Some MACSYMA Programs for Solving Recurrence Relations

JOHN IVIE
University of California, Berkeley

A set of programs to find closed-form solutions to recurrence relations is described. These programs are written in the MACSYMA programming language, a LISP-based system implemented at the Massachusetts Institute of Technology on the PDP-10 mc computer Single variable recurrence relations with constant coefficients are solved as well as some variable coefficient recurrences The use of these programs is illustrated with several examples taken from textbooks.

Key Words and Phrases: recurrence relations, characteristic equation, generating functions
CR Categories: 5 7, 5.30

1. INTRODUCTION

In this paper we are interested in programming techniques to find closed-form solutions to linear recurrence relations of the form

$$a_k u(n + k) + a_{k-1} u(n + k - 1) + \cdots + a_0 u(n) = g(n) \qquad (1)$$

where the coefficients a_i are either constants (constant coefficient case) or polynomials in n (variable coefficient case). In the cases we consider here, either $g(n) = 0$ (homogeneous case) or $g(n)$ is a polynomial in n, a constant raised to such a polynomial power, or sin or cos of a function of n (inhomogeneous case).

Sections 2 and 3 describe methods for the constant coefficient case, while Section 4 examines algorithms for the variable coefficient case. Section 5 gives the results of several different testings of our programs, and Section 7 gives the actual MACSYMA code used in these tests.

2. CONSTANT COEFFICIENT CASE. THE CHARACTERISTIC EQUATION METHOD

We first consider the homogeneous case, that is, when $g(n) = 0$ in eq. (1). The method of attack here is to substitute x^{k-i} for $u(n + k - i)$ in the given recur-

This work was made possible by access to the MACSYMA system at Massachusetts Institute of Technology supported in part by the United States Energy Research and Development Administration under Contract E(11-1)-3070 and in part by the National Aeronautics and Space Administration under Grant NSG 1323

Author's present address: Bell Telephone Laboratories, Warrenville-Naperville Road, Naperville, IL 60540.

rence relation (1) to obtain a polynomial equation

$$a_k x^k + a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 = 0$$

(called the characteristic equation) which can be solved for the k roots (with multiplicities). Then the solution to the recurrence relation $u(n)$ can be written as a linear combination of these roots, namely,

$$u(n) = \sum_{i=1}^{N} \left(\sum_{j=1}^{m(i)} A_{i,j}\, n^{j-1} \right) r_i^n$$

where $r_1, r_2, \ldots, r_N$ are the N distinct roots of the characteristic equation and $m(i)$ is the multiplicity of the ith root in this list. This system of linear equations can then be solved for the constants $A_{i,j}$ by using the given k initial values of the recurrence $u(0), u(1), \ldots, u(k-1)$. All of this is fairly easily accomplished by means of the MACSYMA SOLVE command.

As an example, the Fibonacci sequence is given by the recurrence relation $u(n+2) - u(n+1) - u(n) = 0$, with $u(0) = 0$, $u(1) = 1$. The characteristic equation is $x^2 - x - 1 = 0$ with roots $(1 + \sqrt{5})/2$, $(1 - \sqrt{5})/2$, and with solution $u(n) = (1/\sqrt{5})[((1 + \sqrt{5})/2)^n - ((1 - \sqrt{5})/2)^n]$.

We now consider the inhomogeneous case, that is, when $g(n)$ is a function of n in eq. (1). In our case, this means that $g(n)$ is either a polynomial in n, a constant raised to a polynomial power, or sin or cos of a function of n. (Other possible forms for $g(n)$ can be put into our framework, but these three cases are able to solve a large class of problems, as shown in Section 5.) The method of attack in these cases is to first find the homogeneous solution as previously described and then add to it a particular solution of eq. (1). The method used here to find a particular solution is the principle of undetermined coefficients. That is, using the form of $g(n)$, we assume a solution of the same form which is substituted into the recurrence relation (1); the coefficients for the assumed solution are then found by solving the resulting set of linear equations. Again, this is fairly easily carried out using the SOLVE command. However, it is important to check at each step whether any term of the assumed solution satisfies the homogeneous equation; if so, the assumed solution is multiplied by n and the process is repeated.

