Documentation for the automated LEED–IV package CLEED

Georg Held

September 14, 2001

Contents

List of Figures

List of Tables

Part I

Principles of LEED Theory and I–V Analysis

Chapter 1

Introduction

1.1 History

When Davisson and Germer reported in 1927 that the elastic scatting of low–energy electrons from well ordered surfaces leads to diffraction spots similar to those observed in X–ray diffraction [1, 2, 3], this was the first experimental proof of the wave nature of electrons. A few years before, in 1923, de'Broglie had postulated that electrons have a wave length of

$$\lambda_e = \frac{h}{m_e v} = \sqrt{\frac{150}{E_{kin}[eV]}}[\text{Å}] \tag{1.1}$$

and a corresponding wave vector of the length

$$k = 2\pi/\lambda_e \tag{1.2}$$

where h is Planck's constant, me the electron mass, v the velocity, and E_{kin} the kinetic energy of the electron. Already Davisson and Germer realised that the diffraction of low–energy electrons (LEED) in the energy range between 40 and 500 eV, where their wave length ranges between 0.5 and 2 Å, can be used in order to determine the structure of single crystal surfaces in analogy to X–ray diffraction. Due to their small inelastic mean free path (IMFP) of only a few Å(typically less than 10 Å) electrons in this energy range sample only the top–most i atomic layers of a surface and are, therefore, better suited for the analysis of surface geometries than X–ray photons which have a much larger mean free path (typically a few μm). However, unlike for photon diffraction, multiple scattering plays an important role in the diffraction process of electrons at solid surfaces. Therefore, the analysis of LEED data with respect to the exact positions of atoms at the surface is somewhat more complicated and requires fully dynamical quantum mechanical scattering calculations.

The use of LEED for surface analysis became important when large single crystals became available for surface studies. It was first only used for a qualitative characterisation of surface ordering and the quantitative determination of the two–dimensional surface lattice parameters (e.g. superstructures). The information about the positions of the atoms in the surface is hidden in the energy–dependence of the diffraction spot intensities, the so–called LEED $I--V$, or $I(E)$, curves. In the late 1960's computer programs became available which could perform fully dynamical scattering calculations for simple surface geometries. The comparison of such theoretical $I--V$ curves for a set of model geometries with experimental data allows the determination of the atomic positions within the surface by trial and error. With the immense growth of available computer power and speed since then, LEED $I--V$ structure determination could be applied to a large number of more and more complex surface geometries which had made LEED the standard technique of modern surface crystallography.

For further details about the history, experimental setup, and theoretical approaches of LEED please refer to the books by Pendry, [43], Van Hove and Tong [44], Van Hove, Weinberg and Chan [45], and Clarke [7].

1.2 Instrumentation

The standard modern LEED optics is of the "rear view" type which is schematically shown in Figure **??**. The incident electron beam, accelerated by the potential V_0, is emitted from the electron gun behind the hemispherical fluorescent screen made of glass and hits the sample through a hole in the screen. The surface is in the centre of the hemisphere so that all back–diffracted electrons travel towards the LEED screen on radial trajectories. Before the electrons hit the screen they have to pass a retarding field energy analyser. It typically consists of four (or three) hemispherical grids concentric with the screen, each containing a central hole through which the electron gun is inserted. The first grid (nearest to the sample) is connected to earth ground, as is the sample, to provide an essentially field-free region between the sample and the first grid. This minimises an undesirable electrostatic deflection of diffracted electrons. A suitable negative potential $(V_0 - V_{sup})$ is applied to the second and third (only second) grids, the so–called suppressor grids, to allow a narrow energy range V_{sup} of elastically scattered electrons to be transmitted to the fluorescent screen. The fourth (third) grid is usually grounded in order to reduce field penetration of the suppressor grids by the screen voltage which is of the order of a few kV in order to make the diffraction spots visible. Since the fluorescent screen is transparent, the spots can be observed through a view–port behind the screen without being shadowed by the sample holder. Only the electron gun assembly (diameter < 15 mm) obstructs the view

slightly. Older designs have opaque screens which allow only the observation from behind the sample. A typical diameter of the complete optics is around 140 mm so that it fits into a 150 mm (ID) flange. To date, the usual way of recording the LEED pattern is a light–sensitive video camera with a suitable image processing system. In older systems movable Faraday cups (FC) were used which are detecting the electron current directly. Because of the long data acquisition times and the problems of transferring motion into UHV, these systems are mostly out of use nowadays.

Only one widely used type of LEED optics uses a Faraday cup arrangement as electron detector which is specially designed for an accurate spot profile analysis (SPA-LEED, cf. Figure **??**) [8]. In this design the FC is at a fixed position and the grids are replaced by round apertures defining the lateral resolution of the system. Instead of the position of the electron detector, the angle of the incoming electron beam is varied through an electrostatic octupole lens, thereby deflecting the desired part of the LEED pattern towards the detector without the need of moving parts. This type of LEED optics is specially designed to have a large resolution in k-space. The better the k resolution is the larger are the typical distances on the surface for which effects can be observed in the LEED pattern (spot position, profile). The largest resolvable length on the surface is called the transfer width and characterises the LEED instrument [9]. Typical values are around 150 Åfor conventional rear view LEED systems and up to 1000 Åfor SPA-LEED systems.

The standard experimental setup for collecting LEED I–V curves uses a video camera recording images of the fluorescent screen for each energy (Video LEED) [28]. When a conventional video camera is used the rate at which images are collected is fixed at 50 to 60 Hz. In this case several images (typically between 8 and 256) have to be averaged in order to obtain a satisfying signal–to–noise ratio, especially for weak superstructure spots. Newer systems often use slow-scan CCD cameras which can perform the averaging directly on the cooled CCD chip (exposure times up to several seconds) and therefore avoid multiple readout noise [29]. The images are either stored on a computer hard disk and analysed in a second round or the spot intensities are extracted online by a special software. Usually, the intensity is averaged within a given area at the spot position and the averaged local background outside this area is subtracted from this intensity. Commercially available data acquisition software also performs the control of the electron energy and records the electron beam current which is needed in order to normalise the spot intensities. Older experimental setups were using Faraday cups with small apertures mounted on goniometers which could be moved around the sample in order to collect the back-scattered electron current directly or spot photometers which were directed at one diffraction spot on the fluorescent screen which was then followed by hand while the energy was varied. The data collection is mostly done at normal incidence of the primary electron beam.

In this case, usually several equivalent LEED spots exist due to the surface symmetry. By taking care that the I-V curves of equivalent spots are identical, normal incidence conditions ca be adjusted within a few tenth of a degree.

Chapter 2

Qualitative Information: The LEED Pattern

The most direct information obtained from LEED concerns the periodicity and the intermediate range order within the transfer width of the surface under investigation. It can be gathered by visual inspection of the diffraction pattern and/or through relatively simple mathematical transformations of the spot profiles.

2.1 LEED Pattern

Since the electrons do not penetrate into the crystal bulk far enough to experience its three–dimensional periodicity, the diffraction pattern is determined by the two-dimensional surface periodicity described by the lattice vectors $\vec{a}_1$ and $\vec{a}_2$, which are parallel to the surface plane. A general lattice point within the surface is an integer multiple of these lattice vectors:

$$\vec{R} = m_1\vec{a}_1 + m_2\vec{a}_2 \tag{2.1}$$

The two-dimensional Bragg condition leads to the definition of reciprocal lattice vectors $\vec{a}_1^*$ and $\vec{a}_2^*$ which fulfil the set of equations:

$$\vec{a}_1\vec{a}_1^* = \vec{a}_2\vec{a}_2 = 2\pi; \quad \vec{a}_1\vec{a}_2^* = \vec{a}_2\vec{a}_1 = 0 \tag{2.2}$$

These reciprocal lattice vectors, which have units of Å^{-1} and are also parallel to the surface, define the LEED pattern in k–space. Each diffraction spot corresponds to the sum of integer multiples of $\vec{a}_1^*$ and $\vec{a}_2^*$.

$$\vec{g}(n_1, n_2) = n_1\vec{a}_1^* + n_2\vec{a}_2^* \tag{2.3}$$

The integer numbers (n_1, n_2) are used as indices to label the spot. The parallel component of the corresponding wave vector is:

$$\vec{k}_{\|}(n_1, n_2) = \vec{k}_{\|}(0,0) + \vec{g}(n1, n2) \tag{2.4}$$

where $\vec{k}_{\|}(0,0)$ is the parallel component of the wave vector of the incoming electron beam. The vertical component, k_z, of the back–diffracted electrons is defined by energy conservation:

$$k_z, (n_1, n_2) = \sqrt{\frac{2m_e E_{kin}}{h^2} - k_{\|}^2(n_1, n_2)} \tag{2.5}$$

This equation also limits the set of observable LEED spots by the condition that the expression inside the brackets must be greater or equal to zero. With increasing electron energy the number of LEED spots increases while the polar emission angle with respect to the surface normal, $\vartheta = \arctan(k_{\|}/k_z)$, decreases for each spot except for the specular spot (0,0) whose position does not change. Figure **??** shows examples of common surface unit cells and the corresponding LEED patterns.

