Notes on LAVIB Package Source Codes

Hao Feng[*]

College of Physics, Sichuan University,

Chengdu, Sichuan, 610065, P.R.China

(Dated: March 23, 2006)

Abstract

This document is about all of the source codes of LAVIB package, which is modified from the original one of Prof. Michael A. Morrison's group in Oklahoma University. It is useful to note the detailed reasons behind the changes, so the detailed descriptions of what have been modified are documented here.

[*]Electronic address: `ddsteed@163.com`

Contents

I. CODE CONVENTION

Some original source codes are NOT correctly indented, so I have to format them line by line. I know that it will be difficult to compare the original source code and the new one and find what have been modified if every line is different. However, it is very helpful for the programmer to read and understand the source code if good indentation convention is made consistently. So I formatted all of the source codes by using GNU Emacs editor and the indentation is 3 spaces except for the classic math library function.

I shall account for every modification for each source code in this document. (NOTE: To generalize this package to e-N_2, some arrays are augmented. I do not explain it one by one.)

A. Data Format

In original codes, the potential data are with format of “D23.16”. So the local potential are like

```
  0.01000-0.1749182639341561E+00
```

in which the first is the distance between the incident electron and the target, the second is the local potential. They are too close to be read as free format and some other software, such as xmgr, cannot read them. So I modified the second as “D24.16”, which added one blank between them.

However, Andy reminded me that this package should be compatible with other codes. If I modified the format as “D24.16”, the other codes will not read them correctly. So I kept all the data formats.

II. ALAMN.F

`alamn` is to calcuate a molecular charge distribution and expand it in Legendre polynomials for a linear diatomic or triatomic molecule.

A. Correction to large-r fit of a_λ

In subroutine `ALAM`.

1. *Basic Concepts*

Suppose the charge density of a closed-shell N-electron linear molecule be expanded in Legendre polynomials $P_\lambda(\cos\theta)$, viz.,

$$\rho(r,\theta) = \sum_{\lambda=0}^{\lambda_max} a_\lambda(r) P_\lambda(\cos\theta) \tag{1}$$

From the orthogonality relation for the Legendre polynomials, we find that the λth expansion coefficient in Eq. (1) is given as a function of r by

$$a_\lambda(r) = \frac{2\lambda+1}{2}\int_0^\pi \rho(r,\theta) P_\lambda(\cos\theta)\sin\theta\,\mathrm{d}\theta \tag{2}$$

The integral in Eq. (2) is evaluated for values of r in a user-prescribed n-point mesh $\{r_i,\ i = 1, 2, \ldots, n\}$ by a 32-point Gauss-Legendre quadrature.

As r becomes large, the expansion coefficients $a_\lambda(r)$ of the molecular charge density $\rho(r,\theta)$ [Eq. (2)] can be fitted very accurately to the analytic form

$$a_\lambda(r) = A_\lambda r^{p_\lambda} \exp(-\alpha_\lambda r) \tag{3}$$

where A_λ, p_λ and α_λ are real numbers.

In Ref.[?], these parameters are determined by using the last three values of r at which $a_\lambda(r)$ is evaluated from Eq. (2), r_n, r_{n-1} and r_{n-2}. Letting a_n, a_{n-1} and a_{n-2} denote the values of a_λ at these values of r, we have

$$p_\lambda = \frac{h \ln\left(\dfrac{a_{n-1}}{a_n}\right) + \ln\left(\dfrac{a_{n-1}}{a_{n-2}}\right)}{h \ln\left(\dfrac{r_{n-1}}{r_n}\right) + \ln\left(\dfrac{r_{n-1}}{r_{n-2}}\right)} \tag{4}$$

where

$$h = \frac{r_{n-1} - r_{n-2}}{r_n - r_{n-1}} \tag{5}$$

And

$$\alpha_\lambda = \frac{1}{r_{n-1} - r_n} \ln\left[\frac{a_n}{a_{n-1}} \left(\frac{r_{n-1}}{r_n}\right)^{p_\lambda}\right] \tag{6}$$

$$A_\lambda = a_n r_n^{-p_\lambda} \exp(\alpha_\lambda r_n) \tag{7}$$

NOTICE: There is a print error in Eq. (16) in Ref.[?], the source code of ALAMN is consistent with Eq. (7).