As an illustration of this method, we consider the following examples:

$$u(n+1) + u(n) = 2n^2 - 1, \quad u(0) = 1. \tag{2}$$

The homogeneous solution is $(\text{const})(-1)^n$, and to find a particular solution we assume $u(n) = an^2 + bn + c$, which gives values $a = 1$, $b = -1$, $c = -\frac{1}{2}$. Thus the solution is $u(n) = \frac{3}{2}(-1)^n + n^2 - n - \frac{1}{2}$. As another example, we have

$$u(n+2) - u(n+1) - 2u(n) = 2^n, \quad u(0) = 1, \; u(1) = 0. \tag{3}$$

The homogeneous solution is $c_1 2^n + c_2(-1)^n$; we assume $u(n) = B2^n$, which is a solution of the homogeneous equation, so we then assume $u(n) = nB2^n$, which gives $B = \frac{1}{6}$. Thus the solution is $u(n) = (\frac{2}{9})2^n + (\frac{7}{9})(-1)^n + n2^n/6$.

Several other examples of a similar nature using MACSYMA are given in Section 5.

3. CONSTANT COEFFICIENT CASE. THE METHOD OF GENERATING FUNCTIONS

In this section we describe another method for solving the constant coefficient recurrence relations of Section 2. This method has the advantage of finding the

homogeneous and particular solutions at once, but is slower in our implementation than the characteristic equation method. (It is also more complicated to program.)

This generating function method can be described as follows: define the generating function $F(x)$ of the sequence $\{u(n)\}$ as

$$F(x) = \sum_{n=0}^{\infty} u(n)x^n. \tag{4}$$

Now multiply each term of the recurrence relation (1) by x^{n+k} and sum from zero to infinity; each term $\sum_{n=0}^{\infty} u(n + k - i)x^{n+k}$ of the left-hand side of eq. (1) can be rewritten as $(F(x) - \sum_{j=0}^{K-i-1} u(j)x^j)x^i$.

We then have three cases for $g(n)$ to consider: (1) If $g(n) = 0$, then we can solve for $F(x)$ directly. (2) If $g(n) = a_0 + a_1 n + \cdots + a_m n^m$ is a polynomial in n, then we have the following rules for rewriting n^i in $g(n)$: since $1/(1 - x) = \sum_{n=0}^{\infty} x^n$, substitute $a_0 x^k/(1 - x)$ for $\sum_{n=0}^{\infty} a_0 x^{n+k}$; taking successive derivatives of this formula gives successive substitutions for higher powers of n. (3) If $g(n) = K^{an+b}$, where K is a constant, then by using the geometric series again, we can substitute $x^k K^b/(1 - K^a x)$ for the right-hand side.

In any of the three cases for $g(n)$ which we have considered, the resulting substitution or substitutions leads to an algebraic equation which can be solved for our generating function $F(x)$, so that $F(x)$ can be expressed as a rational function in x. (This step corresponds to taking a discrete Laplace transform.)

Now, we would like to rewrite the rational function expression we have for $F(x)$ in such a way that the coefficients $u(n)$ in $F(x)$ can be identified, thus giving us the solution to the recurrence relation. The method for doing this is a particular partial fraction decomposition; since we know that $1/(1 - Ax)^m = \sum_{n=0}^{\infty} \binom{m+n-1}{n} A^n x^n$, we are seeking a partial fraction decomposition of the form

$$\frac{P(x)}{Q(x)} = \sum_{i,j} \frac{B_{i,j}}{(1 - A_{i,j}x)^{m_j}}. \tag{5}$$

The way to do this is to note that if $R(x)$ is a monic polynomial, with factorization $(x - b_1)(x - b_2) \ldots (x - b_n)$, then substituting x^{n-i} for x^i in $R(x)$ yields a polynomial $R'(x)$ with factorization $(1 - b_1 x)(1 - b_2 x) \ldots (1 - b_n x)$ (since the coefficients of a polynomial are symmetric functions of the roots).

