SIAM J. COMPUT.
Vol. 4, No. 2, June 1975

SOLVING A PROBLEM IN EIGENVALUE APPROXIMATION WITH A SYMBOLIC ALGEBRA SYSTEM*

ANDREW D. HALL, JR.†

Abstract. In a recent paper by R. A. Handelsman and J. S. Lew [1], it is shown that for a certain family of potentials, the eigenvalues of the one-dimensional time-independent Schrödinger equation are proportional to $u(x, y)^{-2}$ with u determined by $u^{1/\beta} = f(xu, yu^{2/\beta})$, where $f(x, y) = \sum_{r=0}^{\infty} \sum_{s=0}^{\infty} f_{rs}x^s y^r$. Both β and the f_{rs} are known constants, with $f_{00} = 1$.

In [2], Lew proposed that a system for symbolic algebra be used to compute the Taylor series expansion for $u(x, y)$, although in [1] what is desired is the expansion of $c(x, y) = u(x, y)^{-2}$. To illustrate how such a system can be used to solve this and similar problems, three solution methods are described and corresponding programs given.

During the attempt to put the output resulting from these programs into a form similar to that given by Lew [1], the general solution was found. We give this solution here but defer the proof to another paper [3].

Key words. symbolic computation, power series

1. Introduction.

In a recent paper by R. A. Handelsman and J. S. Lew [1], it is shown that for the family of potentials

$$V(\chi) = A\chi^m + B\chi^{2m}, \qquad B > 0, \quad m = 2, 4, 6, \cdots, \tag{1}$$

the eigenvalues E_n of the one-dimensional time-independent Schrödinger equation are proportional to $u(x, y)^{-2}$, with u determined by

$$u^{1/\beta} = f(xu, yu^{2/\beta}). \tag{2}$$

Here $\beta = m/(m + 1)$, and the parameters x and y involve negative powers of $(n + \frac{1}{2})$. The function f is defined by

$$f(x, y) = \sum_{r=0}^{\infty} \sum_{s=0}^{\infty} f_{rs}x^s y^r, \tag{3}$$

where the f_{rs} are known constants. In particular, $f_{00} = 1$.

Handelsman and Lew define

$$c(x, y) = u(x, y)^{-2} = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} c_{pq}x^q y^p, \tag{4}$$

and the problem is to express the c_{pq} in terms of β and the f_{rs}. Handelsman and Lew were able to compute the c_{pq} for $p + q \leqq 4$ "by a hand computation lasting about a week" [2].

In this paper we present three methods for solving this problem and illustrate each with a program written in ALTRAN [4], [5]. Alternative methods and solutions are described in [6] and [7].

During the attempt to put the output resulting from these programs in a readable form, the general solution was discovered. We give this solution here but defer the proof to another paper [3].

* Received by the editors May 28, 1973, and in revised form June 11, 1974.

† Bell Telephone Laboratories, Inc., Murray Hill, New Jersey 07974.

2. Method 1. Method 1 is a brute force solution designed to take advantage of the ALTRAN library procedures for manipulating one-dimensional truncated power series. The method illustrates an often useful technique for converting a multidimensional power series problem into a one-dimensional problem that is easily solved.

The problem is considerably simplified if we let $u = v^\beta$ so that (2) becomes

$$v = f(xv^\beta, yv^2). \tag{5}$$

We wish now to determine the coefficients v_{pq} in the expansion

$$v(x, y) = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} v_{pq} x^q y^p, \tag{6}$$

and then use the relation $c = u^{-2} = v^{-2\beta}$ to determine the coefficients c_{pq} in (4).

