The TB-LMTO-ASA program.

G. Krier, O. Jepsen, A. Burkhardt, and O. K. Andersen.
*Max-Planck-Institut für Festkörperforschung, Heisenbergstr. 1,
D-70569 Stuttgart, Federal Republic of Germany .*

This document was written for the course 'Hands-on the TB-LMTO-ASA computer program' held at MPI-FKF in Stuttgart, October 24-28 1994, and organized by the authors. It describes how to install and run the TB-LMTO program version 45 under UNIX.

The development of this program has been managed by Ole Krogh Andersen; the first version was constructed mostly by Mark van Schilfgaarde during 1987-88. Over the years the following people (in alphabetic order) have written, corrected or made additions to the program: Vladimir Antonov, Peter Braun, Armin Burkhardt, Volker Eyert, Ove Jepsen, Georges Krier, Toni A. Paxton, Andrei Postnikov, and Andreas Savin. The latest major revisions and additions are due to the authors of this document.

For use of this program see the Copyright agreement in Section XI.

Version 45 was last updated September 1994.

This documentation was written in LaTeX2e and can be printed with version 3.0 of the REVTEX macros.

The linear muffin-tin orbital (LMTO) method has been described in numerous publications. For a description of the LMTO method in general see Ref. 1, the tight-binding (TB) version see Ref. 2, the down-folding technique Ref. 3, the tetrahedron method Ref. 4, accuracy using overlapping spheres Ref. 5, and a detailed example Ref. 6. Some usefull books are also worth mentioning: *The Mathematical Theory of Symmetry in Solids*[7] contains among other things all the Brillouin zones, *International Tables for Crystallography*[8] contains all crystallographic space groups and *Pearson's Handbook of Crystallographic Data for Intermetallic phases*[9] contains structural data for many compounds and elements.

Contents

I. INSTALLATION OF THE TB-LMTO PROGRAMS

The source code is compressed on the distribution diskette from where this documentation was downloaded.

The uncompression, creation of directories, and copying to directories is done automatically as described in the READ.ME file on the distribution disk. This can be done under MS-DOS or UNIX.

The GNUPLOT graphical package on the distribution diskette can be used for all plots, see Section X.

Options are provided for creation of plot data files appropriate for IBM Visualization Data Explorer version 2, which is available for IBM, Sun, HP, and SGI machines.

The path name in the *makefile* and in all files with the extension *.exec* in the lmto45 directory (hereafter called the main directory, and directories one level below this will be called subdirectories) has to be changed to the appropriate name. Furthermore, the path name in *getmain* and in *getmakefile* in the *utilities* subdirectory should also be changed. Typing 'make all' will then compile and link all programs according to the rules in the *makefile.*

II. EXECUTION OF THE TB-LMTO PROGRAMS

1. The basic input data is inserted in the CTRL file. The CTRL file shown below contains the minimum amount of information to be typed, in order to run the TB-LMTO programs. The rest may be obtained from the default values in the program, which are written into the CTRL file by the program.

```
HEADER  NiSe2 Pyrit (Acta Chem. Scand.
                         v. 23 p. 2325 1969)
VERS    LMASA-46
STRUC   ALAT=11.2683
        PLAT=1 0 0
             0 1 0
             0 0 1
CLASS   ATOM=NI Z=28
        ATOM=Se Z=34
SITE    ATOM=NI POS= .0   .0   .0
        ATOM=NI POS= .5   .0   .5
        ATOM=NI POS= .0   .5   .5
        ATOM=NI POS= .5   .5   .0
        ATOM=Se POS= .383  .383  .383
        ATOM=Se POS= .617  .617  .617
        ATOM=Se POS= .117  .617  .883
        ATOM=Se POS= .883  .383  .117
        ATOM=Se POS= .617  .883  .117
        ATOM=Se POS= .383  .117  .883
        ATOM=Se POS= .883  .117  .617
        ATOM=Se POS= .117  .883  .383
```

The input consists of tokens, like ATOM= or Z=. These are grouped together in categories, like CLASS. The name of a category always starts in the first column of a line. This will be explained in detail in Section V. ALAT= is a lattice constant in a.u. (i.e. Bohr radii), PLAT= are the lattice translation vectors and POS= are the atomic positions. The latter two are in Cartesian coordinates and in units of ALAT.

All Ni atoms and all Se atoms are equivalent i.e. there are two classes. From the information in the CTRL file, the program will find the symmetry i.e. all space group operations and the generators of the space group. Alternatively to the CTRL file above, the symmetry generators and only one of each of the inequivalent atoms could have been given. The program will complete the basis:

```
HEADER  NiSe2 Pyrit (Acta Chem. Scand.
                     v. 23 p. 2325 1969)
VERS    LMASA-46
STRUC   ALAT=11.2683
        PLAT=1 0 0
             0 1 0
             0 0 1
CLASS   ATOM=NI Z=28
        ATOM=Se Z=34
SITE    ATOM=NI POS= .0  .0  .0
        ATOM=Se POS= .383  .383  .383
SYMGRP  NGEN=3 GENGRP=I R3D MZ:(.5,0,.5)
```

The category SYMGRP contains three tokens: NGEN= the number of generators supplied, GENGRP= the names of the generators as defined in the Subsection 'Category SYMGRP'.

A third possibility is to use the interactive program *lminit.run* in the main directory. An example is shown below, where the user's answer to the query is indicated by the string after '–>' and followed by '(Answer)'.

```
| QUERY : SYMBOL OR NUMBER OF SPACE GROUP
|
| Enter: 'nnn': nnn is the new SYMBOL value,
|          'a' to abort program execution.
|        ->205 (Answer)

 Space group: Pa-3       No.:205
 Crystal system: cubic

| QUERY : atomic-units ? (else in Angstrom)
|
| Enter: 'nnn': nnn is the new value for
|              atomic-units,
|              default value=T
|         'a' to abort program execution.
|         '/' to accept default value
|        ->T (Answer)

  Now enter lattice parameters...

| QUERY : a :lattice parameter
|
| Enter: 'nnn': nnn is the new a value,
|         'a' to abort program execution.
|        ->11.2683 (Answer)

LATPAR:             PLAT        ALAT=11.2683
       1.00000    .00000    .00000
        .00000   1.00000    .00000
        .00000    .00000   1.00000

| QUERY : Z = nuclear charge of class 1
|
| Enter: 'nnn': nnn is the new Z value,
|         'a' to abort program execution.
|         'q' to quit
|        ->28 (Answer)

| QUERY : CLABL = label of class 1
|
| Enter: 'nnn': nnn is the new CLABL value
|              default value=Ni
|         'a' to abort program execution.
|         '/' to accept default value
|        ->NI (Answer)

| QUERY : Z = nuclear charge of class 2
|
| Enter: 'nnn': nnn is the new Z value,
|         'a' to abort program execution.
|         'q' to quit
|        ->34 (Answer)

| QUERY : CLABL = label of class 2
|
| Enter: 'nnn': nnn is the new CLABL value
|              default value=Se
|         'a' to abort program execution.
|         '/' to accept default value
|        ->SE (Answer)

| QUERY : Z = nuclear charge of class 3
|
| Enter: 'nnn': nnn is the new Z value,
|         'a' to abort program execution.
|         'q' to quit
|        ->q (Answer)

| QUERY : POS1 = position of NI
|
| Enter: 'nnn': nnn is the new POS1 value
|              default value=0. 0. 0.
|         'a' to abort program execution.
|         '/' to accept default value
|        ->/ (Answer)

| QUERY : POS1 = position of SE
|
| Enter: 'nnn': nnn is the new POS1 value
|              default value=0. 0. 0.
|         'a' to abort program execution.
|         '/' to accept default value
|        ->.383 .383 .383 (Answer)

 ADDBAS: The basis has been extended from
         2 to 12 atoms.
         The new positions are:

        ATOM=NI   POS=    .500  .500  .000
        ATOM=NI   POS=    .000  .500  .500
        ATOM=NI   POS=    .500  .000  .500
        ATOM=SE   POS=   -.383 -.383 -.383
        ATOM=SE   POS=    .117  .883  .383
        ATOM=SE   POS=    .883  .117 -.383
        ATOM=SE   POS=    .383  .117  .883
        ATOM=SE   POS=    .883  .383  .117
        ATOM=SE   POS=   -.383  .883  .117
        ATOM=SE   POS=    .117 -.383  .883

 SHOSYM: Bravais system : cubic
         Bravais lattice: cubic primitive
         Centring type  : P
         Crystal family : cubic
         Crystal system : cubic
         Point group    : m-3
         Space group    : Pa-3      (205)
```

The resulting CTRL file looks like this:

```
HEADER   Ni4Se8, cubic primitive
VERS     LMASA-46
SYMGRP   NGEN=3 GENGRP=I R3D MX:(.5,.5,0)
         SPCGRP=Pa-3
STRUC    ALAT=11.2683
         PLAT=1 0 0
              0 1 0
              0 0 1
CLASS    ATOM=NI Z=28 R=2.558
         ATOM=SE Z=34 R=3.214
SITE     ATOM=NI POS=0.000 0.000 0.000
         ATOM=NI POS=0.500 0.500 0.000
         ATOM=NI POS=0.500 0.000 0.500
         ATOM=NI POS=0.000 0.500 0.500
         ATOM=SE POS=0.383 0.117 -.117
         ATOM=SE POS=-.383 -.117 0.117
         ATOM=SE POS=-.117 0.383 0.117
         ATOM=SE POS=0.117 -.383 -.117
         ATOM=SE POS=0.117 -.117 0.383
         ATOM=SE POS=-.117 0.117 -.383
         ATOM=SE POS=0.383 0.383 0.383
         ATOM=SE POS=-.383 -.383 -.383
```

2. The next step is to find the size of the atomic

spheres. First muffin-tin (MT) radii are found by *lmhart.run* and inserted into the CTRL file. The result of this run is shown below.

```
HEADER  NiSe2 Pyrit (Acta Chem. Scand.
                          v. 23 p. 2325 1969)
VERS    LMASA-46
STRUC   NBAS=12 NCLASS=2 NL=4 ALAT=11.2683
        PLAT=1 0 0
             0 1 0
             0 0 1
CLASS   ATOM=NI Z=28 R=2.41776510
        ATOM=Se Z=34 R=2.28352037
SITE    ATOM=NI POS=0.000 0.000 0.000
        ATOM=NI POS=0.500 0.500 0.000
        ATOM=NI POS=0.500 0.000 0.500
        ATOM=NI POS=0.000 0.500 0.500
        ATOM=Se POS=0.383 0.117 -.117
        ATOM=Se POS=-.383 -.117 0.117
        ATOM=Se POS=-.117 0.383 0.117
        ATOM=Se POS=0.117 -.383 -.117
        ATOM=Se POS=0.117 -.117 0.383
        ATOM=Se POS=-.117 0.117 -.383
        ATOM=Se POS=0.383 0.383 0.383
        ATOM=Se POS=-.383 -.383 -.383
SYMGRP  NGEN=3 GENGRP=I R3D MZ:(.5,0,.5)
SCALE   SCLWSR=T
START
```

The two tokens (R=) with the MT-radii in a.u. have been inserted in category CLASS.

3. The overlap between the WS-spheres is checked by the *lmovl.run* program. By inserting the category SCALE with the token SCLWSR=T the MT-spheres will be scaled to volume filling Wigner-Seitz (WS) spheres. The result written to standard out looks like (the output has been edited to fit into this documentation):

```
IB  JB  CL1  CL2 DIST SUMRS 1/d 1/s1 1/s2
=========================================
 1   5  NI   Se  .42  .55    31  46   49
-----------------------------------------
 5   1  Se   NI  .42  .55    31  49   46
 5   6  Se   Se  .41  .53    31  47   47
-----------------------------------------
```

Each line displays two neighboring atoms with WS-radii s1 and s2, and the distance between them d. Under '1/d', '1/s1', and '1/s2' are the results of $\frac{s1+s2-d}{d}$, $\frac{s1+s2-d}{s1}$, and $\frac{s1+s2-d}{s2}$, respectively, in percent. A 'rule of thumb' is that '1/d' for two atomic spheres should be smaller than 15%, while an atomic and an interstitial sphere are allowed to overlap by 20%. However, if the two sphere radii are very much different, one should check '1/s1', where s1 is the smallest radius. (A big sphere may swallow a small sphere.)

