AN APPLICATION OF SYMBOLIC COMPUTATION TO CRYSTAL PHYSICS

Stuart I. Feldman

Bell Laboratories,
Murray Hill, New Jersey 07974

1. Introduction

This paper describes a practical application of symbolic computation to a problem in chemical physics. We compute a series approximation to a fairly complicated integral using straightforward techniques, allowing the computer to take over when the problem becomes too messy. Even though Altran [1] has no direct way to handle non-rational functions, introducing a few auxiliary variables renders the problem tractable. We certainly hope that more users will attempt to solve standard problems of this sort using symbolic manipulation programs

A. R. Hutson, a chemist at Bell Laboratories, constructed a piezoelectric theory in attempting to understand the fine structure splitting of donor-acceptor pair luminescence spectra of gallium phosphide. To test this theory, he needed to evaluate the piezoelectric potential due to the acceptor, integrated over the electronic ground-state wave function of the donor. Rather than compute the integral numerically for each relevant set of values of five parameters, an analytic series approximation was derived. The potential was expanded in Legendre functions; the wave function was expanded in a power series in the eccentricity of the charge distribution.

It is possible to do all of the resulting integrals analytically, but the terms of the series become quite complicated very quickly. This paper derives the form of the series, and then explains an Altran program to compute the final result. This program is a fairly straightforward transcription of the mathematical analysis.

The output of the Altran program was then converted into a Fortran program that computes the integral for given values of the parameters. This program was used in an attempt to fit the theoretical model to experimental data.

This sort of computation ought to be more common. The example shows the ease of using a computer to do the more repetitive part of the calculation. Altran made it possible to compute the series to a sufficiently high order to use it practically to evaluate the function.

2. Statement of the Problem

We need to integrate the piezoelectric potential over the charge distribution. The experimental data suggested that the non-spherical character of the wave function might be of importance. The effective-mass envelope wave function of the donor for gallium phosphide should be quite similar to that for silicon [2] and consist of the superposition of three orthogonal, oblate spheroids of the form

$$f = \frac{1}{3}[f_1(x,y,z) + f_1(y,z,x) + f_1(z,x,y)],$$

where

$$f_1(x,y,z) = \frac{1}{\pi a^2 b}\exp\left\{-2\left[\frac{x^2+y^2}{a^2}+\frac{z^2}{b^2}\right]^{1/2}\right\},$$

with the semi-axes a and b to be determined empirically.

The lowest order of the piezoelectric potential due to the acceptor at the point (X,Y,Z) is given by

$$\psi = \psi_0 (x-X)(y-Y)(z-Z)/\rho^5,$$

where ψ_0 is a constant related to the strain field and piezoelectric constant and ρ is the Euclidean distance from (x,y,z) to (X,Y,Z):

$$\rho = [(x-X)^2 + (y-Y)^2 + (z-Z)^2]^{1/2}.$$

The energy of interaction is given by

$$E(X,Y,Z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\psi(x,y,z;X,Y,Z)\,dx\,dy\,dz. \quad (1)$$

The obvious approach is to integrate equation (1) numerically for given values of X, Y, Z, a, and b. Considerable work would have to be done before doing the quadrature, since the domain of integration is infinite and the integrand is singular.

However, it is desirable and useful to have the results in symbolic form. We can get a series approximation by analytic methods. Although these methods are usually thought inapplicable because of the amount of calculation that must be done, we will show how a computer can take over most of the tedium.