Thus, given $F(x) = P(x)/Q(x)$, we divide each term by the last term of $Q(x)$, substitute x^{n-i} for x^i in this polynomial, and find the roots of this new polynomial to get our desired partial fraction decomposition. Using our initial values, the appropriate values for the numerators $B_{i,j}$ are then found. Hence the solution $u(n)$ to our recurrence relation can be expressed as

$$u(n) = \sum_{i,j} B_{i,j} \binom{m_j+n-1}{n} A_{i,j}^n, \tag{6}$$

where the $A_{i,j}$ are the roots $b_1, b_2, \ldots, b_n$ (with multiplicities) of the polynomial obtained from $Q(x)$ by substitution. (This step corresponds to taking a discrete inverse Laplace transform.)

As an example of this method, again consider the Fibonacci numbers $u(n + 2) - u(n + 1) - u(n) = 0$, $u(0) = 0$, $u(1) = 1$. We get $(F(x) - u(0) - u(1)x) -$

$x(F(x) - u(0)) - x^2F(x) = 0$; hence $F(x) = x/(1 - x - x^2)$. The polynomial associated with $F(x)$ is $x^2 - x - 1$, and so we get the same solution as before. Other examples using MACSYMA are given in Section 5.

4. VARIABLE COEFFICIENT CASE

In this section we consider the case where the coefficients of the recurrence relation (1) are polynomials in n. One method which can be used to solve such variable coefficient recurrences is that of exponential generating functions; we assume that our generating function for the sequence $\{u(n)\}$ is of the form

$$Y(x) = \sum_{n=0}^{\infty} u(n) \frac{x^n}{n!}. \tag{7}$$

Taking successive derivatives of this equation shows that $Y^{(j)} = \sum_{n=0}^{\infty} u(n + j)x^n/n!$. Hence, if we multiply the recurrence relation (1) by $x^n/n!$ and sum from zero to infinity, we get an ordinary differential equation (ODE) for the generating function $Y(x)$. Once we solve this ODE for $Y(x)$, it follows from eq. (7) that if we expand $Y(x)$ in a Taylor series, then $u(n)$ is found by multiplying the nth term of this Taylor series by $n!$. Thus, this method converts the solution of a recurrence relation to the solution of a differential equation (somewhat like a discrete Fourier transform). This method can be programmed using the MACSYMA commands ODE2 and POWERSERIES.

As a simple example of this technique, consider the following recurrence relation:

$$u(n + 1) - (n + 1)u(n) = 1, \quad u(0) = 1, \tag{8}$$

where $u(n)$ is the total number of permutations of n distinct objects. (This is given as a problem in [5, p. 27].) The ODE is $Y'(1 - x) = Y + e^x$, with solution $Y(x) = e^x/(1 - x) = \sum_{n=0}^{\infty} (\sum_{j=0}^{n} 1/j!)x^n$. Thus the solution is $u(n) = n! \sum_{j=0}^{n} 1/j!$.

One of the problems with this algorithm is that there are no good methods for finding closed-form solutions to general variable coefficient ODEs. Indeed, the usual method of series solution expresses the solution to the ODE in terms of a recurrence relation for the coefficients of the power series. Thus the solutions to (variable coefficient) recurrence relations and the solutions to (variable coefficient) ODEs are intimately related. Another major problem is that of finding the Taylor series expansion of the solution to the ODE. We need to write this series expansion in the form $\sum_{n=0}^{\infty} a_n x^n$, where a_n is some (reasonable) function of n. It seems to be somewhat difficult to do this in any general way.

For first-order recurrence relations, this method is somewhat effective, inasmuch as MACSYMA is well equipped to solve first-order ODEs. However, even in this case, we still have the problem of finding the correct form for the Taylor series expansion of the solution to the ODE. Thus, using the POWERSERIES command, we are only able to solve a small number of first-order recurrences. Fortunately, in this case, there is an explicit closed-form solution available (e.g. see [6] or [7]). If the recurrence is $u(n + 1) - p(n)\,u(n) = g(n)$, then the solution is given by

$$u(n) = \prod_{i=0}^{n-1} p(i) \left(C + \sum_{i=0}^{n} (g(i)/\prod_{j=0}^{i-1} p(j)) \right)$$

where C is a constant determined by the initial condition $u(0)$. Using MACSYMA's facilities for simplification of sums, this can be programmed in a few lines.