In the present case '1/d', is much larger than 15 and interstitial spheres have to be inserted.

4. Interstitial spheres are found by the *lmes.run* program. The resulting CTRL file is shown below. Notice that the token SCLWSR=T is ignored by this program.

```
HEADER  NiSe2 Pyrit (Acta Chem. Scand.
                          v. 23 p. 2325 1969)
VERS    LMASA-46
STRUC   NBAS=36 NCLASS=3 NL=4 ALAT=11.2683
        PLAT=1 0 0
             0 1 0
             0 0 1
CLASS   ATOM=NI Z=28 R=2.41776510
        ATOM=Se Z=34 R=2.28352037
        ATOM=E1 Z= 0 R=1.53150056
SITE    ATOM=NI POS=0.0000 0.0000 0.0000
        ATOM=NI POS=0.5000 0.5000 0.0000
        ATOM=NI POS=0.5000 0.0000 0.5000
        ATOM=NI POS=0.0000 0.5000 0.5000
        ATOM=Se POS=0.3830 0.1170 -.1170
        ATOM=Se POS=-.3830 -.1170 0.1170
        ATOM=Se POS=-.1170 0.3830 0.1170
        ATOM=Se POS=0.1170 -.3830 -.1170
        ATOM=Se POS=0.1170 -.1170 0.3830
        ATOM=Se POS=-.1170 0.1170 -.3830
        ATOM=Se POS=0.3830 0.3830 0.3830
        ATOM=Se POS=-.3830 -.3830 -.3830
        ATOM=E1 POS=-.2148 -.3193 -.0960
        ATOM=E1 POS=0.2148 0.3193 0.0960
        ATOM=E1 POS=-.3193 -.0960 -.2148
        ATOM=E1 POS=-.0960 -.2148 -.3193
        ATOM=E1 POS=0.3193 0.0960 0.2148
        ATOM=E1 POS=0.0960 0.2148 0.3193
        ATOM=E1 POS=-.2851 0.3193 0.4039
        ATOM=E1 POS=-.1806 0.0960 0.2851
        ATOM=E1 POS=0.2851 -.3193 -.4039
        ATOM=E1 POS=0.1806 -.0960 -.2851
        ATOM=E1 POS=0.0960 0.2851 -.1806
        ATOM=E1 POS=0.2148 0.1806 -.4039
        ATOM=E1 POS=-.0960 -.2851 0.1806
        ATOM=E1 POS=-.2148 -.1806 0.4039
        ATOM=E1 POS=0.3193 0.4039 -.2851
        ATOM=E1 POS=-.3193 -.4039 0.2851
        ATOM=E1 POS=0.2851 -.1806 0.0960
        ATOM=E1 POS=0.1806 -.4039 0.2148
        ATOM=E1 POS=-.2851 0.1806 -.0960
        ATOM=E1 POS=-.1806 0.4039 -.2148
        ATOM=E1 POS=0.4039 -.2851 0.3193
        ATOM=E1 POS=-.4039 0.2851 -.3193
        ATOM=E1 POS=-.4039 0.2148 0.1806
        ATOM=E1 POS=0.4039 -.2148 -.1806
SYMGRP  NGEN=3 GENGRP=I R3D MZ:(.5,0,.5)
SCALE   SCLWSR=T
START
```

The program found 24 interstitial spheres, all equivalent, with a sphere radius between 1.25 and 4.5 a.u. These limits are default and can be changed. Notice that the tokens in category STRUC have been updated.

5. Check the overlap again with *lmovl.run*. Standard out now looks like:

```
IB  JB  CL1  CL2 DIST SUMRS 1/d 1/s1 1/s2
=========================================
 1   5  NI   Se  .42  .47    13  22   23
 1  20  NI   E1  .35  .40    13  18   29
-----------------------------------------
```

```
 5   1  Se   NI  .42  .47    13  23   22
 5   6  Se   Se  .41  .46    13  23   23
 5  14  Se   E1  .34  .38    13  19   28
 5  15  Se   E1  .38  .38     1   1    2
 5  17  Se   E1  .34  .38    13  19   28
 5  22  Se   E1  .34  .38    13  19   28
 5  23  Se   E1  .34  .38    13  19   28
 5  29  Se   E1  .38  .38     1   1    2
 5  31  Se   E1  .34  .38    13  19   28
 5  34  Se   E1  .34  .38    13  19   28
 5  35  Se   E1  .38  .38     1   1    2
 5  36  Se   E1  .34  .38    13  19   28
-----------------------------------------
13   2  E1   NI  .35  .40    13  29   18
13   6  E1   Se  .34  .38    13  28   19
13   7  E1   Se  .38  .38     1   2    1
13   8  E1   Se  .34  .38    13  28   19
13  12  E1   Se  .34  .38    13  28   19
13  15  E1   E1  .27  .31    12  21   21
13  25  E1   E1  .30  .31     1   2    2
-----------------------------------------
```

The maximum overlap is 13% which is fine and one can proceed.

6. Complete the CTRL file with the *lmctl.run* program, but first insert the category IO and the token VERBOS=50 to ensure that all default values are inserted.

```
HEADER  NiSe2 Pyrit (Acta Chem. Scand.
                          v. 23 p. 2325 1969)
VERS    LMASA-46
IO      VERBOS=50 HELP=F WKP=F IACTIV=F
        ERRTOL=2 OUTPUT=* ERR=*
STRUC   NBAS=36 NCLASS=3 NL=3 ALAT=11.2683
        PLAT=1 0 0
             0 1 0
             0 0 1
OPTIONS NSPIN=1 REL=T CCOR=T NONLOC=F NRMIX=2
        CORDRD=F NITATOM=5
        CHARGE=F FATBAND=T AFM=F FS=F Q=
CLASS   ATOM=NI Z=28 R=2.72719158 LMX=2
                CONF=4 4 3 4 IDXDN=1 1 1
        ATOM=SE Z=34 R=2.57576616 LMX=2
                CONF=4 4 4 4 IDXDN=1 1 2
        ATOM=E1 Z= 0 R=1.72750257 LMX=2
                CONF=1 2 3 4 IDXDN=1 2 2
SITE    ATOM=NI POS=0.0000 0.0000 0.0000
        ATOM=NI POS=0.5000 0.5000 0.0000
        ATOM=NI POS=0.5000 0.0000 0.5000
        ATOM=NI POS=0.0000 0.5000 0.5000
        ATOM=SE POS=0.3830 0.1170 -.1170
        ATOM=SE POS=-.3830 -.1170 0.1170
        ATOM=SE POS=-.1170 0.3830 0.1170
        ATOM=SE POS=0.1170 -.3830 -.1170
        ATOM=SE POS=0.1170 -.1170 0.3830
        ATOM=SE POS=-.1170 0.1170 -.3830
        ATOM=SE POS=0.3830 0.3830 0.3830
        ATOM=SE POS=-.3830 -.3830 -.3830
        ATOM=E1 POS=0.2851 -.1806 0.0960
        ATOM=E1 POS=-.2851 0.1806 -.0960
        ATOM=E1 POS=0.0960 0.2851 -.1806
        ATOM=E1 POS=-.0960 -.2851 0.1806
        ATOM=E1 POS=0.1806 -.0960 -.2851
        ATOM=E1 POS=-.1806 0.0960 0.2851
        ATOM=E1 POS=0.3193 0.0960 0.2148
        ATOM=E1 POS=0.2148 0.3193 0.0960
        ATOM=E1 POS=-.3193 -.0960 -.2148
        ATOM=E1 POS=-.2148 -.3193 -.0960
        ATOM=E1 POS=0.0960 0.2148 0.3193
        ATOM=E1 POS=-.0960 -.2148 -.3193
        ATOM=E1 POS=0.4039 -.2148 -.1806
        ATOM=E1 POS=-.4039 0.2148 0.1806
        ATOM=E1 POS=-.1806 0.4039 -.2148
        ATOM=E1 POS=0.1806 -.4039 0.2148
        ATOM=E1 POS=0.2148 0.1806 -.4039
        ATOM=E1 POS=-.2148 -.1806 0.4039
        ATOM=E1 POS=0.4039 -.2851 0.3193
        ATOM=E1 POS=0.3193 0.4039 -.2851
        ATOM=E1 POS=-.4039 0.2851 -.3193
        ATOM=E1 POS=-.3193 -.4039 0.2851
        ATOM=E1 POS=-.2851 0.3193 0.4039
        ATOM=E1 POS=0.2851 -.3193 -.4039
SYMGRP  NGEN=3 GENGRP=I R3D MZ:(.5,0,.5)
        USESYM=F
SCALE   SCLWSR=T OMMAX1=.4 OMMAX2=.8
STR     EKAP=0 RMAXS=3.2 NDIMIN=0 NOCALC=F
        IALPHA=1
        DOWATS=F SIGMA=.7 DELTR=.1 LMAXW=8
START   NIT=10 BROY=T WC=-1 NMIX=1 BETA=.5
        FREE=F CNVG=.00001 CNVGET=.00001
        BEGMOM=T CNTROL=T
        ATOM=NI   P=4.67 4.41 3.86
                  Q=0.6 0.0 0.0
                    0.8 0.0 0.0
                    8.6 0.0 0.0
        enu   =-.54025 -.39451 -.29911
        c     =-.41430 0.55748 -.27602
        sqrdel=0.46623 0.58307 0.18957
        p     =0.05289 0.02751 2.34888
        gamma =0.54717 0.24070 -.00923
        ATOM=SE   P=4.95 4.84 4.17
                  Q=2.0 0.0 0.0
                    3.7 0.0 0.0
                    0.3 0.0 0.0
        enu   =-0.90676 -0.32685 -0.45626
        c     =-1.28821 -0.28838  1.52235
        sqrdel= 0.45477  0.46541  0.64709
        p     = 0.17990  0.08697  0.01771
        gamma = 0.44213  0.15420  0.14817
        ATOM=E1   P=1.2 2.2 3.2
                  Q=0 0 0
                    0 0 0
                    0 0 0
        enu   =-1.75789 -0.64181  1.31582
        c     = 0.99040  3.50106  6.98188
        sqrdel= 0.59058  0.48592  0.40034
        p     = 0.00690  0.00328  0.00189
        gamma = 0.35871  0.06503  0.02214
CHARGE  LMTODAT=T ELF=F ADDCOR=F CHARWIN=F
        EMIN=-2 EMAX=2
PLOT    ORIGIN=0 0 0
            R1=1 0 0 NDELR1=0
            R2=0 1 0 NDELR2=0
            R3=0 0 1 NDELR3=0 FORMAT=2
BZ      NKABC=4 4 4 TETRA=T METAL=T
        TOL=.000001
        N=0 W=.005 RANGE=5 NPTS=1001 SAVDOS=F
```

```
EWALD    NKDMX=250 AS=2 TOL=.000001
SCELL    PLAT=1 0 0
              0 1 0
              0 0 1 EQUIV=T
HARTREE BEGATOM=T LT1=2 LT2=2 LT3=2
DOS      NOPTS=2001 EMIN=-1.3 EMAX=0.7
SYML     NQ=30  Q1=0.0 0.0 0.0 LAB1=g
                Q2=0.5 0.5 0.0 LAB2=M
         NQ=20  Q1=0.5 0.5 0.0 LAB1=M
                Q2=0.5 0.0 0.0 LAB2=X
         NQ=20  Q1=0.5 0.0 0.0 LAB1=X
                Q2=0.0 0.0 0.0 LAB2=g
         NQ=35  Q1=0.0 0.0 0.0 LAB1=g
                Q2=0.5 0.5 0.5 LAB2=R
FINDES   RMINES=1.25 RMAXES=4.5
         NRXYZ=72 72 72
```

This is a complete CTRL file. The meaning of only a few of these tokens is necessary in order to run the program, however, a complete list with a description of each token is given in SectionV.

7. Before calculating the TB real space structure constants S^α and $\dot{S}^\alpha$ for the combined correction term by the *lmstr.run* program, the verbosity is lowered by changing VERBOS= to 40 in category IO (VERBOS= determines how much output of the calculation is written to standard out). Furthermore, NL= is set to 3 (NL= is the maximum number of ℓ's on any site, which has the default value of 4), because in this example f-partial waves are not needed. S^α only depends on the structure and the number of partial waves included, while $\dot{S}$ also depends on the WS-radii. If these are changed during the calculations, the structure constants have to be recalculated.

