ALTRAN PROCEDURES FOR THE CHEBYSHEV SERIES SOLUTION OF LINEAR ODE'S

K.O. Geddes
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF WATERLOO

RESEARCH REPORT CS-77-05

February 1977

This research was supported by the National Research Council of Canada under Grant A8967.

Abstract.

If a linear ordinary differential equation with polynomial coefficients is converted into integrated form then the formal substitution of a Chebyshev series leads to recurrence equations defining the Chebyshev coefficients of the solution function. An explicit formula is presented for the polynomial coefficients of the integrated form in terms of the polynomial coefficients of the differential form. Then the symmetries arising from multiplication and integration of Chebyshev polynomials are exploited in deriving a general recurrence equation from which can be derived all of the linear equations defining the Chebyshev coefficients. A method of backward recurrence is described for solving the recurrence equations. A set of ALTRAN procedures is presented which implements both the derivation of the general recurrence equation and the solution of the resulting linear equations. The ALTRAN program is applied to several sample problems. Since the method is algebraic it can also be applied to problems containing indeterminates.

TABLE OF CONTENTS

1. Introduction

The most widely used methods for computing the numerical solution of an ordinary differential equation (ODE), in the form of either an initial-value problem (IVP) or a boundary-value problem (BVP) are discrete-variable methods. That is to say, the solution is obtained in the form of discrete values at selected points. Methods for computing an approximate solution in the form of a continuous function (usually a polynomial or rational function) have received some attention in the literature. Probably the best known continuous-variable method is the Lanczos tau-method [7] which is closely related to the Chebyshev series methods of Clenshaw [1] and Fox [3] for linear ODEs. The Chebyshev series method has also been used for a first-order non-linear ODE ([9], [8]) but the method then requires iteration whereas it is a direct method in the case of linear ODEs. More recently, the Chebyshev series method has been extended to the solution of parabolic partial differential equations ([6], [2]).

The most extensive treatment of Chebyshev series methods is contained in the book by Fox and Parker [4]. The basic approach is series substitution followed by the solution of resulting recurrence equations. All of the authors treat the series substitution and generation of the recurrence equations as a hand computation prior to the application of a numerical procedure for solving the recurrence equations. However, except for particularly simple special cases, the generation of the recurrence equations is a tedious and error-prone hand manipulation which could well be programmed in a language for symbolic computation. In this paper, procedures are described for generating the recurrence equations for arbitrary-order linear ODEs with polynomial coefficients. There is no need to restrict the method to first and second order equations as previous authors have done.

Furthermore, the method can also be applied to problems containing indeterminates (for example, indeterminate initial conditions) and to eigenvalue problems. An attractive feature of the method is that the associated conditions may be of initial-value type, boundary-value type, or any linear combination of function and derivative values at one or more points.

The procedures described here have been implemented in the ALTRAN language. Once the recurrence equations have been generated their solution could, in the standard case, be accomplished by a numerical procedure rather than a symbolic procedure. However, in the potentially powerful application of the method to problems containing indeterminates a symbolic solution of the recurrence equations will sometimes be desired. Therefore this second phase has also been coded in the ALTRAN language. The standard problem without indeterminates is obviously a prime candidate for a hybrid symbolic/numeric computational procedure. In keeping with the potential desire for a symbolic solution, we restrict our attention to a class of problems for which the truncated Chebyshev series can be obtained by a direct method. Thus we consider only linear ODEs with polynomial coefficients. Of course, a linear ODE whose coefficients are rational functions could be converted to one with polynomial coefficients and therefore, in principal, the method can be applied to any linear ODE whose coefficients are functions which can be approximated well by rational functions.

The method assumes that the solution is desired in the interval $[-1, 1]$ (which means that a simple transformation of variables will be required, in general, before applying the method). The truncated Chebyshev series produced by the method is a near-minimax polynomial approximation of the true solution to the problem. This is based on the fact that, for any function continuous in $[-1, 1]$, the minimax error in the truncated Chebyshev series of degree n is never appreciably larger than the error in the best minimax polynomial of degree n (e.g. see [10]). The goodness of the approximate solution obtained therefore depends on the ability of polynomials to approximate the true solution. A more powerful class of approximating functions is the rational functions. However, the computation of near-minimax rational functions would be best accomplished in the form of Chebyshev-Pade approximations [5] which require, as an initial step, the generation of Chebyshev series coefficients. Thus the method discussed in this paper is a basic building block as well as a powerful method in its own right.

2. Conversion to Integrated Form

Consider an ordinary differential equation of order ν with polynomial coefficients:

$$p_\nu(x)\ y^{(\nu)}(x) + \dots + p_1(x)\ y'(x) + p_o(x)\ y(x) = r(x)\ . \qquad (1)$$

We will temporarily ignore the ν associated conditions which would serve to specify a unique solution of (1). We seek a solution of the form

$$y(x) = \sum_{k=0}^{\infty}{}' c_k T_k(x) \qquad (2)$$

where the prime (') indicates the standard convention that the first coefficient is to be halved and where $T_k(x)$ denotes the Chebyshev polynomial of the first kind:

$$T_k(x) = \cos\ (k \arccos\ x).$$

If the series (2) is substituted into the differential equation (1) then the left side of (1) can be expressed in the form of a Chebyshev series. By expressing the right-hand side polynomial $r(x)$ in the Chebyshev form, we can equate coefficients on the left and right to obtain an infinite set of linear equations in the unknowns c_o, c_1, c_2, ... (see [4]). (There will be ν additional equations derived from the associated conditions.) This infinite linear system has the property that the lower triangular part is zero except for a few sub-diagonals and it therefore becomes finite under the assumption $c_k = 0$ $(k > kmax)$, for some chosen kmax. This assumption must be valid, to within some absolute error tolerance, if the solution $y(x)$ is to have a convergent Chebyshev series expansion. Thus one may solve the linear system, for increasing values of kmax, until some convergence criterion has been satisfied.

However, as is noted in [4], the linear system is much simpler if

(1) is first converted to integrated form. This is because the series resulting from indefinite integration of (2) is much simpler than the series resulting from formal differentiation. Specifically, formal differentiation of (2) yields

$$y'(x) = \sum_{k=0}^{\infty}{}' \left[\sum_{i=k}^{\infty} 2(2i+1)\, c_{2i+1} \right] T_{2k}(x)$$

$$+ \sum_{k=0}^{\infty} \left[\sum_{i=k}^{\infty} 2(2i+2)\, c_{2i+2} \right] T_{2k+1}(x) \qquad (3)$$

while indefinite integration of (2) yields

$$\int y(x) = \sum_{k=1}^{\infty} (1/2k)\,(c_{k-1} - c_{k+1})\, T_k(x) \;+\; K \qquad (4)$$

(where K denotes an arbitrary constant). The end result is that in the infinite linear system derived from the integrated form of the differential equation (1), each individual equation contains only a finite number of terms. In the original (differential) form, each individual equation in the infinite linear system is itself infinite. Thus a very substantial reduction in complexity is achieved by considering the integrated form. The coefficients are then specified as the solution of a finite recurrence relation (with non-constant coefficients) rather than an infinite recurrence relation.

The derivation of the recurrence equation is described in detail in the next section. The following theorem gives a formula for the polynomial coefficients of the integrated form of the order ν differential equation (1), in terms of the polynomials in the original form. This formula for the new polynomials is readily incorporated into a program written in any of the computer languages for symbolic computation, since each new polynomial is specified explicitly as a linear combination of derivatives of the original polynomials (and the new right-hand side is obtained by integrating the original right-hand side polynomial).

THEOREM 1:

The ordinary differential equation (1) of order ν with polynomial coefficients $p_\nu(x), \ldots, p_o(x)$ and right-hand side polynomial $r(x)$ is equivalent to the integrated form

$$q_o(x)\, y(x) + \int q_1(x)\, y(x) + \ldots + \int\int \overset{\nu}{\ldots} \int q_\nu(x)\, y(x) = s(x) + K_\nu(x) \qquad (5)$$

where the polynomial coefficients $q_o(x), \ldots, q_\nu(x)$ are given by

$$q_m(x) = \sum_{k=0}^{m} (-1)^{m-k} \binom{\nu-k}{m-k} p_{\nu-k}^{(m-k)}(x), \quad 0 \le m \le \nu \qquad (6)$$

and where the right-hand side polynomial $s(x)$ is given by

$$s(x) = \int\int \overset{\nu}{\ldots} \int r(x) \quad . \qquad (7)$$

In (5) - (7) the notation $K_\nu(x)$ denotes an arbitrary polynomial of degree ν-1 arising from the constants of integration and the notations

$$\int\int \overset{i}{\ldots} \int f(x) \text{ and } f^{(i)}(x)$$

denote the results of applying, respectively, indefinite integration i times and formal differentiation i times to the function $f(x)$.

Proof: The right-hand side (7) is obvious so we consider only the left-hand side of (5). The arbitrary constants of integration will be ignored in this proof.

The proof is by induction on the order ν. For ν=1, the left side of the differential equation (1) is

$$p_1(x)\, y'(x) + p_o(x)\, y(x). \qquad (8)$$

Applying integration once yields, from the first term in (8),

$$p_1(x)\, y(x) - \int p_1'(x)\, y(x)$$

and hence (8) becomes

$$p_1(x)\ y(x) + \int[-p_1'(x) + p_o(x)]\ y(x)$$

which agrees with (5) - (6).