For second-order recurrence relations, we can use results such as given in [8], which states that given the recurrence $u(n + 2) = (an + b)\, u(n + 1) + cu(n)$, with a, b, c integers, the solution $u(n)$ can be written as a linear combination of two Bessel functions. This can be implemented by simply matching the coefficients to the given recurrence relation. It is clear that this technique can be used for many other classes of special functions (e.g. Legendre functions, etc.). We merely indicate here that such a method of matching the given recurrence to a known equation may be the best possible way to approach the problem of finding closed-form solutions to variable coefficient recurrences in general.

Other special techniques for dealing with variable coefficient recurrence relations (e.g. as given in [2] or [7]) could be programmed, but we have not done so in this paper.

5. TESTING THE PROGRAMS

In this section we describe the results of a large number of testings of our programs. The following is a sample interaction with MACSYMA, which consists of solutions to problems and examples taken from [1, 2, 4, 9, 11, 12]. The program implementing the characteristic equation method is CHAR, and the program implementing the generating function method is GENF. The program implementing the exponential generating function method for variable coefficient recurrences is VARC1, while the program for first-order recurrences is VARC2. In all these cases, the argument list is E, G, U, N, K, IV where E is the left-hand side of eq. (1), G is the right-hand side of eq. (1), U and N are bindings, K is the order, and IV is the list of initial values.

```
(C49) CHAR(U(N+3)+6*U(N+2)+12*U(N+1)+8*U(N),0,U,N,3,[U(0)=1,U(1)=-2,U(2)=
8]);
                                          2
                                         N     N              N
(D52)                           U(N) = (-- - - + 1) (- 2)
                                         2     2

(C53) CHAR(U(N+1)-U(N),(1/6)*N*(N-1)*(N-2)+N-1,U,N,1,[U(0)=1]);
                                       3     2
                                      N     N     23 N    7
(D58)                    U(N) = N (-- - -- + ---- - -) + 1
                                      24    4      24     4

(C59) CHAR(U(N+2)-2*U(N+1)+U(N),N**2,U,N,2,[U(0)=0,U(1)=1]);
                                      2
                                  2  N     N    5       5 N
(D64)                    U(N) = N   (-- - - + --) + ---
                                     12    3   12       6

(C65) GENF(U(N+2)-U(N),2**N,U,N,2,[U(0)=1,U(1)=0]);
                                    N          N
                                   2     2 (- 1)
(D71)                      U(N) = -- + --------
                                   3        3

(C72) CHAR(U(N+2)-4*U(N),3+2*N,U,N,2,[U(0)=1,U(1)=0]);
                                 N              N
                              7 2     25 (- 2)      2 N    13
(D77)                  U(N) = ---- + --------- - --- - --
                               4          36         3     9
```

```
(C78) VARC1(U(N+1)-(N+1)*U(N),1,U,N,1,[U(Ø)=1]);
                                          N
                                         ====
                                         \        1
(D79)                        U(N) = N!    >      ----
                                         /        I6!
                                         ====
                                         I6 = Ø

(C8Ø) VARC2(U(N+1)-(N+1)*U(N),1,U,N,1,[U(Ø)=1]);
                              N - 1          N
                              /===\         ====
                               ! !          \             1
(D8Ø)                 U(N) = ( ! !  (I + 1))  >     -------------
                               ! !          /       I - 1
                              I = 1         ====    /===\
                                            I = Ø    ! !
                                                     ! !  (J + 1)
                                                     ! !
                                                    J = 1
```

(Note: A more general product simplifier might be included to rewrite these products as factorials, but this should be independent of our programs here. Moreover, MACSYMA should have better facilities for simplifying products in the near future.)

It is of some interest to compare the running times of CHAR and GENF. Using the 77 problems from the references, we found that CHAR had an average running time of 585 ms, while that for GENF was 1113 ms. Thus our implementation of CHAR is a much more efficient program and is the program to be used for solving constant coefficient recurrence relations. Also, VARC2 is much faster than VARC1, as would be expected.

6. REMARKS

We make the following observations:

(1) There is no simple analysis of the running time of these algorithms as a function of the order k of the recurrence relation. This involves the factorization of a polynomial, which does not depend in a direct way on its degree.

