do
// this loop converts the stream of single numbers into an actual number
if (param_wave0[i] == char2num(".")) // handles the decimal point
decimal = 1
i = i + 1
endif

if (decimal == 0)
number = number*10 + str2num(num2char(param_wave0[i]))
// do the main number
elseif (decimal > 0)
fraction = fraction + ((str2num(num2char(param_wave0[i])))/(10^decimal))
// do the fraction
enddo

i = i + 1
while((param_wave0[i]>47 && param_wave0[i] < 58) || (param_wave0[i] == char2num(".") )
// 47 and 58 are ascii codes for 0 - 9
number = number + fraction
exp_parameters[j] = number // store number in wave
if (j == 7 && exp_parameters[j] < 1) // check to see if this value is energy increment or energy value
par_nums = 11 // as older files are offset by -1 due to the addition of a time variable
elseif (j == 7 && exp_parameters[j] > 1)
par_nums = 12
endif

j = j + 1
while(j<par_nums)
exp_parameters[182] = par_nums
End

function make_energy_scale_4_tiling()
list_of_tiles
variable num_of_pix = Ki -KO
End

// this function generates a full energy scale according to the pixelar energy increments on each tile
//***************************************************************
function new_energy_scale_b4_tile()
list_of_tiles
INVAR
energy_inc
INVAR
calib_energy
INVAR
denergy
NVAR
d_num_of_pix
variable i = 0
variable j = 0
variable tile_num = dimsize(list_of_tiles, 0) - 1
string temp_string
variable num_of_pix = Ki - KO
string new_string
Make/N=O/D/O first_energies
variable energy_max
variable energy_inc
do
// this next bit calculates the maximum energy value for each tile
temp_string = list_of_tiles[i] + ".info" // And stores it in its .info wave
duplicate/O $temp_string temp_info_wave
temp_info_wave[par_nums] = temp_info_wave[par_nums-2] + (93*temp_info_wave[par_nums-1])
//calculate max energy
temp_info_wave[par_nums+1] = (temp_info_wave[par_nums] - temp_info_wave[par_nums-2]) / (K1-K0)
//calculate new pixelar increment after no. of pixels per tile decreases
duplicate/O temp_info_wave $temp_string //overwrite old info wave
i = i + 1
while(i<dimsize(list_of_tiles, 0))
kilwave temp_info_wave
i = 0
do
make/W=(num_of_pix)/D/O temp_wave //create wave that is same size as single tile
temp_wave = 0
temp_string = list_of_tiles[tile_num] + ".info"
duplicate/O $temp_string temp_info_wave //copy the info wave into a temp wave to
prevent access problems
do
  temp_wave[i] = (temp_info_wave[(par_nums+1)] + i) + temp_info_wave[par_nums-2]
  i = i + 1 //using new energy increment
while (i < num_of_pix)
//start concatenate waves.....
i = dimsize(first_energies,0) //measure previous size of final destination wave (where new first point will be)
insertPoints i, num_of_pix, first_energies //add new rows to final wave
do
  first_energies[j+ i] = temp_wave[j]
j = j + 1
while(<num_of_pix)
  //copy new data across
i = 0
j = 0
tile_num = tile_num - 1
while (tile_num>=0)
kilwave temp_info_wave temp_wave
temp_string = list_of_tiles[0] + ".info" //calculate no. of pixels in new energy scale
duplicate/O $temp_string temp_info_wave
num_of_pix = temp_info_wave[par_nums]
energy_inc = temp_info_wave[par_nums-1]
temp_string = list_of_tiles[dimsize(list_of_tiles,0)-1] + ".info"
duplicate/O $temp_string temp_info_wave
num_of_pix = (num_of_pix - temp_info_wave[par_nums-2]) / energy_inc
make/W=(num_of_pix)/D/O second_energies //get first info tile to get energy increments
do
  second_energies[i] = (energy_inc+i) + temp_info_wave[par_nums-2]
i = i + 1
while (i< num_of_pix)
kilwave temp_info_wave temp_wave
End
//************************************************************************************
// This function generates a new full tile with a linear energy scale using interpolation
//************************************************************************************
function interp_new_full_tile()
WAVE first_energies
WAVE second_energies
WAVE tt
variable tile_height
variable new_height
variable i = 0
variable j = 0
tile_height = dimsize(tt,1)
new_height = dimsize(second_energies, 0)
Make/O/N=(183,new_height)/I new_tile