Thus the expansion coefficients and hence the charge density need not be calculated for $r > r_n$ even if $a_\lambda(r)$ is required for large-r values of r.

2. Saha's correction to e-N$_2$

We can get well-behaved large-r a_λ and static potential V_{st} for e-H$_2$ by using Eq. (3), Eq. (4), Eq. (6) and Eq. (7). But for e-N$_2$, if we choose the r-mesh as

0.00	0.01	1.20
1.20	0.02	2.00
2.00	0.04	4.40
4.40	0.08	6.00

and choose LAMMAX = 14, we get $A_{14} = 3.093 \times 10^{18}$, $p_{14} = -65.58$ and $\alpha_{14} = -8.408$ for $R = 2.020a_0$. Meanwhile, the static potential (v_λ) for $\lambda = 14$ are

r	v_λ
...	...
5.04000	-0.1131232954588831E+93
5.12000	-0.1410275599185811E+93
5.20000	-0.1752150266189247E+93
5.28000	-0.2169698823790784E+93
5.36000	-0.2678129626035787E+93
5.44000	-0.3295407561285985E+93
5.52000	-0.4042701040798804E+93
5.60000	-0.4944893200460516E+93
5.68000	-0.6031165389052729E+93
5.76000	-0.7335661891117883E+93
5.84000	-0.8898245787188291E+93
5.92000	-0.1076535689486381E+94
6.00000	-0.1299098386733144E+94

They are overflow! (They are too LARGE for other high $\lambda's$ and other internuclear distances!)

I searched our store directory and found that B. C. Saha wrote the following codes in the subroutine `ALAM`

```
... ... ...
C      PSM=(DLOG(ABY2/ABY1)+RAT*DLOG(ABY2/ABY3))/(DLOG(R2/R1)+RAT*DLOG(R2
C     #/R3))
c---
       PSM=0.D+00
C      IF(IFIX.EQ.1.AND.PSM.GT.1.D+00) PSM=1.D+00
       QSM = -DLOG((ABY3/ABY2)*(R2/R3)**PSM)/(R3-R2)
C See the note for this modification 8/23/85
       ASM = Y3*R3**(-PSM)*DEXP(QSM*R3)
```

We don't have his note, but from his codes I can guess his formula for large-r a_λ is

$$a_\lambda(r) = A_\lambda \exp(-\alpha_\lambda r) \tag{8}$$

By using Saha's formula we get $A_{14} = 1.351 \times 10^{-3}$, $p_{14} = 0.0$ and $\alpha_{14} = 2.595$. The static potential (v_λ)for $\lambda = 14$ are

r	v_λ
...	...
5.04000	.2155357141168185E-08
5.12000	.1772666466746914E-08
5.20000	.1458346809604667E-08
5.28000	.1200442725400119E-08
5.36000	.9890022814558143E-09
5.44000	.8157512135613536E-09
5.52000	.6738244745961844E-09
5.60000	.5575415032036261E-09
5.68000	.4622160475180030E-09
5.76000	.3839942571407144E-09
5.84000	.3197165625943484E-09
5.92000	.2667999764303344E-09
6.00000	.2231387506188567E-09

3. New correction to large-r fit

Although Saha's codes corrected the large-r fit overflow, we can't reproduce e-H$_2$ static potential because Eq. (8) is different from Eq. (3). I guess that for some e-targets, the parameters of A_λ, p_λ and α_λ cannot be chosen ONLY by the last three a_λ. So I will fit them by using more $a'_\lambda s$.