Now for the induction step, assume that the theorem has been proved for order ν-1. For order ν, the left side of equation (1) is

$$p_\nu(x)\ y^{(\nu)}(x) + \sum_{k=1}^{\nu} p_{\nu-k}(x)\ y^{(\nu-k)}(x). \tag{9}$$

The result of applying integration ν times to the <u>first</u> term in (9) can be expressed in the form

$$p_\nu(x)\ y(x) - \iint \overset{\nu}{\ldots} \int \sum_{k=1}^{\nu} \binom{\nu}{k} p_\nu^{(k)}(x)\ y^{(\nu-k)}(x). \tag{10}$$

This is easily verified since ν differentiations of (10) simply strip off the integral signs from the second term of (10) and the first term of (10) becomes

$$p_\nu(x)\ y^{(\nu)}(x) + \{\text{the integrand in (10)}\}.$$

Thus, the result of applying integration ν times to the whole of (9) is

$$p_\nu(x)\ y(x) + \iint \overset{\nu}{\ldots} \int \{ \sum_{k=1}^{\nu} \left[- \binom{\nu}{k}\ p_\nu^{(k)}(x) + p_{\nu-k}(x)\right] y^{(\nu-k)}(x)\}. \tag{11}$$

In the integrand in (11), if the index of summation is changed by $k \leftarrow k+1$ it takes the form

$$\sum_{k=0}^{\nu-1} \left[-\binom{\nu}{k+1}\ p_\nu^{(k+1)}(x) + p_{\nu-1-k}(x) \right]\ y^{(\nu-1-k)}(x) \tag{12}$$

which is the left side of a differential equation of order ν-1 of the form (1) with polynomial coefficients $P_{\nu-1}(x)$,..., $P_o(x)$ defined by

$$P_{\nu-1-k}(x) = -\binom{\nu}{k+1} p_\nu^{(k+1)}(x) + p_{\nu-1-k}(x),\ 0 \le k \le \nu\nu-1\ . \tag{13}$$

Therefore, by the induction assumption, the result of applying integration ν-1 times to (12) is the left side of the integrated form (5) for order ν-1, with p replaced by P in (6). Applying the definition of P given by (13), we have transformed (11) into the form

$$p_\nu(x)\ y(x) + \int \{\ \} \tag{14}$$

where the expression { } is the integrated form of (12), namely

$$\sum_{m=0}^{\nu-1} \int\int \overset{m}{\dots} \int \{ \sum_{k=0}^{m} (-1)^{m-k} \binom{\nu-1-k}{m-k} \left[- \binom{\nu}{k+1}\ p_\nu^{(m+1)}(x) + p_{\nu-1-k}^{(m-k)}(x) \right] \} y(x)\ , \tag{15}$$

Finally, changing the indices of summation in (15) by m ← m-1 and k ← k-1, breaking the k-summation into its two parts, and bringing the extra integral in (14) inside the m-summation, (14) becomes

$$p_\nu(x)\ y(x) + \sum_{m=1}^{\nu} \int\int \overset{m}{\dots} \int \{ (-1)^m \binom{\nu}{m} p_\nu^{(m)}(x) + \sum_{k=1}^{m} (-1)^{m-k} \binom{\nu-k}{m-k} p_{\nu-k}^{(m-k)}(x)\ \}\ y(x) \tag{16}$$

which agrees with (5) - (6).

In the final step above, we have used the fact that the expression

$$- \sum_{k=1}^{m} (-1)^{m-k} \binom{\nu-k}{m-k} \binom{\nu}{k} \tag{17}$$

reduces to $(-1)^m \binom{\nu}{m}$. To see this, simply use the factorial definitions to verify that

$$\binom{\nu-k}{m-k}\binom{\nu}{k} = \binom{m}{k}\binom{\nu}{m}$$

and hence (17) becomes

$$(-1)^m \binom{\nu}{m} \sum_{k=1}^{m} (-1)^{k+1} \binom{m}{k}.$$

But the summation here has the value 1.

3. General Form of the Recurrence Equation

For an ordinary differential equation of order ν in the integrated form (5) we seek a solution in the form of the Chebyshev series (2). Substituting (2) into the left side of (5) and removing the summation sign and the c_k outside the integral signs yields

$$\sum_{k=0}^{\infty}{}' c_k \{ q_0(x)T_k(x) + \int q_1(x)T_k(x) + \ldots + \int\int\overset{\nu}{\ldots}\int q_\nu(x)T_k(x)\} \ . \qquad (18)$$

In order to express (18) in the form of a Chebyshev series (where the coefficient of $T_k(x)$ will be a linear combination of c_i's), the polynomials $q_0(x),\ldots,q_\nu(x)$ are converted into Chebyshev form. Then the following identities (cf.[4]) are applied:

$$T_k(x)T_j(x) = (1/2)\ T_{k+j}(x) + (1/2)\ T_{k-j}(x) \qquad (19)$$

$$\int T_i(x) = (1/2(i+1))\ T_{i+1}(x) - (1/2(i-1))\ T_{i-1}(x) \qquad (20)$$

where, for the moment, we may assume that k is "large enough" in (19) and that i is "large enough" in (20) to avoid non-positive subscripts. This transforms (18) into the following form, for k large enough (i.e. neglecting the first few terms):

$$\sum_k c_k \{v_0 T_{k+h}(x) + v_1 T_{k+h-1}(x) + \ldots + v_{2h} T_{k-h}(x)\} \qquad (21)$$

where the coefficients $v_i (0 \le i \le 2h)$ are rational expressions in k arising from repeated applications of (19) and (20) and h is some positive integer. Then changing the indices of summation in (21), separately in each term, converts (21) into a Chebyshev series of the following form (neglecting the first few terms):

$$\sum_k \{ u_0 c_{k-h} + u_1 c_{k-h+1} + \dots + u_{2h} c_{k+h} \} T_k(x) \tag{22}$$

where the coefficients $u_i (0 \le i \le 2h)$ are rational expressions in k. The first few terms **could** be derived independently. Finally, by converting the right-hand-side polynomial in (5) into Chebyshev form, we are ready to equate coefficients and solve for the c_i's. The coefficients of $T_0(x)$, ..., $T_{\nu-1}(x)$ would not be equated because of the arbitrary term $K_\nu(x)$ appearing in (5). Instead the first ν equations would come from the associated conditions.

The following example will serve to illustrate. Consider the problem:

$$(1+x^2)\ y''(x) - y'(x) + x\ y(x) = (2-x^2) \tag{23}$$

$$y(0) = 0; \quad y'(0) = 1. \tag{24}$$

The integrated form of (23) is, from (5) - (7),

$$(1+x^2)\ y(x) + \int (-1-4x)\ y(x) + \iint (2+x)\ y(x) = x^2 - (1/12)\ x^4 + K_2(x)\ . \tag{25}$$

Substituting (2) into (25) and converting the polynomials into Chebyshev form yields

$$\begin{aligned}\sum_{k=0}^{\infty}{}' c_k \{ &[(3/2)T_0(x) + (1/2)\ T_2(x)]\ T_k(x) \\ &+ \int[-T_0(x) - 4T_1(x)]\ T_k(x) \\ &+ \iint[2T_0(x) + T_1(x)]\ T_k(x)\} \\ &= (11/24)\ T_2(x) - (1/96)\ T_4(x) + K_2(x)\end{aligned} \tag{26}$$

where some constant terms on the right have been absorbed into the arbitrary linear term $K_2(x)$. Applying the identities (19) and then (20) yields, after much manipulation, the following form for the factor { } in (26), for k large enough:

$$\{ [8(k+2)(k+3)]^{-1}\, T_{k+3}(x) + (1/4 - (k+2)^{-1} + [2(k+1)(k+2)]^{-1})\, T_{k+2}(x)$$
$$+ (-[2(k+1)]^{-1} - [8(k+1)(k+2)]^{-1})\, T_{k+1}(x) + (3/2 - [(k-1)(k+1)]^{-1})\, T_k(x)$$
$$+ ([2(k-1)]^{-1} - [8(k-1)(k-2)]^{-1})\, T_{k-1}(x)$$
$$+ (1/4 + (k-2)^{-1} + [2(k-1)(k-2)]^{-1})\, T_{k-2}(x) + [8(k-2)(k-3)]^{-1} T_{k-3}(x)\} \,. \quad (27)$$

