Summary of Li Ju's work at Ames from July 17th to Aug 1st, 96'

Li Ju

July 29, 1996

The purpose of my visit to Ames, initially, was two: 1) To learn and be able to generate LDA database for SiC under various phases. 2) To revise the original TB fitting program from single component to binary systems, such that the code will be workable (as much as possible), and when Lisa comes can concentrate on the fitting part. The latter of course involves which form we are going to use for the binary system.

Now near the end of my visit, it seems that these goals are at mostly fulfilled. I haved already generated the band structure, electronic DOS, cohesive energy curve and charge density plot for diamond (as a test case), and for 3C, CsCl, NaCl, NiAs, anti-NiAs, 2H SiC. The implementations to other cubic and hexagonal systems are quite straightforward. It is worth noting that this high speed is mostly due to the new version of mixed-basis code that had been written at Ames and just been debugged, which is really user-friendly and reliable (most of the time). I'm very grateful to C. T. Chan, whose kind instructions on this and discussions has been most helpful, and to K. M. Ho, who made the Si pseudopotential availble at the right time.

The TB fitting part looks less optimistic at first — mostly because the original code (*band.f*) is a working program and formidably difficult to read. To revise it to accomodate binary system on this basis would be very hard, so I decided to overhaul it. The new program (*new.f*) is almost completely rewritten, which takes standarized input, and is based on the idea of a generic parameter table for binary systems, which has a large capacity for revision without re-structuring the program. The new code is working (although bugs might exist) because it has been tested on pure Si and C and correct band structure were given for all phases.

1 LDA calculation

The input documentation in "/home1/chan/newmixbs/mixb/doku" would be most helpful, the latter German part has been translatd in "document.part2". On the otherhand, there are some options that are not yet in the documentation, such as in plotting out the charge density ("60") and calculating the Hellman-Feynman forces ("55"). In those cases see the main progam "/home1/chan/newmixbs/mixb/src_12.6.96/main3.fp" and corresponding subroutines. The *.fp* source codes were written with C directives in it like "#ifdef" and "#endif". It is then processed by *cpp* and be turned into either *main3.c.f* or *main3.r.f*, which can then be accepted by the Fortran compiler. Here c stands for complex (executable="cmb1") and r stands for real(executable="rmb1"), which is approximately 4 times faster. The real version can be used whenever there is inversion symmetry in the system, such that the plane-wave basis can be decomposed into even- and odd-parity real representations (cos(kx) and sin(kx)). Sometimes the inversion symmetry has to be accomplished by a right choice of origin, most notably in the diamond and graphite structure (this is done automatically by the program). As long as the inversion operation itself doesn't require to be accompanied with any partial translations, it doesn't matter whether the group is symmorphic or not.

Everything is automatic now. To calculate the cohesive energy curve or to get some extra data points, go to the directory and type

```
% runlist coh8 inp 105. 140. 178.
```

The shell script *runlist* will automatically generate the INP file according to the format of *inp*, which is what "rmb1" or "cmb1" takes, with volume 105, 140 and 178 $a.u.^3$ respectively. After finished it will then check the original OUT file and clip down the total energy. In the end everything will be in the file *coh8*.

We usually fit the data points to the Univeral Binding curve by Rose, Ferrante and Smith[1], in order to get the equilibrium lattice constant, total energy, and bulk modulus. This is done by the program /home7/liju/Bin/fit, which takes input file *fitin* like this:

```
3C-SiC nrk=19(5*5*5) 28 Ryd curoff Xc=ca  <--- arbitary heading
 -19.412    2.2    140.<--- guess value of energy, bulk modulus and cell volume at min.
 -19.42    -19.39      <--- lower and upper bound of minimum energy at min.
  0.001     8.         <---                         of bulk modulus at min.
  130.     150.        <---                         of cell volume at min.
   0.0001              <--- DEL: fraction of the parameter range in initial division
    10                 <--- NDIV: number of refinements
   13    1    0        <--- number of data points, 0/1(quadratic/universal binding),
110.    -19.3591203675
```