The first version of this calculation was made by A. R. Hutson and J. A. Morrison. We use Morrison's observation that

$$\psi = -\frac{1}{3}\psi_0 \frac{\partial^3 \rho}{\partial X\, \partial Y\, \partial Z}.$$

If we define

$$I_1(X,Y,Z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_1 \rho\,(x,y,z;X,Y,Z)\,dx\,dy\,dz$$

and

$$E_1(X,Y,Z) = \frac{\partial^3 I_1}{\partial X\, \partial Y\, \partial Z},$$

then

$$E = -\frac{1}{9}\psi_0 [E_1(X,Y,Z) + E_1(Z,X,Y) + E_1(Y,Z,X)]. \tag{2}$$

3. Analysis

To evaluate I_1, we expand f_1 in a power series in the eccentricity of the spheroid, and then expand the distance ρ in Legendre polynomials. Other expansions in terms of Bessel functions are possible, but the simple power series seemed the simplest approach. First, let us scale and transform to spherical coordinates,

$$(x,y,z) \rightarrow (br,\theta,\phi)$$

$$(X,Y,Z) = (bX',bY',bZ') \rightarrow (bR,\Theta,\Phi)$$

where r and R are dimensionless radii and X', Y', and Z' are dimensionless cartesian coordinates. If we define the small parameter

$$\epsilon = 1 - b^2/a^2,$$

then we can expand the distribution

$$\begin{aligned} f_1 &= \frac{1}{\pi a^2 b}\exp\{-2r(1-\epsilon\sin^2\theta)^{1/2}\} \\ &= \frac{1}{\pi a^2 b}e^{-2r}\exp\{-2r[(1-\epsilon\sin^2\theta)^{1/2} - 1]\} \\ &= \frac{1}{\pi a^2 b}e^{-2r}\sum_{i=0}^{\infty}\epsilon^i D_i(r)\sin^{2i}\theta, \end{aligned}$$

where each D_i is a polynomial of degree i with *rational* coefficients d_{il}:

$$D_i = \sum_{l=0}^{i} d_{il} r^l.$$

Thus,

$$I_1 = \frac{b^2}{\pi a^2}\sum_{i=0}^{\infty}\epsilon^i \sum_{l=0}^{i} d_{il} \times \int_0^{\infty} r^{2+l} e^{-2r} dr \int_0^{\pi} \sin^{2i+1}\theta\, d\theta \int_0^{2\pi} \rho\, d\phi.$$

By the law of cosines,

$$\rho^2 = b^2[R^2 + r^2 - 2rR\cos\gamma],$$

where γ is the angle between the vectors (x,y,z) and (X,Y,Z), so we can expand ρ in Legendre polynomials,

$$\rho = b\sum_{n=0}^{\infty} A_n(r,R) P_n(\cos\gamma).$$

Later, we can use the standard expansion

$$\frac{1}{\rho} = \frac{1}{b}\sum_{n=0}^{\infty}\begin{Bmatrix} r^n/R^{n+1}, & \text{for } r \leqslant R \\ R^n/r^{n+1}, & \text{for } r \geqslant R \end{Bmatrix} P_n(\cos\gamma)$$

to compute the A_n.

By the addition theorem for Legendre polynomials,

$$P_n(\cos\gamma) = P_n(\cos\theta)P_n(\cos\Theta) + 2\sum_{l=1}^{\infty}\frac{(n-l)!}{(n+l)!}P_n^l(\cos\theta)\, P_n^l(\cos\Theta)\cos[l(\phi-\Phi)].$$

The ϕ-integral of each term in the infinite series is zero because of the $\cos[l(\phi-\Phi)]$ factor, so

$$I_1 = \frac{2b^3}{a^2}\sum_{i=0}^{\infty}\epsilon^i \sum_{l=0}^{i} d_{il} \sum_{n=0}^{\infty}\int_0^{\infty} A_n(r,R) r^{l+2} e^{-2r} dr \times P_n(\cos\Theta)\int_0^{\pi} P_n(\cos\theta)\sin^{2i+1}\theta\, d\theta.$$

The θ-integral is zero for odd values of n, so we define

$$\beta_{im} = \frac{1}{2}\int_0^{\pi} P_{2m}(\cos\theta)\sin^{2i+1}\theta\, d\theta = \int_0^1 (1-\xi^2)^i P_{2m}(\xi)\, d\xi.$$