In many cases (adsorption, reconstruction) the periodicity at the surface is larger than it is expected for a bulk–truncated surface of the given crystal which leads to additional (superstructure) spots in the LEED pattern for which fractional indices are used. The lattice vectors $\vec{b}_1$ and $\vec{b}_2$ of such superstructures can be expressed as multiples of the (1 x 1) lattice vectors $\vec{a}_1$ and $\vec{a}_2$:

$$\begin{aligned} \vec{b}_1 &= m_{11}\vec{a}_1 + m_{12}\vec{a}_2 \\ \vec{b}_2 &= m_{21}\vec{a}_1 + m_{22}\vec{a}_2 \end{aligned} \tag{2.6}$$

where the numbers m_{ij} are the coefficients of the superstructure matrix, which is a straightforward way of characterising the superstructure. The positions and indices of the additional LEED spots can be calculated directly from this matrix [45] according to the formulae:

$$\begin{aligned} \vec{b}_1^* &= \frac{1}{m_{11}m_{22} - m_{12}m_{21}}(m_{22}\vec{a}_1^* - m_{21}\vec{a}_2^*) \\ \vec{b}_2^* &= \frac{1}{m_{11}m_{22} - m_{12}m_{21}}(-m_{12}\vec{a}_1^* - m_{11}\vec{a}_2^*) \end{aligned} \tag{2.7}$$

The area of the superstructure unit cell, A, in units of the (1 x 1) unit cell area can also easily be calculated from the coefficients of the superstructure matrix:

$$A = (m_{11}m_{22} - m_{12}m_{21}) \tag{2.8}$$

Another, less general way is the notation according to Wood [10] where the lengths of the vectors $\vec{b}_1$ and $\vec{b}_2$ are specified in units of $\vec{a}_1$ and $\vec{a}_2$ together with the rotation angle α between $\vec{b}_1$ and $\vec{a}_1$ (only specified if non-zero):

$$p/c\left(\frac{|\vec{b}_1|}{|\vec{a}_1|} \times \frac{|\vec{b}_2|}{|\vec{a}_2|}\right) R\alpha \tag{2.9}$$

"p" indicates a "primitive" and "c" a "centred" surface unit cell. Examples are "$p(2\times 2)$" "$p(\sqrt{3}\times\sqrt{3})R30°$" and "$c(2\times 2)$". This notation is not applicable in all cases but it is more frequently used than the matrix notation because it is shorter. Figure **??** shows examples of common superstructures with the corresponding matrix and Wood notations.

2.2 Spot Profile Analysis

While the spot positions carry information about the size of the surface unit cell, the shapes and widths of the spots, i.e. the spot profiles, are influenced by the long range arrangement and order of the unit cells at the surface. If vertical displacements of the surface unit cells are involved (steps, facets), the spot profiles do change as a function of electron energy. If all surface unit cells are in the same plane (within the transfer width of the LEED optics), the spot profile is constant with energy. A periodic arrangement of equal steps at the surface leads to a spot splitting at certain energies from which the step height can be determined directly. For a statistical arrangement of steps the analysis of energy dependent changes in the spot profiles allows in many cases the determination of the mean step height and a characterisation of the step distribution [11]. Facets lead to extra spots which move in $k_{||}$ upon changes of the kinetic energy. Point defects, static disorder, and thermally induced displacements lead to an increase of the background intensity between the spots. Depending on the correlation between the scatterers, the background is either homogeneous (no correlation) or structured (correlation). If the coherently ordered surface areas (islands, domains) are smaller than the transfer width of the LEED system and at the same vertical height, the width of these areas, Δd, is directly related to the width of the LEED spots in k-space, $\Delta k_{||}$:

$$\Delta k_{||} = \frac{2\pi}{\Delta d} \tag{2.10}$$

This relation holds for each direction parallel to the surface independently. It is particularly useful for determining the size of adsorbate islands which lead to extra superstructure spots. A good introduction (in German) into spot profile analysis is given by Henzler and Göpel in Ref. [39].

Chapter 3

Theory: Calculating I–V curves

Chapter 4

R factors: Comparing Experiment and Theory

Chapter 5

Optimisation: Finding the best agreement

Part II

The LEED Programs

Chapter 6

Program description

In our *LEED code* fully dynamical scattering theory has been implemented along the lines of the algorithms described by Pendry (layer doubling for multiple scattering between sucessive layers of atoms [43]) and Van Hove/Tong (combined space method for layers with more than one atom per unit cell [44]). The extensive use of dynamical memory allocation – an intrinsic feature of the C programming language – allows the memory requirements, even for large surface unit cells, to be kept small. It also allows a very flexible input format which does not impose any restrictions on the number of bulk layers and overlayers nor on the number of atoms therein (other than the physically available memory). The input from the optimization program to the LEED program is simply a set of atomic coordinates from which the program creates its own set of Bravais and/or composite layers. For these layers scattering matrices are calculated and used to evaluate the amplitudes for multiple scattering between the layers.

While the main emphasis in the first development stage of the code had been put on reducing the time needed for the giant matrix inversion which is part of the combined space method, in the second stage the symmetries among beams were exploited in order to gain additional reductions in computing times. The scattering part of our implementation follows the algorithms developed by Van Hove and Tong [44]. A completely new development is the automated set up of the symmetry–reduced list of beams according to the symmetry of the surface (most possible combinations of rotational and mirror symmetries)

It does not include any linear approximations such as tensor LEED [55, 56] which are in general incompatible with global search strategies such as simulated annealing or the genetic algorithm, and can lead to errors even in downhill–oriented optimizations when the search path enters regimes of the parameter space where the approximation is inaccurate. It is, however, planned to implement this as an option for locally confined searches in the future version.

The input information required from the user is kept as little as possible; the program

creates most of its parameters from the geometry input provided in the input files. The distribution of atoms over different layers is performed by the program based on the atom coordinates and the minimum possible vertical distance (`MIN_DIST` = 1.1 Å) between two layers. The list of beams (length of vectors in plane wave representation) is created on the basis of the final energy and the minimum distance between the layers.

To date there exist two versions of cleed:

The symmetrised version `cleed_sym` makes use of mirror and rotational symmetries in the plane wave field and works thus with a reduced set of beams which can speed up the calculations significantly (up to a factor of 5 with respect to the non–symmetrised version). There are however some restrictions, the most important being that only the case of normal incidence and isotropic vibrations can be treated by the current version.

The non–symmetrised version `cleed_nsym` does not have this restrictions. It is more flexible and can treat most cases of surface geometries, provided, there is at least one bulk inter–layer distance greater than `MIN_DIST` = 1.1 Å.

Both program versions can use the same input files, whereby `cleed_sym` needs some additional information which is ignored by `cleed_nsym`. In any case it is strongly recommended to test the same input file with both programs in order to check the consistency. The relative intensities of equivalent beams should be identical down to 10^{-8}.

Chapter 7

General outline

7.1 Syntax

The general calling syntax of the LEED program is:

```
cleed -i <parameter file> -b <bulk parameter file> -o <results file>
      -r <storage file>   -w <storage file>
```

The first argument (`-i <parameter file>`) specifying the parameter input file is the only mandatory. The file contains all the geometric and non–geometric parameters needed for the LEED calculations. A sample file is shown in Table **??**. Alternatively the input can be split into two files, the parameter file and a bulk parameter file. The latter file (`-b <bulk parameter file>`) contains all the parameters which are not varied during the optimisation. Consequently, the search program has to produce only the parameter file containing the optimised atom positions of the overlayer in each iteration step of an automated search.