Eq. (3) can be rewritten as

$$\ln a_\lambda = \ln A_\lambda + p_\lambda \ln r - \alpha_\lambda r \tag{9}$$

Letting

$$b_\lambda = \ln a_\lambda \tag{10a}$$

$$g_{0\lambda} = \ln A_\lambda \tag{10b}$$

$$g_{1\lambda} = -\alpha_\lambda \tag{10c}$$

$$g_{2\lambda} = p_\lambda \tag{10d}$$

Then

$$b_\lambda = g_{0\lambda} + g_{1\lambda} r + g_{2\lambda} \ln r \tag{11}$$

So b_λ is linearlly dependent on the fitting coefficients. We adopt "General Linear Least Squares" routine — SVDFIT (Signular Value Decomposition fit) [**?**] to fit the coefficients.

By using 10 a_λ to fit Eq. (11) we get $A_{14} = 2.597 \times 10^{-4}$, $p_{14} = 6.346$ and $\alpha_{14} = 4.627$. The static potential (v_λ) for $\lambda = 14$ are

r	v_λ
...	...
5.04000	0.2154651146898829E-08
5.12000	0.1771786323990111E-08
5.20000	0.1457253305351663E-08
5.28000	0.1199088632125510E-08
5.36000	0.9873308802109118E-09
5.44000	0.8136945736411463E-09
5.52000	0.6713014542439685E-09
5.60000	0.5544554312783496E-09
5.68000	0.4584520410108206E-09
5.76000	0.3794161238428322E-09
5.84000	0.3141632320379888E-09
5.92000	0.2600813953949083E-09
6.00000	0.2150311719575100E-09

By using 80 a_λ to fit Eq. (11) we get $A_{14} = 25.275$, $p_{14} = -10.067$ and $\alpha_{14} = 1.217$. The static potential for $\lambda = 14$ are

r	v_λ
...	...
5.04000	0.2155529635889707E-08
5.12000	0.1772881510956653E-08
5.20000	0.1458613984169379E-08
5.28000	0.1200773569363670E-08
5.36000	0.9894106528899811E-09
5.44000	0.8162537098927050E-09
5.52000	0.6744409210914880E-09
5.60000	0.5582955194052478E-09
5.68000	0.4631357026518250E-09
5.76000	0.3851128268835364E-09
5.84000	0.3210734010977414E-09
5.92000	0.2684415191666285E-09
6.00000	0.2251196654564229E-09

I still keep the linear equation method (LEM) to solve A_λ, p_λ and α_λ by solving Eq. (4), Eq. (6) and Eq. (7). So for H_2, we have two methods to calculate large-r a_λ and static potential (v_λ). For $\lambda = 6$, we get

	LEM	SVDFIT
A_λ	0.1601316757808016E-23	0.1189776202324306E-04
p_λ	0.4523276259430859E+02	0.6724227178999224E+01
α_λ	0.7923329439442724E+01	0.3361975965910192E+01

and static potential (v_λ)

	v_λ (LEM)	v_λ (SVDFIT)
...	...	...
8.90000	-0.3738601412980983E-07	-0.3738604465762323E-07
9.00000	-0.3457238352675295E-07	-0.3457241617130511E-07
9.10000	-0.3199833271303255E-07	-0.3199836759524417E-07
9.20000	-0.2964111789027305E-07	-0.2964115513652894E-07
9.30000	-0.2748037669302244E-07	-0.2748041643535470E-07
9.40000	-0.2549785533465599E-07	-0.2549789771093117E-07
9.50000	-0.2367716973439954E-07	-0.2367721488850992E-07
9.60000	-0.2200359606053337E-07	-0.2200364414259235E-07
9.70000	-0.2046388677921551E-07	-0.2046393794575718E-07
9.80000	-0.1904610885409454E-07	-0.1904616326827752E-07
9.90000	-0.1773950121435192E-07	-0.1773955904616741E-07
10.00000	-0.1653434900198538E-07	-0.1653441042846955E-07

B. Fix the bug of calculating the harmonic coefficient of π-orbital

In subroutine SPHPRJ, when the π-orbital is calculated, the original code has a minor bug. The original code is

```
DO MOIND = 1, NMO
... ... ...
if (M .EQ. 2) GO TO 101
ENDDO
101 continue
```