Make/O/N=(tile_height)/D temp
do
do  //copy a column from original full tile to a temp wave
  temp[i] = tt[j][i]
i = i + 1
while(i<tile_height)
i = 0
do  //interpolate new counts from the new calculated energies
  new_tile[j][i] = interp(second_energies[i], first_energies, temp)
i = i + 1
while(i<new_height)
i = 0
j = j + 1
while(j<183)
duplicate/O new_tile tt
tt = new_tile
killwaves temp new_tile
End

//***************************************************************
// This function returns a dimension of the viewport
//***************************************************************
function screen_dim(dimension)
  string dimension
  string height
  string width
  width"=stringfromlist ( 3., stringbykey("SCREEN1", igorinfo(O»., ",")
  height"=stringfromlist ( 4., stringbykey("SCREEN1", igorinfo(O»., ",")
  if (cmpstr(dimension, "height") == 0 )
    return str2num(height)
  else
    return str2num(width)
endif
End

//***************************************************************
// This function checks to see if a wave exists in the table
//***************************************************************
function check_wave_in_table(wave_name)
  string wave_name
  WAVE/T list_of_new_tiles
  variable i = 0
do
    if(cmpstr(wave_name, list_of_new_tiles[i]))
      return 1
    endif
    i = i + 1
while(i<dimsize(list_of_new_tiles,0))
  return 0
End

//***************************************************************
// This function removes a wave and all its trackers
//***************************************************************
function kill_tile(ctrlName)
  String ctrlName
  string temp_string
  SVAR active_wave
  temp_string = active_wave
  string temp_string2 = active_wave + ".info"
variable i = 0
WAVE/T list_of_new_tiles
SVAR list_of_new_tiles_string
try
  killwaves $temp_string; AbortonRTE
  if(waveexists($temp_string2))
    killwaves $temp_string2
  endif
catch
  return 0
endtry
do
  if(tempstr(list_of_new_tiles[i], temp_string))
    DeletePoints i, i, list_of_new_tiles
  Endif
  i = i + 1
while(i < (dimsize{list_of_new_tiles,O}))
list_of_new_tiles_string = removefromlist(temp_string, list_of_new_tiles_string)
refresh_panel()
End
//********************************************************************************
// This function sets the active wave variable to whatever is selected
//********************************************************************************

Function set_active_wave(ctrlName, popNum, popStr)
String ctrlName
Variable popNum
String popStr
  string/G active_wave = popStr
use_active_wave_info()
End

function use_active_wave_info()
  SVAR active_wave
  NVAR par_nums
  string temp_string
  variable/G calib_angle
  temp_string = active_wave + ".info"
  duplicate/O $temp_string temp_info_wave
  par_nums = temp_info_wave[182]
  k3 = temp_info_wave[2] // TEA
  k6 = temp_info_wave[par_nums - 4] // angle per pixel
  k7 = temp_info_wave[par_nums - 1] // energy per pixel
  k8 = temp_info_wave[par_nums - 2] // calib energy
  calib_angle = temp_info_wave[par_nums - 5]
  killwaves temp_info_wave
End

Function save_active_wave(ctrlName)
String ctrlName
String temp_string
SVAR active_wave
  variable i
  duplicate/O $active_wave wave_to_save
  InsertPoints/[i=1 0,1, wave_to_save
  temp_string = active_wave + ".info"
  duplicate/O $temp_string temp_info_wave
do
  wave_to_save[i][0] = temp_info_wave[i]
  i += 1
while (i < 183)
End