The β_{im} are easily computed rational numbers. In particular, note that $\beta_{im} = 0$ if $m > i$, so we can write

$$I_1 = \frac{4b^3}{a^2}\sum_{i=0}^{\infty}\epsilon^i \sum_{m=0}^{i}\beta_{im} P_{2m}(Z'/R)\sum_{l=0}^{i} d_{il} K_{lm}, \tag{3}$$

where

$$\cos\Theta = Z'/R = Z/bR$$

and

$$K_{lm} = \int_0^{\infty} A_{2m}(r,R) r^{l+2} e^{-2r} dr.$$

If we define the exponential integral functions

$$L_k^{(1)}(R) = \int_0^R r^k e^{-2r} dr$$

and

$$L_k^{(2)}(R) = \int_R^{\infty} r^k e^{-2r} dr,$$

then

$$K_{lm} = \frac{1}{4m+3} R^{-(2m+1)} L^{(1)}_{2m+l+4}(R) - \frac{1}{4m-1} R^{-(2m-1)} L^{(1)}_{2m+l+2}(R) + \frac{1}{4m+3} R^{2m+2} L^{(2)}_{l+1-2m}(R) - \frac{1}{4m-1} R^{2m} L^{(2)}_{l+3-2m}(R).$$

Note that we need the $L^{(1)}_k$ only for positive k. Integration by parts yields the following recurrences.

$$L^{(1)}_m = \begin{cases} \tfrac{1}{2}(1 - e^{-2R}) & \text{for } m = 0 \\ -\tfrac{1}{2} R^m e^{-2R} + \tfrac{1}{2} m L^{(1)}_{m-1} & \text{for } m > 0 \end{cases}$$

$$L^{(2)}_m = \begin{cases} \tfrac{1}{2} R^m e^{-2R} + \tfrac{1}{2} m L^{(2)}_{m-1} & \text{for } m > 0 \\ \tfrac{1}{2} e^{-2R} & \text{for } m = 0 \\ E_1(2R) & \text{for } m = -1 \\ -\frac{R^{m+1}}{m+1} e^{-2R} + \frac{2}{m+1} L^{(2)}_{m+1} & \text{for } m < -1 \end{cases}$$

(Here, E_1 is the standard exponential integral.)

4. Simplifications

Although I_1 is a function of X, Y, and Z, X and Y enter only through R. Therefore, the chain rule allows us to write

$$E_1 = \frac{X'Y'Z'}{b^3} \left\{ \frac{1}{R}\frac{\partial}{\partial R} \left[\frac{1}{R}\frac{\partial}{\partial R} \left(\frac{1}{R}\frac{\partial}{\partial R} + \frac{1}{Z'}\frac{\partial}{\partial Z'} \right) \right] \right\} I_1. \quad (4)$$

Since equation (3) shows that I_1 can be written as a power series in Z'^2, E_1 can be written as $X'Y'Z'$ times a power series in Z'^2. If we make the other two symmetric substitutions in equation (2), we find that we have a series in the quantities

$$s_m = \frac{X'^{2m} + Y'^{2m} + Z'^{2m}}{R^{2m}}.$$

Thus, if E_1 is of the form

$$E_1 = \frac{4X'Y'Z'}{a^2} \sum_{i=0}^{\infty} \epsilon^i \sum_{m=0}^{i} H_{im}(R) Z'^{2m},$$

then

$$E = -\frac{4\psi_0}{9a^2} X'Y'Z' H, \quad (5)$$

where

$$H = \sum_{i=0}^{\infty} \epsilon^i \sum_{m=0}^{i} H_{im}(R) R^{2m} s_m.$$

Since $s_0 = 3$ and $s_1 = 1$, we only need to compute the s_m for $m > 1$.

5. Altran Program

The result for ϵ^0 can be obtained quite easily, the ϵ^1 term is fairly messy, and the second order is virtually impossible to do by hand. Therefore, we want the computer to do the manipulations. In this section we present a program written in Altran to compute successive orders of the series H. To get the energy, it is only necessary to multiply by the factor in equation (5).