[1]PRL 47, 675, 1981

```
115.    -19.3798026532
120.    -19.3947307605
125.    -19.4046077868
130.    -19.4103123655
135.    -19.4125788902
140.    -19.4119553075 <--- datas: volume in a.u.^3, energy in Ryd.
145.    -19.4088759887
150.    -19.4036211206
155.    -19.3967326799
160.    -19.3883006834
165.    -19.3786777010
170.    -19.3680663447
 -0.                   <--- origin of energy: for absolute cohesive energy
                        we should use isolated atomic total energy using the
                        same pseudopotential and exchange-correlation functional,
                        with spin-polarization correction for open-shell systems.
```

and then

```
% fit < fitin
```

The *fit* code actually use the same optimization subroutine *mina*, that we are going to use in TB fitting. Its strategy is to devide the parameter range into meshes, initially defined by DEL: the smaller the DEL, the finer the mesh. It then refine the mesh by NDIV times in order to locate the minima. fit will give two output files "fitout", which contains bulk modulus etc., and "fitdat", that contains a continuos curve which is the fit.

In using *fit*, it is imperative to always keep an eye on the error: usually it's considered to be an acceptable fit only when the error is smaller than 0.003 Ryd. The best way to check is to plot out the "fitdat" upon the raw data and compare. It is not very easy to achieve good fit, either, so do not put a collection of data with too big volume variations in the input (we shouldn't expect the universal binding curve to be so universal!). For instance, if the minimum is at 150. $a.u.^3$, although you can generate data points from 70. to 300. $a.u.^3$, that is, approximated a span of 4 in total range, you may only need to use datas between 100. and 200. $a.u.^3$ to get the fit. If you want better bulk modulus, lattice constants, etc., you should try to get more points inside this range.

After you have located the minima, you can proceed to get more detailed information at this point. For instance in 3C-SiC, after you know that the minimum is at 137.813 $a.u.^3$ by fitting the ab initio data generated from using 25 Ryd plane wave cutoff and 5*5*5 mesh, you can upgrade the accuracy by using the following INP file at this volume:

```
Get 3C-SiC's converged charge density at equilibrium volume.
```

```
%define MB_SMALL_TOL
%define MB_TIMING
%define PRT_SYM_R
10
    2    2     Zinc_Blend
    1    C  <--- C must proceed before Si.
   0.125          0.125          0.125   <--- NOTE: displacement in lattice vectors,
    1   Si       not Cartesian coordinates!
   -.125          -.125          -.125
  -137.813                               <--- minimum volume
   0.0            0.5            0.5
   0.5            0.0            0.5     <--- Lattice vectors in terms
   0.5            0.5            0.0          of Cartesian coordinates.
phonon=No     The program will automatically
  nops=-1   c=1     zinc                    normaliza the length according to the
20              volume.
  35.00   24.0     0.0       0.0       corr=ca  <-- upgrade to 35 Ryd. plane wave cutoff.
    2    1
    1    0   20   1.500   1.200  <-- local orbital cutoff at 1.2 a.u.,
    1    1   20   1.500   1.200      such that don't touch at high density.
  300    1
    C s=lo p=nl d=nl 1200  100  100   0.020 .070000 .070000  06.000
   Si s=lo p=nl d=nl  600   60   60   0.020 .083330 .083330  14.000
30
    0cc  <-- Cohen-Chadi sampling of charge density. Delete cc would become Monkhost-Pack!
   10    1   04       10   10   10  0.5   0.5   0.5 <--- upgrade to 10*10*10.
40
  niter= 10    scr=tf    ifmet= 00i    gamma=   0.000g
  nsprt= 20                      ^-- change to -1m if no gap.
  nsmix=  0    ekmix=  36.0
00
```

The most important result after running, using this INP, is the converged charge density, stored in file CD. Be sure to do this

```
% cp CD CD.converged
% cat INP OUT > OUT.converged
```

immediately after because CD is refreshed every time you run "rmb1" and you will be running the risk of losing this valuable file if you are to do anything in this directory. The INP and OUT file are also important because 1) INP is also easy to lose. 2) OUT contains the the Fermi energy and the band eigenvalues (although may not be your desired k points), which can be used to check your later results.