(2) The method of generating functions is often presented as an algorithm for solving recurrence relations. Our experience here seems to indicate that this is a poor algorithm for implementation purposes.

(3) After this paper was written, we became aware of a similar paper by Cohen and Katcoff [3]. Their methods seem somewhat more general (they deal with systems of recurrences also); however, our programs are much shorter and seem to have faster running times.

7. THE MACSYMA PROGRAMS

In this section we give a listing of the actual MACSYMA code for our programs.

```
/*THIS BLOCK CHECKS FOR A POLYNOMIAL IN N*/
POLYP(G,N):=BLOCK([D,F,C],
     G:RATEXPAND(G), IF FREEOF(N,G) THEN RETURN(TRUE),
     D:HIPOW(G,N), F:TRUE,
     FOR I:D STEP -1 THRU Ø DO
     (C:COEFF(G,N,I), IF NOT(FREEOF(N,C)) THEN F:FALSE,
      G:RATEXPAND(G-C*N**I)),
     RETURN(IS(G=Ø AND F)))$
```

```
/*THIS BLOCK CHECKS FOR A CONSTANT TO A POLYNOMIAL POWER*/
POLYINN(X,N):=BLOCK([B,E],
     IF INPART(X,0)="*" THEN
     RETURN(POLYINN(INPART(G,1),N) AND POLYINN(INPART(G,2),N)),
     IF INPART(X,0)#"**" THEN RETURN(FALSE),
     B:INPART(X,1),
     E:INPART(X,2),
     IF NOT FREEOF(N,B) THEN RETURN(FALSE),
     RETURN(POLYP(E,N)))$


/*THIS BLOCK IMPLEMENTS THE CHARACTERISTIC EQUATION METHOD*/
CHAR(E,G,U,N,K,IV):=BLOCK([GENSOL,HOMSOL,PARSOL,LOS,MULTIPLICITIES,
H,V,L,SS,DISPFLAG],
LOCAL(A,AA,B,R,M),
DISPFLAG:FALSE,
  FOR I:0 THRU K DO
    AA[I]:COEFF(E,U(N+K-I)),
    H:0,
  FOR I:0 THRU K DO
    H:H+AA[I]*U(N+K-I),
  IF H#E THEN RETURN("ERRONEOUS INPUT"),

  FOR I:0 THRU K DO
    H:SUBST(U**(K-I),U(N+K-I),H),

  MULTIPLICITIES:TRUE,
  LOS:SOLVE(H,U),
  FOR I:1 THRU LENGTH(LOS) DO
    (R[I]:LOS[I], R[I]:RHS(EV(R[I]))),
     M[I]:MULTIPLICITIES[I]),
  HOMSOL:
    SUM(SUM(A[I,J]*N**(M[I]-J),J,1,M[I])*R[I]**N,I,1,LENGTH(LOS)),
IF G=0 THEN
   (V:[ ],
    FOR I:1 THRU LENGTH(LOS) DO
    FOR J:1 THRU M[I] DO V:CONS(A[I,J],V),
       L:[ ],
    FOR Q:0 THRU K-1 DO L:CONS(SUBST(Q,N,HOMSOL)=U(Q),L),
    SS:EV(SOLVE(L,V),IV),
    RETURN(U(N)=(EV(HOMSOL,SS))))

ELSE IF POLYP(G,N) = TRUE THEN
   (G:RATEXPAND(G), PARSOL:SUM(B[J]*N**J,J,0,HIPOW(G,N)),
    FOR J:0 THRU K DO
     (L:0, V:E,
       FOR I:0 THRU K DO
       (L:RATEXPAND(SUBST(N+K-I,N,B[J]*N**J)),
        V:RATEXPAND(SUBST(L,U(N+K-I),V))),
        V:RATSIMP(V),
         IF V#0 THEN RETURN(V) ELSE PARSOL:N*PARSOL),
       V:E),
    FOR I:0 THRU K DO (L:RATEXPAND(SUBST(N+K-I,N,PARSOL)),
       V:RATEXPAND(SUBST(L,U(N+K-I),V))),
       L:[ ],
    FOR I:0 THRU HIPOW(PARSOL,N) DO
       L:CONS(COEFF(V=G,N,I),L),
       V:[ ],
    FOR J:0 THRU HIPOW(PARSOL,N) DO
       V:CONS(B[J],V),
    SS:SOLVE(L,V),
    PARSOL:EV(PARSOL,SS))

ELSE IF POLYINN(G,N) = TRUE THEN
   (PARSOL:B1*G,
    FOR J:0 THRU K DO