`-o <results file>` specifies the file name where the beam intensities will be written to. If this argument is not used, the results will be written to the file `leed.res`. Any existing files with the same name as the results file will be overwritten.

`-r <storage file>` and `-w <storage file>` specify files where the bulk reflexion matrices can be read from (`-r`) or written to (`-w`). This feature can be used in order to save computer time. It is, however, only recomended for very time consuming bulk matrix calculations, i.e. when the bulk unit cells contain more than one atom.

7.2 UNIX–Environment

The program has been developed in a UNIX environment and uses a few UNIX specific features. The most important is the use of environment variables:

`CLEED_PHASE`: directory where the files are stored which contain the energy dependent atomic phase shifts.

These variables have to be set using the export or setenv UNIX commands, respectively before the program is called for the first time.

Chapter 8

Input files

8.1 Parameter Input

Both program versions can use the same input files, whereby `cleed_sym` needs some additional information which is ignored by `cleed_nsym`. Table **8.1** shows an example for a single parameter input file for a $p(\sqrt{3} \times \sqrt{3})$ H_2O / Ru(0001) overlayer structure.

Each parameter or set of parameters, respectively, is specified by two letters and a colon at the beginning of a line. There can only be one parameter specifier per line. Comments can either be indicated by a leading "`#`" or by "`c:`". In the first case the comment is ignored by the LEED program, in the latter case it is stored by the program and written to the output file.

The meaning and the syntax of the parameter specifiers is (`n`: integer number, `f`: floating point number `c`: character):

`a1:`, `a2:`, `a3: f f f`
: Lattice vectors of the three–dimensional bulk unit cell (x, y, z in Å). a1 and a2 are parallel to the surface plane, i.e. they define the two–dimensional (1×1) unit cell. a3 must contain a component perpendicular to the surface. For the symmetrised program version a3 must not have any parallel components. In some cases (e.g. fcc lattice) a non–primitive bulk unit cell must be used in order to fulfill this condition.

`m1:`, `m2: f f`
: Super structure matrix defining the relationship between the superstructure lattice vectors b1 and b2 and the (1×1) lattice vectors a1 and a2:

$$\begin{aligned} \vec{b}_1 &= m1(1) \cdot \vec{a}_1 + m1(2) \cdot \vec{a}_2 \\ \vec{b}_2 &= m2(1) \cdot \vec{a}_1 + m2(2) \cdot \vec{a}_2 \end{aligned}$$

```
# Sample input file for
c: Ru(0001) + p(r3xr3)-2O
# bulk unit cell parameters
a1:       1.3525       -2.3426    0.0000
a2:       1.3525        2.3426    0.0000
a3:       0.0000        0.0000   -4.2800
# superstructure matrix
m1:  2. 1.
m2: -1. 1.
# symmetry
sr: 3 0.0 0.0
# Overlayers:
# O atoms on hcp site
po: O_H2O                     2.7050  0.0000  4.3100  dr3 0.05 0.05 0.05
po: O_H2O                     1.3525  2.3426  4.1900  dr3 0.05 0.05 0.05
# Ru atoms of the first layer
po: /home/CLEED/PHASE/Ru.phs  0.0000  0.0000  2.0400  dr1 0.0707
po: /home/CLEED/PHASE/Ru.phs  2.7050  0.0000  2.0500  dr1 0.0707
po: /home/CLEED/PHASE/Ru.phs  1.3525  2.3426  2.1100  dr1 0.0707
# bulk layers:
pb: /home/CLEED/PHASE/Ru.phs  0.0000 -1.5617  0.0000  dmt 400. 101. 200.
pb: /home/CLEED/PHASE/Ru.phs  0.0000  0.0000 -2.1400  dmt 400. 101. 200.
# NON-GEOMETRIC PARAMETERS:
# optical potentials
vr:   -13.00
vi:     4.50
# energies
ei: 32.
ef: 260.1
es: 4.
# angles of incidence
it: 0.
ip: 0.
# epsilon, lmax
ep: 1.e-4
lm: 8
```

Table **8.1**: Sample parameter input file for $p(\sqrt{3} \times \sqrt{3})$ $(H_2)O$ / Ru(0001).

`sr: n f f`
Rotational symmetry: degree (n–fold axis), position of axis (x, y in Å). (Ignored by the non–symmetrised program version.)

`vr: f`
Real part of optical potential (in eV).

`vi: f`
Imaginary part of optical potential (in eV).

`po: <phasestring> f f f ccc f {f f}`
Atom parameters in overlayer (super structure unit cell):
`<phasestring>`: Name of phase shift file. It can either be specified by the full path (starting with a leading slash '/', as in `/home/CLEED/PHASE/Ru.phs`) or by the file name body (no leading '/', e.g. `O_H2O`) which will then be expanded into `CLEED_PHASE/<phasestring>.phs` by using the environment variable `CLEED_PHASE`. If `CLEED_PHASE = /home/CLEED/PHASE` in the current example, the input of `O_H2O` is equivalent to `/home/CLEED/PHASE/O_H2O.phs`.
`f f f`: atom position x, y, z (in Å);
`ccc f {f f}`: three character specifier for input of vibrational displacements:

`dtm f f f`: input of Debye temperature (Θ_D in K), temperature (T in K), mass (in amu). From these values the isotropic radial root mean square displacement

$$\sqrt{<\Delta r^2>} = \frac{9}{2m\ k_B\Theta_D} \cdot \sqrt{\frac{1}{16} + \frac{T^2}{\Theta_D^2}}$$

will be calculated and used inside the program.

`dr1 f`: input of radial root mean square displacement in Å.

$$\sqrt{<\Delta r^2>} = \sqrt{<\Delta x^2> + <\Delta y^2> + <\Delta z^2>}$$

`dr3 f f f`: separate input of root mean square displacements along the coordinates $\sqrt{<\Delta x^2>}$, $<\sqrt{\Delta y^2}>$, and $<\sqrt{\Delta z^2}>$ in Å. From these values the average radial root mean square displacement $\sqrt{<\Delta r^2>}$ (see above) will be calculated and used as isotropic displacement inside the program, thus the inputs "`dr1 0.0866`" and "`dr3 0.05 0.05 0.05`" are equivalent.

`nd3 f f f`: separate input of root mean square displacements along the coordinates $\sqrt{<\Delta x^2>}$, $<\sqrt{\Delta y^2}>$, and $<\sqrt{\Delta z^2}>$ in Å (Only non–symmetric version). The displacements are used in order to set up a non–diagonal atomic t–matrix which is used in the program.

`pb: <phasestring> f f f ccc f {f f}`
Atom parameters in bulk layers ((1×1) unit cell); for syntax see `po:`.

`ei: f`
First energy in energy loop for which intensities are calculated (in eV).

`ef: f`
Last energy in loop (in eV).

`es: f`
Energy step in loop (in eV).

`it: f`
Polar angle of incidence with respect to surface normal ($0° \leq \vartheta < 90°$).

`ip: f`
Azimuthal angle of incidence with respect to x–axis ($0° \leq \varphi < 360°$).

`ep: f`
Epsilon: criterion for convergence of lattice sums and layer doubling.

`lm: n`
Maximum angular momentum quantum number (l_{max}).

The number of overlayer atoms specified by `po:` must be exactly the same as the number of atoms within one two–dimensional overlayer unit cell given by the overlayer matrix. However, they can lie in different unit cells. The bulk atoms specified by `pb:` must be exactly those within the topmost three–dimensional bulk unit cell specified by `a1`, `a2`, and `a3`. All overlayer atoms must have larger z coordinates than the top–most bulk atom. The program will produce unreliable results if the vertical distance between the top–most bulk atom and the bottom–most overlayer atom is shorter than `MIN_DIST` = 1.1 Å.

8.2 Separate bulk and overlayer parameter input

If the parameter input is split into two files, all parameters except for the overlayer atom parameters are read from the bulk parameter input file specified by the `-b` option. The overlayer atom parameters (`po:`) are read from the parameter input file (option `-i`). The two files corresponding to the above example of $p(\sqrt{3}\times\sqrt{3})$ $(H_2)O$ / Ru(0001) are shown in Tables **8.2** and **8.3**.