To plot out the valence charge density (see source code *denplt.fp*), you can change to "niter=00", because you don't need to diagnolize the plane-wave matrix even once, and "scr=cd", meaning that you will read the screening potential from file CD (be sure that it is the converged CD!), and you need to add the following option to INP:

```
60       <-- valence charge density. "61" will be total minus atomic CD
    1    <-- only one plane
    2    0    0   -1    1    1  <-- (200) and (-1,1,1) in lattice vectors, which
 -.125 -.125  -.25 0.002 0.002    in Cartesain coordinates would be (011) and (100)
    ^                             direction in the zig-zag plane.
    ^____ origin of the plot.
```

It will then generate a file RHOV1, which when ftped to a machine that has Matlab, can be process by the Matlab program *den.m*.

To plot out the band structure, you only need to revise an input file called con_band:

```
Number of Important K-points:
5

Gamma
0  0  0   <--- in terms of the reciprocal lattice vectors.

X
0  0.5  0.5

W
0.25  0.5  0.75

K
0.375 0.375 0.75

L
0.5  0.5  0.5


Total Number of Links:
5

From       To       Sections
 1          2         50   <--- Gamma to X
 2          3         50   <--- X to W
 3          4         50
 4          1         50
 1          5         50
       ^---------- how many k points during the inteval.
constant of length (a.u.)
```

```
8.1994249   <---- lattice constant, or any convienient length scale. It should be
          the same quantity as the "Constant of length (A):" in the input file
                  for the TB fitting program new.f, although new.f require you to
                  convert it to angstrom as input. You can read this quantity from OUT.converge
                  but be sure to multiply by a factor (like 2) because they may not be the
  lattice constant! For instance, you may see this in OUT.converged,

                  lattice vectors (a.u.)
                  0.00000000     4.09971245     4.09971245
                  4.09971245     0.00000000     4.09971245
                  4.09971245     4.09971245     0.00000000

  but 4.09971245 a.u. is not the lattice constant!

reciprocal basis (a.u.^-1)
     -0.76629585     0.76629585     0.76629585
      0.76629585    -0.76629585     0.76629585
      0.76629585     0.76629585    -0.76629585 <--- this can be taken from the front
           part of the OUT.converged.
```

and then just type

```
% band con_band
```

and the band structure file "KM.out" would be ready as result. How simple! But the process it went through is not simple. The Shell script *band* looks like

```
#
# Shell script that combine the efforts to plot out
# the band structure.
#
# In order for it to be correct, the converged charge
# density must be in CD.converged

rm KM.out
kp < $1
# first generate the k-points using the Fortran program.
cp INP.band INP
cp CD.converged CD
run
cat OUT | select '  kp      ms      energies   and   (kinetic energy)   both in [
ev]' | cut -b 16-100 > 1
sortk 1
paste line 1 >> KM.out
```

```
cp CD.converged CD
#
```

and INP.band looks like

```
Plot out 3C-SiC's band with k-points in IBZKPTS (generated by kp.f)
%define MB_SMALL_TOL
%define MB_TIMING
%define PRT_SYM_R
10
    2    2     Zinc_Blend
    1    C
   0.125          0.125          0.125
    1   Si
   -.125          -.125          -.125
  -137.813
   0.0            0.5            0.5
   0.5            0.0            0.5
   0.5            0.5            0.0
phonon=No
  nops=-1   c=1     zinc
20
  30.00   24.0     0.0       0.0       corr=ca
    2    1
    1    0   20   1.500   1.200
    1    1   20   1.500   1.200
  300    1
    C s=lo p=nl d=nl 1200  100  100   0.020 .070000 .070000  06.000
   Si s=lo p=nl d=nl  600   60   60   0.020 .083330 .083330  14.000
30
   00ff
   10    1   04
40
  niter= 01    scr=cd    ifmet= 00i    gamma=   0.000g
  nsprt= 20
  nsmix=  0    ekmix=  36.0
00
```