To obtain the general coefficient of $T_k(x)$ on the left side of (26), the index of summation must be changed separately in each term of the factor (27). For example, for the first term

$$\sum_k c_k\, [8(k+2)(k+3)]^{-1}\, T_{k+3}(x)$$

the desired change of index is $k \leftarrow k-3$, which yields

$$\sum_k [8(k-1)k]^{-1}\, c_{k-3}\, T_k(x)$$

where again we are neglecting the first few terms in the series. After changing the indices of summation appropriately, the left side of equation (26) becomes

$$\sum_k \{ [8k(k-1)]^{-1} c_{k-3} + (1/4 - 1/k + [2k(k-1)]^{-1}) c_{k-2}$$

$$+ (-[2k]^{-1} - [8k(k+1)]^{-1}) c_{k-1} + (3/2 - [(k-1)(k+1)]^{-1}) c_k$$

$$+([2k]^{-1} - [8k(k-1)]^{-1}) c_{k+1} + (1/4 + 1/k + [2k(k+1)]^{-1}) c_{k+2}$$

$$+ [8k(k+1)]^{-1} c_{k+3}\} T_k(x) \quad . \qquad (28)$$

Working out the first few terms using special cases (see section 4) of identities (19) and (20), and obtaining the first two equations from the two associated conditions (24), we obtain the following infinite set of linear equations which define the Chebyshev coefficients of the solution function y(x):

$$1/2\ c_0 - c_2 + c_4 - c_6 + \dots \qquad = 0$$

$$c_1 - 3c_3 + 5c_5 - 7c_7 + \dots \qquad = 1$$

$$-5/24\ c_1 + 7/6\ c_2 + 3/16\ c_3 + 5/6\ c_4 + 1/48\ c_5 \qquad = 11/24 \qquad (29)$$

$$1/48\ c_0 + 0\ c_1 - 17/96\ c_2 + 11/8\ c_3 + 7/48\ c_4 + 15/24\ c_5 + 1/96\ c_6 \quad = 0$$

$$1/96\ c_1 + 1/24\ c_2 - 21/160\ c_3 + 43/30\ c_4 + 11/96\ c_5 + 21/40\ c_6 + 1/160\ c_7$$

$$\begin{matrix} . & . & . \\ . & . & . \end{matrix} \qquad = -1/96$$

The remaining equations are obtained by equating to zero the coefficient of $T_k(x)$ in (28), for k = 5,6,7, ... Note that (29) is a 7-diagonal system starting from the fourth equation.

In general, the desired Chebyshev coefficients satisfy a (2h+1) - term linear recurrence equation of the form

$$u_0 c_{k-h} + u_1 c_{k-h+1} + \dots + u_{2h} c_{k+h} = 0 \qquad , \qquad (30)$$

where the coefficients u_i are rational expressions in k. Equation (30) will be valid for $k \geq h$ except that the first few right-hand-sides may be nonzero depending on the degree of the right-hand-side polynomial in (5). The value of h depends on the order ν of the differential equation and on the degree of the left-hand-side polynomials in the integrated form (5). Each application of the product formula (19) and each application of the integration formula (20) increases the value of h by one. Thus in the example of equation (25), the second-degree polynomial in the first term indicates that h will be at least 2, the first-degree polynomial in the second term indicates that h will be at least 1 + 1 = 2 (1 for the product and 1 for the integration), and the first-degree polynomial in the third term indicates that h will be at least 1 + 2 = 3 (1 for the product and 2 for the double integration). The final value of h will be max $\{2,2,3\} = 3$ as is verified by expression (28). In general, lower and upper bounds on h can be readily determined from the original order-ν differential equation (1); namely, if maxdeg is the maximum of the degrees of the left-hand-side polynomials in (1) then

$$\nu \leq h \leq \nu + \text{maxdeg} \quad . \qquad (31)$$

The first ν equations in the infinite linear system come from the associated conditions and will be equations containing an infinite number of terms. If $h > \nu$ then there will follow $\nu-h$ "special" cases of the general recurrence equation (30), with nonzero right-hand-sides in general, resulting from equating the coefficients of the terms $T_\nu(x), \ldots, T_{h-1}(x)$ on the left and right of the transformed form of equation (16). The remaining linear equations result from equating the coefficients of $T_k(x)$, $k = h, h+1, \ldots$ on the left and right and will all be in the form of recurrence equation (30) except that there will be a few more nonzero right-hand-sides if

$$\deg[s(x)] \geq h,$$

where $s(x)$ is the right-hand-side polynomial in the integrated form (5).

4. Special Cases of the Recurrence Equation

The derivation of the general recurrence equation (30) as described in section 3 is not difficult to implement in a symbolic language. We now consider the derivation of the "special" equations which require the application of modified versions of the product formula (19) and the integration formula (20). In other words, we now want to consider what happens when we drop the assumption that k is "large enough" which was assumed in the derivation of equation (30).

The product formula (19) is in fact correct for all values of k and j if the subscript k-j is replaced by $|k-j|$. The integral formula (20) has a special form for the cases i = 0 and i = 1, namely

$$\int T_0(x) = T_1(x) \quad \text{and} \quad \int T_1(x) = 1/4T_2(x) \qquad (32)$$

where an arbitrary constant of integration is implied. These special cases could be incorporated into a program for generating the recurrence equations but the cost of deriving each individual "special" equation would be approximately equal to the cost of deriving the one general equation (30). Fortunately, the form of the special equations can be deduced immediately from the general equation without extra work. Referring to the example in section 3, the third equation of (29) arises from equating coefficients of $T_2(x)$ in the transformed form of (26). If we "blindly" obtain the left-side coefficient of $T_2(x)$ by setting k=2 in the general formula (the bracketed expression in (28)) we obtain the equation

$$1/16\ c_{-1} + 0\ c_0 - 13/48\ c_1 + 7/6\ c_2 + 3/16\ c_3 + 5/6\ c_4 + 1/48\ c_5 = 11/24\ . \tag{33}$$

If the negative subscript is interpreted in absolute value - i.e. if we equate c_{-1} with c_1 - then the third equation of (29) results. Our task is now to prove that this "rule" holds in general.

The main point is that negative subscripts may be carried throughout the derivation and their interpretation in absolute value may be postponed until the final step. We prove firstly, in Theorem 2, that negative subscripts are valid when applying identities (19) and (20) to transform expression (18) into the form (21). Secondly, we prove via Theorems 3 and 4 that in the substitutions used to transform (21) into (22) the occurrence of terms c_i with i negative are valid if the subscripts are interpreted in absolute value.

<u>THEOREM 2:</u>

Identities (19) and (20) are valid when non-positive subscripts occur on the left and/or right in the sense that $T_i(x)$ represents $T_{|i|}(x)$.

<u>Proof:</u> For the product formula (19) this is a well-known result which is a simple consequence of the cosine definition of $T_k(x)$.

The integration formula (20) is certainly valid for expressing the integral of $T_i(x)$ for $i \geq 2$. Consider the case $i = -j$ for $j \geq 2$. (The negative subscripts will arise from previous applications of either (19) or (20). $T_{-j}(x)$ is interpreted as $T_j(x)$ and we are proving the validity of explicitly carrying the negative subscripts). Formula (20) would give

$$\int T_{-j}(x) = (1/2(-j+1))\, T_{-j+1}(x) - (1/2(-j-1))\, T_{-j-1}(x) \quad . \tag{34}$$

The right side of (34) is interpreted as

$$-(1/2(j-1))\, T_{j-1}(x) + (1/2(j+1))\, T_{j+1}(x)$$

which agrees with formula (20) applied to $\int T_j(x)$.

Next consider the application of formula (20) when $i = 0$. Direct application gives

$$\int T_0(x) = (1/2)\, T_1(x) - (1/2(-1))\, T_{-1}(x) \quad . \tag{35}$$

The right side of (35) is interpreted as

$$(1/2)\, T_1(x) + (1/2)\, T_1(x) = T_1(x)$$

which agrees with the special formula in (32).

Finally, consider the application of formula (20) when $i = 1$ and $i = -1$. Direct application for $i = 1$ appears to present a problem since we get

$$\int T_1(x) = (1/4)\, T_2(x) - (1/0)\, T_0(x) \quad . \tag{36}$$

However, recalling that there is an implied arbitrary constant in the integration formula (20), the "troublesome" term here is simply absorbed into the arbitrary constant. (In practice, this means that we will never have cause to evaluate the coefficient of $T_0(x)$.) Formula (36) therefore agrees with the special formula in (32). The case $i = -1$ follows immediately. Furthermore, if a subsequent integration is applied to the right side of (36) a "troublesome" coefficient of $T_1(x)$ will arise, but then there will be an arbitrary linear term (due to the double integration) which absorbs all linear terms. Q.E.D.