We can make the problem one-dimensional by grouping terms according to the sum of the exponents of x and y. Formally, this is accomplished by setting $x = \lambda x$ and $y = \lambda y$ and viewing (3), (4) and (6) as power series in λ. These become, respectively,

$$f(\lambda x, \lambda y) = \sum_{k=0}^{\infty} f_k(x, y)\lambda^k, \qquad f_k(x, y) = \sum_{s=0}^{k} f_{k-s,s} x^s y^{k-s} \tag{7}$$

$$c(\lambda x, \lambda y) = \sum_{k=0}^{\infty} c_k(x, y)\lambda^k, \qquad c_k(x, y) = \sum_{s=0}^{k} c_{k-s,s} x^s y^{k-s} \tag{8}$$

$$v(\lambda x, \lambda y) = \sum_{k=0}^{\infty} v_k(x, y)\lambda^k, \qquad v_k(x, y) = \sum_{s=0}^{k} v_{k-s,s} x^s y^{k-s}. \tag{9}$$

Equation (5) becomes

$$v(\lambda x, \lambda y) = f(\lambda x v^\beta, \lambda y v^2). \tag{10}$$

The problem can now be solved in the following way. First represent $v(\lambda x, \lambda y)$ as a power series in λ to order n with unknown coefficients $v_0, v_1, \cdots, v_n$. Letting $\lambda = 0$ in (9) and (10), we find immediately that $v_0 = 1$. Now use (7) to compute to order n the power series for $f(\lambda x, \lambda y)$. Then substitute xv^β and yv^2 for x and y, respectively. The result will be a power series for $f(\lambda x v^\beta, \lambda y v^2) = v(\lambda x, \lambda v)$. Normally, we would have to equate the coefficients of these two series and solve the resulting system of equations for the unknowns $v_0, v_1, \cdots, v_n$. However, from (10) it is easy to show that the coefficient of λ^k in $f(\lambda x v^\beta, \lambda y v^2)$ depends only on x, y, β, f_{rs} and the unknown coefficients $v_1, \cdots, v_{k-1}$. Since we also know from (10) that these are precisely the previous coefficients in the series, simple substitution can be used to eliminate them. The result will be the desired power series for $v(\lambda x, \lambda y)$.

Using the relation $c = v^{-2\beta}$, we can now compute $c(\lambda x, \lambda y)$. From (8) we see that we can easily extract the c_{pq} of (4) from the coefficients of $c(\lambda x, \lambda y)$.

An ALTRAN program for carrying out this computation to order n is shown in Fig. 1. The program is straightforward except for the integer array-valued function maxexp which is used to compute the maximum exponent of each f_{rs} that can occur. For small n, we could have simply used n in place of the procedure call maxexp (n, n), but for large n we need a better bound to conserve space. For completeness, the procedure maxexp is discussed in detail in Appendix A.

```
        procedure main
        integer n = 4
        integer k, s
        integer array altran maxexp

# Declare the indeterminates and their maximum exponents.

        algebraic ( x:n, y:n, f(0:n,0:n):maxexp(n,n), b:n, v(0:n):n )

# Declare arrays for the coefficients of the truncated power series.
# In ALTRAN, the power series variable is not explicitly represented.

        long algebraic array (0:n) vtps, ctps, ftps = 0
        long algebraic array altran tpspwr, tpssbs

# Step 1.   As in (9), let vtps be a series with unknown coefficients,
#           v(0), v(1), . . . , v(n), but noting that v(0) = 1.

        vtps = v; vtps(0) = 1

# Step 2.   Using (7), compute the power series for f. Note that
#           f(0) = 1 and that ftps(k) is initially 0.

        ftps(0) = 1
        do k = 1, n
           do s = 0, k
              ftps(k) = ftps(k) + f(k-s,s) * x**s * y**(k-s)
           doend
        doend

# Step 3.   Compute the right side of (10), using truncated power series
#           substitution to replace x and y by x*vtps**b and y*vtps**2.

        ftps = tpssbs( ftps, x*tpspwr(vtps,b) , x )
        ftps = tpssbs( ftps, y*tpspwr(vtps,2) , y )

# Step 4.   Form vtps(k) by replacing the v(1), v(2) , . . . , v(k-1) in
#           ftps(k) with vtps(1), vtps(2), . . . , vtps(k-1).

        do k = 1, n
           vtps(k) = ftps(k) (v = vtps)
        doend

# Step 5.   Compute c and write out the coefficients.

        ctps = tpspwr( vtps, -2*b )

        do k = 0, n
           write ctps(k)
        doend

        end
```