Function load_saved_wave(ctrlName)
    String ctrlName
    String tempstring
    String tempstring2
    Variable i = 0
    WAVE/T list_of_new_tiles
    SVAR list_of_new_tiles_string
    LoadWave/O/J/M/D/A=temp_thing/K=O
    tempstring2 = S_filename
    Do
        // remove the .2d.data. from the file name as it is too long otherwise
        If ((cmpstr(tempstring2[i, i + 8], ".*.2d.data."))
            tempstring2[i, i + 8] = "."
        Endif
        i += 1
    While(i + 7 < strlen(tempstring2)
    i = 0
    Do
        // remove the .txt from the file name as it is too long otherwise
        If ((cmpstr(tempstring2[i, i + 3], ".*.txt"))
            tempstring2[i, i + 3] = ""
        Endif
        i += 1
    While(i + 3 < strlen(tempstring2)
    i = 0
    tempstring = tempstring2 + ".info"
    Make/N=183/D/O tempwave
    duplicate temp_thingO temp_mat.
    Do
        tempwave[i] = temp_mat[i][0] //extract info wave
        i += 1
    While(i < 183)
    DeletePoints/M=1 0,1, temp_mat
    duplicate/O temp_mat $tempstring2
duplicate/O tempwave $tempstring
    killwaves temp_thingO tempwave temp_mat
    If((check_wave_in_table(S_filename)))//make a tracker of wave
        InsertPoints dimsize(list_of_new_tiles,0),1, list_of_new_tiles
        list_of_new_tiles[dimsize(list_of_new_tiles,0)-1] = tempstring2
        list_of_new_tiles_string = list_of_new_tiles_string + tempstring2 + ":"
    Endif
    refresh_panel()
End

Function display_active_wave(ctrlName)
    String ctrlName
    SVAR active_wave
    duplicate/O $active_wave k_corr_tile
    execute ("display_kcorr()")
End
#pragma rtGlobals=1 // Use modern global access method.
SetVariable Smooth_Amount proc=Smooth_Raw_Data
// Data program by Chris Howe Loughborough University

// V1 11th March 2005

/******************
This function creates the main control panel
******************

Window panel0(): Panel
PauseUpdate; Silent 1 // building window...
NewPanel/Axis=(15.75,44.267,404.75)
SetDrawLayer UserBack
DrawRect 207.3,3,8,112
DrawRect 10,226,207,117
Button button0,pos=(20,10),size={150,20},proc=load_raw_data,title="Load Raw Data (substrate)"
SetVariable Smooth_Amount,pos=(19,40),size={150,16},proc=Smooth_Raw_Data,title="Smoothing Amount"

SetVariable Smooth_Amount,limits={0,Inf,1},value=0
Button button0,pos=(20,44),size={150,20},proc=find_peak,title="Find Peaks (Substrate)"

CheckBox rutherford_correct,pos={20,64},size={96,14},proc=checkbox_rutherford_correction,title="Rutherford Correct"
CheckBox rutherford_correct,value=0

Button load_sim_data,pos=(20,124),size={150,20},proc=load_sim_data,title="Load Sim Data"

PopupMenu start_layer,pos={9,147},size={139,21},proc=sim_start_layer_popup,title="Start - End Layer"
PopupMenu start_layer,node=7,popvalue="1",value="#calc_list_of_no_of_sim_layers()"

PopupMenu end_layer,pos={153,146},size={52,21},proc=sim_end_layer_popup
PopupMenu end_layer,node=29,popvalue="29",value="#calc_list_of_no_of_sim_layers()"

SetVariable sim_data_smooth,pos={27,172},size={150,16},proc=Smooth_sim_Data_button,title="Smoothing Amount"

SetVariable sim_data_smooth,FSize=9,limits={0,Inf,1},value=1
Button find_sim_peaks,pos=(20,199),size={150,20},proc=find_sim_peak,title="Find Simulation Peaks"

Button compare_data,pos=(20,237),size={150,20},proc=cmp_sim_raw_peaks,title="Compare Peak Locations"

Button load_data_2b_corrected,pos={19,266},size={150,20},proc=load_data_to_be_corrected,title="Load Data to be Corrected"

Button correct_data,pos=(19,294),size={150,20},proc=correct_data,title="Correct Data"

Button save_data,pos=(18,321),size={170,20},proc=save_data,title="Save Calibrated (Rebinned Data)"
EndMacro

/*************************************************
This function loads the raw data from a 2 column angle/counts text file
/*************************************************

Macro load_raw_data(ctrlName) : ButtonControl
String ctrlName
string /G raw_data_file_name = "Data may not be loaded"
variable /G have_raw_peaks Been_found = 0
//LoadWave/G/A/G/0/N = rawdata
LoadWave/GRAY/N = rawdata
raw_data_file_name = a_fileName
Duplicate/O rawdatal rawdata1_smoothed
Duplicate/O rawdata0 rawdata0_smoothed
smooth_raw_data("",0,"","")
Display_Raw_Data()
TextBox/0/X=3.71/Y=0.55 "\Z08\(\{raw_data_file_name}"
DoWindow/K panel0
panel0()
This function displays the raw data on a graph on the top left of the screen.