The following simple example will clarify the application of Theorem 2. Consider the differential equation

$$y'(x) + y(x) = 0$$

or, in integrated form,

$$y(x) + \int y(x) = 0 \quad . \tag{37}$$

Substituting the series (2) into (37) yields

$$\sum_{k=0}^{\infty}{}' \; c_k \{ T_k(x) + \int T_k(x) \} = 0 \quad . \tag{38}$$

Applying formula (20) gives

$$\sum_{k=0}^{\infty}{}' \; c_k \{ T_k(x) + (1/2(k+1)) \; T_{k+1}(x) - (1/2(k-1)) \; T_{k-1}(x) \} = 0 \; . \tag{39}$$

The third term in brackets would cause trouble if we evaluated it for k = 1 but we will never do so because we do not equate coefficients of $T_0(x)$. Continuing with the example, the next step is to change indices of summation in (39) yielding

$$\sum_{k=0}^{\infty}{}' \; c_k T_k(x) + \sum_{k=1}^{\infty}{}' \; (1/2k) \; c_{k-1} T_k(x) - \sum_{k=-1}^{\infty}{}' \; (1/2k) \; c_{k+1} T_k(x) = 0 \, . \tag{40}$$

Equating coefficients of $T_k(x)$ on the left and right of (40) gives the general recurrence equation:

$$(1/2k)\ c_{k-1} + c_k - (1/2k)\ c_{k+1} = 0 \quad . \tag{41}$$

For this first-order differential equation we must equate coefficients of $T_k(x)$ for $k \geq 1$. Theorem 2 gives a valid interpretation to (39) for each value of the index k but we have yet to prove that (41) is valid when, for example, $k = 1$. In this example, examination of the lower limits of summation in (40) reveals that (41) is clearly valid for $k \geq 2$. The case $k = 0$ will not be required. For $k = 1$ (i.e. equating coefficients of $T_1(x)$), the middle summation in (40) has a factor 1/2 associated with the first term in its sum and the third summation will contribute two terms to the coefficient of $T_1(x)$ - namely, the terms with $k = -1$ and $k = 1$. Thus the coefficient of $T_1(x)$ comes from the terms

$$c_1T_1(x) + (1/2)(1/2)\ c_0T_1(x) - (1/2)(1/2(-1))\ c_0T_{-1}(x) - (1/2)\ c_2T_1(x)$$

$$= (c_1 + 1/4\ c_0 + 1/4\ c_0 - 1/2\ c_2)\ T_1(x).$$

The special form of the recurrence equation corresponding to $k = 1$ should therefore be

$$1/2\ c_0 + c_1 - 1/2\ c_2 = 0 \quad . \tag{42}$$

But (42) is precisely the result of setting $k = 1$ in the general recurrence equation (41).

The following two theorems prove that the left side of the general recurrence equation is valid for all $k \geq 1$, in the sense that negative subscripts are to be interpreted in absolute value. Recall that the left side of the general recurrence equation is obtained by transforming (21) into (22). By Theorem 2, the range of the index of summation in (21) may be indicated by $\sum\limits_{k=0}^{\infty}{}'$. Changing the indices of summation in the terms of (21) transforms (21) into the form

$$\sum_{k=h}^{\infty}{}' v_o(k \leftarrow k-h)\; c_{k-h}\; T_k(x) + \sum_{k=h-1}^{\infty}{}' v_1(k \leftarrow k-h+1)\; c_{k-h+1}\; T_k(x)$$
$$+ \ldots + \sum_{k=-h}^{\infty}{}' v_{2h}\; (k \leftarrow k+h)\; c_{k+h}\; T_k(x) \tag{43}$$

where the notation $v_i(k \leftarrow f(k))$ denotes, in an obvious way, an operation of substitution in the rational expression v_i. Collecting terms, (43) takes the general form (22) where the new rational expressions u_i are given by

$$u_i = v_i\; (k \leftarrow k-h+i),\; 0 \le i \le 2h. \tag{44}$$

Theorem 3 gives a symmetry property of the rational expressions v_i and then Theorem 4 uses this symmetry property to prove the validity of the general recurrence for $k \ge 1$. In the substitution operations appearing in the following theorems and proofs, the symbol " = " is used in place of the symbol " $\leftarrow$ " in order to emphasize the fact that they are arithmetic evaluations in contrast to the change of indices occurring in (43) and (44).

THEOREM 3:

The rational expressions v_i $(0 \le i \le 2h)$ appearing in (21) satisfy the following symmetry property:

$$v_i(k=\ell) = v_{2h-i}\; (k = -\ell),\; 0 \le i \le h$$

for any value of ℓ.

Proof: Expression (21) is derived by applying identities (19) and (20) to (18). Performing all applications of the product formula (19) first, it is clear that for every term in $T_{k+j}(x)$ there is a corresponding term in $T_{k-j}(x)$ and, moreover, the coefficients of corresponding terms are identical. Now when the integral formula (20) is applied to a term in $T_{k+j}(x)$:

$$\int T_{k+j}(x) = (1/2(k+j+1))\ T_{k+j+1}(x) - (1/2(k+j-1))\ T_{k+j-1}(x) \quad (45)$$

there will be a corresponding application to a term in $T_{k-j}(x)$:

$$\int T_{k-j}(x) = (1/2(k-j+1))\ T_{k-j+1}(x) - (1/2(k-j-1))\ T_{k-j-1}(x). \quad (46)$$

The symmetry expressed in the statement of the theorem is seen by comparing in (45) - (46) the coefficient of $T_{k+j+1}(x)$ when $k=\ell$ with the coefficient of $T_{k-j-1}(x)$ when $k = -\ell$:

$$1/2(\ell+j+1) = -1/2(-\ell-j-1)$$

and by comparing in (45) - (46) the coefficient of T_{k+j-1} when $k=\ell$ with the coefficient of $T_{k-j+1}(x)$ when $k = -\ell$:

$$-1/2(\ell+j-1) = 1/2(-\ell-j+1).$$

Repeated applications of the integral formula (20) will preserve this symmetry. Q.E.D.

THEOREM 4:

The expression (22), which defines the general form of the recurrence equation, is valid for values of the index $k \geq 1$ in the sense that negative subscripts are to be interpreted in absolute value.

Proof: Expression (22) is derived from (21) via (43) - (44). Examining the lower limits of the indices of summation in (43) reveals that (22) is clearly valid for $k \geq h+1$. For $k=\ell$ with $1 \leq \ell \leq h$, direct application of (22) would give the following expression as the coefficient of $T_\ell(x)$:

$$u_o(k=\ell)\ c_{-(h-\ell)} + u_1(k=\ell)\ c_{-(h-\ell)+1} + \dots + u_{2h}(k=\ell)\ c_{h+\ell}\ . \quad (47)$$

The actual coefficient of $T_\ell(x)$, from (43), will consist of the following terms. The first $h-\ell$ summations in (43) contain no term in $T_\ell(x)$. The next 2ℓ summations each contribute one term in $T_\ell(x)$. The remaining $h-\ell+1$ summations each contribute two terms (the terms $k = -\ell$ and $k=\ell$) to the coefficient of $T_\ell(x)$. The $k=\ell$ terms are:

$$1/2\ v_{h-\ell}(k=0)\ c_o + v_{h-\ell+1}(k=1)\ c_1 + \dots + v_{2h}(k=h+\ell)\ c_{h+\ell} \qquad (48)$$

which, by (44), corresponds directly with the terms in $c_o, \dots, c_{h+\ell}$ of (47) except for the factor 1/2 in the first term of (48). The $k = -\ell$ terms are :

$$1/2 v_{h+\ell}(k=0)\ c_o + v_{h+\ell+1}(k=1)\ c_1 + \dots + v_{2h}(k=h-\ell)\ c_{h-\ell}\ . \qquad (49)$$

By the symmetry property of Theorem 3, (49) is equivalent to

$$1/2\ v_{h-\ell}(k=0)\ c_o + v_{h-\ell-1}(k=-1)\ c_1 + \dots + v_o(k=-h+\ell) c_{h-\ell} . (50)$$

The first term in (50) is identical with the first term in (48) so their combination corresponds to the c_o-term given by (47). Using (44), the remaining terms in (50) take the form

$$u_{h-\ell-1}(k=\ell)\ c_1 + u_{h-\ell-2}(k=\ell)\ c_2 + \dots + u_o(k=\ell)\ c_{h-\ell}\ . \qquad (51)$$

(51) corresponds to the terms in $c_{-1}, c_{-2}, \dots, c_{-(h-\ell)}$ of (47) if the negative subscripts in (47) are interpreted in absolute value. Q.E.D.

Finally in this section, we mention the interpretation of the term k=0 in (22) which would be required in equating coefficients of $T_o(x)$. Of course for any differential equation (1) of order $\nu \geq 1$ the coefficient of $T_o(x)$ is undetermined because of the constants of integration.

However, the method discussed in this paper can be applied directly to a differential equation of order 0:

$$p_o(x)\ y(x) = r(x) \qquad (52)$$

in order to compute the Chebyshev series coefficients for an explicit rational function $r(x)/p_o(x)$. In this case the coefficients of $T_k(x)$ on the left and right must be equated for all $k \geq 0$. The coefficient of $T_o(x)$ on the left of the transformed form of (52) is not that obtained by direct application of the general expression in (22):

$$u_o(k=0)\ c_{-h} + u_1(k=0)\ c_{-h+1} + \dots + u_{2h}(k=0)\ c_h. \tag{53}$$

Rather, the correct coefficient of $T_o(x)$ comes from the last h+1 summations in (43) and it is

$$1/2\ v_h(k=0)\ c_o + v_{h+1}(k=1)\ c_1 + \dots + v_{2h}(k=h)\ c_h. \tag{54}$$

Using (44), (54) becomes

$$1/2\ u_h(k=0)\ c_o + u_{h+1}(k=0)\ c_1 + \dots + u_{2h}(k=0)\ c_h. \tag{55}$$

Comparing (55) with (53) we see that, for the special case k=0 in (22), the terms with negative subscripts must be ignored and the term in c_o must have a factor 1/2 associated with it.

5. Solution of the Recurrence Equations

We have seen that the Chebyshev series coefficients for the solution function are specified as the solution of an infinite system of linear equations. As noted in section 2, the linear system becomes finite under the assumption $c_k=0$ $(k > \mathrm{kmax})$, for some choice of kmax. The method of solution recommended by Fox and Parker [4] in a numerical context is a variant of Gaussian elimination applied to successively larger linear systems. A second method discussed in [4] uses a backward recurrence process. The latter method has been coded in ALTRAN for the purposes of this paper. A study of the relative efficiencies of the two possible solution methods in the symbolic context would be useful.