8. A self-consistent-field (SCF) calculation can now be performed using the program *lm.run*. If self-consistency is not achieved within the specified number of iterations (10 in the present case, the token NIT= in category START), the iterations can be continued using as start values, the information (potentials, potential parameters, etc.) in the atomic files which were generated by *lm.run*. If it is decided to start the SCF calculation from scratch, then the atomic files and the MIXM file must be erased. In the present case the program was converged before the 10th iteration. However, the default values for the k-mesh in the Brillouin zone are small (4 divisions along each of the primitive reciprocal translation vectors, see token NK-ABC= in category BZ).

9. *lm.run* is executed again with a larger number of k-points. This is only necessary for a metal where the Fermi energy and the Fermi surface must be accurately determined. For a semiconductor, a few k-points suffice. Change token NKABC= to e.g. 16 16 16 in category BZ. In category START, the tokens BEGMOM= and CNTROL= are set to F. This tells the program to start with a band structure calculation and with the self consistent potential parameters in the atomic files instead of the guesses in the CTRL file.

10. After the SCF solution to the problem has been found, the results are analyzed. First, the band structure along lines in k-space is calculated using *lmbnd.run*. The default values for the lines are given in category SYML. If orbital projected bands (fat bands) are needed the token FATBAND= must be changed to T in category OPTIONS. The band structure is plotted by *gnubnd.exec* as described in Section X E.

11. The atom and orbital projected densities of states (DOS) are calculated by *lmdos.run*. The default values for the energy window and the mesh points are given in category DOS and they must be changed appropriately. To get nice curves also for semiconductors an iteration with more k-points should be performed by *lm.run* as described above. The DOS is plotted by *gnudos.exec* as described in Section X F.

12. The full charge density is calculated by *lm.run* in the charge mode i.e. the token CHARGE= is set to T in category OPTIONS. If the electron localization function (ELF) is also needed, the token ELF= is set to T in category CHARGE. The r-mesh on which these functions are calculated is given in category PLOT which must be changed. The plots are performed by *gnucharge.exec* as described in Section X C if token FORMAT=1 in category PLOT or with Data Explorer if FORMAT=3.

13. The Fermi surface data is calculated by *lm.run* if the token FS= is set to T in category OPTIONS. This can be plotted by *gnufs.exec* as described in Section X G (Note that not all structures are implemented yet!), or by Data Explorer.

III. ORGANIZATION OF TB-LMTO PROGRAMS

A. Content of the main directory

All executable programs are in the main directory. The TB-LMTO package consists of four classes of programs.

(*i*) Programs to construct and check the data in the main input file (CTRL):

lminit.run generates a CTRL file with the structural data.

lmhart.run generates overlapping potentials from atomic Hartree potentials or finds MT-radii.

lmovl.run calculates and displays sphere overlaps.

lmes.run finds interstitial spheres.

lmctl.run rewrites the CTRL file according to the parameters in the original CTRL and atomic files and inserts default values.

lmscell.run increases the basis for super cell calculations.

lmnghbr.run produces a table of neighbors for each atom.

(*ii*) Programs to calculate the structure constants and performe self-consistent calculations:

lmstr.run generates the structure constants.

lm.run is the main LMTO program performing self-consistency iterations.

(*iii*) Programs to calculate the data to be displayed:

lmbnd.run generates bands for plotting.

lmdos.run generates DOS for plotting.

lm.run generates output for *lmdos.run* and charge density for plotting. (This is the same as under (*ii*) above, but run with different options.)

(*iv*) Plot programs. The programs with extension *exec* will (1) execute the program with the same name but extension *run*, which produces data files for the GNUPLOT program, (2) call the GNUPLOT program to make the plot, and (3) delete the temporary data files.

gnudos.exec: For density of states plot.

gnubnd.exec: For energy bands plot.

gnupot.exec: For contour plot of Hartree potentials.

gnucharge.exec: For charge density plots.

gnufs.exec: For Fermi surface plots.

All these executable programs are made by the UNIX *make* command which compiles and links the source codes in the subdirectories. All information about how this should be done is in the *makefile*, which is also located in the main directory. The *main* programs for the first three classes of programs above are made from the content of the file *lmall.f* in the main directory. This is done by *make* using the information in *makefile*, *getmain* in the subdirectory *utilities*, and the program *ccomp*. *ccomp* and its source *ccomp.c* are in the main directory, and the latter is the only program written in the language *C*. For a description of *ccomp.c* see Appendix A. The reason why the *main* programs are constructed in this way is that when changes are made, this only has to be done once, namely in *lmall.f*.

1. Dimensions/size of the program

Since the program is written in FORTRAN77 no true dynamical memory allocation can be performed. However, a 'semi-dynamical memory allocation' procedure is implemented. Apart from small fixed size arrays all arrays are stored in a work-array called *w* in the main programs. At the time a new array is needed, say the Hamiltonian, its size is calculated and space for it is allocated in the work array. When the array it not needed anymore, the space is freed and other arrays can be stored in the same place. This prevents 'over-dimensioning'. The size of the work array, however, depends on the size of the problem i.e. number of atoms, number of k-points, number of partial waves, etc. The size of the work array is given by *wksize* in the main programs. If *wksize* is too small for the problem at hand the program will abort, requesting increase of *wksize*.

B. Content of the subdirectories

After installation of the package each subdirectory contains files of source codes (.f or .c) and object codes (.o). Each file contains one main program or one subroutine. These are linked together by the UNIX *make* command to executable programs (.run) using the *makefile*.

1. Main programs

MAIN: All main programs. These are generated by *make* as described above.

MAINA: Some subroutines which have to be linked to the main program and some routines to calculate the Hartree potential.

2. Structure related routines

TBSTR: Real-space tight-binding structure constants S^α and $\dot{S}^\alpha$ for combined-correction term.

FINDES: Programs to find interstitial spheres.

HARTREE: Programs to calculate Hartree potentials.

3. Atomic programs

ATOM: Produces potential parameters (pp) and other atomic information from the moments. Non-relativistic calculations may be performed by setting REL=F in category OPTIONS. One can choose between the von Barth-Hedin or the Cheperly-Alder local density exchange correlation potential. The Langreth-Mehl-Hu and Perdew-Wang gradient corretions are also implemented.

4. Solid state programs

BNDASA: Calculates energy bands and evaluates the moments.

TETRA: Sets up the k-points and weights for k-space integration using the tetrahedron method.

DOS: Evaluates the density of states.

MAD: Evaluates the Madelung potential and energy.

DENS: Calculates full charge density and the ELF.

SYM: Generates all operations in a space group from a set of generators. This is used to determine the k-points in the irreducible part of the Brillouin zone, and to symmetrize the density matrix which is only calculated in the irreducible BZ.

5. Utility subdirectories

LMIO: Input of the CTRL file using library in IOLIB.

IOLIB: Fortran support for input and file handling.

IOCTRL: Programs to write the CTRL file.

EISP: Matrix diagonalization routines.

LINP: Linear algebra routines.

BLAS: Basic linear algebra routines.

EXTENS: Various utility routines and machine constants.

utilities: Some utility programs used by *makefile*.

6. Plots

PLOT: Programs to produce the data files used by GNUPLOT. This is the only subdirectory where the programs are not split into subroutines.

IV. DATA FILES

The names of the data files given below should be used, though the actual name depends on the operating system. The name of the program which produces the data file is given in parentheses.

CTRL: An input and documentation file for all programs. All data in this file are described in detail in Section V.

POT: Overlapping Hartree potential. (*lmhart.run*)

ATOM-NAME: There is one file assigned to each inequivalent atom. A complete file (one generated in the atomic program) contains some general information, the moments, potential parameters and the ASA or Hartree potential and core density within the sphere. The moments and potential parameters are most commonly read from this file, but it is possible to input the moments or the potential parameters from the CTRL file instead, in which case these files need not be present at the start of a run. The name ATOM-NAME is fixed by the string following ATOM= in the CLASS category of the CTRL file.

The atomic file also has categories delimited by a nonblank character in the first column. The categories are: GEN: A table of output generated by the ATOM program. MOMNTS: The P's and moments Q. PPAR: The potential parameters. POT: The potential without the nuclear part (-2Z/r). The *lmhart.run* stores the Hartree atomic potential here, but after the run the atomic files are removed in order that they are not used by mistake as starting potentials in the SCF run. COR: The core electron density. PHI: ϕ and $\dot{\phi}$.

None of these categories is essential unless the information is required. For example, if you begin with a band calculation, it is essential that the potential parameters be present either here or in the CTRL file. If you begin with the moments, the moments must be present either here or in the CTRL file. Like in the CTRL file, the data is formatted. Having the potential present, facilitates the iterations towards self-consistency within the sphere. (*lmhart.run and lm.run*)

STR: Unformatted structure constant fiel and $\dot{S}$ file for combined correction term. (*lmstr.run*)

MIXM: This file contains moments and potential parameters from previous iterations for the Anderson or Broyden mixing. If you want to start again, and avoid using any previous iterations towards the next iteration, then erase the mixing files. This should be done if you change the input in such a way that it is expected to make a significant change in the self-consistent moments, such as if you change the IDMOD parameters, for example. (*lm.run*)

MOMS: Moments belonging to each eigenvalue (eigenvectors are only summed over the different magnetic states m) (*lm.run*)

BAND: Eigenvalues and angular momentum weights for density of states calculation. (*lm.run*)

LMDM: Density matrix used for full charge density calculation. (*lm.run*)

RHO: Full charge density (Including core if the token ADDCOR=T in category CHARGE). (*lm.run*)

RHOv: Valence charge density (Only produced if token ADDCOR=T in category CHARGE). (*lm.run*)

RHOc: Core charge density (Only produced if token ADDCOR=T in category CHARGE). (*lm.run*)

RHOS: spin density (Only produced if token SPINDENS=T in category CHARGE). (*lm.run*)

BNDS: Energy bands. (*lmbnd.run*)

EIGN: Eigenvalues and full eigenvectors in the orthogonal representation - for 'fatbands' plot. Note that this file can be big and in case 'fatbands' are not needed, set token FATBAND=F in category OPTIONS. (*lmbnd.run*)

DOS: Atom and orbital projected densities of states. (*lmdos.run*)

V. DESCRIPTION OF THE CTRL FILE, CATEGORIES AND TOKENS

Examples of CTRL files are shown in Section II.

The control file is the main input file, which can also serve to document a calculation. Data is read from the control file by categories, one category at a time. A category begins with a nonblank character in the first column; it ends with the next occurrence of one. The name of the category is the string that begins the category; e.g. the category 'STRUC' begins with 'STRUC' and ends before 'CLASS'. Data within a category is identified by a token, e.g. NSPIN=. Categories can occur in any order, but only the first category of a given name is used. Apart from a mild exception described later, the order of tokens within a category is also irrelevant. Only the 72 first characters in each line are read and a # works as an end-of-line character. See the documentation in IOLIB for a detailed description of the input procedures.

Four programs modify the CTRL file: *lmhart.run* inserts MT-radii, *lmes.run* inserts interstitial spheres, *lmctl.run* inserts default values for tokens not already specified in the CTRL file, and *lmscell.run* inserts data from the single cell to the super cell. The latter two changes the sphere radii to space filling spheres if token SCLWSR=T in category SCALE.

A complete list of categories and an explanation of each token is given below. Each subsection contains the name of the category and each 'subsubsection' contains the name of the token.

A. Category HEADER (optional)

Provides space to describe the contents of the control file.

B. Category VERS

1. Token LMASA- of cast integer

Is the version number to ensure compatibility of the CTRL file with the executing program. The program will send a warning message if the version number is incorrect, but it will not stop.

C. Category IO (optional)

1. Token HELP= of cast logical (optional)

T: print a list of tokens and data appropriate for this program, without reading anything.

2. Token VERBOS= of cast integer (optional)

Is an integer that determines the verbosity of output a program sends to standard out. The output is roughly as follows:
0: nearly nothing is printed 10: very terse 20: terse 30: normal 40: normal 50: verbose 100: low-level debugging 110: intermediate level debugging 120: high-level debugging.