Before writing the program, we must make a few tactical decisions. It may appear impossible for a language like Altran to handle this problem, since Altran can only manipulate rational functions, while our problem is fundamentally exponential. In reality, this presents no difficulty, since the solution is rational in the auxiliary variable e^{-2R}, called EXP2R in the program. We also note that $L^{(2)}_{-1}(R)$ is a transcendental function independent of the exponential, so we introduce another indeterminate called EXPINT.

In the rest of this section, we present the program a few lines at a time. This program is a direct transcription of the mathematical analysis above, and demonstrates the ease of doing the calculation symbolically on a computer. Even though very little effort was expended to save time or space, the entire run to compute H through order ϵ^3 cost less than fifteen dollars on the HIS 6070.

It is not necessary to know Altran to understand the code; a rudimentary knowledge of Fortran, Algol, or PL/I and a little imagination should suffice.

As usual, the program begins with a number of declarations. In finite time, we can only compute a finite number of orders of ϵ, so we read an integer parameter MAXI using the built-in function SIREAD. We then declare the two types of algebraic quantities we will be using. The Legendre polynomials will be defined in terms of a single indeterminate, XI. All other algebraic quantities will be built from the indeterminates EXP2R, EXPINT, R, Z, and the S(M), corresponding to e^{-2R}, $L^{(2)}_{-1}(R)$, R, Z', and the s_m, respectively. (The numbers after the colons in the algebraic declarations are the maximum values of the exponents of the corresponding indeterminates.)

```
procedure main

external integer maxi = imax(2,siread( ) )

long algebraic (xi : 4*maxi) array(0 : 2*maxi) p
long algebraic (expint:1, exp2r:1, s(2:maxi):1,
```

```
r:100, z:100)
```

We must now declare all of the other variables we intend to use. The names of most of the arrays are the same as used in the analysis above.

```
long algebraic array (0:maxi) il, h
long rational array (0:maxi, 0:maxi) beta, d

long algebraic array (0 : 3*maxi+4) L1
long algebraic array (1-2*maxi : maxi+3) L2
long algebraic array (0:maxi, 0:maxi) k
```

As explained below, we will be making use of a package of library procedures for operating on truncated power series. We must declare these procedures, since they are not built-in, and must also declare some temporary arrays for holding the series involved.

```
altran algebraic array tpssbs, tpspwr
long algebraic array (0:maxi) texp, tsqrt,t
```

Now we declare the temporary algebraic variables and integer indices that we will use:

```
long algebraic sum, deriv
integer i, L, m
```

Finally, we declare a function that we will write ourselves to compute $\partial/\partial R$ using the chain rule (see below)

```
altran algebraic diffr
```

We are done with the preliminaries. Next, we must calculate a number of the quantities used in the analysis. First, we compute the Legendre polynomials, using the classic recurrence:

```
p(0) = 1
p(1) = xi

do i = 2, 2*maxi
   p(i) = ( (2*i-1)*xi*p(i-1)-(i-1)*p(i-2))/i
doend
```

Note that we have computed these polynomials *symbolically* as functions of ξ. To compute the β_{im}, we integrate symbolically from 0 to 1 using the built-in function PINT:

```
do i = 0, maxi
do m = 0, maxi
   beta(i,m) = pint( (1-xi**2)**i * p(2*m) ,
                     xi) (1)
doend
doend
```

Next, we evaluate the $L^{(1)}$ and $L^{(2)}$ functions, using the recurrence relations (28) and (29).

```
L1(0) = (1 - exp2r) / 2
do m = 1, 3*maxi+4
   L1(m) = - r**m * exp2r / 2  +
           m * L1(m-1) / 2
doend

L2(0) = exp2r / 2
do m = 1, maxi+3
   L2(m) = r**m * exp2r / 2 + m * L2(m-1) / 2
doend

L2(-1) = expint
do m = -2, 1-2*maxi, -1
   L2(m) = ( 2*L2(m+1) - exp2r * r**(m+1) ) /
                  (m+1)
doend
```