FIG. 1. *Method* 1

3. Method 2. In Method 2, we derive a recurrence relation for the v_k defined by (9). We first replace x and y in (7) by xv^β and yv^2, respectively, to arrive at

$$v(\lambda x, \lambda y) = \sum_{k=0}^{\infty} \left(\sum_{s=0}^{k} f_{k-s,s} x^s y^{k-s} v^{\beta s + 2(k-s)} \right) \lambda^k. \tag{11}$$

Now let the γth power of v be represented by

$$v^\gamma = \sum_{k=0}^{\infty} (v^\gamma)_k \lambda^k, \tag{12}$$

where $(v^\gamma)_k$ simply denotes the coefficient of λ^k in the power series expansion of v^γ.

Substituting (12) into the right side of (11), we have

$$\sum_{k=0}^{\infty} v_k \lambda^k = \sum_{k=0}^{\infty} \left[\sum_{s=0}^{k} f_{k-s,s} x^s y^{k-s} \sum_{i=0}^{\infty} (v^{\beta s + 2(k-s)})_i \lambda^i \right] \lambda^k. \tag{13}$$

Equating terms of order n in λ, we have

$$v_n = \sum_{k=0}^{n} \sum_{s=0}^{k} f_{k-s,s} x^s y^{k-s} (v^{\beta s + 2(k-s)})_{n-k}. \tag{14}$$

Note that the terms with $k = 0$ do not contribute to the sum when $n > 0$, because $\beta s + 2(k - s) = 0$ and $(1)_n = 0$. Thus for $n > 0$, (14) expresses v_n in terms of the first $n - 1$ coefficients of the expansion of v^γ for various values of γ.

Furthermore, for $n > 0$ and γ arbitrary, $(v^\gamma)_n$ can be computed from $(v^\gamma)_0, \cdots, (v^\gamma)_{n-1}$ and $v_0, \cdots, v_n$ by the powering formula (see Appendix B)

$$(v^\gamma)_n = \frac{1}{nv_0} \sum_{i=1}^{n} [(\gamma + 1)i - n] v_i (v^\gamma)_{n-i}. \tag{15}$$

An ALTRAN program for computing v_n, $0 \leqq n \leqq \text{nmax}$ based on (14) and (15) is shown in Fig. 2. Since the procedure also computes $(v^\gamma)_n$, the c_n are easily obtained by setting $\gamma = -2\beta$.

4. Method 3. In Method 3, we use a technique similar to that used in Method 2 to derive a recurrence for the v_{pq} defined by (6). We first replace x and y in (3) by xv^β and yv^2, respectively, to arrive at

$$v(x, y) = \sum_{r=0}^{\infty} \sum_{s=0}^{\infty} f_{rs} x^s y^r v^{\beta s + 2r}. \tag{16}$$

Now let the γth power of v be represented by

$$v^\gamma = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} (v^\gamma)_{pq} x^q y^p, \tag{17}$$

where $(v^\gamma)_{pq}$ is used to denote the coefficient of $x^q y^p$ in the expansion of v^γ. Substituting (17) into (16) and equating this to (6), we have

$$\sum_{p=0}^{\infty} \sum_{q=0}^{\infty} v_{pq} x^q y^p = \sum_{r=0}^{\infty} \sum_{s=0}^{\infty} f_{rs} x^s y^r \left[\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (v^{\beta s + 2r})_{ij} x^j y^i \right]. \tag{18}$$

```
      procedure main

      integer nmax = 4
      integer i, k, n, s
      integer array altran maxexp

# Declare the indeterminates and their maximum exponents.

      algebraic(x:nmax,y:nmax,f(0:nmax,0:nmax):maxexp(nmax,nmax),
             b:nmax,g:nmax)

      long algebraic array (0:nmax) c, v = 0, vg = 0

# Step 1.    Initialize.

      v(0) = 1; vg(0) = 1

      do n = 1, nmax

# Step 2.    Using (14), compute v(n).

        do k = 1, n
          do s = 0, k
            v(n) = v(n) + f(k-s,s) * x**s * y**(k - s) *
            vg(n-k) (g = b*s+2* (k-s) )
          doend
        doend

# Step 3.    Using (15), compute vg(n). Note that v(0) = 1 and that
#            vg(n) is initially 0.

        do i = 1, n
          vg(n) = vg(n) + ((g+1)*i-n) * v(i) * vg(n-i)
        doend
        vg(n) = vg(n)/n

# Step 4.    Compute c(n), write it out, and recover the space.

        c(n) = vg(n) (g = -2*b)
        write c(n); c(n)=.null.

      doend

      end
```