     (L:0, V:E,
       FOR I:0 THRU K DO
       (L:SUBST(N+K-I,N,PARSOL), V:SUBST(L,U(N+K-I),V)),
        V:RATSIMP(V),
         IF V#0 THEN RETURN(V) 'ELSE PARSOL:N*PARSOL),
    SS:SOLVE(V=G,B1),
    PARSOL:EV(PARSOL,SS))
```

```
  ELSE IF INPART(G,0)=SIN OR INPART(G,0) = COS THEN
     (PARSOL:B[1]*SIN(INPART(G,1)) + B[2]*COS(INPART(G,1)),
      FOR J:0 THRU K DO
       (L:0, V:E,
         FOR I:0 THRU K DO
         (L:EXPAND(SUBST(N+K-I,N,PARSOL)),
          V:EXPAND(SUBST(L,U(N+K-I),V))),
      V:TRIGEXPAND(V),
      IF V#0 THEN RETURN(V) ELSE PARSOL:N*PARSOL),
         V:E,
      FOR I:0 THRU K DO(L:EXPAND(SUBST(N+K-I,N,PARSOL)),
         V:EXPAND(SUBST(L,U(N+K-I),V))),
      V:TRIGEXPAND(V),
         L:[ ],
      LT:[SIN(INPART(G,1)),COS(INPART(G,1))],
      FOR JJ:1 THRU 2 DO
         L:CONS(COEFF(V=G,LT[JJ]),L),
         V:[ ],
      FOR J:1 THRU 2 DO
         V:CONS(B[J],V),
      SS:SOLVE(L,V),
      PARSOL:EV(PARSOL,SS))
  ELSE RETURN("CAN'T BE SOLVED IN CLOSED FORM BY PROGRAM"),

  GENSOL:HOMSOL + PARSOL,
     V:[ ],
  FOR I:1 THRU LENGTH(LOS) DO
  FOR J:1 THRU M[I] DO V:CONS(A[I,J],V),
     L:[ ],
  FOR Q:0 THRU K-1 DO
  L:CONS(SUBST(Q,N,GENSOL)=U(Q),L),
  SS:EV(SOLVE(L,V),IV),
  RETURN(U(N)=(EV(GENSOL,SS)))$



/*THIS BLOCK IMPLEMENTS THE GENERATING FUNCTION METHOD*/
GENF(E,G,U,N,K,IV):=BLOCK([MULTIPLICITIES,L,V,SS,VV,LOS,
NR,F,SOL,P,DISPFLAG],
LOCAL(A,AA,B),
DISPFLAG:FALSE,
  FOR I:0 THRU K DO
    AA[I]:COEFF(E,U(N+K-I)),
    H:0,
  FOR I:0 THRU K DO
    H:H+AA[I]*U(N+K-I),
  IF H#E THEN RETURN("ERRONEOUS INPUT"),

     L:E,
  FOR I:0 THRU K DO
    L:SUBST((F-SUM(U(J)*X**J,J,0,K-I-1))*X**I,U(N+K-I),L),

  IF G=0 THEN
     (S:SOLVE(L,F),
      F:EV(F,S))

  ELSE IF POLYP(G,N) = TRUE THEN
     (G:RATEXPAND(G),
      V:SUBST(X**K/(1-X)*COEFF(G,N,0),COEFF(G,N,0),G),
      VV:RATSIMP(DIFF(1/(1-X),X)),
      FOR I:1 THRU HIPOW(G,N) DO
         (V:SUBST(X**K*X*VV*COEFF(G,N,I),COEFF(G,N,I)*N**I,V),
          VV:RATSIMP(DIFF(X*VV,X))),
      V:RATSIMP(V),
      SS:SOLVE(L=V,F),
      F:EV(F,SS))

  ELSE IF POLYINN(G,N) = TRUE AND HIPOW(INPART(G,2),N) < 2 THEN
     (G1:(X**K)*(INPART(G,1)**COEFF(INPART(G,2),N,0)),
      G2:1 - X*(INPART(G,1)**COEFF(INPART(G,2),N,1)),
      V:RATSIMP(G1/G2),
      SS:SOLVE(L=V,F),
      F:EV(F,SS))
```

```
  ELSE RETURN("CAN'T BE SOLVED IN CLOSED FORM BY PROGRAM"),


  MULTIPLICITIES:TRUE,
  LOS:SOLVE(NEWRAT(F),X),
  FOR I:1 THRU LENGTH(LOS) DO
     (R[I]:LOS[I], R[I]:RHS(EV(R[I]))),
      M[I]:MULTIPLICITIES[I]),

     V:[ ],
  B:PRODUCT((1-R[I]*X)**M[I],I,1,LENGTH(LOS)),
  FOR I:1 THRU LENGTH(LOS) DO
  FOR J:1 THRU M[I] DO
     (P[I,J]:B*A[I,J]/((1-R[I]*X)**J),  V:CONS(A[I,J],V)),
  P:SUM(SUM(P[I,J],J,1,M[I]),I,1,LENGTH(LOS)),

     L:[ ],
  NF:RATEXPAND(NUM(F)/ABS(COEFF(DENOM(F),X,0))), P:RATEXPAND(P),