The method of backward recurrence can be applied to a general recurrence equation:

$$u_o z_k + u_1 z_{k+1} + \dots + u_n z_{k+n} = 0, \; k \geq 0, \tag{56}$$

where the coefficients $u_i (0 \leq i \leq n)$ are rational expressions in k. Associated with (56) there will be a number m of initial conditions. Under the assumption $z_k=0$ for $k > \mathrm{kmax}$ we wish to compute an m-parameter family of solutions for (56). The m initial conditions will then be used to specify values for the m parameters.

The solution we seek is a vector of the form

$$(z_o, z_1, \dots, z_{\mathrm{kmax}}, 0, 0, \dots). \tag{57}$$

Set $z_{\mathrm{kmax}}=1$ and use (56) to define the values of z_k for $k = \mathrm{kmax}-1, \dots, 0$; next set $z_{\mathrm{kmax}}=0$, $z_{\mathrm{kmax}-1}=1$ and use (56) to define the values of z_k for $k = \mathrm{kmax}-2, \dots, 0$; and so on until m solutions have been generated. If these m vectors are denoted by $v_1, \dots, v_m$ then the general m-parameter solution of the homogeneous equation (56) is given by

$$v = \sum_{i=1}^{m} \lambda_i v_i \tag{58}$$

where λ_i $(1 \le i \le m)$ are indeterminates.

In general, the equation (56) will be non-homogeneous with non-zero right-hand sides $r_o,\ldots,r_\ell$, for some $\ell \le$ kmax, corresponding to the cases $k=0,\ldots,\ell$ of equation (56). In this case, a particular solution v_p must be added to the general solution (58). The particular solution may be chosen of the form

$$v_p = (z_o, z_1, \ldots, z_\ell, 0, 0, \ldots). \tag{59}$$

It can be obtained by setting

$$z_\ell = r_\ell \; / \; u_o(k=\ell)$$

(where the denominator denotes the result of substituting the value $k=\ell$ in the leading coefficient of (56)) and then using (56) to define the values z_k for $k=\ell-1,\ldots,0$.

Having obtained the m-parameter solution $v + v_p$, it can be substituted into the m initial conditions. There results a linear system to be solved for the m parameters $\lambda_1,\ldots,\lambda_m$. The desired solution of the form (57) is then fully specified.

In the context of solving the differential equation (1), equation (56) comes from the general recurrence equation (30) for the standard cases $k \ge h$. Thus $n=2h$ in (56). The coefficients u_i in (56) correspond to the coefficients u_i in (30) after the change of index $k \leftarrow k+h$ and z_k in (56) corresponds to c_k in (30). The first h equations of our infinite linear system come from the ν associated conditions of the differential equation and from $h-\nu$ special cases of the recurrence equation (30), namely the equations resulting from equating coefficients of $T_\nu(x)$,..., $T_{h-1}(x)$. These h equations become the initial conditions of equation (56) and therefore we have

$$m = h. \tag{60}$$

It is possible for the leading coefficient in (30) to vanish for one or more values of $k \geq h$. In such a case, we choose to seek a solution of (56) of the form

$$(z_{kmin}, \ldots, z_{kmax}, 0, 0, \ldots) \tag{61}$$

rather than of the form (57). We then assign kmin additional parameters $\lambda_{m+1}, \ldots, \lambda_{m+kmin}$ directly to the solution elements $z_k (0 \leq k \leq kmin - 1)$. The value of kmin may be determined by simply evaluating the coefficient u_o in (30) for values of k in an appropriate range and setting kmin such that division by u_o will always be valid in solving (56) in the form (61). The equations corresponding to values of k in the range $h \leq k \leq h + kmin - 1$ in (30) are then treated as additional special equations. The number of "initial equations" is therefore $h + kmin$ which, by (60), corresponds to the number of parameters.

Note that the linear system to be solved for λ_i $(1 \leq i \leq m + kmin)$ in the above method is relatively small and is independent of the size of kmax. From (60) and (31) we see that

$$m \leq \nu + maxdeg$$

and kmin will usually be zero. For example, for second-order differential equations with second-degree polynomial coefficients the linear system would usually be 4 by 4. The main computational work is in the backward recurrence process to compute the solution (61) of the recurrence equation (56). It should also be noted that the linear system may happen to be singular for a particular choice of kmax in which case the solution of (56) should be re-started with a larger value of kmax.

Finally, the conditions associated with the differential equation (1) can be quite arbitrary. In general, any linear combination of function and derivative values (up to derivatives of order ν-1) at one or more points is allowed. The series solution represented by the coefficients

$$(c_o, c_1, \ldots, c_{kmax}, 0, 0, \ldots)$$

given by (57) can be evaluated at any point and can also be formally differentiated.

As an example, consider the problem:

$$(1+x^2)\ y''(x) - y'(x) + x\ y(x) = (2-x^2) \tag{62}$$

$$y(0) = 1;\ y'(0) + 2\ y(1) - 1/2\ y(-1) = 0. \tag{63}$$

Equation (62) is the same as equation (23) of the example in section 3 but the initial conditions have been changed. The "complicated" form of the second condition in (63) poses no difficulties for the method. Of course the most common "complicated" form of associated conditions in practice would be two-point boundary conditions.

The infinite linear system for (62) - (63) will be identical with (29) except for the first two equations. The first step in solving the system is to solve the general recurrence equation under the assumption c_k=0 for k > kmax. For example, when kmax = 3 the relevant equations are the 4th, 5th, and 6th in (29):

$$\begin{aligned} &1/48\ c_o + 0\ c_1 - 17/96\ c_2 + 11/8\ c_3 = 0 \\ &1/96\ c_1 + 1/24\ c_2 - 21/160\ c_3 = -\ 1/96 \\ &1/160\ c_2 + 3/40\ c_3 = 0 \quad . \end{aligned} \tag{64}$$

Since the number of initial equations is 3 we want a 3-parameter solution. The three independent solutions of the <u>homogeneous</u> form of (64) are

$$v_1 = (-168,\ 303/5,\ -12,\ 1)$$

$$v_2 = (17/2,\ -4,\ 1,\ 0)$$

$$v_3 = (0,\ 1,\ 0,\ 0).$$

For the particular solution of (64) we get

$$v_p = (0, -1, 0, 0).$$

Thus the desired 3-parameter solution is

$$\sum_{i=1}^{3} \lambda_i v_i + v_p$$

which is

$$(-168\ \lambda_1 + 17/2\ \lambda_2,\ 303/5\ \lambda_1 - 4\ \lambda_2 + \lambda_3 - 1,\ -12\ \lambda_1 + \lambda_2,\ \lambda_1). \quad (65)$$

Note that the solution (65) does not satisfy all three equations in (64) but only the first one, since v_3 was obtained without reference to the other two equations. The solution we will obtain is identical to that obtained by solving the first four equations in (29) for the four coefficients c_o, c_1, c_2, c_3 under the assumption c_k=0 for k > 3. In the general case, the m-parameter solution obtained for (56) will satisfy (56) for $0 \le k \le kmax - m$ and the values of the m parameters will be chosen so that the m initial equations are also satisfied. The solution obtained will therefore satisfy the first kmax + 1 equations in the infinite linear system. In practice, kmax should be chosen to be greater than or equal to the degree of the right-hand side polynomial so that the non-homogeneous part of the right side of the linear system will be satisfied.

Continuing with the example, we now have the parametric solution

$$y(x) = \sum_{k=0}^{3}{}' \ c_k\ T_k(x) \quad (66)$$

where the four coefficients are given by the vector (65). The parameters must be chosen by satisfying three initial equations. The first equation comes from the first condition in (63) by evaluating (66) at x=0:

$$1/2\ c_o - c_2 = 1 \quad . \quad (67)$$

To obtain the second equation from the second condition in (63), we must differentiate (66) to get:

$$y'(x) = 1/2\ (2\ c_1 + 6\ c_3)\ T_o(x) + 4\ c_2\ T_1(x) + 6\ c_3\ T_2(x). \quad (68)$$

(See [4] for the differentiation of finite Chebyshev series.)