This way of input is used within the automated search where only the optimised `po:` parameters are supplied by the search program whereas the unchanged parameters are read from a separate bulk file provided by the user.

```
# sample bulk geometry input file
c: Ru(0001) + p(r3xr3)
# bulk unit cell parameters
a1:       1.3525       -2.3426    0.0000
a2:       1.3525        2.3426    0.0000
a3:       0.0000        0.0000   -4.2800
# superstructure matrix
m1:  2. 1.
m2: -1. 1.
# symmetry
sr: 3 0.0 0.0
#
# NON-GEOMETRIC PARAMETERS:
# optical potentials
vr:   -13.00
vi:     4.50
# energies
ei: 32.
ef: 260.1
es: 4.
# angles of incidence
it: 0.
ip: 0.
# epsilon, lmax
ep: 1.e-4
lm: 8
```

Table **8.2**: Sample bulk input file for any $p(\sqrt{3}\times\sqrt{3})$ Ru(0001).

```
# sample overlayer geometry input file
c: Ru(0001) + p(r3xr3)-2O
# Overlayers:
# O atoms on hcp site
po: O_H2O                    2.7050  0.0000  4.3100  dr3 0.05 0.05 0.05
po: O_H2O                    1.3525  2.3426  4.1900  dr3 0.05 0.05 0.05
# Ru atoms of the first layer
po: /home/CLEED/PHASE/Ru.phs 0.0000  0.0000  2.0400  dr1 0.0707
po: /home/CLEED/PHASE/Ru.phs 2.7050  0.0000  2.0500  dr1 0.0707
po: /home/CLEED/PHASE/Ru.phs 1.3525  2.3426  2.1100  dr1 0.0707
```

Table **8.3**: Sample overlayer input file for $p(\sqrt{3} \times \sqrt{3})$ $(H_2)O$ / Ru(0001).

8.3 Phase shifts input

The energy dependent atomic phase shifts $\delta_l(E)$ are read from the files specified in the atom parameter sets (`po:` and `pb:`). These files have the format as shown in Table **8.4**

```
40 9 neng, lmax (Ru, m= 101)
 0.5000
 2.2748-0.3054-0.5527 0.0180 0.0007 0.0000 0.0000 0.0000 0.0000 0.0000
 1.0000
 1.8705-0.5465-0.6358 0.1689 0.0115 0.0009 0.0001 0.0000 0.0000 0.0000
 1.5000
 1.5527-0.7538-0.7190 0.5972 0.0497 0.0056 0.0005 0.0000 0.0000 0.0000
 2.0000
 1.2909-0.9420-0.7769 1.1747 0.1270 0.0190 0.0024 0.0003 0.0000 0.0000

                    ..................................

19.0000
-1.2839-3.0896-1.8887 2.6390 1.6041 1.0384 0.6948 0.4697 0.3216 0.2165
19.5000
-1.3173-3.1180-1.9057 2.6399 1.6129 1.0490 0.7059 0.4795 0.3300 0.2243
20.0000
-1.3499-3.1458-1.9223 2.6407 1.6212 1.0593 0.7166 0.4891 0.3382 0.2319
```

Table **8.4**: Sample phase shifts input file for Ru.

The first line specifies the number of energy values and maximum angular momentum quantum number l_{max} for which phase shifts are stored in the file. The rest of the first line is treated as comment and ignored by the program. The rest of the file consists of pairs of lines containing the energy value (in Hartree) in the first and the phase shifts δ_l (in the order $l = 0, 1, 2, ..., l_{max}$) for this energy in the second line. The phase shifts are either separated by blanks or by '–' which is interpreted as sign of the following number. Inside the program the phase shifts are interpolated for each energy of the I–V curve or extrapolated, respectively, when the I–V curve surpasses the energy range of the input file. It is, however not extrapolated to energies below the lowest energy value in the input list.

Chapter 9

Output files

9.1 Results - IV curves

Table **9.1** shows a sample output file for a $p(\sqrt{3}\times\sqrt{3})$ H_2O / Ru(0001) overlayer structure.

The header of the file consits of lines beginning with '`#`'. They contain information needed e.g. by the R factor program:

`vn` program version.

`ts` creation date and time.

`en` number of energy points, first energy, last energg, energy step.

`bn` total number of beams in output files.

`bi` beam counter, 1st beam index, 2nd beam index, beam set (for each beam).

The header is followed by lines containing the energy (in eV) and the relative beam intensities in the same order as the beam list separated by blanks.

```
# ######################################## #
#            output from CLEED             #
# ######################################## #
#vn 0.20 (version GH/30.08.97)
#ts Thu Jul  2 07:18:21 1998
#
#en 58 32.000000 260.100000 4.000000
#bn 109
#bi 0 0.000000 0.000000 0
#bi 1 -1.000000 0.000000 0
#bi 2 -1.000000 1.000000 0
#bi 3 0.000000 -1.000000 0
#bi 4 0.000000 1.000000 0
#bi 5 1.000000 -1.000000 0
#bi 6 1.000000 0.000000 0
#bi 7 -2.000000 1.000000 0
#bi 8 -1.000000 -1.000000 0
#bi 9 -1.000000 2.000000 0
#bi 10 1.000000 -2.000000 0

.......................

#bi 104 -3.333333 2.666667 2
#bi 105 0.666667 -3.333333 2
#bi 106 0.666667 2.666667 2
#bi 107 2.666667 -3.333333 2
#bi 108 2.666667 0.666667 2
32.00 1.758462e-02 1.557583e-03 6.515479e-04 6.515850e-04 1.557595e-03
1.557651e-03 6.515733e-04 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
............ 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

                        ..............

260.00 1.303590e-02 7.845590e-05 5.554994e-03 5.555435e-03 7.845676e-05
7.851732e-05 5.555437e-03 8.115542e-04 7.950326e-04 7.949291e-04 8.117350e-04
............ 1.905631e-05 1.905916e-05 1.286778e-05 1.286476e-05 1.906064e-05
```

Table **9.1**: Sample output file for $p(\sqrt{3} \times \sqrt{3})$ $(H_2)O$ / Ru(0001).

Chapter 10

LEED–specific Functions

10.1 Introduction

10.2 Multiple scattering Matrix: Single Bravais layer

```
(lmsbravl.c:)
int ms_bravl       ( mat *p_Tpp, mat *p_Rpm,
                     struct var_str * v_par, struct layer_str * layer,
                     struct beam_str * beams)
(lmsbravlnd.c:)
int ms_bravl_nd    ( mat *p_Tpp, mat *p_Tmm mat *p_Rpm, mat *p_Rmp,
                     struct var_str * v_par, struct layer_str * layer,
                     struct beam_str * beams)
(lmslsumii.c:)
mat ms_lsum_ii     ( mat Llm, real k_r, real k_i, real *k_in, real *a,
                     int l_max, real epsilon )
(lmstmatii.c:)
mat ms_tmat_ii     ( mat Tii, mat Llm, mat Tl, int l_max)
(lmstmatndii.c:)
mat ms_tmat_nd_ii  ( mat Tii, mat Llm, mat Tlm_in, int l_max)
(lmsymat.c:)
mat ms_ymat        ( mat Ymat, int l_max,
                     struct beam_str *beams, int n_beams)
(lmsypy.c:)
mat ms_yp_yxp      ( mat Yxmat, mat Ymat)
mat ms_yp_yxm      ( mat Yxmat, mat Ymat)
mat ms_yp_ym       ( mat Ymmat, mat Ypmat)
```

The function `ms_bravl` calculates the multiple scattering matrix for a single Bravais layer of scatterers. According to equation (??) in Ref. [43]. This equation can be written in Matrix notation as:

$$M = \frac{1}{\kappa \cdot A} \; Y^{out\,\pm} \left\{ (1 - \tau \cdot G)^{-1} \tau \right\} Y^{in\,\pm} \tag{10.1}$$

The matrices Y^{in} and Y^{out} perform the transformation from the plane wave representation (used between the layers) into the spherical wave representation (used inside the layers). They are defined as:

$$Y^{in\,\pm}_{lm,\vec{k}} = Y^{*}_{lm}(\vec{k}^{\pm}) \tag{10.2}$$

computed by the functions `ms_ymat` and `ms_yp_ym` and

$$Y^{out\,\pm}_{\vec{k},lm} = \frac{8i}{k_z} \, Y_{lm}(\vec{k}^{\pm}) \tag{10.3}$$

computed by the functions `ms_yp_yxp` and `ms_yp_yxm`.