  FOR I:0 THRU HIPOW(RATEXPAND(B),X)-1  DO
     L:CONS(COEFF(NF=P,X,I),L),
  SSS:EV(SOLVE(L,V),IV),

  SOL:SUM(SUM(A[I,J]*COEFF(DENOM(F),X,0)/ABS(COEFF(DENOM(F),X,0))*
     BINOMIAL(J+N-1,N)*R[I]**N,J,1,M[I]),I,1,LENGTH(LOS)),

  RETURN(U(N)=(EV(SOL,SSS))))$


/*THIS BLOCK FINDS THE NEW POLYNOMIAL ASSOCIATED TO F*/
NEWRAT(F):=BLOCK([HD,CP,DP],
     HD:HIPOW(DENOM(F),X),
     CP:COEFF(DENOM(F),X,HD),
     DP:SUM((COEFF(DENOM(F),X,I))/CP*X**I,I,0,HD),
     RETURN(SUM(COEFF(DP,X,HD-I)*X**I,I,0,HD)))$




/*THIS BLOCK IMPLEMENTS THE VARIABLE COEFFICIENT METHOD*/
VARC1(E,G,U,N,K,IV):=BLOCK([V,VV,EQ,Y,CAUCHYSUM,FINSOL,SERSOL,DISPFLAG],
LOCAL(A,B),DISPFLAG:FALSE,
  FOR I:0 THRU K DO
    (A[I]:COEFF(E,U(N+I)),
     A[I]:RATEXPAND(A[I]),
     IF POLYP(A[I],N)=FALSE THEN RETURN("CAN'T DO IT")),
  IF K=2 AND (B:BESSELCHECK(E,K) # FALSE) THEN RETURN(B),
  V:RATEXPAND(E),
  FOR I:K STEP -1 THRU 0 DO
     FOR J:HIPOW(A[I],N) STEP -1 THRU 0 DO
       (V:RATSUBST(X**J*'DIFF(Y,X,I+J),N**J*U(N+I),V),
        V:RATEXPAND(V)),
  V:RATSUBST(Y,'DIFF(Y,X,0),V),
  V:RATEXPAND(V),
  IF POLYP(G,N) = TRUE THEN
    (G:RATEXPAND(G), VV:G,
     FOR I:0 THRU HIPOW(G,N) DO
        VV:SUBST(X**I,N**I,VV),
        VV:%E**X*VV)
  ELSE RETURN("CAN'T DO IT"),
     EQ:V-VV,
  DEPENDENCIES(Y(X)),
  SOL:ODE2(EQ=0,Y,X),
        IF K=1 THEN FINSOL:IC1(SOL,X=0,Y=EV(U(0),IV))
        ELSE IF K=2 THEN FINSOL:IC2(SOL,X=0,Y=EV(U(0),IV),'DIFF(Y,X)=EV(U(1),IV!
))
  ELSE RETURN("O.D.E. CAN'T BE SOLVED AT PRESENT BY MACSYMA"),
     CAUCHYSUM:TRUE,
  SERSOL:POWERSERIES(RHS(FINSOL),X,0), SERSOL:EXPAND(SERSOL),
  B:INPART(SERSOL,1),
  B:EV(B,X=1),
  IF ATOM(B)=FALSE THEN B:SUBSTPART(N,B,4),
  RETURN(U(N)=((N!)*B)))$


/*THIS BLOCK CHECKS FOR A BESSEL RECURRENCE RELATION*/
```

```
BESSELCHECK(E,K):=BLOCK([A,ANS],
LOCAL(A),
     FOR I:0 THRU K DO
     (A[I]:COEFF(E,U(N+I)),
      A[I]:RATEXPAND(A[I])),
     IF NOT(INTEGERP(A[0])) THEN RETURN(FALSE),
     IF NOT(INTEGERP(EV(A[1],N=0))) THEN RETURN(FALSE),
     IF NOT(HIPOW(A[1],N)=1) THEN RETURN(FALSE),
     IF NOT(INTEGERP(COEFF(A[1],N,1))) THEN RETURN(FALSE),
     IF NOT(A[2]=1) THEN RETURN(FALSE),
     ANS:"A LINEAR COMBINATION OF BESSEL FUNCTIONS",
     /*EXACT DETAILS ARE OF NO SIGNIFICANCE,SINCE WE ARE MERELY
       DEMONSTRATING THE FEASIBILITY OF THIS APPROACH*/
     RETURN(ANS))$



/*THIS BLOCK IMPLEMENTS THE FIRST ORDER METHOD*/
VARC2(E,G,U,N,K,IV):=BLOCK([H,P,V,C,SOL],
LOCAL(AP,P),
     P:(-1)*COEFF(E,U(N))/COEFF(E,U(N+1)),
     V:G/COEFF(E,U(N+1)),
     S[J]:SUBST(J,N,P),
     S[I]:SUBST(I,N,P),
     P[N]:PRODUCT(S[I],I,1,N-1),
     H[I]:SUBST(I,N,V)/PRODUCT(S[J],J,1,I-1),
     V1:SUM(H[I],I,0,N),
     AP:EV(U(0)-SUBST(0,N,V),IV),
     RETURN(U(N)=AP*P[N]+P[N]*V1))$
```