Evaluating (68) at x=0 and evaluating (66) at x=1 and x= -1, we get for the second equation:

$$3/4\ c_o + 7/2\ c_1 + 3/2\ c_2 - 1/2\ c_3 = 0 \quad . \quad (69)$$

The third of the "initial equations" is precisely the third equation in (29) which is a special case of the general recurrence:

$$-5/24\ c_1 + 7/6\ c_2 + 3/16\ c_3 = 11/24. \quad (70)$$

Substituting the parametric solution (65) into the three linear equations (67), (69), (70) and solving the resulting 3 by 3 system yields the solution

$$(\lambda_1, \lambda_2, \lambda_3) = (-196/60353,\ 14228/60353,\ 445188/301765). \quad (71)$$

Substituting these values for the parameters into the solution (65) then gives the desired solution for the coefficients (c_o, c_1, c_2, c_3):

$$(153866/60353,\ -40105/60353,\ 16580/60353,\ -196/60353) \quad (72)$$

or to three decimal places:

$$(2.549, -.665, .275, -.003). \quad (73)$$

Computing the solution with larger values of kmax shows that the correct values for the first four Chebyshev coefficients are, to three decimal places:

$$(2.586, -.666, .282, -.001). \quad (74)$$

6. Sample Problems

In this section, the output from the ALTRAN program which implements the method of the preceeding sections is presented for several sample problems. In this output, the following indeterminates appear:

X - the indeterminate in the polynomial coefficients of the differential equation;

Y - the unknown solution function (implicitly a function of X);

DY(i) - the i-th derivative of Y;

YX(j) - used in the associated conditions to mean the value of Y at a point x_j which is specified later as XSUB(j);

DYX(i,j) - used in the associated conditions to mean the value of the i-th derivative of Y at a point x_j which is specified later as XSUB(j);

MU(j) - an unspecified indeterminate (μ_j) appearing in the problem;

K - the indeterminate appearing as the independent variable in the recurrence equation;

CK(j) - represents the term c_{k+j} in the recurrence equation where k is an indeterminate.

The output is given for the following sample problems.

Problem 1: (Simple initial-value problem for exp(x))

$y' = y;\ y(0) = 1$.

Solution: $y(x) = \exp(x)$.

Problem 2: (Simple initial-value problem for arctan(x))

$(1 + x^2)\ y' = 1;\ y(0) = 0.$

Solution: $y(x) = \arctan(x)$,

Problem 3: (Order 0 differential equation)

$(1 + x^2)\ y = 1$.

Solution: $y(x) = 1/(1 + x^2)$.

Problem 4: (Simple boundary-value problem)

$y'' + 16\, y = 0;\ y(-1) = 1;\ y(1) = 0.$

Solution: $y(x) = \cos(4x)/2\cos(4) - \sin(4x)/2\sin(4).$

Problem 5: (Second-order initial-value problem)

$(1 + x^2)\, y'' - y' + x\, y = 2 - x^2;\ y(0) = 0;\ y'(0) = 1\ .$

Solution: unknown.

Remark: example in section 3.

Problem 6: (Complicated boundary-value problem)

$(1 + x^2)\, y'' - y' + x\, y = 2 - x^2;\ y(0) = 1;$

$y'(0) + 2\, y(1) - \frac{1}{2}\, y(-1) = 0.$

Solution: unknown.

Remark: example in section 5 .

Problem 7: (Indeterminate initial conditions)

$(1 + x^2)\, y'' - y' + x\, y = 2 - x^2;\ y(0) = \mu_1;\ y'(0) = \mu_2.$

Solution: unknown.

Remark: Each Chebyshev coefficient c_k is a bilinear polynomial of the form $c_k = a_k\, \mu_1 + b_k\, \mu_2 + d_k$ for some constants $a_k,\ b_k,\ d_k$.

Problem 8: (Fourth-order initial-value problem)

$y^{(4)} - y = 0;\ y(0) = 3/2;\ y'(0) = -1/2;\ y''(0) = -3/2;$

$y'''(0) = 1/2.$

Solution: $y(x) = 3/2 \cos(x) - 1/2 \sin(x).$

Problem 9: (Fourth-order boundary-value problem)

$y^{(4)} - y = 0;\ y(0) = 0;\ y(1) = 1;$

$y''(0) = 0;\ y'''(-1) - y'(1) = 0.$

Solution: $y(x) = \sinh(x)\ /\ \sinh(1).$

Problem 10: (Problem with polynomial solution)

$(x - x^2)\, y'' + (1/2 - x)\, y' + 4y = 0;\ y(0) = 1;\ y(1) = 1.$

Solution: $y(x) = T_2(1 - 2x).$

Remark: Special case of the hypergeometric equation.

Problem 11: (Indeterminates in differential equation)

$(x - x^2)\, y'' + (\mu_3 - (\mu_1 + \mu_2 + 1)x)\, y' - \mu_1\mu_2\, y = 0;$

$y(0) = 1;\ y(1) = -1.$

Solution: This is the hypergeometric equation.

Remark: Only the recurrence equation was computed. Solving the recurrence equation in terms of μ_1, μ_2, μ_3, would require a large amount of time and space and the solution would be intractable.

Problem 12: (Eigenvalue problem)

$y'' + \mu_1\, x\, y = 0;\ y(-1) = 0;\ y(1) = 0.$

Solution: unknown.

Remark: Only the recurrence equation was computed. The problem is thus reduced to a matrix eigenvalue problem (see [4]).

Remarks on the Output

The first part of the output is an echo of the input. DIFFEQ is the differential equation represented as a polynomial in the indeterminates described above; more correctly, the differential equation is: DIFFEQ = 0. CONDN(i), $1 \le i \le \nu$, are the associated conditions represented as polynomials (again CONDN(i) = 0 is implied) and XSUB(j), j = 1,2,... are the values (as rational numbers) of the points x_j appearing in the associated conditions (see above). Note that ALTRAN re-orders the indeterminates in the polynomials according to its own rules; the actual input would use a more natural arrangement of the terms forming DIFFEQ and CONDN(i). Sample input data for Problem 6 and Problem 9 is given at the end of the program listing in section 7.

The output specifies the recurrence equation (EQN) and its "half-length" (HALFN) and gives timing information for the "setup" phase and for generating EQN. The "setup" phase here includes reading in the problem, extracting the polynomial coefficients in DIFFEQ, and computing an array containing the Chebyshev form of the powers x**k for an appropriate range of exponents k. Then the values of kmax and kmin used in computing the parametric solution (see section 5) are given and the time required to compute the parametric solution is specified. Finally, the values of the Chebyshev coefficients are printed out, as reals unless indeterminates are present, and the time required to compute the values of the parameters (and substitute these values into the parametric solution and print out the coefficients) is specified.

Some of the problems deserve a few comments. Problem 7 has two indeterminate initial conditions; the Chebyshev coefficients are therefore polynomials in these indeterminates and they are specified here as polynomials with rational coefficients. In order to see the "size" of these coefficients the rational numbers should be converted into reals; for example, the last coefficient for Problem 7 is

$$c_{10} = .45\ (10^{-5})\ \mu_1 + .20\ (10^{-5})\ \mu_2 + .24\ (10^{-6}).$$

Problem 10 has the polynomial solution

$$T_2(1 - 2x) = 10\ T_o(x)/2 - 8\ T_1(x) + 4\ T_2(x)$$

and this solution is obtained exactly.

Problem 11 contains three indeterminates in the hypergeometric differential equation; the general recurrence equation is computed without difficulty but one would want to assign values to these indeterminates prior to the solution phase since the indeterminates enter into the coefficients in a very complicated manner (in contrast to Problem 7). Problem 12 is an eigenvalue problem; the recurrence equation computed could be used to set up a matrix eigenvalue problem whose solution would approximately solve the given eigenvalue problem (cf. [4]).

Accuracy of the Chebyshev Coefficients

For each problem where the true solution is known, Table 1 gives the relative error in each of the Chebyshev coefficients computed by our program. Of course, for greater accuracy the value of kmax (see section 5) should be increased. All of these problems were solved using the assumption $c_k = 0$ for $k > 10$ (i.e., kmax = 10). In this table, numbers in brackets refer to powers of 10.

			Problem				
k	#1	#2	#3	#4	#8	#9	#10
0	.4(-8)	0	.1(-8)	.2(-4)	.5(-10)	0	0
1	.5(-9)	.3(-7)	0	.9(-4)	.4(-8)	.2(-6)	0
2	.5(-8)	0	.2(-7)	.2(-4)	.2(-9)	0	0
3	.2(-8)	.8(-6)	0	.9(-4)	.3(-7)	.4(-5)	0
4	.2(-8)	0	.8(-6)	.2(-4)	.7(-11)	0	0
5	.2(-8)	.3(-4)	0	.9(-4)	.4(-9)	.5(-6)	0
6	.2(-8)	0	.3(-4)	.2(-4)	.1(-9)	0	0
7	.9(-10)	.9(-3)	0	.8(-4)	.4(-7)	.4(-5)	0
8	.2(-7)	0	.9(-3)	.2(-4)	.7(-10)	0	0
9	.6(-5)	.3(-1)	0	.2(-2)	.7(-7)	.6(-6)	0
10	.2(-2)	0	.3(-1)	.1(-2)	.4(-7)	0	0

Table 1: Relative Errors in Chebyshev Coefficients c_k.

Summary of Timing Statistics

The timing statistics in Table 2 were obtained on a Honeywell 66/60. All of the problems were run with an execution time core specification of 55K and no attempt was made to determine the minimum core requirement. The times measured were:

T_1 - time to generate the general recurrence equation (via procedure RECURR);

T_2 - time to compute the parametric solution (via procedure SOLVER);

T_3 - time to assign values to the parameters (via procedure VALUES);

TOTAL - total time elapsed .

All times are in seconds.

Problem #	T_1	T_2	T_3	TOTAL
1	4	6	4	15
2	7	15	9	33
3	3	10	6	19
4	17	26	10	55
5	160	46	31	240
6	166	45	35	249
7	156	43	30	231
8	255	86	28	373
9	266	95	30	393
10	44	39	31	117
11	143	-	-	-
12	30	-	-	-

Table 2: Timing Statistics (in seconds) .