The expression embraced by the Y matrices describes the multiple scattering process inside the layer. G is the propagator matrix

$$G_{l_1 m_1, l_2 m_2} = \sum_{l_3 m_3} i^{l_1 - l_2} (-1)^{(l_2 - l_1 - l_3)/2 - m_1)} \cdot C_{l_1 m_1, l_3 m_3, l_2 m_2} \cdot L_{l_3 m_3} \tag{10.4}$$

with $C_{l_1 m_1, l_3 m_3, l_2 m_2}$ being a Clebsh–Gordan–coefficient and $L_{l_3 m_3}$ the lattice sum:

$$L_{l_3 m_3} = (-1)^{m_3} 4\pi \cdot Y_{l_3 m_3}(\cos\frac{\pi}{2}, 0) \cdot \sum_{\vec{R}_j} \exp[i\vec{k}_{in}\vec{R}_j - i\varphi(\vec{R}_j)] \cdot h^{(1)}_{l_3}(\kappa \cdot |\vec{R}_j|) \tag{10.5}$$

computed in the function `ms_lsum_ii`. $\vec{R}_j$ is the position of the jth atom inside the layer; $h^{(1)}_{l_3}$ the Hankle function of the first kind, and $\kappa = \sqrt{2E}$ the length of the wave vector.

τ is the atomic scattering matrix t times $-\kappa$. The functions `ms_bravl` and `ms_bravl_sym` assume isotropic scatterers with a diagonal scattering matrix which only depends on l and not on the m quantum numbers.

$$\tau_{l_1 m_1, l_2 m_2} = -\kappa \cdot t_{l_1 m_1, l_2 m_2} = e^{i\delta_{l_1}} \sin(\delta_{l_1}) \;\; \delta_{l_1 l_2} \delta_{m_1 m_2} \tag{10.6}$$

The diagonality of τ is explicitly used in the function `ms_tmat_ii` called by `ms_bravl` and `ms_bravl_sym` which computes the matrix expression

$$(1 - \tau \cdot G)^{-1} \tau \tag{10.7}$$

The function `ms_tmat_nd_ii` and the calling function `ms_bravl_nd` treat the more general case of anisotropic scatterers (e.g. anisotropic thermal vibrations) with non–diagonal scattering matrices which can also depend on m (see below).

Before performing any calculations, the function checks whether some quantities needed in the calculation can be reused from the last call of `ms_bravl`. In the nex steps the function calculate the lattice sum (if needed) by calling `ls_lsum_ii`, $(1 - t \cdot G)^{-1} t$ by calling `ls_tmat_ii`, and the transformation matrices $Y^{\pm}$ by calling `ms_ymat` first and then `ms_yp_yxp` and `ms_yp_yxm`, respectively. After multiplying the outgoing wavefunctions (Y^{+}) with $i8\pi/(|k|Ak'_{\perp})$, the reflection and transmission are put together.

10.3 Multiple scattering: combined space method

10.4 Layer Doubling

Chapter 11

Quantum Mechanical Functions

11.1 Introduction

11.2 Clebsh–Gordan Coefficients

```
(qmcgc.c:)
int mk_cg_coef(int l_max)
double cg(int l1, int m1, int l2, int m2, int l3, int m3)
double blm(int l1, int m1, int l2, int m2, int l3, int m3)
double gaunt(int l1, int m1, int l2, int m2, int l3, int m3)
```

All necessary Clebsh–Gordan coefficients (or Gaunts coefficients) up to l_{max} are calculated once by the function `mk_cg_coef`. The coefficients are stored in a list and can be read from there by using the functions `cg`, `blm`, or `gaunt`. These functions read from the same list but have return different values:

`double cg(int l1, int m1, int l2, int m2, int l3, int m3)` returns the value of the integral described in Slater's book:

$$(-1)^{m_2} \int Y_{l_1 m_1}(\Omega) Y^*_{l_2 m_2}(\Omega) Y_{l_3 m_3}(\Omega) d\Omega$$

Allowed (non–zero) values:
$0 \le l_1 \le 2 \cdot l_{max}, \;\; 0 \le l_{2,3} \le l_{max}$;
$m_1 = m_2 + m_3 \;\; \Leftrightarrow \;\; m_1 - m_2 - m3 = 0$,
$|l_2 - l_3| \le l_1 \le l_2 + l_3$;
Transformations:
$cg(l_1, m_1, l_2, m_2, l_3, m_3) = cg(l_1, m_1, l_3, m_3, l_2, m_2) = cg(l_2, m_2, l_1, m_1, l_3, -m_3)$

`double blm(int l1, int m1, int l2, int m2, int l3, int m3)` returns the value of the integral used in Pendry's book:

$$\int Y_{l_1 m_1}(\Omega) Y_{l_2 m_2}(\Omega) Y_{l_3 m_3}(\Omega) d\Omega$$

Allowed (non–zero) values:
$0 \le l_2 \le 2 \cdot l_{max}, \;\; 0 \le l_{1,3} \le l_{max}$;
$m_1 + m_2 + m_3 = 0$,
$|l_1 - l_3| \le l_2 \le l_1 + l_3$;
Transformations:
$blm(l_1, m_1, l_2, m_2, l_3, m_3) = cg(l_2, -m_2, l_1, m_1, l_3, m_3)$

`double gaunt(int l1, int m1, int l2, int m2, int l3, int m3)` returns Gaunt's integral:

$$(-1)^{m_3} \int Y_{l_1 m_1}(\Omega) Y_{l_2 m_2}(\Omega) Y^*_{l_3 m_3}(\Omega) d\Omega$$

Allowed (non–zero) values:
$0 \le l_2 \le 2 \cdot l_{max}, \;\; 0 \le l_{1,3} \le l_{max}$;
$m_3 = m_1 + m_2 \;\; \Leftrightarrow \;\; m_1 + m_2 - m_3 = 0$,
$|l_2 - l_3| \le l_1 \le l_2 + l_3$;
Transformations:
$gaunt(l_1, m_1, l_2, m_2, l_3, m_3) = cg(l_2, m_2, l_1, -m_1, l_3, m_3)$

The memory requirements of the list created by `mk_cg_coef` are for a given value of l_{max}:

$$(2 \cdot l_{max} + 1)(2 \cdot l_{max} + 2)/2 \cdot (l_{max} + 1)^2 (l_{max}/2 + 1) \cdot \;\texttt{sizeof(double)}$$

l_{max}	memory (bytes)
6	142,688
8	495,720
10	1,341,648
12	3,075,800
14	6,264,000
16	11,673,288

11.3 Spherical Harmonics

```
(qmylm.c:)
mat r_ylm( mat Ylm, real x, real phi, int l_max )
mat c_ylm( mat Ylm, real z_r, real z_i, real phi, int l_max )
(lmsymat.c:)
(lmsypy.c:)
```

11.4 Spherical Hankle Functions

```
(qmhank.c:)
mat r_hank1 ( mat Hl, real x, int l_max )
mat c_hank1 ( mat Hl, real z_r, real z_i, int l_max )
```

Chapter 12

Matrix Functions

12.1 Introduction

In contrast to the conventional definition of matrices in C as arrays of pointers to arrays of numbers (integer, float, etc.), in our programs matrices are structures `struct mat_str` (defined in `"mat.h"`, see table **12.1**). This has the advantage that all relevant information such as matrix dimensions (`rows,cols`) and the type of matrix elements (`num_type`) can be exchanged between functions together with the matrix elements themselves through a single variable of type `mat`, which is a pointer to this structure (`typedef struct mat_str * mat` in `"mat.h"`).

```
struct mat_str     /* real or complex matrix */
{
 int mag_no;       /* magic number */
 int mat_type;     /* type of matrix (square, etc.) */
 int num_type;     /* type of matrix elements (read, complex) */
 int rows;         /* 1st dimension of matrix (number of rows) */
 int cols;         /* 2nd dimension of matrix (number of columns) */
 real *rel;        /* pointer to real parts of matrix elements */
 real *iel;        /* pointer to imag. parts of matrix elements */
};
```

Table **12.1**: Matrix structure

In order to make the matrix type independent of the type of the matrix elements, complex matrices are effectively stored as two matrices, one containing the real part of the elements (`*rel`) and one containing the imaginary part (`*iel`). For real matrices

`iel` is a null pointer, in addition, the structure element `num_type` is an integer number representing the type of the matrix elements.