FIG. 2. *Method* 2

Equating the coefficients of $x^q y^p$, we have

$$v_{pq} = \sum_{r=0}^{p} \sum_{s=0}^{q} f_{rs}(v^{\beta s+2r})_{p-r,q-s}. \tag{19}$$

Note that the term with $r = s = 0$ does not contribute to the sum when $p > 0$ or $q > 0$, because $\beta s + 2r = 0$ and $(1)_{pq} = 0$. Hence (19) expresses v_{pq} in terms of $(v^\gamma)_{ij}$ with $i \leqq p$, $j \leqq q$ and $i + j < p + q$.

Now for $p > 0$ or $q > 0$ and arbitrary γ, we can express $(v^\gamma)_{pq}$ in terms of the

```
        procedure main

        integer pmax = 4, qmax = 4
        integer i, j, p, q, r, s
        integer array altran maxexp

# Declare the indeterminates and their maximum exponents.

        algebraic (f(0:pmax,0:qmax):maxexp(pmax,qmax), b:pmax+qmax,
                 g:pmax+qmax)

        long algebraic array (0:pmax, 0:qmax) c, v = 0, vg = 0

# Step 1.   Initialize.

        v(0,0) = 1; vg(0,0) = 1

        do p = 0, pmax
           do q = 0, qmax

              if (p+q .eq. 0) go to skip

# Step 2.   Using (19), compute v(p,q).

              do r = 0, p
                 do s = 0, q
                    v(p,q) = v(p,q) + f(r,s) * vg(p-r,q-s) (g=b*s+2*r)
                 doend
              doend

# Step 3.   Using (20), compute vg(p,q). Note that v(0,0) = 1 and that
#           because vg(p,q) is initially 0, the term with i = j = 0 does
#           not contribute to the sum

              do i = 0, p
                 do j =0, q
                    vg(p,q) = vg(p,q) + (g+1)*(i+j) - (p+q))*v(i,j)*
                    vg(p-i,q-j)
                 doend
              doend
              vg(p,q) = vg(p,q)/(p+q)

# Step 4.   Compute c(p,q), write it out, and recover the space.

skip:         c(p,q) =vg(p,q) (g = -2*b)
              write c(p,q); c(p,q) = .null.

           doend
        doend

        end
```

FIG. 3. *Method* 3

lower order coefficients of v^γ and the v_{ij}, $i \leqq p$ and $j \leqq q$, by the powering formula (see Appendix C)

$$(20) \qquad (v^\gamma)_{pq} = \frac{1}{(p+q)v_{00}} \sum_{\substack{i=0 \\ i+j\neq 0}}^{p} \sum_{j=0}^{q} [(\gamma+1)(i+j) - (p+q)]v_{ij}(v^\gamma)_{p-i,q-j}.$$

An ALTRAN program for computing v_{pq}, $p \leqq$ pmax, $q \leqq$ qmax, is shown in Fig. 3. Since the procedure also computes $(v^\gamma)_{pq}$, c_{pq} can be obtained by setting $\gamma = -2\beta$.

5. Results. Methods 1 and 2 would normally be used to compute c_{pq} for all p, q such that $p + q \leqq n$, whereas Method 3 would be used to compute c_{pq} for $p \leqq$ pmax, $q \leqq$ qmax. To compare the speeds of these techniques, Method 3 was modified slightly so that it also computes c_{pq} for $p + q \leqq n$.