Notice that this time after we read in the converged charge density by "scr=cd", we have to do one iteration to get the eigenvalues under this effective Kohn-Sham one-body potential. Option "30" with "ff" will read in the k points from a file IBZKPTS, which is generated by /home7/liju/Bin/kp (source code is /home7/liju/Bin/Src/kp.f) by analysing the input file *con_band*. The resulting single-particle eigenvalues would be in file OUT, which the program *select* will pick out (source code /home7/liju/Bin/Src/C/select.c) and packed into one row for each k by the program *sortk* (source code /home7/liju/Bin/Src/C/sortk.c). It is then pasted with a *line* file which is generated by *kp*. In the end KM.out looks like:

```
.

.00000    .00000    .00000  1   .00000  -2.4877   12.6248   12.6248   12.6248   ...
 ^---(kx, ky, kz) in        ^    ^          ^---- eigenvalues for this k. ----
.   Cartesian coord.        ^    ^------ a "ruler", that is, to measure how long
.   and in terms of Pi/A,   ^ -- nonzero denotes that    it has traversed in the
.   which is directly usable     this is a special pt,   band structure plot, with
.   for the input file con       in this case Gamma.     Cartesian measure, to be
    for new.f.  used in band.m
```

This standarized KM.out, which is also one of the outputs of the TB fitting program *new.f*, using the optimized TB parameters after optimization, is acceptable by the Matlab program band.m. At each "special point" it will draw a vertical line, and it scale the x-axis according to "ruler", which measures how long in Cartesian distance that the k-points has traversed.

To calculate the total electronic DOS, use INP.totdos.plot

```
Plot out 3C-SiC's total electronic DOS
%define MB_SMALL_TOL
%define MB_TIMING
%define PRT_SYM_R
10
    2    2     Zinc_Blend
    1    C
   0.125          0.125          0.125
    1   Si
   -.125          -.125          -.125
  -137.813
   0.0            0.5            0.5
   0.5            0.0            0.5
   0.5            0.5            0.0
phonon=No
  nops=-1   c=1     zinc
20
  30.00   24.0     0.0       0.0       corr=ca
    2    1
    1    0   20   1.500   1.200
    1    1   20   1.500   1.200
  300    1
    C s=lo p=nl d=nl 1200  100  100   0.020 .070000 .070000  06.000
   Si s=lo p=nl d=nl  600   60   60   0.020 .083330 .083330  14.000
30
    Occ
   10    1   04       12   12   12  0.5   0.5   0.5
40
  niter= 01    scr=cd    ifmet= 00i    gamma=   0.000g
```

```
  nsprt= 20
  nsmix=  0    ekmix=  36.0
70
  300  -2.488  20.000   0.200
       yes ^     ^       ^---------- Gaussian smearing width (eV).
    0      ^     ^---maximum energy
00         ^-- minimum energy (in eV), usually the
               lowest eigenvalue at Gamma point.
```

Notice that the charge density/eigenvalue sampling mesh has been made denser, to $12 \times 12 \times 12$, because otherwise we will get pretty oscillatory DOS (if the Gaussian smearing width is made too big we'll lose structure), as all DOS calculations do. The output TOTDOS contains three columns, the first column is the energy, the second is the DOS, the third is the integral of DOS. It can be plotted out by the Matlab program *totdos.m*. If this material is an insolator ("ifmet= 00i"), then we can find out the Fermi energy (highest occupied eigenvalue) in OUT.converged. If this thing is a metal ("ifmet= -1m"), then the energy scale of TOTDOS will be automatically shifted to make the Fermi level positioned at 0 eV.