The structure element `mat_type` is another integer number representing the matrix type (rectangular, square, diagonal) which decides upon the storage scheme of the matrix elements and various other things. For example, the matrix inversion routine checks first of all, if the matrix to be inverted is square at all, and sends an error message if not.

The storage scheme for rectangular and square matrices is:

$$\Re[M(m,n)] \rightarrow *(\texttt{rel} + (\texttt{m} - 1) * \texttt{rows} + \texttt{n}) \tag{12.1}$$

$$\Im[M(m,n)] \rightarrow *(\texttt{iel} + (\texttt{m} - 1) * \texttt{rows} + \texttt{n}) \tag{12.2}$$

diagonal matrices are stored as:

$$\Re[M(m,m)] \rightarrow *(\texttt{rel} + \texttt{m}) \tag{12.3}$$

$$\Im[M(m,m)] \rightarrow *(\texttt{iel} + \texttt{m}) \tag{12.4}$$

Note that the first element in the arrays `rel` and `iel` is never used, therefore these arrays have the length $[(n \cdot m + 1) \cdot \texttt{sizeof(real)}]$ for a rectangular or square $n \times m$ matrix and $[(n+1) \cdot \texttt{sizeof(real)}]$ for a diagonal matrix.

The general format for matrix functions which have a matrix as results is:

```
(mat) result = matfunction( (mat) storage, operand_1, operand_2, ...)
```

The first parameter of the function call (`storage`) is usually a matrix where the result will be written to (type `mat`, i.e. a pointer to a matrix structure). If the dimensions or matrix type of this matrix do not match with those of the result, the necessary adjustments in memeory will be done automatically. In particular, if `storage` is the null–pointer (`NULL`), a new matrix will be created. In any case, the return value `result` is a pointer to this updated matrix which is not necessarily identical with `storage`. The types of the other parameters (operands) depend on the purpose of the particular matrix function. They can be either matrices or single numbers.

12.2 Basic Matrix Functions

The matrix functions described in this section are mainly called by other "high level" matrix functions in order to check, allocate, free, etc. memory space used to store a matrix. They are listed in table **12.2**.

function name	description	source file
`mat matalloc(mat, int, int, int)`	allocate memory for matrix	matalloc.c
`int matcheck(mat)`	check validity of pointer	matcheck.c
`mat matcop(mat, mat)`	copy matrices	matcop.c
`int matfree(mat)`	free memory	matfree.c

Table **12.2**: Basic matrix functions

12.2.1 `matalloc, matfree`

12.2.2 `matcheck`

12.2.3 `matcop`

12.3 Matrix Inversion (`matinv`)

12.4 Display and Control Functions for Matrices

The matrix functions described in this section are not thought to be used in the "production state" of a LEED program since they produce very large outputs. During the development of LEED (or other electron scattering) programs it is however useful, to display the contents of matrices in the right format. Two functions are available for this purpose. `int matshow(mat)` displays the real and imaginary part of complex matrix elements while `int matshowabs(mat)` diplays only the modulus of complex matrix elements and therefore reduces the amount of output. For example the matrix:

$$\begin{pmatrix} 1 & 1+i \\ 1-i & i \end{pmatrix}$$

would be displyed by `matshow` as:

```
(2 rows) x (2 columns):
(1.00, 0.00)  (1.00, 1.00)
(1.00,-1.00)  (0.00, 1.00)
```

while `matshowabs` would produce the following output:

```
*** abs *** (2 rows) x (2 columns):
1.00  1.41
1.41  1.00
```

Note that both functions also display the matrix dimensions which is particularly useful, when the rows are too long to be diplayed in one line of the screen.

Part III

The R factor Program

Chapter 13

General outline

13.1 Program description

The R factor program performs the comparison of calculated and experimental IV curves. The agreement is quantified in terms of the R(eliability) factor. It offers the choice between four different R factors that can be used for the search: R_1, R_2 [**?**], R_P [53], and R_B [54]. The output R factor value is the optimum achieved by shifting the energy axes of experimental and theoretical I–V curves with respect to each other. This shift acts as a correction for any non–optimum value of the optical potential, V_{0r}, in the LEED calculations, which need therefore not be optimized by the search program. In this way, one dimension is eliminated from the search parameter space on which the search program operates, hence reducing the number of LEED calculations to be performed. The assignment of experimental and theoretical I–V curves (or average of curves) for comparison, and of the relative weight that a particular I–V curve has in the overall R factor, is performed by the user prior to the search.

13.2 Syntax

The general calling syntax of the R factor program is:

```
crfac -a <ID flag>  -c <control file>   -h    -o <output file>
      -r <R factor> -s <shift1,shift2,shift3> -t <theoretical file>
      -v <optical potential>  -w <IV output file>
```

The options `-c <control file>` and `-t <theoretical file>` are mandatory for normal use of the program.

`-a <ID flag>` defines whether only the average R factor is calculated (argument `average`, default) or partial R factors for each subset of IV curves sharing a common ID number (argument `all`). Only the first two characters of the argument are significant.

`-c <control file>` specifies the control file which defines the correlation between experimental and theoretical IV curves. A sample control file is shown in Table **14.1**.

`-h` causes the program to show a short list of arguments.

`-o <output file>` specifies the output file where the R factor values are written to (default: standard output).

`-r <R factor>` specifies the R factor to be calculated. Valid arguments are: `r1` (R_1), `r2` (R_2), `rb` (R_{B1} and R_{B2}), and `rp` (R_P, default)

`-s <shift1,shift2,shift3>` defines the range (arguments `shift1` and `shift2`) and stepwidth (`shift3`) of energy shifts between experimental and theoretical IV curves.

`-t <theoretical file>` specifies the file containing the theoretical IV curves, i.e. the results file from the LEED program.

`-v <optical potential>` specifies the value of the optical potential V_{0i} used in the evaluation of Pendry's R–factor (default: 4 eV).

`-w <IV output file>` causes the program to write all normalised IV curves as energy/intensity pairs to separate files so that they can be plotted. `<IV output file>` specifies the body of the file names to which the letters `e` (experimental) or `t` (theoretical) and the number of the pair of curves is added.

13.3 UNIX–Environment

The program has been developed in a UNIX environment and uses a few UNIX specific features. The most important is the use of environment variables:

`RF_HELP_FILE`: file to be shown, when the `-h` option is chosen.

This variable has to be set using the `export` or `setenv` UNIX commands, respectively before the program is called for the first time.

Chapter 14

Input / output files

14.1 Control file

Table **14.1** shows an example R factor control file for a $p(1 \times 1)$ Er / Si(111) structure. Each line defines the correlation of one pair of theoretical and experimental IV curves using the following syntax:

`ef=<expt. file>:ti=<index list>:id=<ID number>:wt=<weight>`

`<expt. file>` contains one experimental IV curve in the format:

energy 1	intensity 1
energy 2	intensity 2
...	...

Lines beginning with a `#` and blank lines are ignored.

`<index list>` is a list of indices of beams to be averaged and compared with the experimental IV curve. The two indices of each beam are in round brackets and can be preceeded by a weighting factor followed by $*$. Different beams are connected through `+`. The general form is:

`{<weight>}*(<index_1>,<index_2>){+{<weight>}*(<ind_1>,<ind_2>)}`

`<ID number>` is a integer number attached to this pair of IV curves. When the option `-a all` is specified, the partial R factors will be calculated for all groups of IV curves sharing the same ID numbers.