$M = 2$ means the orbital is π_y. In this code, the xz-plane can be chosen so that π_y does not contribute. So we skip π_y orbital. However, N_2 has 7 bound orbitals which include one π_x and one π_y. If the π_x is before π_y, such as for $R = 2.020a_0$, the original code is right. But if the π_y is before the π_x, like $R = 1.700a_0$, the original code will NOT calculate the a_λ of π_x. To fix the bug, there's a simple way,

```
DO MOIND = 1, NMO
... ... ...
```

```
if (M .EQ. 2) GO TO 101
101 continue
ENDDO
```

i.e., for π_y orbital, we do not jump out of the do loop. So whether π_y is before π_x or not, we can get the correct a_λ of π-orbital.

However, someone else has calculated e-N_2 by using `alamn`. Why is there still a bug? I think that it would be right if all `ENDDO` are replaced by `CONTINUE`. i.e.,

```
DO 110 MOIND = 1, NMO
... ... ...
if (M .EQ. 2) GO TO 101
101 continue
```

the loop would exit only when `MOIND` = `NMO`.

C. Input of GTO Basis

In the original codes, the GTO basis sets are generated by POLYATOM package. The basis sets are like

```
3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1
H1         S           33.644400   0.0253740
H1         S           5.0579600   0.1896830
H1         S           1.1468000   0.8529300
H1         S           0.3211440   1.0000000
H1         S           0.1013090   1.0000000
H1         X           2.2280000   1.0000000
H1         X           0.5183000   1.0000000
H1         Y           2.2280000   1.0000000
H1         Y           0.5183000   1.0000000
H1         Z           2.2280000   1.0000000
H1         Z           0.5183000   1.0000000
H2         S           33.644400   0.0253740
```

```
H2          S             5.0579600   0.1896830
H2          S             1.1468000   0.8529300
H2          S             0.3211440   1.0000000
H2          S             0.1013090   1.0000000
H2          X             2.2280000   1.0000000
H2          X             0.5183000   1.0000000
H2          Y             2.2280000   1.0000000
H2          Y             0.5183000   1.0000000
H2          Z             2.2280000   1.0000000
H2          Z             0.5183000   1.0000000
```

However, we are currently using GAMESS package to calculate the bound orbitals and the electronic density of the target. Moreover, the polarization potential (BTAD/DSG) are also calculated by the modified GAMESS. So I modified the format of `ITYPE` considering the output of GAMESS, (for example, for N_2 the basis sets are like)

```
 4 2 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 2 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1      S         5909.440000        .006240
 1      S          887.451000        .047669
 1      S          204.749000        .231317
 1      S           59.837600        .788869
 1      S           19.998100        .792912
 1      S            2.686000        .323609
 1      S            7.192700       1.000000
 1      S             .700000       1.000000
 1      S             .213300       1.000000
 1      S             .060000       1.000000
 1      X           26.786000        .038244
 1      X            5.956400        .243846
 1      X            1.707400        .817193
 1      Y           26.786000        .038244
 1      Y            5.956400        .243846
```

```
1      Y           1.707400      .817193
1      Z          26.786000      .038244
1      Z           5.956400      .243846
1      Z           1.707400      .817193
1      X            .531400     1.000000
1      Y            .531400     1.000000
1      Z            .531400     1.000000
1      X            .165400     1.000000
1      Y            .165400     1.000000
1      Z            .165400     1.000000
1      X            .050000     1.000000
1      Y            .050000     1.000000
1      Z            .050000     1.000000
1     XX            .980000     1.000000
1     YY            .980000     1.000000
1     ZZ            .980000     1.000000
1     XY            .980000     1.000000
1     XZ            .980000     1.000000
1     YZ            .980000     1.000000
1     XX            .160000     1.000000
1     YY            .160000     1.000000
1     ZZ            .160000     1.000000
1     XY            .160000     1.000000
1     XZ            .160000     1.000000
1     YZ            .160000     1.000000
2      S        5909.440000      .006240
2      S         887.451000      .047669
2      S         204.749000      .231317
2      S          59.837600      .788869
2      S          19.998100      .792912
2      S           2.686000      .323609
2      S           7.192700     1.000000
```

```
2    S          .700000    1.000000
2    S          .213300    1.000000
2    S          .060000    1.000000
2    X        26.786000     .038244
2    X         5.956400     .243846
2    X         1.707400     .817193
2    Y        26.786000     .038244
2    Y         5.956400     .243846
2    Y         1.707400     .817193
2    Z        26.786000     .038244
2    Z         5.956400     .243846
2    Z         1.707400     .817193
2    X          .531400    1.000000
2    Y          .531400    1.000000
2    Z          .531400    1.000000
2    X          .165400    1.000000
2    Y          .165400    1.000000
2    Z          .165400    1.000000
2    X          .050000    1.000000
2    Y          .050000    1.000000
2    Z          .050000    1.000000
2   XX          .980000    1.000000
2   YY          .980000    1.000000
2   ZZ          .980000    1.000000
2   XY          .980000    1.000000
2   XZ          .980000    1.000000
2   YZ          .980000    1.000000
2   XX          .160000    1.000000
2   YY          .160000    1.000000
2   ZZ          .160000    1.000000
2   XY          .160000    1.000000
2   XZ          .160000    1.000000
```

```
 2    YZ           .160000      1.000000
```