2 Tight-Binding Optimization

We first realized that currently there is still not a very reliable theory to provide a first-principle account of the scaling functional form for the hopping integrals and on-site energies. It would be very interesting to do some theoretical research in this direction to see at least how they come in and take which asymptotic behaviour. Empirical tight-binding formalism assumes that 1) there does exist such an functional. 2) is simple enough that can be closely approximated by a family of functions, which takes the form of:

$$\alpha_2 R^{\alpha_2} \times \exp(-\alpha_3 R^{\alpha_4}) \times \text{screening} \tag{1}$$

in the current environmental-dependent model, i.e., power law+exponential+screening, and R is the rescaled distance related to the coordination number on two sides

$$R = r f(g_i, g_j) \tag{2}$$

$f(g_i, g_j)$ is a linear function now controlled by δ. Since this form is complicated enough, and makes rough physical sense, we believe that whatever the true functional is, by adjusting these parameters we can achieve good approximation to it. And indeed this model had been quite successful on carbon.

The screening factor is the central idea of the environmental-dependent TB model. To better understand it, let's draw the so called "constant-screening contours"(CSC). For instance, the functional form $f((r_{il} + r_{jl})/r_{ij})$ implies CSC of ellipses with symmetric focuses on the two atoms, because $r_{il} + r_{jl} =$ constant is just an definition of ellipse for $\mathbf{r}_l$. Generally speaking, the CSC of $ss\sigma, sp\sigma, pp\sigma$ must be axially symmetric (the $pp\pi$ bond is an exception), no matter in what situation. So we can use a general functional form that represents these axially-symmetric screenings. In the first order, I believe that the following form can be used:

$$(r_{il} + r_{jl} + f \cdot r_{lf})/r_{ij}) \tag{3}$$

where $\mathbf{r}_f$ is an extra force center lying in the axis between i and j:

$$\mathbf{r}_f = \mathbf{r}_i + d(\mathbf{r}_j - \mathbf{r}_i) \tag{4}$$

By adding two more "form factor" parameters, d and f, we can describe the screening in non-symmetric situations like Si-C or $sp\sigma$ hopping integrals. Even in symmetric situations like Si-Si $ss\sigma$, although d has to be 0.5, we would still have some control over the shape of CSC by varying f.

Once we accept the general family of scaling functionals to be of the form (1), we can construct a table including all the possible paramters that could occur in a binary system.

Generic Parameter Table for Si, C systems

		strength					screening					
		α_1	α_2	α_3	α_4	δ		β_1	β_2	β_3	d	f
Si-Si	1:ΔE_s											
	2:ΔE_p						Si					
	3:$ss\sigma$											
	4:$sp\sigma$						—	—	—	—	—	—
	5:$pp\sigma$											
	6:$pp\pi$						C					
	7:ϕ_{Si-Si}											
Si-Si	8:ΔE_s											
	9:ΔE_p						Si					
	10:$ss\sigma$											
	11:$sp\sigma$						—	—	—	—	—	—
	12:$pp\sigma$											
	13:$pp\pi$						C					
	14:ϕ_{C-C}											
Si-C	15:ΔE_s(Si)											
	16:ΔE_p(C)											
	17:ΔE_s(Si)						Si					
	18:ΔE_p(C)											
	19:$ss\sigma$											
	20:$sp\sigma$					—	—	—	—	—	—	—
	21:$sp\sigma$											
	22:$pp\pi$						C					
	23:$pp\pi$											
	24:ϕ_{Si-C}											

3) Software Enviroments

When Lisa comes, she'll use C.Z.'s account on the "vincent" network (which is a clone of Athena) of Iowa State Univ., most probably in Room 510. It has emacs and Matlab ("add matlab; matlab"). Our directory is in "liju.dir".