ACKNOWLEDGMENT

I would like to thank Richard Fateman for suggesting the problem discussed here, as well as for all of his helpful conversations during the writing of this paper.

REFERENCES

1 ANDERSON, I. *A First Course in Combinatorial Mathematics* Oxford U. Press, London, 1974
2 BRAND, L *Differential and Difference Equations* Wiley, New York, 1966.
3 COHEN, J., AND KATCOFF, J. Symbolic solution of finite-difference equations *ACM Trans. Math. Software 3*, 3 (Sept. 1977), 261–271.
4. GOLDBERG, S *Introduction to Difference Equations.* Wiley, New York, 1958.
5 HALL, M , JR , *Combinatorial Theory.* Blaisdell, Waltham, Mass. 1967.
6. HILDEBRAND, F B *Finite Difference Equations and Simulations.* Prentice-Hall, Englewood Cliffs, N J., 1968.
7. JORDAN, C. *Calculus of Finite Differences* Chelsea, New York, reprint, 1950.
8. LEHMER, D.H. Arithmetical periodicities of Bessel functions. *Annals of Math. 33*, 1 (Jan. 1932), 143–150
9. LIU, C.L. *Introduction to Combinatorial Mathematics* McGraw-Hill, New York, 1968.
10 Mathlab Group. MACSYMA Reference Manual. Lab. for Comptr. Sci., M.I.T., Cambridge, Mass
11. RIORDAN, J *An Introduction to Combinatorial Analysis* Wiley, New York, 1958.
12 RYSER, H.J. Combinatorial Mathematics. Carus Monographs 14, Math. Assoc. Amer., Washington, D.C., 1963.

Received May 1977