TABLE 1
Total processor time in seconds required for the computation of all c_{pq} for $p + q \leqq n$

n	*Method* 1	*Method* 2	*Method* 3
1	4.6	1.3	1.9
2	8.6	3.1	5.2
3	16.0	6.8	13.1
4	26.6	13.3	29.8

The processor time in seconds required for computing all c_{pq} with $p + q \leqq n$ for $n = 1, \cdots, 4$ is shown in Table 1. The ALTRAN system used was installed on the Honeywell 6070 computer (36 bit words, 1 μs cycle time) at Bell Telephone Laboratories, Murray Hill, New Jersey.

In each case, no more than 10,000 words of workspace were used. Using Method 2 with 24,000 words of workspace, we were able to compute all c_{pq} for $p + q \leqq 6$ in 53.1 seconds.

Part of the output resulting from Method 3 (modified) is shown in Fig. 4.

To put the c_{pq}, $p + q \leqq 6$, in a form similar to that given in [1], an attempt was made to devise a heuristic program to collect terms of the same degree in the f_{rs} and factor the resulting coefficient which is a polynomial in β. For example, c_{21} can be written

$$(21) \qquad C_{21} = -2\beta[f_{21} + (f_{10}f_{11} + f_{20}f_{01})(3-\beta) + \tfrac{1}{2}f_{10}^2 f_{01}(3-\beta)(2-\beta)].$$

This effort led to the discovery of the general form of the $(v^\gamma)_{pq}$ appearing in (6). Derivation of this solution will be deferred to a subsequent paper [3]. The formula is as follows:

$$(22) \qquad (v^\gamma)_{pq} = \frac{\gamma}{2p + \beta q + \gamma} \sum_{t=0}^{p+q} F_t(p, q)(2p + \beta q + \gamma)_t,$$

```
# C(2,0)
        2*F(1,0)**2*B**2 - 3*F(1,0)**2*B - 2*F(2,0)*B
# C(0,3)
        ( - F(0,1)**3*B**3 + 3*F(0,1)**3*B**2 - 2*F(0,1)**3*
        B - 6*F(0,1)*F(0,2)*B**2 + 6*F(0,1)*F(0,2)*B -
        6*F(0,3)*B) / 3
# C(1,2)
        - 2*F(0,1)*F(1,1)*B - 2*F(0,2)*F(1,0)*B - 2*F(1,2)*B
# C(2,1)
        - F(0,1)*F(1,0)**2*B**3 + 5*F(0,1)*F(1,0)**2*B**2 -
        6*F(0,1)*F(1,0)**2*B + 2*F(0,1)*F(2,0)*B**2 -
        6*F(0,1)*F(2,0)*B + 2*F(1,0)*F(1,1)*B**2 - 6*F(1,0)*
        F(1,1)*B - 2*F(2,1)*B
# C(3,0)
        ( - 4*F(1,0)**3*B**3 + 18*F(1,0)**3*B**2 -
        20*F(1,0)**3*B + 12*F(1,0)*F(2,0)*B**2 - 30*F(1,0)*
        F(2,0)*B - 6*F(3,0)*B ) / 3
```

FIG. 4. *Output from Method* 3

where the $F_t(r, s)$ are defined by

$$\sum_{r=0}^{\infty} \sum_{s=0}^{\infty} \sum_{t=0}^{\infty} F_t(r, s) x^s y^r z^t = e^{(f(x,y)-1)z} \tag{23}$$

and the notation $(\delta)_t$ is the falling factorial

$$(\delta)_0 = 1, \qquad (\delta)_t = \delta(\delta - 1) \cdots (\delta - t + 1), \qquad t > 0.$$

From (23) and the fact that $f_{00} = 1$, it is easy to derive the following identities which permit the computation of $F_t(r, s)$ for all t, r and s.

$$F_0(0, 0) = 1,$$

$$F_0(r, s) = 0, \qquad r + s > 0,$$

$$F_t(r, s) = 0, \qquad t > r + s,$$

$$F_{t+1}(r, s) = \frac{1}{r + s} \sum_{i=0}^{r} \sum_{j=0}^{s} (i + j) f_{ij} F_t(r - i, s - j), \qquad r + s > 0.$$

For example, we find that

$$F_0(2,1) = 0, \qquad F_2(2,1) = f_{10}f_{11} + f_{20}f_{01},$$
$$F_1(2,1) = f_{21}, \qquad F_3(2,1) = \tfrac{1}{2}f_{10}^2 f_{01}.$$

Thus

$$c_{21} = (v^{-2\beta})_{21} = -2\beta[f_{21} + (f_{10}f_{11} + f_{20}f_{01})(3-\beta) + \tfrac{1}{2}f_{10}^2 f_{01}(3-\beta)(2-\beta)],$$

in accordance with (21).