The K_{lm} are then easily calculated from equation (27):

```
do L = 0, maxi
do m = 0, maxi
   k(L,m) = (r**(-2*m-1)*L1(2*m+L+4) +
      r**(2*m+2)*L2(L+1-2*m) )/(4*m+3) -
      (r**(-2*m+1)*L1(2*m+L+2) +
       r**(2*m)*L2(L+3-2*m) ) / (4*m-1)
doend
doend
```

Next we compute the power series for the distribution. The program is a little tricky, since we require the truncated power series package. A truncated power series is represented as an array of algebraics, each implicitly multiplied by a power of the hidden expansion variable (in our case, ϵ). To compute the D_i, we first create a truncated power series for the exponential function:

```
texp(0) = 1
do i = 1, maxi
   texp(i) = texp(i-1) / i
doend
```

Then we compute a series for $[(1-x)^{1/2}-1]$. In an Altran list, a dollar sign implies repetition; in the following, these repetitions pad lists with zeroes to ensure that the truncated power series are computed at the right order.

```
tsqrt = tpspwr( (1,-1,(maxi-1)$0), 1/2) -
                (1,maxi$0)
```

Finally, we compute the series in equation (13) by substituting the series just computed into the exponential series TEXP:

```
t = tpssbs(texp, -2*r*tsqrt)
```

The built-in procedure GETBLK then extracts the coefficients d_{il}:

```
do i = 0, maxi
do L = 0, i
   d(i,L) = getblk( t(i), r, L)
doend
doend
```

Finally, we are ready to compute the triple sum in equation (3):

```
do i = 0, maxi
   i1(i) = 0
   do m = 0, i
      sum = 0
      do l = 0, i
         sum = sum + d(i,l)*k(l,m)
      doend
      i1(i) = i1(i) + sum * beta(i,m)*p(2*m)(z/r)
   doend
```

Now that we have I1(I), we take the triple derivative in equation (4). We have written our own function DIFFR (see below) to compute $\partial/\partial R$; we use the built-in procedure DIFF to compute $\partial/\partial Z'$.

```
deriv = (1/r)*diffr( (1/r)*diffr (
      (1/r)*diffr(i1(i)) + (1/z)*diff(i1(i),z) ))
```

Finally, we do the simplifications described in Section 4, again using GETBLK to extract the coefficient of various powers of Z:

```
h(i) = 3 * getblk(deriv,z,0) +
   getblk(deriv,z,2) * r**2
do m = 2,i
   h(i) = h(i) +
      getblk(deriv,z,2*m) * r**(2*m) * s(m)
doend
```

Having computed all of the derivatives and simplified, we write out the results for this order and proceed:

```
      write h(i)
   doend
end
```

Since the indeterminates EXP2R and EXPINT are implicitly functions of R, we used the following special-purpose procedure DIFFR to compute the derivative with respect to R.

```
procedure diffr(f)
external integer maxi
long algebraic (expint:1, exp2r:1, s(2:maxi):1,
        r:100, z:100) value f
   return( diff(f,r) - 2*exp2r*diff(f,exp2r) -
      (exp2r/r)*diff(f,expint) )
end
```

6. Altran Results

The program described above was run. The first few terms are fairly simple, but the higher orders are very complicated. (If this were not true, the whole job could have been done with pencil and paper). The computer output begins:

```
# H(0)
    3*(4*EXP2R*R**4+14*EXP2R*R**3+
    27*EXP2R*R**2 + 30*EXP2R*R +
     15*EXP2R + 3*R**2 - 15 ) / ( 4*R**7 )
# H(1)
     ( 8*EXP2R*R**5 + 32*EXP2R*R**4 +
    82*EXP2R*R**3 + 141*EXP2R*R**2 +
     150*EXP2R*R +
    75*EXP2R + 9*R**2 - 75 ) / ( 4*R**7 )
```

The higher orders are so bulky as to be of little direct use to a human reader. However, this result can be used to compute the energy. Altran has a facility for substituting real values for the indeterminates of an algebraic, so we can evaluate the energy directly in Altran. Since real substitution is fairly expensive, the Altran output was converted to a set of Fortran function subprograms. (A general-purpose package to do this for any rational function is available). These Fortran functions were evaluated thousands of times in attempting to fit the theoretical model to real data.