3. Token WKP= of cast logical (optional)

Is a switch that turns on the 'debug' mode in the dynamic memory routines. An effective heap is declared as a single integer array in the main program. Pieces are apportioned dynamically by calls to routine defdr.f and others. They can be 'released' to be reused. You can watch the memory grow and shrink if this is turned on.

4. Token IACTIV= of cast logical (optional)

Is a switch that turns on the 'interactive' mode. When this mode is on, you have an option to abort program execution, change the verbosity, turn on the work array debug switch WKP= , or sometimes to change a value of a single variable that may be passed to query. There is also the option to turn off the interactive mode. At the prompt, enter either a simple carriage return, or one of the following: 'Vnnn', where nnn is the new verbosity; 'W' to toggle printing of work array; 'I' to turn off interactive prompt; 'A' to abort program execution or, in the case when a variable is passed, you may also enter 'Snnn', where nnn is a number (or T or F for logical variable); to change the value of that variable.

5. Token ERRTOL= of cast integer (optional)

An error or inconsistency discovered by the program is assigned an integer error code from 0 to 4. If the error code is larger or equal to the value of ERRTOL, the program will stop. The meaning of the error codes are: 0; information, 1; warning, 2; error (this is the default), 3; severe error, meaning that the results cannot be trusted, 4; fatal error, the program will always stop. For ERRTOL > 4, ERRTOL is set to 4. For ERRTOL=-1 all errors will be ignored and the program will continue.

6. Token OUTPUT= of cast character (optional)

A file name to which standard out is routed. OUTPUT=* means standard out.

7. Token ERR= of cast character (optional)

A file name to which standard error is routed. ERR=* means standard error.

D. Category STRUC

1. Token NBAS= of cast integer (optional)

Is the number of atoms in the basis. This is not used as an input to the calculation.

2. Token NCLASS= of cast integer (optional)

Is the number of classes (inequivalent atoms not related by symmetry). This is not used as an input to the calculation.

3. Token NL= of cast integer

Maximum number of ℓ values on any sphere. This need not be given.

4. Token ALAT= of cast double

Is a scaling factor, in atomic units (Bohr radii), of the primitive translation and basis vectors.

5. Token PLAT= of cast double and length 9

Are primitive translation vectors, $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$, in units of ALAT and in Cartesian coordinates. The order is as follows: $t_{1x}t_{1y}t_{1z}t_{2x}t_{2y}t_{2z}t_{3x}t_{3y}t_{3z}$.

6. Token FIXLAT= of cast logical (optional)

The programs may rotate and translate the lattice in order to obtain the standard choice of the cell and the origin. FIXLAT=T prevents teh rotation and translation if it is not wanted. (default FIXLAT=F)

E. Category OPTIONS (optional)

Is self explanatory.

1. Token NSPIN= of cast integer (optional)

Must be 1 for non-spin-polarized or 2 for spin-polarized calculations.

2. Token REL= of cast logical (optional)

T: The scalar relativistic wave equation is solved. F: The non-relativistic Schrödinger equation is solved.

3. Token CCOR= of cast logical (optional)

Turns on the combined correction. Program *lmstr.run* always calculates the $\dot{S}^{\alpha}$. They are stored in a file SDOT for use by program *lm.run* and *lmbnd.run*, which will include the combined correction into the LMTO Hamiltonian and overlap matrices when this switch is set to T.

4. Token NONLOC= of cast logical (optional)

Turns on the non-local exchange correlation. NRXC=1 and NONLOC=T Langreth, Mehl and Hu nonlocal correction to the v. Barth, Hedin parametrization. NRXC=2 and NONLOC=T Perdew-Yue correction to the Vosko-Ceperley-Alder parametrization. NRXC=4 and NONLOC=T the Perdew-Wang 91 GGA. Note: NRXC=3 and NONLOC=T not possible.

5. Token NRMIX= of cast integer (optional)

Determines the number of previous iterations in the charge density to mix when making a sphere self-consistent. Anderson or Broyden mixing can be used to mix the charge density to facilitate convergence. Practical experience has shown that NRMIX=2 works well (this is the default), and normally you should not need to worry about this parameter. It may occasionally happen that the sphere program will not converge with Anderson mixing, in which case you should set NRMIX to 0.

6. Token Q= of cast character (optional)

BAND, MAD, ATOM, SHOW make the program stop after the band structure calculation, the Madelung calculation etc.

7. Token CORDRD= of cast logical (optional)

T: reads the core density from the atomic file. The core density is not calculated inside the atomic loop but kept frozen. (Default is F.)

8. Token NITATOM= of cast integer (optional)

Number of atomic iterations. To make any atom self-consistent NITATOM=80 suffices. However, there is no need for fully self-consistent atoms in the beginning of the band structure loops since the moments are not fully self-consistent. The default is 5.

9. Token NRXC= of cast integer (optional)

With the token NRXC in category it is possible to select the exchange correlation functional. Supplied are: (default NRXC=1)
NRXC=1 —> v. Barth & Hedin
NRXC=2 —> Vosko / Ceperley & Alder
NRXC=3 —> Perdew & Zunger / Ceperley & Alder
NRXC=4 —> Perdew & Wang 91

10. Token CHARGE= of cast logical (optional)

T: Full charge density is calculated. Other Tokens are automatically set to the right values. (Only active for *lm.run*.)

11. Token FATBAND= of cast logical (optional)

T: a file (EIGN) is produced by *lmbnd.run* containing the eigenvalues and eigenvectors. This is used by the plot program to give the energy bands a width proportional to specified orbital characters. The default is F.

12. Token AFM= of cast logical (optional)

T: Anti ferromagnetic calculation. I.e. the electronic structure is only calculated for one spin direction. It should be noticed that the atoms in the two spin sublattices must occur in exactly the same order in the CTRL file as explained in Section IX.

13. Token FS= of cast logical (optional)

T: The program *lm.run* produces a file LMFS, which can be used to plot the Fermi surface.

14. Token OVLCOR= of cast logical (optional)

T: A correction for the sphere overlap is added to the hamiltonian and overlap matrices. This is not well optimized, therefore, it is recommended to use the default value F.

15. Token WRIBAS= of cast logical (optional)

T: The *lmctl.run* program will create a file cstruc.dx with all atomic positions within the volume defined in category PLOT. This can be used by DATA EXPLORER to make a plot of the structure. If *lmctl.run* is executed interactively (token IACTIV=T in category IO) the colors and the 'sticks' connecting the atoms can be chosen, otherwise they are chosen by the program.

16. Token CARTESIAN= of cast logical (optional)

T: express the basis atoms in cartesian coordinates (default). F: express the basis atoms in units of the conventional lattice vectors PLATCV. In the later case the token POS= is replaced by POS1=. Also the translational part of the symmetry operations are expressed in units of PLATCV (':' is replaced by '::'). Example: I::(0,0,.5) is an inversion with translation vector platcv(3)/2.

F. Category CLASS

Contains information relevant to parameters inside atomic spheres for each inequivalent atom.

1. Token ATOM= of cast character

A one– (or more) character string naming the atom of that class. This string names the disk file – at least for some operating systems – that will hold important information about that atom (potential parameters, moments, potential etc.), and is used elsewhere in the input (see categories SITE and START below) to identify a particular atom. Following the token ATOM=, several tokens are sought:

2. Token NEWNAM= of cast logical (optional)

T: the labels of the atoms are changed and unified (just try it out).

3. Token Z= of cast double

The nuclear charge of the atom. For interstitial spheres Z=0.

4. Token R= of cast double (optional)

The sphere radius, in atomic units (Bohr radii).

5. Token R/W= of cast double (optional)

Alternatively to token R=, the ratio of the sphere radius to the average Wigner Seitz radius. The average WSR is defined such that if all R/W were 1, the sum of all sphere volumes would fill space.

6. Token LMX= of cast integer (optional)

The maximum ℓ quantum number inside the sphere.

7. Token IDXDN= of cast integer and length 4 (optional)

Can take the value 0, 1, 2, or 3, for each ℓ. This determines if the partial wave should be treated as low (1), intermediate (2) (i.e. it will be downfolded), or high (3) (i.e. it will be thrown away). If IDXDN=0; then the program decides if the partial wave should be treated as low, intermediate, or high.

8. Token CONF= of cast integer and length 4 (optional)

Configuration i.e. a principal quantum number. This determines which states will be treated as core states. For each ℓ all states with principal quantum number smaller than that given by CONF will be treated as core states.

9. Token IDMOD= of cast integer and length 4 (optional)

Is a set of integers, one for each ℓ-channel. IDMOD controls how the potential parameter E_ν changes from one iteration to the next in a self-consistency cycle.

IDMOD=0 lets the E_ν's shift to the center of gravity of the occupied part of the band (this is the default);

IDMOD=1 essentially freezes the logarithmic derivative of the wave function phi at the sphere radius;

IDMOD=2 freezes the E_ν's from one iteration to the next.

Because data must be read for each atom, tokens are repeated (once for each class). For that reason, there is some restriction as to the order of tokens. Tokens Z=, R=, R/W=, LMX= and CONF= must follow the token ATOM= they are associated with and precede the next token ATOM=.

G. Category SITE

Holds site information.

1. Token ATOM= of cast character

Is needed to identify which atom in category CLASS that site belongs to. Similarly to category CLASS, one set of tokens is read for each site in the basis, and there is a similar restriction on the order of tokens.

2. Token POS= of cast double and length 3

Is the basis vector, in units of ALAT and in Cartesian coordinates. Some of the programs require the most compact basis position i.e. positions closest to the origin. These new positions are calculated by the programs and written back into the CTRL file.

H. Category SYMGRP (optional)

Provides information to confine integrations to a smaller part of the Brillouin zone. One enters a small number of space group operations; subroutines in the subdirectory SYM will generate all inequivalent products of these operations. For example, the three operations R4X, MX and R3D are sufficient to generate all 48 elements of the cubic octahedron symmetry. Symbols have the form O:(nx,ny,nz) where O is one of M, I or Rj for mirror, inversion and j-fold rotation; and nx,ny,nz are a triplet of indices specifying the axis of operation. You may use X, Y, Z or D as shorthand for (1,0,0), (0,1,0), (0,0,1), (1,1,1). You may also enter products, such as I*R4X. Since these operations restrict numerical integration to a portion of the BZ, it is important that the class definitions in category CLASS are compatible with these operations. If these operations must assume two atoms to be equivalent but are not so defined in CLASS, the electron density in those two atoms may not be equivalent. For non-symmorphic space groups the translational part of the group operation must also be specified (this is only used for calculating the density matrix, i.e. for charge density calculation). This is done by adding after the rotational part :(rx,ry,rz). I. e. for the NiSe2 example above MZ:(0.5,0,0.5).

1. *Token NGEN= of cast integer (optional)*

The number of generators that follows.

2. *Token GENGRP= of cast integer (optional)*

The names of the generators separated by a blank.

3. *Token USESYM= of cast logical (optional)*

T: The space group generated from the generators is used. This may be useful if the calculation has to be performed for a lower symmetry than the actual symmetry of the solid. F: The space group of the solid is found by the program and used in the calculation. Furthermore, if some atoms are missing in category SITE they will be added to the basis. I.e. only one of the equivalent atoms must be supplied in SITE. After the basis has been completed it is recommended to use USESYM=F.

I. Category SCALE

Contains information about scaling the sphere radii.

1. *Token SCLWSR= of cast logical (optional)*

T: Scale the sphere radii. F: Do not scale.

2. *Token FACVOL= of cast double (optional)*

Scale the sphere volumes to FACVOL$\times$ the unit cell volume. Default is FACVOL=1.

3. *Token OMMAX1= of cast double (optional)*

Scale to an overlap (s1+s2–d)/d $<$ OMMAX1, keeping s1/s2 ratio. s1 and s2 are the two sphere radii and d is the distance between the centers of the spheres.

4. *Token OMMAX2= of cast double (optional)*

Scale to an overlap (s1+s2–d)/s $<$ OMMAX2, keeping s1/s2 ratio. s1 and s2 are the two sphere radii, s is the smallest of these, and d is the distance between the centers of the spheres.

5. *Token GAMMA= of cast double (optional)*

Linear scaling of radii to space filling volume. s(new) = a(s(old)+b), where ab=GAMMA(a-1) $\times$ the average Wigner-Seitz radius.