6. Conclusion. Although the need for a program to solve this particular problem has been obviated by the discovery of the general solution, it is clear that algebraic manipulation systems—and in particular, procedures for manipulating truncated power series—can greatly simplify the computation of solutions to seemingly difficult problems. Of the three methods described for solving this particular problem, Method 1, which uses a package specifically designed for the manipulation of truncated power series, was by far the easiest to derive and program. Methods 2 and 3 required considerably more work for only a small improvement in performance. This illustrates the fallacy of using processor time as the sole measure of the value or quality of an algebraic manipulation system. More often, the relevant comparison is the ease with which a given problem is solved, provided the cost is not exorbitant.

Appendix A. Maximum exponents. In order to write ALTRAN programs that make reasonably economical use of storage, it is sometimes necessary to obtain tight bounds for the exponents of the indeterminates. This is particularly true when a large number of indeterminates are present.

It is easy to show by induction from (19) and (20) that the $(v^\gamma)_{pq}$ are "homogeneous" in the sense each term satisfies the identities

$$\sum_{r=0}^{p}\sum_{s=0}^{q} r\cdot\varepsilon_{rs} = p, \qquad \varepsilon_{rs} \leqq p,$$
$$\sum_{r=0}^{p}\sum_{s=0}^{q} s\cdot\varepsilon_{rs} = q, \qquad \varepsilon_{rs} \leqq q,$$

where ε_{rs} is the exponent of f_{rs}.

It follows immediately that in $(v^\gamma)_{pq}$,

$$\varepsilon_{rs} \leqq \min\left(\left\lfloor\frac{p}{\max(1,r)}\right\rfloor, \left\lfloor\frac{q}{\max(1,s)}\right\rfloor\right).$$

Since $f_{00} = 1$, it does not appear formally in any of our computations, and we take $\varepsilon_{00} = 1$ (ALTRAN does not allow maximum exponent declarations to be 0). The procedure maxexp used in each of our programs is shown in Fig. 5.

It is also easy to show from (19) and (20) that for $p + q > 0$, the exponent of β in v_{pq} cannot exceed $p + q - 1$. Taking $\gamma = -2\beta$ in (20), we can therefore show that the maximum exponent of β in c_{pq} cannot exceed $p + q$.

```
procedure maxexp ( pmax, qmax )
integer pmax, qmax, r, s
integer array ( 0:pmax, 0:qmax ) exp
do r = 0, pmax
   do s = 0, qmax
      exp (r,s) = imin ( iquo(pmax,imax(1,r)), iquo(qmax,imax(1,s)))
   doend
doend
exp (0,0) = 1
return (exp)
end
```

FIG. 5. *The procedure* maxexp

Appendix B. Powering of one-dimensional series. We derive here the formula attributed to J. C. P. Miller in [8] for powering a one-dimensional series. Using the previous notation, let

$$v = \sum_{k=0}^{\infty} v_k \lambda^k \tag{B.1}$$

and

$$w = v^\gamma = \sum_{k=0}^{\infty} (v^\gamma)_k \lambda^k. \tag{B.2}$$

Taking the derivative of w with respect to λ, we have

$$w' = \gamma v^{\gamma-1} v', \tag{B.3}$$

$$vw' = \gamma v' w. \tag{B.4}$$

Replacing v, w, v' and w' by their power series representations and equating coefficients of λ^n, we have

$$\sum_{i=0}^{n} v_i(n-i)(v^\gamma)_{n-i} = \gamma \sum_{i=0}^{n} i v_i (v^\gamma)_{n-i}, \tag{B.5}$$

and finally,

$$0 = \sum_{i=0}^{n} [(\gamma+1)i - n] v_i (v^\gamma)_{n-i}. \tag{B.6}$$

For $n > 0$ and $v_0 \neq 0$, we can solve (B.6) for $(v^\gamma)_n$. Thus

$$(v^\gamma)_n = \frac{1}{nv_0} \sum_{i=1}^{n} [(\gamma+1)i - n] v_i (v^\gamma)_{n-i}, \tag{B.7}$$

in accordance with (15). For $n = 0$, we obviously have $(v^\gamma)_0 = v_0^\gamma$.