So, the `ITYPE` are modified from

```
 DATA  ITYPE/'S  ','X  ','Y  ','Z  ','XX ','YY ','ZZ ','XY ','XZ ',
X'YZ ','XXX','YYY','ZZZ','XXY','XXZ','XYY','YYZ','XZZ','YZZ','XYZ'/
```

to

```
 DATA  ITYPE/'  S','  X','  Y','  Z',' XX',' YY',' ZZ',' XY',' XZ',
$      ' YZ','XXX','YYY','ZZZ','XXY','XXZ','XYY','YYZ','XZZ','YZZ',
$      'XYZ'/
```

III. VLAM.F

IV. VIBKER.F

`vibker` is to calculate the exchange kernels at a given internuclear geometries so they may be used in the construction of a vibrational exchange kernel. Now, it is ONLY for *homonuclear* diatomic molecule.

A. Minor Modifications

- In Original codes, the names of input/output files are read directly through the shell script. In subroutine `indata`

```
            read(5,*)sphnam
            indsph=index(sphnam,' ')
            read(5,*)kernfil
            indker=index(kernfil,' ')
            read(5,*)(geomnam(i),i=1,ngeom)
            write(6,*)
            write(6,*)' The kernel filenames (symmetry label will be added):'
            do i=1,ngeom
              indgeo(i)=index(geomnam(i),' ')
```

```
       if(indgeo(i).eq.0) indgeo(i)=9
       write(6,*)sphnam(1:indsph-1)//geomnam(i)(1:indgeo(i)-1)//dat,
+      ' ',kernfil(1:indker-1)//geomnam(i)(1:indgeo(i)-1)//dat
     end do
```

and in the `main` function,

```
     open(unit=10,file=sphnam(1:indsph-1)//
+          geomnam(igeom)(1:indgeo(igeom)-1)//dat,status='old',
+          form='unformatted')
     ... ... ...
     open(unit=11,file=kernlfil(1:indker-1)//
+           geomnam(igeom)(1:indgeo(igeom)-1)//
+           symlab(isymind)(1:2)//ker,
+           form='unformatted')
```

For example, `sphnam` is "h2sphrj", `kernfil` is "h2" and `geomnam` is "r14". However, it is not convenient since every name should be less than 8 characters—it is often forgotten. So I copy the input file as `fort.*` and read the data through the corresponding channel in the code.

- In original codes, all the geometries to be calculated are read as an array `geomnam`. I remove the number of geometries and calculate one given geometry once so that I can loop different geometry in the shell script — it is clearer and more understandable.