J. Category STR

Contains information for the structure constant program. Real space tight-binding structure constants are generated for a cluster of atoms inside RMAXS$\times$W around each basis atom by 'inversion' using Cholesky decomposition. See the explanation in the subdirectory TBSTR.

1. *Token EKAP= of cast double (optional)*

Kappa square. The default is zero, but the program works for positive as well as negative kappas. Alternatively EW**2 may be used as input.

2. *Token EW**2= of cast double (optional)*

(Kappa$\times$average Wigner-Seitz radius) squared.

3. *Token IALPHA= of cast integer (optional)*

0; $\alpha = j_l(kappa \times SIGMA \times s_R)/n_l(kappa \times SIGMA \times s_R)$ is used. IALPHA= 1, α's are calculated in the program. IALPHA= 2, α and $\dot{\alpha}$ must be supplied in the CTRL file (ALPHA0, ADOT, and NOALPH must be given). Explanation(IALPHA=1): α and $\dot{\alpha}$ depend on kappa. The canonical α's only applies for kappa=0. Optimal α's (leading to screening for non-zero kappa's) have been calculated as a function of kappa and a fit formula has been found. This is used to calculate the α's. If IALPHA<0 (i.e. -1 or -2) and *lmopt.run* is run, then new α's are not calculated, but the two-center integrals are calculated for the interpolated or input α's.

4. *Token ALPHA0= of cast double (optional)*

A list of different α values are read into ALPHA0. They are assigned to sites and orbitals by the pointers NOALPH below.

5. *Token ADOT= of cast double (optional)*

Same as ALPHA0, but for $\dot{\alpha}$. Only used if CCOR=T.

6. Token NOALPH= of cast double (optional)

NCLASS*NL (number of classes times max. number of ℓ's) pointers associating the α's and $\dot{\alpha}$'s in the ALPHA0 and ADOT list with each non-equivalent site and ℓ. I.e. with three non-equivalent sites, nl= 3 and NOALPH= 1 2 3 1 2 4 1 2 5: The s-orbitals on all sites have the first α from the list, the p's have the second alpha, the d on the first site has the third α etc.

7. Token SIGMA= of cast double (optional)

Screening core radii used to make α's (see Token IALPHA).

8. Token DOWATS= of cast logical (optional)

T: A 'Watson sphere' with a radius enclosing all atomic spheres in the cluster plus DELTR (Token DELTR below) $\times$ W (=the average Wigner-Seitz radius) is placed around each cluster of atoms. On this sphere the envelope functions expanded in spherical harmonics is zero for all l's below LMAXW (Token LMAXW below).

9. Token DELTR= of cast double (optional)

See Token DOWATS above.

10. Token LMAXW= of cast integer (optional)

See Token DOWATS above.

11. Token RMAXS= of cast double (optional)

Is the maximum sphere radius (in units of the average WS) in which neighbors will be included in the generation of structure constants. This takes a default value and is not required as input. For 'strange' structures one should calculate the structure constants for several RMAXS and check if the structure constants are converged.

12. Token NDIMIN= of cast integer (optional)

For the default value of 0, this is ignored. For positive values of NDIMIN, the number of sites in the cluster around each site is determined as the total number of orbitals in the cluster = NDIMIN. The number of orbitals on each site is weighted by WSR(tail)/WSR(head).

13. Token ITRANS= of cast integer (optional)

This token is only effective for the *lmstr.run* program. It can take three values: 0, the usual TB-structure constants; 1 or 2, new structure constants. Default is 0.

14. Token DONALP= of cast logical (optional)

T: Plot a screened envelope function: $||N\rangle + \epsilon||\dot{N}\rangle$. See Ref. 10. The following tokens: LDN, MDN, JBASDN and EPS must be supplied.

15. Token LDN= of cast integer (optional)

See token DONALP. LDN is the ℓ-value of the function.

16. Token MDN= of cast integer (optional)

See token DONALP. MDN is the m-value of the function.

17. Token JBASDN= of cast integer (optional)

See token DONALP. JBASDN is the atom number on which the function is centered.

18. Token EPS= of cast double (optional)

See token DONALP. EPS is the ϵ-value of the function.

19. Token NOCALC= of cast logical (optional)

Is available if it is desired to read the structure constants from the file without recalculating them. This is mostly useful if you would like to look at them (by turning the verbosity above 40) without recalculating them.

K. Category BZ

Holds information concerning the numerical integration of energy bands, etc. over the Brillouin Zone. The LMTO programs are not tied to any particular method, so the desired method must be specified by a token. In all methods, the BZ is split into a mesh of parallelepipeds.

1. Token NKABC= of cast integer and length 3

Is the number of divisions of the three primitive reciprocal translation vectors. This is preferable to the alternative token MAXKP below, since the divisions can be chosen so that the step length is about the same along the three vectors.

2. Token MAXKP= of cast integer

Is the maximum number of k-points in the entire Brillouin zone.

3. Token TETRA= of cast logical (optional)

T: k-space integration by the tetrahedron method. F: integration by sampling. (Don't use F!!! O.J.)

4. Token METAL= of cast logical (optional)

T: for metals. F: for insulators. If not known or for the first iterations choose T.

5. Tokens N=, W=, RANGE=, and NPTS=

Parameters used in k-space integration by sampling.

6. Token TOL= of cast double (optional)

Is the accuracy with which the Fermi level will be determined.

L. Category EWALD (optional)

Holds information controlling the Madelung sums. The defaults are almost always adequate; for a detailed description the reader is referred to the documentation on the Madelung sums in the program in the subdirectory MAD. For 'strange' structures one should check the convergence by decreasing the token TOL= (see below).

1. Token AS= of cast double (optional)

Controls the relative number of lattice vectors in real and reciprocal space.

2. Token NKDMX= of cast integer (optional)

Maximum number of terms in the Ewald sum (used for both real space and reciprocal space sums)

3. Token TOL= of cast double (optional)

Is the error criterion for the Ewald sums.

M. Category DOS

Holds information about the energy mesh in the density of states calculations.

1. Token NOPTS= of cast integer

Number of energy mesh points.

2. Token EMIN= of cast double

First energy mesh point.

3. Token EMAX= of cast double

Last energy mesh point.

N. Category SYML

Holds information about lines in k-space along which the band structure should be calculated and plotted.

1. Token NQ= of cast integer

The number of points along the line.

2. Tokens Q1, Q2= of cast double and length 3

The first point and last point on the line in Cartesian coordinates and in units of $2\pi/\mathrm{ALAT}$, where ALAT is the lattice constant given in the CTRL file.

3. Tokens LAB1, LAB2= of cast character

The labels of the first and last point on the line. Because of limitations in the present version of gnuplot the Γ point should be given the label g.

O. Category START (optional)

Controls the flow of execution and the mixing scheme used in the iterations towards self-consistency in the *lm.run* program. Either the modified Broyden mixing or Anderson mixing scheme can be used to get an optimal input vector of P and Q for the next iteration to self-consistency. For more information, see subroutines *mixpq.f*, *mixpqa.f* and *mixpqb.f* in the subdirectory MAINA.

1. Token NIT= of cast integer (optional)

Is the number of iterations *lm.run* executes before quitting (unless self-consistency is achieved earlier).

2. Token BROY= of cast logical (optional)

T: Modified Broyden mixing. F: Anderson or linear mixing.

3. Token WC= of cast double (optional)

Specifies how strong previous iteration steps are weighted in the modified Broyden scheme. For WC large, the last iteration step is weighted most. When starting with bad moments it is useful to cancel the stored Jacobi matrix after some iterations to let the calculation forget its history (delete file MIXM). You can also choose an arbitrary negative value for WC. In this case, WC is adjusted internally according to the inverse root mean square difference of the components of the mixed vector.

4. Token NMIX= of cast integer (optional)

Is the number of previous iterations mixed in with the present one using the Anderson scheme. Use NMIX=0 for linear mixing and the the token BETA= the feedback parameter.

5. Token BETA= of cast double (optional)

Is the proportion of the new iteration (as obtained from the Anderson mixing) admixed with the (1-BETA) of the old iteration. When difficulties with convergence occur BETA should be reduced.

6. Token CNVG= of cast double (optional)

Is the tolerance in the RMS change in the zeroth moments before self-consistency is achieved.

7. Token CNVGET= of cast double (optional)

Is the tolerance in the total energy before self-consistency is achieved.

8. Token FREE= of cast logical (optional)

Is intended to facilitate a self-consistent free-atom calculation. When FREE is true, the program uses R=30 for the sphere radius rather than whatever R is passed to it; the boundary conditions at R are taken to be value=slope=0 (R=30 should be large enough that these boundary conditions are sufficiently close to that of a free atom.); subroutine *atscpp.f* in subdirectory ATOM does not calculate potential parameters or save anything to disk; and program *lm.run* terminates after all the atoms have been calculated. Token FREE= T, and running *lmbnd.run*, free-electron bands are produced. In this case the structure constants are not used.

9. Token EFERMI= of cast double (optional)

Fermi energy.

10. Token VMTZ= of cast double (optional)

Muffin-tin zero potential.

11. Token BEGMOM= of cast logical (optional)

T: Causes program *lm.run* to begin with moments from which potential parameters are generated. F: the potential parameters are used and the program proceeds directly to the band calculation.

12. Token CNTROL= of cast logical (optional)

T: Read and use moments or potential parameters supplied in the CTRL file. F: The data following CNTROL are not used. Both the 'continuous principal quantum numbers' P (as described in the section 'Iterations Towards Self-Consistency') and the Q's (the moments) must be present for a given atom as displayed in the example; however, it is not necessary to have an input for each atom. Moments are read in Q0, Q1, Q2, once for each ℓ channel, and then once for each spin, if there are two spins. In the case ATOM=NI in the above example, 0.6 electrons is put in the s orbital, 0.8 in the p and 8.6 in the d.

P. Category HARTREE

Gives information about the calculation of the overlapping Hartree potential.

1. Tokens LT1, LT2, LT3= of cast integer

The solid Hartree potential is a superposition of atomic Hartree potentials. Atoms in unit cells translated from -LT1 to +LT1 (and equivalently for LT2 and LT3) are taken into account.

2. Token BEGATOM= of cast logical (optional)

T: Calculation of the SCF potential for each atom (Notice that the token FREE is effective) and keep the Hartree part. F: The potentials are read from the atom files. After this an overlapping Hartree potential for the solid is constructed and stored in the file POT.

Q. Category PLOT

Gives the mesh on which the full charge density, the ELF, or the Hartree potential has to be calculated.

1. Token ORIGIN= of cast double and length 3

Origin of the mesh in Cartesian coordinates and in units of ALAT.

2. Tokens R1, R2, and R3= of cast double and length 3

Are three vectors spanning a parallelepiped in which the charge density is calculated. Units of ALAT.

3. Tokens NDELR1, NDELR2, and NDELR3= of cast integer

Number of mesh points along the R1, R2, and R3 vectors respectively. If NDELR3=0 the charge density is evaluated in the plane spanned by R1 and R2. If NDELR2= is also 0 the charge density is evaluated along the line R1.

4. Token FORMAT= of cast integer (optional)

Selects the format of the output file. FORMAT=1, the format is appropriate for the GNUPLOT. =3, the format appropriate for DATA EXPLORER. =2, the format for 2D plots: x, y, f(x,y) and for 3D plots x, y, z, f(x,y,z), where x, y, and z are the x-, y-, and z-coordinate of the mesh point and f is the value of the function at this point. There is one mesh point per line. This format is appropriate for other plot packages.

5. Token POLAR= of cast logical (optional)

It is possible to use polar coordinates if you want to make a plot. The center is defined by ORIGIN=.

6. Token RMIN= RMAX= of cast double (optional)

Range of r.

7. Token TMIN= TMAX= of cast double (optional)

Range of θ.

8. Token PMIN= PMAX= of cast double (optional)

Range of Φ.

R. Category CHARGE

Information about the full charge density calculation.

1. Token LMTODAT= of cast logical

T: for the first plane in which the charge density has to be evaluated. For subsequent planes some data are already present (i.e. LMDM) and need not be recalculated, therefore LMTODAT= F. Default is T.

2. Token CHARWIN= of cast logical (optional)

T: Only states in the energy window from EMIN to EMAX are included. F: All occupied states are included (default).