Function Display_Raw_Data():

PauseUpdate; Silent 1 // building window...
DoWindow /K Display_Raw_Data // Remove any previous window to keep workspace tidy
Display /K=(200,39,520,210) rawdata1_smoothed vs rawdata0_smoothed as "Raw Data"
ModifyGraph marker=19
ModifyGraph msize=2
ModifyGraph mrkThick=0.05
ModifyGraph fSize=0
ModifyGraph axOffset(left)=-1.5,axOffset(bottom)=-0.5
wavestats /Q rawdata1_smoothed
SetAxis left V_min=(V_min*0.05), V_max+(V_max*0.05)

EndMacro

This function smooths the raw data.

Function Smooth_Raw_Data(ctrlName, varNum, varStr, varName):

String ctrlName
Variable varNum
String varStr
String varName
NVAR bave_raw_peaks_been_found
variable i = 0
WAVE rawdata1_smoothed =rawdata1_smoothed // Update smoothed data from original data so you can...
WAVE rawdata1 = rawdata1 // reverse the smoothing
rawdata1_smoothed = rawdata1
if (kO>O)
 Smooth kO, rawdata1_smoothed // Smooth the data to a given amount
Endif
wavestats /Q rawdata1_smoothed
if(!cmpstr( winlist("display_raw_data",**"","Display_Raw_Data")))
 SetAxis /W= Display_Raw_Data left V_min=(V_min*0.1), V_max+(V_max*0.05)
ENDIF
if(have_raw_peaks_been_found=1)
 rescale_dip_marker_size()
ENDIF // Adjust the dip marker size after smoothing
CheckBox rutherford_correct value = 0

End

This function finds all the dips in the raw data and makes a full wave of these (182 points).

Function find_peak(ctrlName): ButtonControl

String ctrlName
Variable i = 1
Variable j = 10
Variable abort = 0
Variable peak_number = 0
Make/N=0/D=0 peak_locations
WAVE rawdata1_smoothed = rawdata1_smoothed
NVAR have_raw_peaks_been_found
have_raw_peaks_been_found = 1
do findpeak /Q/N=1,10] rawdata1_smoothed
if(V_Flag != 0)  //If peak isn't found then increase the search window
j = j + 1
else
  InsertPoints peak_number, peak_locations  //If peak is found then make a note of it
  peak_locations[peak_number] = V_PeakLoc
  i = V_PeakLoc + 1
  peak_number = peak_number + 1
  j = 10
endif
if ((i + j) > 182)  //If search window exceeds data set then abort
  abort = 1
endif
while (abort == 0)
  rescale_dip_marker_size()
End

/********************************************
** This function rescales the blue dip marker points (shown on graph) after rutherford corrections
** and smoothing operations
*********************************************/
function rescale_dip_marker_size()
  WAVE peak_locations = peak_locations
  WAVE rawdata1_smoothed = rawdata1_smoothed
  variable i = 0
  Make /G=153 dip_find
dip_find = 0
wavestats /G peak_locations
do
  dip_find(peak_locations[i]) = rawdata1_smoothed(peak_locations[i])
i = i + 1
while (i < V_Pnts)
  if (((cmpstr( winlist("display_raw_data", ",", ",", "Display_Raw_Data")) )//Update graph if it exists
    RemoveFromGraph /W= Display_Raw_Data /Z dip_find
    AppendToGraph /W= Display_Raw_Data dip_find vs rawdata0
    ModifyGraph /W= Display_Raw_Data mode(dip_find)=1,rgb(dip_find)=(0.43520,65280)
  Endif
End

/****************************************************
** END OF RAW DATA BIT
****************************************************
/****************************************************
** This function loads simulation data straight from VEGAS output
****************************************************
macro load_sim_data(ctrName) : ButtonControl
  String ctrlName
  string/G sim_data_file_name
  variable i = 0
  variable /G have_peaks_been_found = 0
  LoadWave/C/H/D/A=simdata
duplicate/G simdata0 sim_2d_angle
Redimension/N=(-1,1) sim_2d_angle
DeletePoints/W=1 0,1, simdata0
duplicate/G simdata0 sim_2d_cut_smoothed sim_2d_cut_smoothed_old
sim_data_file_name = a_filename
killwaves simdata0
display_sim_data()
End
This macro displays the simulation graph in the top right of the screen.