`<weight>` is a positive real number defining the weight of this pair of IV curves when calculating the average R factor. This weight (default: 1) will be multiplied with the fraction of the total energy range covered by this pair of IV curves.

```
#  Si(111)-Er (1x1)
#  (control file for R factor program CRFAC)
#
#  Theor. indices are listed for 3-fold rot. symmetry output
#  and a calculation up to 250 eV. Delete if no experimental
#  data are available for certain spots.
#
#
#  ef=<experimental input file>
#  ti=<corresponding indices in theoretical input file>
#  id=<group ID> (does not really matter)
#  wt=<weight>
#  : = separator
#
#
ef=Si111_Er_expt_01:ti=(0.00,1.00):id=01:wt=1.
ef=Si111_Er_expt_02:ti=(1.00,0.00):id=02:wt=1.
ef=Si111_Er_expt_03:ti=(1.00,1.00)+(2.00,-1.00):id=03:wt=1.
ef=Si111_Er_expt_04:ti=(0.00,2.00):id=04:wt=1.
ef=Si111_Er_expt_05:ti=(2.00,0.00):id=05:wt=1.
ef=Si111_Er_expt_06:ti=(-1.00,3.00)+(1.00,2.00):id=06:wt=1.
ef=Si111_Er_expt_07:ti=(3.00,-1.00)+(2.00,1.00):id=07:wt=1.
ef=Si111_Er_expt_08:ti=(0.00,3.00):id=08:wt=1.
ef=Si111_Er_expt_09:ti=(3.00,0.00):id=09:wt=1.
ef=Si111_Er_expt_10:ti=(2.00,2.00)+(4.00,-2.00):id=10:wt=1.
ef=Si111_Er_expt_11:ti=(-1.00,4.00)+(1.00,3.00):id=11:wt=1.
ef=Si111_Er_expt_12:ti=(4.00,-1.00)+(3.00,1.00):id=12:wt=1.
ef=Si111_Er_expt_13:ti=(0.00,4.00):id=13:wt=1.
ef=Si111_Er_expt_14:ti=(4.00,0.00):id=14:wt=1.
```

Table **14.1**: Sample parameter input file for $p(1 \times 1)$ Er / Si(111).

14.2 Output

A typical output line is shown below:

```
0.318127 0.185645 3.00 928.50          #  Rp  RR  shift  range
```

The first value is the actual R factor, followed by Pendry's RR factor defining the statistical error. The third number is the energy shift between the experimental and theoretical IV curves, the last value is the total overlaqp (in eV) between experimental and theoretical IV curves. These values are followed by a comment specifying them.

Part IV

The SEARCH Program

Chapter 15

Program description

The search program performs the structure optimisation. It uses standard optimisation algorithms such as the downhill simplex method [48, 49], Powell's method [48, 50], simulated annealing [48] and the genetic algorithm [51, 52] which can be selected by the user in order to perform the search for the best fit geometry. While the first two algorithms are strictly downhill–oriented, i.e. will find only the nearest local R factor minimum, the latter two algorithms should in principle provide a means of locating the global R factor minimum within the given constraints, at the expense of many more search steps. Within each search step a set of geometrical parameters is chosen by the algorithm depending on the R factor values achieved before and in accordance with user–specified symmetry constraints. The parameters are written to a file serving as input for the LEED program whose output is then fed into the R factor program in order to calculate an R factor value for this parameter set.

Chapter 16

General outline

16.1 Syntax

The general calling syntax of the search program is:

```
csearch -i <parameter file> -d <initial displacement>
        -s <search type>    -v <vertex file>
```

`-i <parameter file>` specifying the parameter input file is the only mandatory option. The file contains all the geometric and non–geometric parameters needed for the search. A sample file is shown in Table **17.1**. The parameter file name without extension is used as "project name" by the program in order to create names for output files and the control file of the R factor program.

`-d <initial displacement>` specifies the initial displacements from the start geometry for all parameters.

`-s <search type>` specifies the search algorithm to be used for the structure optimisation. Possible arguments are: `ga` (genetic algorithm [**?**], not implemented yet), `po` (Powells method [**?**]), `sa` (Simulated annealing [**?**]), and `si`, `sx` (both simplex algorithm [**?**], default).

`-v <vertex file>` allows to restart the search with the current simplex, if the simplex algorithm is used. The argument `<vertex file>` is the `*.ver` file produced by the program.

16.2 UNIX–Environment

The program has been developed in a UNIX environment and uses a few UNIX specific features. The most important is the use of environment variables:

`CSEARCH_LEED`: name of the program used for the LEED calculations.

`CSEARCH_RFAC`: name of the program used for the R factor evaluation.

These variables have to be set using the export or setenv UNIX commands, respectively before the program is called for the first time.

Chapter 17

Input files

17.1 Parameter Input

Table **17.1** shows an example input file for a $p(1 \times 1)$ Er / Si(111) overlayer structure.

Each parameter or set of parameters, respectively, is specified by two letters and a colon at the beginning of a line. There can only be one parameter specifier per line. Comments can either be indicated by a leading "`#`" or by "`c:`". In the first case the comment is ignored by the LEED program, in the latter case it is stored by the program and written to the output file.

The meaning and the syntax of the parameter specifiers is (`n`: integer number, `f`: floating point number `c`: character):

`a1:`, `a2: f f f`
: Lattice vectors of the two–dimensional (1×1) unit cell (x, y, z in Å). a1 and a2 have to be parallel to the surface plane.

`m1:`, `m2: f f`
: Super structure matrix defining the relationship between the superstructure lattice vectors b1 and b2 and the (1×1) lattice vectors a1 and a2:

$$\begin{aligned} \vec{b}_1 &= m1(1) \cdot \vec{a}_1 + m1(2) \cdot \vec{a}_2 \\ \vec{b}_2 &= m2(1) \cdot \vec{a}_1 + m2(2) \cdot \vec{a}_2 \end{aligned}$$

`po: <phasestring> f f f ccc f {f f}`
: Atom parameters in overlayer (super structure unit cell): same syntax as for the LEED program, see chapter 8.

`rm: <phasestring> f`
: minimum radius of atoms specified by `<phasestring>`. If the distance between two

```
# input file for SEARCH
#  Si(111)-Er (1x1)
#  atomic positions according to Spence et al. Phys. Rev. B 61 (2000) 5707.
#
#
a1:      1.9162240  -3.3189974   0.0
a2:      1.9162240   3.3189974   0.0
#
m1:  1. 0.
m2:  0. 1.
#
# bulk: (see *.bul)
# pb: Si   0.0         2.2126649  -0.7822951  dr3 0.05  0.05  0.05
# pb: Si   0.0         0.0         0.0        dr3 0.05  0.05  0.05
#
# overlayer:
#
po: Si   0.0         0.0        +2.353      dr3 0.05  0.05  0.05
po: Si   0.0        -2.2126649  +3.153      dr3 0.05  0.05  0.05
po: Ru   0.0         0.0        +5.453      dr3 0.05  0.05  0.05
po: Si   0.0        +2.2126649  +7.233      dr3 0.05  0.05  0.05
po: Si   0.0        -2.2126649  +8.053      dr3 0.05  0.05  0.05
#
# minimum radii:
#
rm: Si 1.00
rm: Ru 1.00
# z range
zr: 1.00  9.00
# sz: 0 (xyz search), 1 (z only)
sz: 1
# sr: rotational axis
sr: 3 0.0 0.0
```

Table **17.1**: Sample parameter input file for $p(1 \times 1)$ Er / Si(111)

atoms is smaller than the sum of their miniumum radii, an additional number is added to the actual R factor in order to repell the search from this un–physical geometry.

`zr: f f`

z–range. The atoms are forced to stay within this range of z coordinates. If an atom exceeds this range, an additional number is added to the actual R factor in order to repell the search from this un–physical geometry.

`sz: n`

variation of vertical parameters (1) or vertical and lateral parameters (0) in agreement with the specified symmetry.

`sr: n f f`

Rotational symmetry: degree (n–fold axis), position of axis (x, y in Å).

17.2 Bulk parameters

`<project>.bul`. See chapter 8

17.3 R factor control file

`<project>.ctr`. See chapter 14

Chapter 18

Output files

18.1 Log file

A protocol of the search is stored in the file `<project>.log`, where `<project>` is the name of the parameter input file without extension. The log file starts with a list of atom positions in the start geometry followed by a listing of the parameters and the corresponding R factor values of each search step.

18.2 LEED files

The input parameter file for the LEED program is called `<project>.par`. In order to call the LEED program successfully, there must also exist a file called `<project>.bul` containing the bulk parameters, i.e. the geometrical and non–geometrical parameters which are not varied within the structure search. The calculated IV curves will be stored in `<project>.res`. `<project>.out` contains the control output from the LEED program. The input parameter file and the calculated IV curves of the current best fit geometry are copied to `<project>.pmin` and `<project>.rmin`, respectively.

18.3 R factor files

The output from the R factor program is written to the file `<project>.dum`, which contains only one line with the current R factor value.

18.4 Vertex file

If the simplex method is used, the vertices of the simplex are stored in the file `<project>.ver` after each change of the simplex. The file `<project>.vbk` contains a backup of the iteration step before.