3. Token EMIN= of cast double

For CHARWIN=T, EMIN is the minimum energy in the energy window.

4. Token EMAX= of cast double

For CHARWIN=T, EMAX is the maximum energy in the energy window.

5. Token ELF= of cast logical

T: in addition to the charge density, the electron localization function is calculated.

6. Token ADDCOR= of cast logical

T: The core charge density is added to the valence charge density. F: The core is not added, which is the default.

7. Token SPINDENS= of cast logical

T: a file RHOS containing the spin density is created.

S. Category FINDES

Information about the search for interstitial spheres.

1. Token RMINES= of cast double (optional)

No sphere with radius smaller than RMINES (in a.u.) should be found. The default value is 1.25 a.u.

2. Token RMAXES= of cast double (optional)

No sphere with radius larger than RMAXES (in a.u.) should be found. The default value is 4.5 a.u.

3. Token MAXPT= of cast integer (optional)

A real space mesh is used in the search for interstitial spheres. MAXPT is approximately the total number of points in the unit cell. Alternatively the token NRXYZ= below can be used.

4. Token NRXYZ= of cast integer and length 3 (optional)

A real space mesh is used in the search for interstitial spheres. The three numbers NRXYZ are the divisions of the three translation vectors.

T. Category SCELL

Information for supercell calculations.

1. Token PLAT= of cast double and length 9

Primitive translation vectors for the supercell. This is equivalent to the token PLAT in category STRUC and should be compatible with this i.e. the supercell translations should be a multiplum of those in STRUC.

2. Token EQUIV= of cast logical (optional)

T: The atomic data in category CLASS and SITE for the supercell will be completed from the single cell data.

U. Category RHOFIT

This category holds information about fitting the charge density. It is not recommended to use it!

1. Token FIT= of cast logical

T: Fit the charge density.

2. Token EKAP= of cast double

Energy of the fitting functions.

*3. Token EW**2= of cast double*

EKAP (see above) times the average Wigner-Seitz radius squared.

4. Token RMAXS= of cast double

Size of cluster used for calculating the structure constants. Same as in category STR.

5. Token SIGMA= of cast double

Fitting radius.

VI. ITERATIONS TOWARDS SELF-CONSISTENCY

Just as the LMTO method breaks naturally into an atomic part and a ‘solid’ part, so do the programs. In general the ‘solid’ part requires potential parameters and structure constants as its input, from which it generates bands, energy moments, densities-of-states, etc. The ‘atomic’ part takes moments as its input and calculates potential parameters from it. The atomic part requires very little information beyond the moments and E_ν's or boundary conditions to completely specify the electronic structure within an atomic sphere.

An LMTO calculation is self-consistent when the atomic part produces, from moments generated by the solid part, once again the same potential parameters that the solid part used to generate the moments in the first place. The self-consistency works by alternating between the solid part and atomic part, generating moments, then potential parameters, then moments again until the process converges. The program can be started either with the solid part, specifying potential parameters, or with the atomic part, specifying the moments.

Because the method is a linear one, only three functions can carry charge inside a sphere ($\phi^2, \phi\dot{\phi}, \dot{\phi}^2$) and therefore the properties of a sphere, within the approximations of the linear method are completely determined by the first three moments, the atomic number and the E_ν's or boundary conditions at the surface of the sphere. In some sense these numbers are ‘fundamental’ to a sphere; the atomic program will generate a self-consistent potential for a specified set of moments and E_ν's and generate potential parameters from it.

From the point of view of the bands, the Hamiltonian is completely specified by the potential parameters (and the structure constants). These are fundamental to the band program; it will generate moments from the eigenvectors of the Hamiltonian. Full self-consistency is achieved when the ‘input moments’ coincide with the ‘output moments’, or equivalently when the input pp's coincide with the output pp's.

The ASA program can start equally as well from either potential parameters or moments, though it is generally easier to start from the moments, if no prior information is available. This is because one can usually begin with a very simple starting guess (choosing the zeroth moment to be the charge of the free atom, the first and second to be zero) that is usually good enough to iterate towards self-consistency.

If potential parameters are available, you may choose to begin directly with a band calculation and need not worry about the moments. If you wish to make a self-consistent calculation, you must also supply the principal quantum numbers for each ℓ channel in the so-called P number described in the following paragraphs.

To make a sphere self-consistent one needs the moments and also to specify the boundary condition on the wave function at the sphere radius, or the E_ν of the wave function ϕ. For a given potential, there is a unique correspondence between the logarithmic derivative D_ν at the sphere radius and E_ν, so in principle, it is possible to specify either one.

There is a set of numbers called P (one for each ℓ channel) that supplies the information about both the principal quantum number and the logarithmic derivative, D. P is defined as: P = .5 - atan(D_ν)/π + n where n (its integer part) is the principal quantum number; its fractional part varies smoothly from 0 (for the bottom extreme of the band for that principal quantum number) to 1 (the top extreme of the band), and can be thought of as a 'continuously variable' principal quantum number. Here is a table of P_n as function of D_ν: D_ν 10 5 1 0 -1 -2 -3 -4 -10 P_n .03 .06 .25 .5 .75 .85 .90 .92 .97

P must be supplied to the atom program. The fractional part of P is automatically supplied by the output of a band calculation (provided the token IDMOD in category CLASS is not 1), but you must supply P (in addition to the moments Q) if you choose to begin with moments. P, together with the moments Q can be input directly through the control file. A word on choices for the fractional part of P: .3 is quite free- electron-like and suitable for free-electron-like ℓ channels such as Si d electrons, while .8 is quite tight-binding-like and suitable for deep states like Cu d orbitals. If there is no information from the very beginning, .5 is a suitable choice.

In self-consistency cycles, you have a choice, through the parameter IDMOD described below, regarding the related pair of parameters P and E_ν. You may let P and E_ν float to the center of gravity of the occupied part of the band (most accurate for self-consistent calculations); this is the default. You may also freeze alternatively P or E_ν in the self-consistency cycle.

The problem of 'ghost' bands can be overcome by using the downfolding procedure. Orbitals are separated in lower, intermediate and higher sets. By switching on the automatic downfolding procedure, this choice is done automatically by the program. Lower waves contribute to the dimension of the Hamiltonian matrix, H, and the overlap matrix, O, and carry charge. Intermediate waves do not contribute to the dimension of H and O, but carry charge. Higher waves are thrown away altogether. If it is not known how to separate the orbitals the automatic downfolding can be switched on before the self-consistency cycle. After a few iterations, the downfolding indices don't change and if it is desired the cycle can be stopped and started again with the proper division into lower, intermediate, and higher set. Calculations using the automatic downfolding should be checked carefully. The token which makes the decomposition is IDXDN.

IDXDN = 0 : automatic downfolding
IDXDN = 1 : low orbitals
IDXDN = 2 : intermediate orbitals
IDXDN = 3 : higher orbitals
More about this below under category CLASS and token IDXDN.

VII. ACCURATE BAND STRUCTURE AWAY FROM THE CENTER OF GRAVITY

Sometimes it is necessary to get accurate bands in a small energy range e.g. around the Fermi energy, for an accurate description of the Fermi Surface or in an energy range far from the center of gravity of the partial waves. In such cases the potential parameters have to be recalculated with new E_ν's for the SCF potential.

After the self-consistent calculation which has generated atomic files with potential parameters and potentials, one 'iteration' is performed with the following input files: In the atomic files the E_ν's are changed, which can be done by the program *chenu.run* in the main directory. Furthermore, the values for ID which appear after the moments are set to 2 (normally ID=0). In the control file (CTRL) the tokens NITATOM=1 and Q=BAND in category OPTIONS. Furthermore, the tokens BETA=0 BROY=F NIT=1 BEGMOM=T and CNTROL=F in category START. With these parameters *lm.run* produces a BAND file with the new bands and eigenvectors, which may be used for a DOS calculation. It also produces atomic files with the new potential parameters which may be used for calculation of energy bands along symmetry lines.

VIII. FROZEN POTENTIAL CALCULATIONS

In the ASA it is easy to calculate for instance structural energy differences. This is done by the so-called frozen potential technique. Assume that we want to calculate the structural energy difference between fcc and bcc copper. First a self-consistent calculation for fcc copper is performed in the usual way and the total energy and a potential is obtained. The potential is inserted in the bcc structure with the same atomic volume and the total energy is calculated without iterating to self-consistency i.e. keeping the potential frozen. The total energy difference in the two calculations is the structural energy difference.

The second step is performed in the following way: The atomic file containing the frozen potential and potential parameters is not changed, but the CTRL file is.

In category START: ITER=1, so that only one iteration is performed and BEGMOM=F, so that the program will start by making the hamiltonian and overlap matrices. Eigenvalues and eigenvectors will be calculated and from the latter the moments are obtained.

In category START: NMIX=0 so that linear mixing is effective and BETA=1 to ensure that the moments are not mixed with the previous ones.

In category CLASS: IDMOD=2 for all ℓ channels so that the E_ν's are not shifted to the new center of gravity.

In category OPTIONS: NITATOM=0 so that only one incomplete loop in the atomic program is performed. This produces the contribution to the total energy from the sphere.

IX. SPIN POLARIZED CALCULATIONS

A self-consistent spin-polarized calculation can be started just like a non spin-polarized (hereafter called NM) calculation, except that NSPIN=2 in Category OPTIONS and the starting moments for up and down spins in Category START are made different.

Often it is, however, faster first to perform a self-consistent NM calculation and then using this to start the spin-polarized calculation.

After the self-consistent NM calculation has been performed the atomic files contain the self-consistent moments, potential parameters and potentials. These have to be doubled (one set for each spin) and changed, before the spin-polarized iterations can begin. This can be done by changing NSPIN=1 to NSPIN=2 in Category OPTIONS and VERBOS=50 in Category IO and running *lmctl.run*, which inserts the doubled moments (devided by two) and potential parameters from the atomic files into the CTRL file. The potential parameters or the moments then have to be changed so that the two spin directions become different. Probably the most predictable is to change the moments. Example: Assume paramagnetic Ni has 9 *d*-electrons, then after *lmctl.run* the zeroth *d* moments for up and down spin are 4.5. We guess that the magnetic moment is 0.6 Bohr magnetons and the up-spin moment is changed to 4.8 and the down-spin moment is changed to 4.2. The spin polarized iterations are then started using these values in the CTRL file by setting BEGMOM=T and CNTROL=T in Category START.

The antiferromagnetic (AFM) calculation is different from the ferromagnetic and antiferrimagnetic calculations because in the latter two cases two band structure calculations have to be performed, one for spin-up and one for spin-down. In the AFM case the two band structures are identical, but the moments and potential parameters for one sublattice and say spin-up are the same as those for the other sublattice and spin-down. The program can take advantage of this by performing only one band structure calculation and interchanging the resulting moments. This is achieved by setting the token AFM=T in category OPTIONS. It should be noticed that one is *not* free to chose the order in which the atoms occur in the CTRL file. With the atomic order in the first sublattice chosen, the order of the second sublattice must be the same. Example: In NiO the spin-up sublattice consists of Ni-spin-up and O-spin-up and the spin-down sublattice consists of Ni-spin-down and O-spin-down. If Ni-spin-up is the first and O-spin-up the second in the CTRL file, then Ni-spin-down must be the third and O-spin-down must be the fourth atom. As described above it may be advantageous to start the spin-polarized calculations from a self-consistent NM calculation. This could be done as indicated above, but since usually there are twice as many atoms in the AFM case as in the NM case, it may be more convenient to use the program *chspin.run* in the main directory to produce 'spin-polarized' atomic files from NM ones. To use this, just type *chspin.run*. It will ask for the name of the 'non-spin-polarized' input file (let us call it NI) and the spin direction in the output file (U, for up or D, for down). It will produce a file with the name NIU or NID and with the same atom name in the file (NIU or NID). In this file the moments, the potential parameters, and the potentials are doubled (the moments divided by two). The potential parameters or the moments then have to be changed just as in the ferro-magnetic case described above. The extra atoms are then inserted in category SITE and CLASS BEGMOM=T and CNTROL=F in category START.

X. PLOT PROGRAMS

All programs for simple two-dimensional plots are based on the public domain program Gnuplot (Copyright (C) 1986 - 1993 Thomas Williams, Colin Kelley), which is part of the LMTO distribution.