**Window display_sim_data()** : Graph

```plaintext
PauseUpdate; Silent 1 // building window...
String fldrSav = GetDataFolder()
DoWindow /K Display_simData
Display /K=1/W=(525,39,660,210) sim_2d_cut_smoothed vs sim_2d_angle as "Simulated Data"
ModifyGraph grid(bottom)=1
ModifyGraph fSize=8
ModifyGraph axOffset(left)=-3.14286,axOffset(bottom)=-1.07692
ModifyGraph axisEnab(left)=0.0,0.065
ModifyGraph rgb(sim_2d_cut_smoothed)=(0,0,66280)
wavestats /O sim_2d_cut_smoothed
TextBox/N=text1/A=LT/X=70.54/Y=-5.82 "\Z08\{ "Cut from layer "+/g to layer "+/g", start_layer,
end_layer)"
TextBox/N=text2/A=LT/X=70.54/Y=-6.35 "\Z08\{sim_data_file_name)"
SetAxis /W=display_sim_data left V_min=(V_min+0.05), V_max=(V_max+0.05)
EndMacro
```

**Function smooth_sim_data_button(ctrlName, varNum, varStr, varName)** : SetVariableControl

```plaintext
String ctrlName
Variable varNum
String varStr
String varName

Smooth_sim_Data()

End function Smooth_sim_Data() : SetVariableControl
```

variable i = 0
NVAR have_peaks_been_found
WAVE sim_2d_cut_smoothed_old //...reverse the smoothing
WAVE sim_2d_cut_smoothed
sim_2d_cut_smoothed = sim_2d_cut_smoothed_old
if (kl>0)
Smooth kl, sim_2d_cut_smoothed
Endif
if (have_peaks_been_found == 1)
recale_sim_dip_marker_size()
endif
if((cmpstr(winlist("display_sim_data",",","), "Display_sim_data")))/Update graph if it
exists
wavestats /O sim_2d_cut_smoothed
SetAxis /W=display_sim_data left V_min=(V_min+0.1), V_max=(V_max+0.05)
Endif
End

**Function find_sim_peak(ctrlName)** : ButtonControl

```plaintext
String ctrlName
variable i = 1
variable j = 10
variable abort_findpeak = 0
variable peak_number = 0
NVAR have_peaks_been_found
have_peaks_been_found = 1
Make/N=0/D/O sim_peak_locations
```

This function finds all the dips in the sim data and makes a full wave of these (182 points).

**Function find_sim_peak(ctrlName)** : ButtonControl

```plaintext
String ctrlName
variable i = 1
variable j = 10
variable abort_findpeak = 0
variable peak_number = 0
NVAR have_peaks_been_found
have_peaks_been_found = 1
Make/N=0/D/O sim_peak_locations
```
WAVE sim_2d_cut_smoothed = sim_2d_cut_smoothed
sim_peak_locations = 0
do
    findpeak /Q/N/N=\[i,i+j\] sim_2d_cut_smoothed
    if(Y_Flag = 0) //If peak isn't found then increase the search window
        j = j + 1
    else
        insertpoints peak_number, i, sim_peak_locations //If peak is found then make a 
        note of it
        sim_peak_locations[peak_number] = Y_PeakLoc
        i = Y_PeakLoc + 1
        peak_number = peak_number + 1
        j = 10
    endif
    if ((i + j) > 182) //If search window exceeds data set then abort
        abort_findpeak = 1
    endif
endwhile(abort_findpeak == 0)
rescale_sim_dip_marker_size()

//**********************************************************************************************
// This function rescales the blue dip marker points (shown on graph) after rutherford corrections
// and smoothing operations
//**********************************************************************************************
function rescale_sim_dip_marker_size()
WAVE sim_peak_locations
WAVE sim_2d_angle
WAVE sim_2d_cut_smoothed
variable i = 0
variable number_of_peaks = 0
number_of_peaks = dimsize(sim_peak_locations, 0)
duplicate/O sim_2d_angle sim_dip_find
sim_dip_find = 0
D
    sim_dip_find = sim_peak_locations[i] = sim_2d_cut_smoothed[sim_peak_locations[i]]
    i = i + 1
while(i < number_of_peaks)
    if compile Trần('display_sim_data"","","","Display_Sim_Data") // Look to see if graph is 
        displayed
        if(i == 0) //If it is then...
            removefromgraph /W display_sim_data /Z sim_dip_find
            appendtograph /W display_sim_data sim_dip_find vs sim_2d_angle //display where the 
                dips are
            modifygraph /W display_sim_data mode(sim_dip_find)=1, rgb(sim_dip_find)=(0,43520,65280)
        endif
    wavestats /Q sim_2d_cut_smoothed
    setaxis /W display_sim_data left V_min-(V_min.O.1), V_max+(V_max.O.05)
End