B. π-orbital

The original code could only deal with σ-orbital, so we have to augment it to include π-orbital for e-N_2 scattering. The general exchange kernel is

$$\mathrm{K}^{(\Lambda)}(\ell\ell'|r_1 r_{N+1};R_\alpha) = \sum_{i=1}^{N_{occ}} \sum_{\ell''\ell'''} u_{\ell''}^{(i)}(r_{N+1};R_\alpha) u_{\ell'''}^{(i)}(r_1;R_\alpha) \sum_\lambda \mathrm{g}_\lambda(\ell\ell'\ell''\ell''';\Lambda,m_i) \frac{{r_<}^\lambda}{{r_>}^{\lambda+1}} \tag{12}$$

with

$$\begin{aligned}\mathrm{g}_\lambda(\ell\ell'\ell''\ell''';\Lambda,m_i) &= \sqrt{\frac{(2\ell+1)(2\ell'+1)}{(2\ell''+1)(2\ell'''+1)}}\,\mathrm{C}(\ell\lambda\ell'';-\Lambda,\Lambda-m_i)\,\mathrm{C}(\ell\lambda\ell'';00)\\ &\otimes \mathrm{C}(\ell'\lambda\ell''';\Lambda,m_i-\Lambda)\,\mathrm{C}(\ell'\lambda\ell''';00)\end{aligned} \tag{13}$$

If the isolated molecule only contains σ and π orbitals, both of which are fully occupied, we have,

$$\begin{aligned}\mathrm{K}^{(\Lambda)}(\ell\ell'|r_1 r_{N+1};R_\alpha) = & \sum_{i=1}^{N_{occ}(\sigma)} \sum_{\ell''\ell'''\lambda} u_{\ell''}^{(\sigma_i)}(r_{N+1};R_\alpha) u_{\ell'''}^{(\sigma_i)}(r_1;R_\alpha) \frac{{r_<}^\lambda}{{r_>}^{\lambda+1}} \mathrm{A}(\ell\ell'\ell''\ell''';0,\Lambda) \\ & + \sum_{i=1}^{N_{occ}(\pi)} \sum_{\ell''\ell'''\lambda} u_{\ell''}^{(\pi_i)}(r_{N+1};R_\alpha) u_{\ell'''}^{(\pi_i)}(r_1;R_\alpha) \frac{{r_<}^\lambda}{{r_>}^{\lambda+1}} \mathrm{B}(\ell\ell'\ell''\ell''';1,\Lambda)\end{aligned} \tag{14}$$

where

$$\mathrm{A}(\ell\ell'\ell''\ell''';0,\Lambda) = \mathrm{g}_\lambda(\ell\ell'\ell''\ell''';\Lambda,0) \tag{15}$$

and

$$\mathrm{B}(\ell\ell'\ell''\ell''';1,\Lambda) = \mathrm{g}_\lambda(\ell\ell'\ell''\ell''';\Lambda,+1) + \mathrm{g}_\lambda(\ell\ell'\ell''\ell''';\Lambda,-1) \tag{16}$$

We must note that in Eq. (14) *there is only one π orbital for N_2* since $\pi(x)$ and $\pi(y)$ are summed up through Eq. (16).

For e-H_2, we only calculate $\mathrm{A}(\ell\ell'\ell''\ell''';0,\Lambda)$ and $N_{occ} = 1$. However, for e-N_2, both $\mathrm{A}(\ell\ell'\ell''\ell''';0,\Lambda)$ and $\mathrm{B}(\ell\ell'\ell''\ell''';1,\Lambda)$ have to be calculated where $N_{occ}(\sigma) = 5$ and $N_{occ}(\pi) = 1$. To do this, another two arrays `clebx4(nbound,nexdim,nlamex,nexdim)` and `clebx5(nbound,nexdim,nlamex,nexdim)` are defined.

1. *clebexch*

In subroutine `clebexch`, the Clebsch-Gordon elements are calculated. From Eq. (13), 3 C-G coefficients have to be calculated,

- $\mathrm{C}(\ell\lambda\ell'';-\Lambda,\Lambda-m_i,-m_i)$
- $\mathrm{C}(\ell\lambda\ell'';0,0,0)$
- $\mathrm{C}(\ell\lambda\ell'';\Lambda,m_i-\Lambda,m_i)$

V. LAVIB.F