Part V

Auxiliary Programs

Chapter 19

Pattern

Chapter 20

Lesen (Extract IV curves)

Chapter 21

Plot Geometry

Bibliography

[1] C. Davisson, L. H. Germer, Nature (London) 119 (1927), 558.

[2] C. Davisson, L. H. Germer, Phys. Rev. 29 (1927), 908.

[3] C. Davisson, L. H. Germer, Phys. Rev. 30 (1927), 705.

[4] J. B. Pendry, "Low Energy Electron Diffraction", Academic Press, London, 1974.

[5] M. A. Van Hove and S. Y. Tong, "Surface Crystallography by LEED", Springer, Berlin, 1979.

[6] M. A. Van Hove, W. H. Weinberg, C.-M. Chan, "Low-Energy Electron Diffraction", Springer, Berlin, 1986

[7] L. J. Clarke, "Surface Crystallography - An Introduction to Low Energy Electron Diffraction", Wiley, Chichester, 1985.

[8] U. Scheithauer, G. Meyer, M. Henzler, Surf. Sci. 178 (1986) 441.

[9] R. L. Park, J. E. Houston, D. G. Schreiner, Rev. Sci. Instrum. 42 (1971) 60.

[10] E. A. Wood, J. Appl. Phys. 35 (1964) 1306.

[11] M. Henzler, in "Electron Spectroscopy for Surface Analysis", Ed. H. Ibach, Springer Berlin (1977) 117.

[12] M. Henzler, W. Göpel, "Oberflächenphysik des Festkörpers", Teubner, Stuttgart, 1991.

[13] P. R. Watson, M. A. Van Hove, K. Hermann, "NIST Surface Structure Database" (SSD) Version 3.0, NIST, Gaithersburg, 1999.

[14] H. Pfnür, P. Piercy, Phys. Rev. B40 (1989) 2515.

[15] H. Pfnür, P. Piercy, Phys. Rev. B41 (1990) 582.

[16] G. Hoogers, D. A. King, Surf. Sci. 286 (1993) 306.

[17] E. Bauer, H. Poppa, Y. Visvanath, Surf. Sci. 58 (1976) 517.

[18] C. Zhang, M. A. Van Hove, G. A. Somorjai, Surf. Sci. 149 (1985) 326.

[19] S. Fölsch, U. Barjenbruch, M. Henzler, Thin Solid Films 172 (1989) 123.

[20] H. Neureiter, S. Spranger, M. Schneider, U. Winkler, M. Sokolowski, E. Umbach, Surf. Sci. 388 (1997) 186.

[21] M. Sokolowski, H. Neureiter, M. Schneider, E. Umbach, S. Tatarenko, Appl. Surf. Sci., 123–124 (1998) 71.

[22] H. Neureiter, S. Schinzer, M. Sokolowski, E. Umbach, Journal of Crystal Growth, 201–202, (1999), 93.

[23] J. B. Pendry, J. Phys. C 13 (1980) 937.

[24] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, "Numerical Recipes in C", Cambridge University Press, Cambridge, 1988.

[25] D. E. Goldberg "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, Mass., 1989.

[26] K. Heinz, U. Starke, F. Bothe, Surf. Sci. Lett. 243 (1991) L70.

[27] K. Heinz, U. Starke, M. A. Van Hove, G. A. Somorjai, Surf. Sci. 261 (1992) 57.

[28] E. Lang, P. Heilmann, G. Hanke, K. Heinz, K. Müller, Appl. Phys. 19 (1979) 287.

[29] G. Held, S. Uremovic, C. Stellwag, D. Menzel, Rev. Sci. Instrum. 67 (1996) 378.

[30] P. J. Rous, J. B. Pendry, Surf. Sci. 219 (1989) 355.

[31] P. J. Rous, J. B. Pendry, Surf. Sci. 219 (1989) 373.

[32] P. J. Rous, Prog. Surf. Sci. 39 (1992) 3.

[33] LEED I-V program packages are distributed by:
G. Held: georg.held@chemie.uni-erlangen.de.
V. Blum, K. Heinz: Comp. Phys. Comm. 134 (2001) 392.
M. A. Van Hove: vanhove@lbl.gov.

[34] K. Heinz, Surf. Sci. 299/300 (1994) 433.

[35] K. Heinz, S. Müller, L. Hammer, J. Phys.: Condens. Matter 11 (1999) 9437.

[36] H. Over, Prog. Surf. Sci. 58 (1998) 249.

[37] A. Kahn, Surf. Sci. 299/300 (1994) 469.

[38] W. Braun, G. Held, H.-P. Steinrück, C. Stellwag, D. Menzel, Surf. Sci. 475 (2001) 18.

[39] M. Henzler, W. G"opel, *Oberf"achenphysik des Festk"orpers*, Teubner Stuttgart, 1991.

[40] D. P. Woodruff, T. A. Delchar, *Modern techniques in surface science*, Cambridge University Press, 1989.

[41] G. A. Somorjai, *Chemistry in two Dimensions*, Cornell University Press, Ithaca and London, 1981.

[42] C. Davisson and L. H. Germer, *Phys. Rev.* **30** (1927), 705.

[43] J. B. Pendry, *Low Energy Electron Diffraction*, Academic Press London and New York, 1974.

[44] M. A. Van Hove, S.Y. Tong, *Surface Crystallography by LEED*, Springer–Verlag Berlin, Heidelberg, New York, 1979.

[45] M. A. Van Hove, W. H. Weinberg, C. M. Chan, *Low-Energy Electron Diffraction*, Springer Series in Surface Science 6, 1986.

[46] G. Ertl, J. K"uppers, *Low Energy Electrons and Surface Chemistry*, 2. Auflage, VCH Weinheim, 1985.

[47] J.M. Cowley, *Diffraction Physics*, 2.Auflage, North–Holland Amsterdam, 1986.

[48] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988.

[49] J. A. Nedler and R. Mead, *Computer Journal* **7**, (1965), 308.

[50] R. P. Brent, *Algorithms for Minimization without Derivatives*, Englewood Cliffs, NJ: Prentice–Hall, 1973, chapter 7.

[51] J. H. Holland, *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.

[52] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley, Reading, Mass., 1989.

[53] J. B. Pendry, *J. Phys.* **C 13**, (1980), 937.

[54] G. Held, H. Pfnür and D. Menzel, *Surf. Sci.* **271** (1992), 21.

[55] P. J. Rous and J. B. Pendry, *Surf. Sci.* **219** (1989), 355.

[56] P. J. Rous, M. A. Van Hove, and G. A. Somorjai, *Surf. Sci.* **226** (1990), 15.

[57] A. Wander, J. B. Pendry, M. A. Van Hove, *Phys. Rev.* **B 46** (1990), 9897.

[58] V. L. Moruzzi, J. F. Janak, A. R. Williams, *Calculated Electronic Properties of Metals*, Pergamon Press, New York, 1978.

[59] M. A. VanHove, S. Y. Tong et. al., *LEEDTP – Sammlung von Computerprogrammen zur Berechnung von LEED–IV–Kurven und verwandten Problemen*, Stand:1990.

[60] J. M. MacLaren, J. B. Pendry, P. J. Rous, D. K. Saldin, G. A. Somorjai, M. A. Van Hove, D. D. Vvedensky, *Surface Crystallographic Information Service – A Handbook of Surface Structures*, Reidel Publishing Company, Dordrecht, Holland, 1987.

[61] P. R. Watson, M. A. Van Hove and K. Hermann, *NIST Surface Structure Database (SSD)*, U. S. Department of Commerce Gaithersburg, 1993.

[62] Landold B"ornstein, *Zahlenwerte und Funktionen aus Physik, Chemie, Astronomie, Geophysik und Technik*, Springer Berlin, ab 1951.

[63] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover Publications New York, 1972.

[64] I. N. Bronstein, K. A. Semendjajew, *Taschenbuch der Mathematik*, Frankfurt/Main, 1981.

[65] Rupert Lasser, *Funktionalanalysis I,II*, Technische Universit"at M"unchen, 1985.

[66] K. K"onigsberger, *Analysis I–III*, Technische Universit"at M"unchen, 1981–1985.

[67] F. Reinhardt und H. Soeder, *dtv–Atlas zur Mathematik* , dtv M"unchen, 1982.