//**********************************************************************************************
// END OF SIM DATA BIT
//**********************************************************************************************
// This function moves the data and produces waves listing the dips angular locations in the root 
// data folder, and calls another function to display a graph
//**********************************************************************************************
function cmn_sim_raw_peaks(ctrlName): ButtonControl
    String ctrlName
    WAVE peak_locations = peak_locations
    WAVE rawdata_smoothed = rawdata_smoothed
WAVE rawdataO = rawdataO
DoWindow /K show_peak_comparison
NVAR have_raw_peaks_been_found
if (have_raw_peaks_been_found == 0)
    Abort "Must find raw data peaks to compare first! Click on find peaks(substrate)"
Endif
NVAR have_peaks_been_found
if (have_peaks_been_found == 0)
    Abort "Must find simulation peaks to compare first! Click on find simulation peaks"
Endif
WAVE sim_2d_cut_smoothed = sim_2d_cut_smoothed
WAVE sim_peak_locations = sim_peak_locations
WAVE sim_2d_angle = sim_2d_angle
variable i = 0
duplicate /0 sim_peak_locations sim_peak_locations_copy
duplicate /0 sim_peak_locations sim_peak_angular_locations
duplicate /0 sim_peak_locations sim_peak_heights
sim_peak_angular_locations = 0
sim_peak_angular_locations = 0
do
    //find heights of each peak, make a wave of that peak
    sim_peak_heights[i] = sim_2d_cut_smoothed[sim_peak_locations[i]]
    sim_peak_angular_locations[i] = sim_2d_angle[sim_peak_locations[i]]
    i = i + 1
while (i < dimsize(sim_peak_locations, 0))
duplicate /0 peak_locations peak_angular_locations
duplicate /0 peak_locations peak_heights
peak_angular_locations = 0
peak_heights = 0
i = 0
do
    //find heights of each peak, make a wave of that peak
    peak_heights[i] = rawdataI_smoothed[peak_locations[i]]
    peak_angular_locations[i] = rawdataO[peak_locations[i]]
    i = i + 1
while (i < dimsize(peak_locations, 0))
execute "show_peak_comparison()"
execute "match_data_panel()"
End

//***************************************************************
// This function makes a graph of the locations of dips of sim data and raw data
// and displays them as bars of different colours
//***************************************************************
Window show_peak_comparison() : Graph
PauseUpdate; Silent 1 // building window...
Display /K=W=(525.75,230,857.25,428) peak_heights vs peak_angular_locations as "Comparison of Sim and Raw data dips"
AppendToGraph/R sim_peak_heights vs sim_peak_angular_locations
ModifyGraph mode=9
ModifyGraph marker(peak_heights)=9
ModifyGraph lSize=2
ModifyGraph rgb(peak_heights)=(65280,0,0),rgb(sim_peak_heights)=(0,0,65280)
ModifyGraph msize=3
ModifyGraph textMarker(peak_heights)="(peak_angular_locations,"default",0,0,5,0.00,5.00)
ModifyGraph textMarker(sim_peak_heights)="(sim_peak_angular_locations,"default","0,0,5,0.00,5.00"
ModifyGraph tick(bottom)=2
ModifyGraph fSize=0
ModifyGraph lb1Margin(bottom)=3
ModifyGraph standoff(bottom)=0
ModifyGraph axOffset(Left)=−3.14288,axOffset(bottom)=−0.346154,axOffset(right)=−3.33333
ModifyGraph lb1LastPos(bottom)=−30

************************************************************
**********
This function makes a graph of the locations of dips of sim data and raw data and displays them as bars of different colours
************************************************************
**********
ibis function reads the contents of a wave and returns it as a string separated by ;

function/S return_string(wave_name)

wave wave_name

string list_of_elements = ""

variable i = 0
do

list_of_elements = list_of_elements + num2str(wave_name[i]) + ";" 

i = i + 1
while(i < dimsize(wave_name, 0))

return list_of_elements
end

This function creates a panel to select where the raw data angles should in relation to the sim data

Window match_data_panel() : Panel
dowindow/k match_data_panel

variable i = 0
variable j = 0

string alphabet = "abcdefghijklmnopqrstuvwxyz" //for popdown menu auto naming
make/7/0/26 Alphabet_wave // This is done because popups cant have names beginning with a number

do

Alphabet_wave[i] = Alphabet[i]
i = i + 1
while(i < 26)

if(dimsize(peak_angular_locations, 0) <= dimsize(sim_peak_angular_locations, 0))

j = dimsize(peak_angular_locations, 0) //Find the smallest no. of dips
else

j = dimsize(sim_peak_angular_locations, 0)
endif

PauseUpdate; Silent 1 // building window...