7. Data Fitting

In a real crystal, the splitting of a spectral line is due to the effect of the donor field on the acceptor and the effect of the acceptor field on the donor. Equation (5) and the definition of ϵ allow us to write the total theoretical energy in the form

$$E_{th} = \psi_{0,donor} f(X,Y,Z; b_{acceptor}, \epsilon_{acceptor}) + \psi_{0,acceptor} f(X,Y,Z; b_{donor}, \epsilon_{donor}).$$

An attempt was made to fit such a function to experimental data for gallium phosphide doped with carbon as the acceptor and another impurity as donor, using assumed values for $\epsilon_{acceptor}$, $b_{acceptor}$, and ϵ_{donor}. Even though ϵ_{donor} is fairly large, the power series coefficients fall off for moderate values of R, so the approximation of ignoring all terms that are $o(\epsilon^3)$ probably gives sufficient accuracy for the available data. The theoretical curve was then fit directly to experimental data to determine values of $\psi_{0,donor}$, $\psi_{0,acceptor}$, and b_{donor}. Unfortunately the fit between the experimental data and the theoretical curve was not very good. Later experimental work showed that a more complicated model involving additional physical phenomena will be required to explain the observations.

8. Acknowledgement

This memo was printed on a phototypesetter attached to the Unix operating system, using Kernighan and Cherry's mathematical typesetting program [3].

References

1 Brown, W. S., *ALTRAN User's Manual,* Bell Telephone Laboratories, Murray Hill, New Jersey, 1971. Third Edition, 1973.

2. Kohn, W., *Solid State Physics* 5 (1957), 288 (Academic Press: New York)

3. Kernighan, B. W. and Cherry, L. L., 'A System for Typesetting Mathematics', *Communications of the ACM* **18**, (1975), 151.

Computing Real Zeros of Polynomials with SAC-1

(continued from page 10)

8. Conclusion.

In the SAC-1 polynomial real zero system the refinement of isolating intervals to arbitrary small length is done by bisection. This method is known to be very slow. It is not at all surprising that for many inputs to the system, this refining procedure turns out to be the most time-consuming one. We have presented an alternative refinement algorithm we used Newton's method and only if necessary bisection. We also introduced a method for computing approximations of rational numbers to control the growth of number lengths. Empirical results show a considerable gain in efficiency.

This work was done independently of similar work by G.E. Collins [3]. He too recognized the inefficiency of bisection and replaced it by an adaptation from Newton's method. He also introduced a method to control the growth of number lengths. Although he has presented his method at a seminar in Leuven, a comparison between both approaches was impossible because of a lack of corresponding empirical results.

9. References.

[1] Collins G.E., The SAC-1 list processing System, University of Wisconsin Computing Center, Technical Report No 129, July 1971.

[2] Collins G.E., The SAC-1 Integer arithmetic system - Version III, University of Wisconsin Computing Center, Technical Report No 156, March 1973.

[3] Collins G.E., High precision calculation of Real Algebraic Numbers, abstract in Sigsam vol. 8, No 4, November 1974.

[4] Collins G.E. and Heindel L.E., The SAC-1 polynomial real zero system, University of Wisconsin Computing Center, Technical Report No. 18, August 1970.

[5] Dekker J.C., Zeroin, Constructive aspects of the fundamental theorem of algebra.

[6] Ralston A. and Wilf H., Mathematical methods for digital computers, John Wiley New York.