Appendix C. Powering of two-dimensional series. In a manner similar to that used in Appendix B, we derive a formula for the powering of two-dimensional series. Let

$$v = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} v_{pq} x^q y^p \tag{C.1}$$

and

$$w = v^\gamma = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} (v^\gamma)_{pq} x^q y^p. \tag{C.2}$$

Taking the derivative of w with respect to x, we have

$$w' = \gamma v^{\gamma-1} v', \tag{C.3}$$

$$vw' = \gamma w v'. \tag{C.4}$$

Replacing v, w, v' and w' by their power series representations and equating the coefficients of $x^q y^p$, we have

$$\sum_{i=0}^{p} \sum_{j=0}^{q} v_{ij}(q-j)(v^\gamma)_{p-i,q-j} = \gamma \sum_{i=0}^{p} \sum_{j=0}^{q} j v_{ij}(v^\gamma)_{p-i,q-j}, \tag{C.5}$$

or

$$0 = \sum_{i=0}^{p} \sum_{j=0}^{q} [(\gamma+1)j - q] v_{ij}(v^\gamma)_{p-i,q-j}. \tag{C.6}$$

Unfortunately, (C.6) is trivial for $q = 0$. However, by repeating the above process using derivatives with respect to y, we obtain

$$0 = \sum_{i=0}^{p} \sum_{j=0}^{q} [(\gamma+1)i - p] v_{ij}(v^\gamma)_{p-i,q-j}. \tag{C.7}$$

Adding (C.6) and (C.7), we get

$$0 = \sum_{i=0}^{p} \sum_{j=0}^{q} [(\gamma+1)(i+j) - (p+q)] v_{ij}(v^\gamma)_{p-i,q-j}. \tag{C.8}$$

Now for $p + q > 0$ and $v_{00} \neq 0$, we can solve (C.8) for $(v^\gamma)_{pq}$. Thus

$$(v^\gamma)_{pq} = \frac{1}{(p+q)v_{00}} \sum_{\substack{i=0 \\ i+j\neq 0}}^{p} \sum_{j=0}^{q} [(\gamma+1)(i+j) - (p+q)] v_{ij}(v^\gamma)_{p-i,q-j}, \tag{C.9}$$

in accordance with (20). For $p + q = 0$, we obviously have $(v^\gamma)_{00} = v_{00}^\gamma$.

REFERENCES

[1] R. A. Handelsman and J. S. Lew, *Analytical evaluation of energy eigenvalues for a class of anharmonic oscillators*, J. Chem. Phys., 50 (1969), pp. 3342–3354.

[2] J. S. Lew, *Problem 3—Reversion of a double series*, SIGSAM Bull. 23, 1972, pp. 6–7.

[3] A. J. Goldstein and A. D. Hall, *Solutions to a problem in power series reversion*, SIAM J. Math. Anal., 6 (1975), pp. 192–198.

[4] W. S. Brown, *ALTRAN User's Manual*, Bell Telephone Laboratories, Murray Hill, N.J., 1971; 2nd ed., 1972.

[5] A. D. HALL, *ALTRAN Installation and Maintenance*, Bell Telephone Laboratories, Murray Hill, N.J., 1971; 2nd ed., 1972.

[6] R. LOOS, *A user's solution of Problem #3 with REDUCE 2*, SIGSAM Bull. 26, 1973, pp. 12–14.

[7] J. FITCH, *A solution to Problem #3*, SIGSAM Bull. 26, 1973, pp. 24–27.

[8] P. HENRICI, *Automatic computation with power series*, J. Assoc. Comput. Mach., 3 (1956), pp. 10–15.