NewPanel /k=1/\w=(15.75,410,360,490+25;i)

CheckBox select_fit pos=(200,15).title="Cubic Spline (or linear)", proc = what_spline

Button Create_calibration title="Create Calibration", pos=(16,16)

SetDrawLayer UserBack

DrawText 118,64,"should be at this" 

SetDrawEnv fstyle=1,textrgb= (65280,0,0)

DrawText 4,64,"Raw Data dip angle"

SetDrawEnv fstyle=1,textrgb= (0,0,65280)

DrawText 215,64,"Sim data angle"
do

//produces an arrow pointing from a left popdown to a right popdown

PopupMenu $alphabet_wave[i] + num2str(i) + " _Left" ,pos=(20,70+25+i),size=(100,20),
proc =popup_match_angles
PopupMenu $alphabet_wave[i] + num2str(i) + ",Left\",node=1,popvalue="Select",value="\"Select\";\" + return_string(peak_angular_locations)
PopupMenu $alphabet_wave[i] + num2str(i) + ",Right\",pos=(220,70+25*i),size={100,20},
proc =popup_match_angles
PopupMenu $alphabet_wave[i] + num2str(i) + ",Right\",node=1,popvalue="Select",value="\"Select\";\" + return_string(sim_peak_angular_locations)

while (i<j)
KillWaves Alphabet_wave
EndMacro

// This function is called every time the match data panel popups are used. It produces a matrix
// that matches the user selected angle conversions.
// This function produces a calibration (angular correction) wave and graphs it
// from the popup menus selected
Function popup_match_angles(ctrlName,popNum,popStr) : PopupMenuControl
String ctrlName
Variable popNum
String popStr
WAVE match_angles_wave = match_angles_wave
variable i = 0
variable j = 0
if ((cmpstr(ctrlName[3], "R") == 0) //Work out if it is a left or right popup being used.
  j = 1
endif
i = str2num(ctrlName[1]) // Work out which popup row is being used
match_angles_wave[i][j] = str2num(popStr) // Put the selected value into the matrix
End

Function create_calibration(ctrlName) : ButtonControl
String ctrlName
WAVE match_angles_wave = match_angles_wave
WAVE peak_locations = peak_locations
variable i = 0
variable j = 0
Make/N=(O)/D/O correction_points_offset
Make/N=(O)/D/O correction_points_bin_number
do // Convert popup menu data to bin number and equivalent offset (ignoring mismatched points)
  if (( match_angles_wave[i][0] & match_angles_wave[i][1]) > 0)
    InsertPoints i,1, correction_points_offset
    InsertPoints i,1, correction_points_bin_number
    correction_points_offset[j] = match_angles_wave[i][1] - match_angles_wave[i][0]
    findvalue/T=0.005/V=(match_angles_wave[i][0]) peak_angular_locations
    correction_points_bin_number[j] = peak_locations[V_Value]
    j = j + 1
  endif
i = i + 1
while (i < dimsize(match_angles_wave, 0))
InsertPoints i,1, correction_points_offset //deal with end points
correction_points_offset[j] = correction_points_offset[j-1]
InsertPoints 0,1, correction_points_offset //deal with start points
correction_points_offset[0] = correction_points_offset[1]
InsertPoints i,1, correction_points_bin_number //deal with end points
correction_points_bin_number[j] = 162
InsertPoints 0,1, correction_points_bin_number //deal with start points
correction_points_bin_number(0) = 0
dowindow/K Calibration_graph
Execute "InterpolateITs(0/to_spline_or_not)/Y=183/X=2/Y=offset_interp/X=bin_interp/
correction_points_offset /K=correction_points_bin_number"
Execute "Calibration_graph()"

End

This function is called when the cubic spline tick box is selected, and then sets a global variable
in accordance with that.