A. Gnuplot

Gnuplot is a command-line driven interactive plotting utility for UNIX, MSDOS and VMS platforms. The software is copyrighted but freely distributed (i.e., you don't have to pay for it). It was originally intended for visualization of mathematical functions and data. Gnuplot supports many different types of terminals, plottersx, and printers (including many color devicesx, and pseudo-devices like LaTeX) and is easily extended to new devices. The 'GNU' in gnuplot is NOT related to the Free Software Foundation, the naming is just a coincidence (and a long story). Thus Gnuplot is not covered by the GNU copyleft, but rather by its own copyright statement, included in all source code. The Gnuplot package comes with makefiles for the following UNIX platforms:

1. Apollo running SR10.3 with Apollo's X11
2. DEC3100/5000 running DEC OSF/1 v1.0
3. DEC3100/5000 running Ultrix 3.1d with MIT's X11
4. HP/9000 700 series running HP/UX 8.0 with MIT's X11R4
5. Sun sparcstation running SunOS 4.1 with suntools (with and without X11)
6. Silicon Graphics IRIS4D machines (with and without X11)
7. NeXT Cube and Slab running NeXTOS 2.0+ (with and without X11)
8. ATT 3b1 machines (with and without X11)
9. 386 machines running 386/ix or ISC 2.2 (with and without T.Roell X386)
10. IBM RS/6000 running AIX 3.2 with xlc 1.2 or xlc 1.1
11. Cray Y-MP or Cray-2 running Unicos 6.0 or 6.1 (with and without X11)
12. Sequent Dynix/PTX with MIT X11
13. Sequent Symmetry (DYNIX 3) with X11
14. Convex 9.0 with MIT X11
15. KSR1 running DEC OSF/1 v1.0 (use make -j 16)
16. LINUX with XFree86-1.2
17. VAX-VMS

The following table summarizes most of the terminals (output formats) supported by Gnuplot Release 3.5.

table
Dump ASCII table of X Y [Z] values to

output
corel
EPS format for CorelDRAW
eepic
EEPIC – extended LaTeX picture

environment
emtex
LaTeX picture environment with

emTeX specials
epson-180dpi
Epson LQ-style 180-dot per inch

(24 pin) printers
epson-lx800
Epson LX-800, Star NL-10, NX-1000,

PROPRINTER ...
hp2623A
HP2623A and maybe others
hp2648
HP2648 and HP2647
hp7580B
HP7580, and maybe other HPs (4 pens)
hp500c
HP DeskJet 500c, [75 100 150 300]

[rle tiff]
hpgl
HP7475 and (hopefully) others (6 pens)
hpljii
HP Laserjet series II, [75 100 150 300]
hpdj
HP DeskJet 500, [75 100 150 300]
hppj
HP PaintJet and HP3630

[FNT5X9 FNT9X17 FNT13X25]
kc/km-tek40xx
MS-DOS Kermit Tek4010 terminal

emulator - color/monochrome
latex
LaTeX picture environment
mif
Frame maker MIF 3.00 format
nec-cp6
NEC printer CP6, Epson LQ-800

[monocrome color draft]
pbm
Portable bitmap [small medium large]

[monochrome gray color]
pcl5
HP LaserJet III [mode] [font] [point]
postscript
PostScript graphics language

[mode fontname font-size]

pslatex
LaTeX picture environment with

PostScript specials
pstricks
LaTeX picture environment with

PSTricks macros
qms
QMS/QUIC Laser printer

(also Talaris 1200 and others)
regis
REGIS graphics language
tek410x
Tektronix 4106, 4107, 4109 and 420X

terminals
tek40xx
Tektronix 4010 and others;

most TEK emulators
texdraw
LaTeX texdraw environment
tpic
TPIC – LaTeX picture environment

with tpic specials
vttek
VT-like tek40xx terminal emulator
X11,x11
X11 Window System

If there is access to one of the platforms listed above and one of the supported terminals connected to it, all that needs to be done is to compile and install Gnuplot on the target machine (read the File 0INSTALL for details).

B. Common features of the plotting programs

Five interface programs convert the output of the LMTO package to a format suitable for Gnuplot and generate a command file where all options are set:

gnucharge.run
Charge density or ELF plot
gnupot.run
Hartree potential plot
gnubnd.run
Band structure plot
gnudos.run
Density of states plot
gnufs.run
Fermi surface plot

These programs will be described in detail below. There are, however, some common features which will be described first.

Out of the variety of output devices and media, the most common ones are supported in the programs. Starting one of the programs results in the following prompt:

```
Enter output device:
 1 = Postscript (default)
 2 = HP-GL pen plotter
 3 = HP Laserjet III (PCL5)
 4 = PC screen (vt220 emulation)
 5 = X-Windows
```

Depending on the choice the following output is created:

1. A Postscript file named *charge.ps, pot.ps, bnds.ps, dos.ps* or *fs.ps.* This file can be send to any Postscript printer or be viewed with Display Postscript or Ghostview.
2. A file named *charge.hpgl, pot.hpgl, bnds.hpgl, dos.hpgl* or *fs.hpgl* suitable for all pen-plotters and laserprinters that understand HPGL (Hewlett-Packard Graphics Language).
3. A file named *charge.pcl, pot.pcl, bnds.pcl, dos.pcl* or *fs.pcl* suitable for all Hewlett-Packard laserprinters and compatibles.
4. Direct output to any vt220 compatible terminal including PCs emulating a vt220 (Tektronics 4014 and compatibles are also supported).
5. Direct output to a local or remote X-Window. If the program is used remotely, make sure that the remote host has permission to use the local display (xhost +localdisplayname) and set the environment variable DISPLAY to 'localdisplayname:screennumber'.

If the parameters have to be changed (e.g. Encapsulated Postscript instead of Postscript) or another terminal supported by Gnuplot has to be added, this can easily be done by changing the source code where the line "set term ..." is written to the Gnuplot command file. Refer to the Gnuplot manual for details.

All programs generate intermediate files containing the plot data and a commandfile with all the commands and options necessary. The names of these files are listed in the detailed description for each program below. These files will remain on the disk if the executables (*.run) are used directly, and that will allow some fine-tuning by editing the command file (*.GNU).

For routine plots, however, it is better to run the shellscripts *.exec, which call Gnuplot immediately after generating the plot data and delete all intermediate files afterwards.

C. Charge density and ELF plot

The program *gnucharge.run* generates plots of the calculated charge density or the Electron Localization Function (ELF). The program reads the files RHO and ELF created by *lm.run* in charge mode (CHARGE=T in category OPTIONS, ELF=T in category CHARGE) and creates the plot data file CHARGE.DAT and the command file CHARGE.GNU. After the selection of the output device follows the question:

```
generate a charge density(t) or ELF plot(f)?
```

Entering t (logical true) or / (End-of-record) selects a charge density plot, whereas f (logical false) selects a plot of the ELF. In the next step the plot data are read in and the minimum and maximum are determined:

```
 Number of grid-points:  2821
 the charges range from .660E-03 to .581E+01
```

By default the plot range extends from the minimum to the maximum of the data. However, you can modify the range by entering two scalars bounding the new interval:

```
  enter new min, max or / for default
```

The following question concerns the number of isocontour lines to be drawn

```
 how many contours?(default is 11)
```

You may enter any number between 2 and 98 or / to accept the default. The iso contours are now equally spaced from minimum to maximum including the upper and lower bound. Assuming 11 lines for a minimum of 0.0 and a maximum of 1.0 results in contours representing the values 0, 0.1, 0.2 ... , 0.9, 1.0. The next question defines the nature of the plot:

```
 draw a 3d-surface(t) or not(f)?(default: f)
```

Entering t (logical true) places a 3d surface with hidden-line-removal on top of the contour plot, whereas f (logical false) or / lead to a simple contour plot. Now the plotfile or the plot window should be generated.

Important notice: Gnuplot does not retain proper x-y-scaling for most output formats unless your plot is quadratic. You generally have to scale the plot manually by editing the directive "set size ..." in CHARGE.GNU.

D. Hartree potential plot

The program *gnupot.run* generates plots of the superimposed atomic Hartree potentials of the free atoms. This plot can be useful to check the estimate of the sphere radii performed by *lmhart.run.* The program reads the file POT created by *lmhart.run* for the plane specified in the PLOT category in the CTRL file and creates the plot data file GPOT.DAT and the command file POT.GNU. After the selection of the output device the plot data are read in and the minimum and maximum are determined:

```
lines:  2700
time for reading pot:  1.85
the potential range is from -.1E+7 to -.829E-1
```

By default the plot range extends from the minimum to the maximum of the data. Since these potentials usually cover several orders of magnitude, it is advisable to modify the range by entering two scalars bounding the new interval (e.g. -5.0 0.0) after this prompt:

```
  enter new min, max or "/" for default
```

The following question concerns the number of isocontour lines to be drawn

```
 how many contours? (default is 11)
```

You may enter any number between 2 and 98 or / to accept the default. The iso contours are now equally spaced from minimum to maximum including the upper and lower bound as explained in the *gnucharge* section. The next question defines the nature of the plot:

```
 draw a 3d-surface(t) or not(f)?(default: f)
```

Entering t (logical true) places a 3d surface with hidden-line-removal on top of the contour plot, whereas f (logical false) or / lead to a simple contour line plot. Finally you are prompted for a 40-character title string for the plot:

```
 enter plot title :
```

Now the plotfile or the plot window should be generated. As stressed in the *gnucharge* section, Gnuplot may change the x-y-scaling.

E. Band structure plot

The program *gnubnd.run* generates plots of the band structure along the symmetry lines defined in the SYML category of the CTRL file. The program reads the CTRL file, the BNDS file and, if FATBAND=T in the category options in the CTRL file, the file EIGN. The latter two are created by *lmbnd.run.* It creates the plot data files BNDS.DAT, FERMI.DAT and the command file BNDS.GNU. For an orbital projected bands plot BNDS2–4.DAT are created in addition.

After the selection of the output device one is prompted for a 40-character title string for the plot:

```
 enter title:
```

followed by the definition of the energy unit:

```
 energies in Ryd.(f) or eV(t)? (default f)
```

Entering t (logical true) selects electron volts, whereas f (logical false) or / selects Rydberg. The next prompt permits you to shift the origin of the energy scale to the Fermi energy:

```
  energies relative to EF (t)? (default f)
```

Depending on the choice of symmetry lines and energy range it may be advantageous to plot the band structure in portrait mode instead of the default landscape mode:

```
  landscape plot (t) ? (default t)
```

To achieve this enter f (logical false) after this prompt. If subsequent points of the same band should not be connected by a line, enter f (logical false) at the next prompt:

```
  energies connected by lines (t)? (default t)
```

The decision whether orbital projected bands are to be plotted or not is the next issue:

```
  plot orbital character(t)? (default f)
```

Enter t (logical true) to plot orbital projected bands, whereas f (logical false) or / result in an ordinary band plot. If the fat bands option is not used you may skip to the modification of the energy range below. If the fatband plot option is chosen, the prompted is:

```
Change coordinate system? Enter Euler angles:
alpha, beta, gamma(units of Pi).
If nochange: enter "/"
```

If the internal coordinate system should be changed, enter the Euler angles[11]. Enter / to retain the current orientation. Now the atoms whose orbitals should be displayed fat should be given:

```
no coordinate transformation!
Enter orbital character plotted as "fatbands"
First, enter name(s) of atom(s) format(20a4)
followed by <enter> and "/" on next line
```

Enter the names of the atoms as specified in the CLASS category of the CTRL file separated by blanks. Enter '/' on the next line. If there are several atoms of this class, can be selected:

```
RB1
Following atom(s) selected RB1
There are  6  atoms of type RB1
Specify which one(s) to select, e.g. 1 3 7 for
the first, third, and seventh, followed by "/"
```

Enter a list of integers followed by '/'. Next the orbitals to be included in the fatband plot should be given:

```
For each atom specify orbital number from list:
NB! Orbitals are in the new coordinate system!
s  y  z  x  xy  yz  3z^2-1  xz  x^2-y^2
1  2  3  4   5   6     7     8     9
y(3x^2-y^2) xyz y(5z^2-1) z(5z^2-3)
     10      11   12         13
 x(5z^2-1) z(x^2-y^2) x(3y^2-x^2)
     14          15          16
RB1 number 1 enter number(s) followed by "/":
```

Again, enter a list of integers followed by '/'. This step is repeated for all selected atoms of all selected classes. Now all fat band specific options have been set. The input files are read in and some information about them is displayed:

```
 Bands= 44 Fermi Energy= -.1426
 Lattice const.=29.589 Spins=1
 nq1=   35
 nq1=   40
 nq1=   20
 nq1=   30
 nq1=   25
 nq1=   45
 nq1=    0
 ebot=-.3864 etop=.3131 eferm=-.14258
   nq=  195   nline=    5
 default emin and emax =  -1.0 1.0
```

If band structure plot should be limited to a fraction of the calculated energy range the new boundaries may be entered after this prompt:

```
 enter emin, emax
```

Entering / instead of two scalars selects the whole range shown above. Now the plotfile or the plot window should be generated.