<u>Output for Problem #1</u>:

```
 # DIFFEQ

       - ( Y - DY(1) )

 # CONDN(1)

       YX(1) - 1

 # XSUB(1)

       0

 # SETUP TIME IN SECONDS WAS

 # TNEW

       1.241688

 # HALFN

       1

 # EQN

       ( 2*K*CK(0) - CK(-1) + CK(1) ) / ( 2*K )

 # TIME IN SECONDS TO GENERATE EQN WAS

 # TNEW

       3.865141

 # KMAX

       10

 # KMIN

       0

 # TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

 # TNEW

       5.739844

 # I

       0

 # CIREAL

       2.5321317442553344
```

```
# I
      1
# CIREAL
      1.1303182073880147
# I
      2
# CIREAL
      2.7149533830866716D-1
# I
      3
# CIREAL
      4.4336849934398059D-2
# I
      4
# CIREAL
      5.4742404285122567D-3
# I

      5
# CIREAL
      5.4292631056695242D-4
# I
      6
# CIREAL
      4.4977322842732531D-5
# I
      7
# CIREAL
      3.1984364621045916D-6
```

```
# I

      8

# CIREAL

      1.9921248446385429D-7

# I

      9

# CIREAL

      1.1036702740379739D-8

# I

      10

# CIREAL

      5.5183513701898694D-10

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      4.241438

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      1.516305E1
```

Output for Problem #2:

```
# DIFFEQ

      X**2*DY(1) + DY(1) - 1

# CONDN(1)

      YX(1)

# XSUB(1)

      0

# SETUP TIME IN SECONDS WAS

# TNEW

      1.374453

# HALFN

      2

# EQN

      ( K*CK(-2) + 6*K*CK(0) + K*CK(2) - 2*CK(-2) + 2*CK(2) ) / ( 4*K )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      7.225719

# KMAX

      10

# KMIN

      1

# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      1.475083E1
```

```
# I
        0
# CIREAL
        0.
# I
        1
# CIREAL
        8.2842710329210939D-1
# I
        2
# CIREAL
        0.
# I
        3
# CIREAL
        -4.7378505486848981D-2
# I
        4
# CIREAL
        0.
# I
        5
# CIREAL
        4.8771990942344539D-3
# I
        6
# CIREAL
        0.
```

```
# I

      7

# CIREAL

      -5.9720805235523926D-4

# I

      8

# CIREAL

      0.

# I

      9

# CIREAL

      7.7415858638642126D-5

# I

      10

# CIREAL

      0.

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      9.438375

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      3.286706E1
```

Output for Problem #3:

```
# DIFFEQ

      X**2*Y + Y - 1

# SETUP TIME IN SECONDS WAS

# TNEW
      9.559844E-1

# HALFN

      2

# EQN

      ( CK(-2) + 6*CK(0) + CK(2) ) / 4

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      2.602844

# KMAX

      10

# KMIN

      0

# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      9.685938

# I

      0

# CIREAL

      1.4142135605326259

# I

      1

# CIREAL

      0.
```

```
# I

      2

# CIREAL

      -2.4264068159787766D-1

# I

      3

# CIREAL

      0.

# I

      4

# CIREAL

      4.1630529054640069D-2

# I

      5

# CIREAL

      0.

# I

      6

# CIREAL


      -7.1424927299962757D-3

# I

      7

# CIREAL

      0.

# I

      8

# CIREAL

      1.2244273251364726D-3
```

```
# I

     9

# CIREAL

     0.

# I

     10

# CIREAL

     -2.0407122085607877D-4

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

     6.086547

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

     1.940552E1
```

Output for Problem #4:

```
# DIFFEQ

      16*Y + DY(2)

# CONDN(1)

      YX(1) - 1

# CONDN(2)

      YX(2)

# XSUB(1)

      -1

# XSUB(2)

      1

# SETUP TIME IN SECONDS WAS

# TNEW

      1.686328

# HALFN

      2

# EQN

      ( K**3*CK(0) + 4*K*CK(-2) - 9*K*CK(0) + 4*K*CK(2) + 4*CK(-2) -

      4*CK(2) ) / ( ( K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      1.732269E1

# KMAX

      10

# KMIN

      0
```

```
# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

        2.55397E1

# I

        0

# CIREAL

        6.0758284851235024D-1

# I

        1

# CIREAL

        -8.727407774201535D-2

# I

        2

# CIREAL

        5.5706438919328949D-1

# I

        3

# CIREAL

        -5.6845753899480069D-1

# I

        4

# CIREAL

        -4.3008757255382698D-1

# I

        5

# CIREAL

        1.745481554840307D-1
```

```
# I
      6
# CIREAL
      7.5097020636780473D-2
# I
      7
# CIREAL
      -2.0054468927952463D-2
# I
      8
# CIREAL
      -6.1632655825640539D-3
# I
      9
# CIREAL
      1.2379301807378064D-3
# I
      10
# CIREAL
      2.9800405014595426D-4
# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS
# TNEW
      1.037458E1
# TOTAL ELAPSED TIME IN SECONDS WAS
# TNEW
      5.500364E1
```

<u>Output for Problem #5</u>:

```
# DIFFEQ

      X**2*DY(2) + X**2 + X*Y - DY(1) + DY(2) - 2

# CONDN(1)

      YX(1)

# CONDN(2)

      DYX(1,1) - 1

# XSUB(1)

      0

# SETUP TIME IN SECONDS WAS

# TNEW

      2.624063

# HALFN

      3

# EQN

      ( 2*K**3*CK(-2) + 12*K**3*CK(0) + 2*K**3*CK(2) - 8*K**2*CK(-2) -

      4*K**2*CK(-1) + 4*K**2*CK(1) + 8*K**2*CK(2) + K*CK(-3) + 2*K*CK(-2) -

      K*CK(-1) - 20*K*CK(0) - K*CK(1) + 2*K*CK(2) + K*CK(3) + CK(-3) +

      12*CK(-2) + 5*CK(-1) - 5*CK(1) - 12*CK(2) - CK(3) ) /

      ( ( 8*K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      1.603859E2

# KMAX

      10

# KMIN

      0
```

```
# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      4.590784E1

# I

      0

# CIREAL

      1.3103210883040351

# I

      1

# CIREAL

      1.2518273563200095

# I

      2

# CIREAL

      6.2578023488704703D-1

# I

      3

# CIREAL

      6.7580968605735406D-2

# I

      4

# CIREAL

      -2.8128516370876885D-2

# I

      5

# CIREAL

      -8.50766923580579D-3
```

```
# I

      6

# CIREAL

      1.193339899046909D-3

# I

      7

# CIREAL

      8.263359368835608 3D-4

# I

      8

# CIREAL

      -5.625523059188 4712D-5

# I

      9

# CIREAL

      -8.463920647624 3919D-5

# I

      10

# CIREAL

      2.198124588957858 2D-6

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      3.073631E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      2.397361E2
```

Output for Problem #6:

```
# DIFFEQ

      X**2*DY(2) + X**2 + X*Y - DY(1) + DY(2) - 2

# CONDN(1)

      YX(1) - 1

# CONDN(2)

      ( 4*YX(2) - YX(3) + 2*DYX(1,1) ) / 2

# XSUB(1)

      0

# XSUB(2)

      1

# XSUB(3)

      -1

# SETUP TIME IN SECONDS WAS

# TNEW

      2.876594

# HALFN

      3

# EQN

      ( 2*K**3*CK(-2) + 12*K**3*CK(0) + 2*K**3*CK(2) - 8*K**2*CK(-2) -

      4*K**2*CK(-1) + 4*K**2*CK(1) + 8*K**2*CK(2) + K*CK(-3) + 2*K*CK(-2) -

      K*CK(-1) - 20*K*CK(0) - K*CK(1) + 2*K*CK(2) + K*CK(3) + CK(-3) +

      12*CK(-2) + 5*CK(-1) - 5*CK(1) - 12*CK(2) - CK(3) ) /

      ( ( 8*K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      1.663187E2

# KMAX

      10
```

```
# KMIN

      0

# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      4.471497E1

# I

      0

# CIREAL

      2.5864002340529057

# I

      1

# CIREAL

      -6.6599579752459272D-1

# I

      2

# CIREAL

      2.8178328741588448D-1

# I

      3

# CIREAL

      -8.3193206117771278D-4

# I

      4

# CIREAL

      -1.0757249622261317D-2

# I

      5

# CIREAL

      -2.0446971804471966D-3
```

```
# I

        6

# CIREAL

        6.137753799358776D-4

# I

        7

# CIREAL

        2.254026764266O495D-4

# I

        8

# CIREAL

        -4.242795046069561 3D-5

# I

        9

# CIREAL

        -2.456105680304342 1D-5

# I

        10

# CIREAL

        3.376659126541022D-6

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

        3.523827E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

        2.492322E2
```