Function what_spline(ctrlName, checked) : CheckBoxControl
String ctrlName
Variable checked
variable /G to_spline_or_not = 1
to_spline_or_not = checked + 1
End

This function displays the calibration graph.

This function loads the raw data that needs to be corrected.

Function load_data_to_be_corrected(ctrlName) : ButtonControl
String ctrlName
string /G corr_raw_data_file_name = "Data may not be loaded"
LoadWave/G/A/B/C/N = rawdata_2b_corrected
corr_raw_data_file_name = s_filename
 dowindow /K=graph_data_to_be_corrected
execute "graph_data_to_be_corrected()"
End

This function creates the new angular scale, and then finds what corrected angular points
the new angular point is in between. This loops for every point. The interpolation between the 2 points
is a separate algorithm.

Function correct_data(ctrlName) : ButtonControl
String ctrlName
variable i = 0
variable j = 0
WAVE rawdata_2b_corrected0 = rawdata_2b_corrected0
WAVE rawdata_2b_corrected1 = rawdata_2b_corrected1
WAVE offset_interp = offset_interp
Make/N=183/D/O temp_angular_scale, temp_counts_scale, final_corrected_angle,
final_corrected_counts
temp_angular_scale = 0
final_corrected_angle = 0
final_corrected_counts = 0
do // Add the offsets to the angular scale
    temp_angular_scale[i] = offset_interp[i] + rawdata_2b_corrected[i]
    final_corrected_angle[i] = temp_angular_scale[i] + (i*0.15)
    i = i + 1 // Create new FINAL angular scale with equal increments
while (i < dimsize(offset_interp, 0))
    i = 0
    do
        if(final_corrected_angle[i] == temp_angular_scale[i+j])
            final_corrected_counts[i] = rawdata_2b_corrected[i+j]
        elseif(final_corrected_angle[i] > temp_angular_scale[i])
            j = j + 1
            if (final_corrected_angle[i] < temp_angular_scale[i+j])
                final_corrected_counts[i] = interpolate_2_points(i+j-1, i + j, i)
            endif
        elseif(final_corrected_angle[i] < temp_angular_scale[i])
            j = j - 1
            if (final_corrected_angle[i] > temp_angular_scale[i+j])
                final_corrected_counts[i] = interpolate_2_points(i+j, i+j+1, i)
            endif
        endif
    endwhile
    AppendToGraph final_corrected_counts vs final_corrected_angle
    ModifyGraph rgb(final_corrected_counts)=(0,0,66280)
End

//***********************************************************************
// This function is the algorithm that interpolates between 2 points pi and p2. i is the
// index to the new value that we need a value for.
//**************************************************************************
function interpolate_2_points(pi, p2, i)
    variable pi
    variable p2
    variable i
    variable result = 0
    variable m = 0
    variable c = 0
    temp_angular_scale hand new angular scale with equal
    increments
    temp_angular_scale = temp_angular_scale // Old angular scale + offset
    rawdata_2b_corrected = rawdata_2b_corrected // Data counts wave from which new data
    is to be extracted
    m = rawdata_2b_corrected[i] - rawdata_2b_corrected[i]
    m = m/(temp_angular_scale[i] - temp_angular_scale[i])
    c = rawdata_2b_corrected[i] - (m*temp_angular_scale[i])
    result = (m*final_corrected_angle[i]) + c
    return result
End

//***********************************************************************
// This macro displays the data to be corrected
//**************************************************************************
Window graph_data_to_be_corrected() : Graph
PauseUpdate; Silent 1 // building window...
Display /ki/ (299.25,206.75,693.75,615.25) rawdata_2b_corrected1 vs rawdata_2b_corrected0
ModifyGraph tick=1
ModifyGraph fSize=8
ModifyGraph standoff(left)=0
ModifyGraph axOffset(left)=-2.33333, axOffset(bottom)=-0.75
Label left "Counts"
Label bottom "\Z08\{corr_raw_data_file_name}"
TextBox/W=text0/X=3.71/Y=0.55 "\Z08\{corr_raw_data_file_name}
EndMacro

//******************************************************************************
// This function saves the final data - one column counts, one column angle
//******************************************************************************
Function save_data(ctrlName) : ButtonControl
  String ctrlName
  Save/J/W="\r\n"/W/F final_corrected_counts,final_corrected_angle as "final_corrected_counts++.dat"
End