Important notice: Gnuplot is currently not capable of handling more than one font per plot. Therefore it is impossible to mix greek and roman letters and e.g. the Γ *point is represented by G. If the output is postscript, however, it is possible to edit the file bnds.ps and replace the sequence*

```
( G ) Cshow
```

by

```
/Symbol findfont 180 scalefont setfont
( G ) Cshow
/Times-Roman findfont 180 scalefont setfont.
```

One can move the label of the y-axis closer to the plot, simply by changing the line

```
currentpoint gsave translate 90 rotate 0 0
moveto ( Energy eV) Cshow
```

to

```
currentpoint gsave translate 90 rotate 0 -700
moveto ( Energy eV) Cshow.
```

F. Density of states plot

The program *gnudos.run* generates total or partial density of states plots. The program reads the file DOS created by *lmdos.run* and creates the plot data files DATA.DOS, FERMI.DOS and DOS.GNU. After the selection of the output device one is prompted for the energy units to be used:

```
 energies in Ryd.(f) or eV(t)? (default f)
```

Entering t (logical true) selects electron volts, whereas f (logical false) or / selects Rydberg. The next prompt allows a shift of the energy origin to the Fermi energy:

```
  energies relative to EF (t)? (default is f)
```

Now the DOS file is read and information about it's content is displayed:

```
 emin,emax= -1.0 .0, nopts=1001,
 nd=24 efermi=-.1426

 classes are: NA1  NA2  RB1  RB2  SB1  SB2
 l's     are: s  p  d  f
```

Next the projected DOS which should be added and plotted are selected:

```
 Enter class1-l1 class2-l2 .. to be added to DOS
 Examples:   all    s    p    NA1-f  SB2-s
```

If 'all' is entered, then the total density of states is plotted. Entering the name of one or several atoms yields a plot of the partial density of states associated with these atoms. Entering the orbital character (s,p,d,f) makes a ℓ-projected DOS which is summed over all atoms. If the name of a class of atoms augmented by '-' and an orbital character symbol is entered, only the partial density of states associated with these atoms and this specific ℓ-value is plotted. Additive combinations are also possible (e.g. RB1 + s generates a plot of all s-orbitals present and all additional orbitals on the RB1 atoms).
Following a table of all selected orbitals, a choice of the energy range is prompted:

```
Take: NA1-s NA1-p NA1-d NA2-s NA2-p NA2-df
```

Following the table of all selected orbitals you are prompted for the relative weight of each partial DOS:

```
 Now enter weights for partial DOS
 (default=1,1,..)
 A weight of 1.0 for each partial DOS gives
 the correct total DOS.
```

Entering / for the default yields the correct total DOS, but you may enter a number for each partial DOS. Next you are prompted for the energy range:

```
 emin   , emax   =  -1.000    .000
 if desired, enter new emin emax, / for default
```

Entering / instead of two numbers selects the whole range shown above. The energy units are the ones selected in the first step. The default range is defined in the DOS category in the CTRL file. The next prompt permits changing the range of the density of states in exactly the same way:

```
 dosmin, dosmax =.0 1070.419
 if desired, enter new min max,/ for default
/
  new values for emin, emax, dmin, dmax are:
    -1.000       .000      .000     1080.0

 title = all
 if desired, enter new title / for default
```

Finally a title for the plot can be given: Now the plotfile or the plot window should be generated.

G. Fermi Surface plot

Important notice: This program is not yet fully tested for all lattices! Please cross-check the results carefully!

The program *gnufs.run* generates a plot of Fermi surface contours in the unfolded irreducible part of the Brillouin zone. The program reads the file LMFS created by *lm.run* if FS=T in category OPTIONS and creates the plot data files Y1.DAT, Y2.DAT and the command file Y.GNU. After the selection of the output device one is prompted for a 40-character title string for the plot:

```
 enter title:
```

followed by a list of lattices and the prompt for a general scaling factor and the selection of the lattice:

```
Number  Lattice type
1       sc
2       bcc
3       fcc
4       orthor primitive
5       orthor based centered
        different setting for 1248
6       orthor primitive,diff. sett.
7       orthor based centered-real
8       tetrag primitive kz=0 plane
x9      tetrag primitive more planes
```

```
        x=1->4 for different planes
enter: scale and lattice
      e.g.: 8.  4 (orthorhombic)
      or    3.  8 (tetrag)
      or    4. x9 (tetrag for several planes,
               x selects which one)
```

After the scale and the lattice is entered, a short summary of the input is printed and the plot is generated.

XI. COPYRIGHT

APPENDIX A: THE PREPROCESSOR

A preprocessor *ccomp* provides a simplified, FORTRAN compatible version of C conditional compilation. FORTRAN statements beginning with C# are preprocessor directives; the ones implemented now are C#ifdef, C#ifndef, C#else, C#elseif, C#endif (also C#define defines a name). Directives C#ifdef, C#ifndef, C#elseif, and C#endif are followed by a name, e.g. C#ifdef CRAY. when C#ifdef is false (either name is not defined or it lies within an #if/#endif block that is false), *ccomp* comments out until a change of state (new C#ifdef, C#ifndef, C#else, C#elseif, C#endif encountered); C#ifdef is true, *ccomp* uncomments lines following until another conditional compilation directive is encountered.

Conditional compilation blocks may be nested. As with C, *ccomp* distinguishes case. Output is to standard out.

There is a primitive facility to make logical expressions using the AND (&) and OR (|) operators, such C#ifdef john & bill, or C#ifdef john | bill, is provided. Precedence of operators is strictly left to right, so that john | bill & mike is equivalent to (john | bill) & mike, whereas john & bill | mike is equivalent to (john & bill) | mike

How *ccomp* determines whether to modify code:

Whether the lines following a C#ifdef, C#ifndef, C#else, C#elseif, C#endif need to be commented out or uncommented depends on whether they have been commented out in a previous pass. This information is contained in a 'C' following the directive, e.g. C#ifdefC, C#ifndefC, C#elseC, C#elseifC, C#endifC. The preprocessor will set this, it is best advised to create any new blocks uncommented and let the preprocessor do the commenting.

The main programs in directory MAIN: *lm.f, lmstr.f, lmbnd.f, lmdos.f, lmovl.f,* and *lmhart.f* are all obtained from and *lmall.f* in the main directory, by running the latter *ccomp* with different keywords defined. To modify a main program change *lmall.f* and execute *make all.* This will create all the main programs in MAIN and compile and link them. Similarly for the atomic program a non-relativistic version may be generated using the keyword NONREL.

The following line illustrates the use of the preprocessor (after it has been compiled by the C-compiler) IBM (CMS) machine: ccomp -uLM -dLMSTR LM LMSTR. '-uLM' means undefine LM i.e. remove LM specific lines. '-dLMSTR' means define LMSTR i.e. uncommend LMSTR specific lines. 'LM' is the filename of the input file with file type FOR and file mode A. 'LMSTR' is the output file. Multiple defines and undefines are possible.

APPENDIX B: MACHINE DEPENDENCIES

The LMTO programs have been written with portability in mind, and some effort was made to keep within ANSI-77 standards. If you do find any compilation errors in the code as it stands, please report them, so that the code can be made as portable as possible. There is some system dependence lying outside the purview of ANSI. File handling is very operating-system dependent, and LMTO files have been kept as simple as possible to make the code as portable as possible. All files are opened in function *fopna* in directory IOLIB, so whatever machine dependence there is should be confined to that subroutine. Also, different machines have differing internal representations of their numbers. In some places this is important, such as the function *derfc.* There is a collection of three functions in the math library, r1mach, d1mach) and *i1mach* that hold machine-specific information. These must be set to the appropriate machine before compilation. Besides the four routines, *fopna, i1mach, r1mach* and *d1mach*, a few occasional machine-dependencies are found. Perhaps the most important exception occurs in routine *cnvt* in the IOLIB directory. There data read from the control file is converted from ASCII representation to the binary representation, using the FORTRAN internal read facility. This read differs on different machines. The unformatted read should be used if possible; if not an alternate formatted read is supplied. This should be checked when installing this routine on a new compiler. Routine *finits* (also in IOLIB) does some machine-dependent initialization.

1. Machine constants

The following machine dependent constants are used in the TB-LMTO prog. They are supplied in functions in *extens.f*:

dmach(1)=1.d-99 [smallest double precision number]
dmach(2)=1.d+99 [largest double precision number]

dmach(3)=1.d-15 [smallest number which added to 1.0 gives something different from 1.0]
imach(2)=6 [standard output channel]
imach(4)=6 [standard error channel]
imach(6)=4 [no. characters per integer]
imach(17)=1 [no. integer words per real word]
imach(18)=2 [no. integer words per double precision word]

---

[1] [1] O.K. Andersen, Phys. Rev. B **12**, 3060 (1975); O.K. Andersen, Europhysics News **12**, 4 (1981); O.K. Andersen, in *The Electronic Structure of Complex Systems*, edited by P. Phariseau and W.M. Temmerman (Plenum Publishing Corporation, 1984); O.K. Andersen, O. Jepsen, and M. Sob, in *Electronic Band Structure and Its Applications*, edited by M. Yussouff (Springer-Verlag, Berlin, 1986); H.L. Skriver, *The LMTO Method*, (Springer-Verlag, Berlin, 1984).

[2] [2] O.K. Andersen and O. Jepsen, Phys. Rev. Lett. **53**, 2571 (1984); O.K. Andersen, O. Jepsen, and D. Glötzel, in *Highlights of Condensed-Matter Theory*, edited by F. Bassani, F. Fumi, and M.P. Tosi (North-Holland, New York, 1985); O.K. Andersen, Z. Pawlowska, and O. Jepsen, Phys. Rev. B **34**, 5253 (1986); H.J. Nowak, O.K. Andersen, T. Fujiwara, O. Jepsen, and P. Vargas, Phys Rev. B **44**, 3577 (1991).

[3] [3] W.R.L. Lambrecht and O.K. Andersen, Phys. Rev. B **34**, 2439 (1986); O.K. Andersen, T. Paxton, O. Jepsen, and M. van Schilfgaarde (to be published).

[4] [4] O. Jepsen and O.K. Andersen, Solid State Commun. **9**, 1763 (1971) and Phys. Rev. B **29**, 5965 (1984); P. Blöchl, O. Jepsen, and O.K. Andersen, Phys. Rev. B **49**, 16223 (1994).

[5] [5] O.K. Andersen, A.V. Postnikov, and S.Yu. Savrasov, Mat. Res. Symp. Proc. **253**, 37 (Materials Research Society 1992) Chapter V.

[6] [6] O. Jepsen and O.K. Andersen, Z. Phys. B (submitted).

[7] [7] C.J. Bradley and A.P. Cracknell, *The Mathematical Theory of Symmetry in Solids* (Clarendon, Oxford, 1972).

[8] [8] *International Tables for Crystallography*, edited by Theo Hahn (Riedel, Boston, 1987) Vol. A

[9] [9] *Pearson's Handbook of Crystallographic Data for Intermetallic phases*, by P. Villars and L.D. Calvert (American Society for Metals, Metal Park, Oh44073)

[10] [10] O.K. Andersen, O. Jepsen, and G. Krier, (World Scientific, Singapor, 1994) in press

[11] [11] M. Tinkham, *Group Theory and Quantum Mechanics* (McGraw-Hill Book Company, New York, 1964) p. 102