Output for Problem #7:

```
# DIFFEQ

      X**2*DY(2) + X**2 + X*Y - DY(1) + DY(2) - 2

# CONDN(1)

      - ( MU(1) - YX(1) )

# CONDN(2)

      - ( MU(2) - DYX(1,1) )

# XSUB(1)

      0

# SETUP TIME IN SECONDS WAS

# TNEW

      2.750297

# HALFN

      3

# EQN

      ( 2*K**3*CK(-2) + 12*K**3*CK(0) + 2*K**3*CK(2) - 8*K**2*CK(-2) -

      4*K**2*CK(-1) + 4*K**2*CK(1) + 8*K**2*CK(2) + K*CK(-3) + 2*K*CK(-2) -

      K*CK(-1) - 20*K*CK(0) - K*CK(1) + 2*K*CK(2) + K*CK(3) + CK(-3) +

      12*CK(-2) + 5*CK(-1) - 5*CK(1) - 12*CK(2) - CK(3) ) /

      ( ( 8*K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      1.555058E2

# KMAX

      10

# KMIN

      0
```

```
# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      4.306044E1

# C(0)

      ( 21_0547435509_8623733708*MU(1) + 4_4702227539_4233242729*MU(2) +

      9_4585562410_3555401191 ) / 10_6300502241_1190143842

# C(1)

      - ( 5571748110_3512052608*MU(1) - 5_7511040545_4997541555*MU(2) -

      9023897802_5241121150 ) / 5_3150251120_5595071921

# C(2)

      - ( 2575602633_3634042432*MU(1) - 4_2111341444_3754169617*MU(2) -

      9_0930165149_7915697151 ) / 21_2601004482_2380287684

# C(3)

      - 4 * ( 1278187474_2516703376*MU(1) - 864765940_1664884758*MU(2) -

      1829193150_6517017541 ) / 15_9450753361_6785215763

# C(4)

      - 4 * ( 137624071_9711816020*MU(1) + 770943497_4538133479*MU(2) +

      1097853643_8457309569 ) / 26_5751255602_7975359605

# C(5)

      4 * ( 6931313_1394551760*MU(1) - 12947844_1445108738*MU(2) -

      24734218_6907272553 ) / 1_7716750373_5198357307

# C(6)

      ( 120619415_3776349040*MU(1) + 194553902_6717647067*MU(2) +

      228288199_6995532117 ) / 35_4335007470_3967146140

# C(7)

      - ( 1865053_2159914032*MU(1) - 5241070_8413904289*MU(2) -

      9398916_6993553341 ) / 1_7716750373_5198357307
```

```
# C(8)

      - ( 4157803_8208257444*MU(1) + 3358743_9829835467*MU(2) +

      2621215_2844239333 ) / 10_6300502241_1190143842

# C(9)

      ( 1317769_1471312944*MU(1) - 4930842_6448138751*MU(2) -

      8564942_5957359215 ) / 15_9450753361_6785215763

# C(10)

      ( 949373_8117352880*MU(1) + 417073_6266786805*MU(2) +

      50249_8690515483 ) / 21_2601004482_2380287684

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      2.971733E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      2.311168E2
```

<u>Output for Problem #8</u>:

```
# DIFFEQ

      - ( Y - DY(4) )

# CONDN(1)

      ( 2*YX(1) - 3 ) / 2

# CONDN(2)

      ( 2*DYX(1,1) + 1 ) / 2

# CONDN(3)

      ( 2*DYX(2,1) + 3 ) / 2

# CONDN(4)

      ( 2*DYX(3,1) - 1 ) / 2

# XSUB(1)

      0

# SETUP TIME IN SECONDS WAS

# TNEW

      3.020562

# HALFN

      4

# EQN

      ( 16*K**7*CK(0) - 224*K**5*CK(0) - K**3*CK(-4) + 4*K**3*CK(-2) +

      778*K**3*CK(0) + 4*K**3*CK(2) - K**3*CK(4) - 6*K**2*CK(-4) +

      12*K**2*CK(-2) - 12*K**2*CK(2) + 6*K**2*CK(4) - 11*K*CK(-4) -

      16*K*CK(-2) - 522*K*CK(0) - 16*K*CK(2) - 11*K*CK(4) - 6*CK(-4) -

      48*CK(-2) + 48*CK(2) + 6*CK(4) ) /

      ( ( 16*K ) * ( K - 3 ) * ( K - 2 ) * ( K - 1 ) * ( K + 1 ) * ( K + 2

      ( K + 3 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      2.551718E2
```

```
# KMAX

      10

# KMIN

      0

# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      8.6468E1

# I

      0

# CIREAL

      2.2955930595685642

# I

      1

# CIREAL

      -4.4005058389317173D-1

# I

      2

# CIREAL

      -3.4471045484991001D-1

# I

      3

# CIREAL

      1.9563354643780871D-2

# I

      4

# CIREAL

      7.4299168922810961D-3
```

```
# I

      5

# CIREAL

      -2.4975773010151088D-4

# I

      6

# CIREAL

      -6.2815014016567472D-5

# I

      7

# CIREAL

      1.5023258717627605D-6

# I

      8

# CIREAL

      2.8267032515938945D-7

# I

      9

# CIREAL

      -5.249249830289044D-9

# I

      10

# CIREAL

      -7.8918450757498436D-10

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      2.812091E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      3.728641E2
```

Output for Problem #9:

```
# DIFFEQ

      - ( Y - DY(4) )

# CONDN(1)

      YX(1)

# CONDN(2)

      YX(2) - 1

# CONDN(3)

      DYX(2,1)

# CONDN(4)

      - ( DYX(1,2) - DYX(3,3) )

# XSUB(1)

      0

# XSUB(2)

      1

# XSUB(3)

      -1

# SETUP TIME IN SECONDS WAS

# TNEW

      2.81425

# HALFN

      4

# EQN

      ( 16*K**7*CK(0) - 224*K**5*CK(0) - K**3*CK(-4) + 4*K**3*CK(-2) +

      778*K**3*CK(0) + 4*K**3*CK(2) - K**3*CK(4) - 6*K**2*CK(-4) +

      12*K**2*CK(-2) - 12*K**2*CK(2) + 6*K**2*CK(4) - 11*K*CK(-4) -

      16*K*CK(-2) - 522*K*CK(0) - 16*K*CK(2) - 11*K*CK(4) - 6*CK(-4) -

      48*CK(-2) + 48*CK(2) + 6*CK(4) ) /
      ( ( 16*K ) * ( K - 3 ) * ( K - 2 ) * ( K - 1 ) * ( K + 1 ) * ( K + 2 )
      ( K + 3 ) )
# TIME IN SECONDS TO GENERATE EQN WAS
# TNEW
      2.659498E2
# KMAX
      10
# KMIN
      0
# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS
# TNEW
      9.462813E1
# I
      0
# CIREAL
      0.
# I
      1
# CIREAL
      9.6180809569416168D-1
# I
      2
# CIREAL
      0.
# I
      3
# CIREAL
      3.7727187701446803D-2
```

```
# I

      4

# CIREAL

      0.

# I

      5

# CIREAL

      4.619855935148573 6D-4

# I

      6

# CIREAL

      0.

# I

      7

# CIREAL

      2.721619404076467 2D-6

# I

      8

# CIREAL

      0.

# I

      9

# CIREAL

      9.391383951184423 9D-9

# I

      10

# CIREAL

      0.
```

```
# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      2.973214E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      3.932103E2
```

Output for Problem #10:

```
# DIFFEQ

      - ( 2*X**2*DY(2) + 2*X*DY(1) - 2*X*DY(2) - 8*Y - DY(1) ) / 2

# CONDN(1)

      YX(1) - 1

# CONDN(2)

      YX(2) - 1

# XSUB(1)

      0

# XSUB(2)

      1

# SETUP TIME IN SECONDS WAS

# TNEW

      2.434891

# HALFN

      2

# EQN

      - ( K**3*CK(-2) - 2*K**3*CK(-1) + 2*K**3*CK(0) - 2*K**3*CK(1) +

      K**3*CK(2) - 3*K**2*CK(-2) + 3*K**2*CK(-1) - 3*K**2*CK(1) +

      3*K**2*CK(2) - 4*K*CK(-2) + 2*K*CK(-1) + 4*K*CK(0) + 2*K*CK(1) -

      4*K*CK(2) - 3*CK(-1) + 3*CK(1) ) / ( ( 4*K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      4.37198E1

# KMAX

      10

# KMIN

      3
```

```
# TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS

# TNEW

      3.927061E1

# I

      0

# CIREAL

      1.D1

# I

      1

# CIREAL

      -8.

# I

      2

# CIREAL

      4.

# I

      3

# CIREAL

      0.

# I

      4

# CIREAL

      0.

# I

      5

# CIREAL

      0.
```

```
# I

      6

# CIREAL

      0.

# I

      7

# CIREAL

      0.

# I

      8

# CIREAL

      0.

# I

      9

# CIREAL

      0.

# I

      10

# CIREAL

      0.

# TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS

# TNEW

      3.129861E1

# TOTAL ELAPSED TIME IN SECONDS WAS

# TNEW

      1.168089E2
```

Output for Problem #11:

```
# DIFFEQ

      - ( X**2*DY(2) + X*DY(1)*MU(1) + X*DY(1)*MU(2) + X*DY(1) - X*DY(2) +

      Y*MU(1)*MU(2) - DY(1)*MU(3) )

# CONDN(1)

      YX(1) - 1

# CONDN(2)

      YX(2) + 1

# XSUB(1)

      0

# XSUB(2)

      1

# SETUP TIME IN SECONDS WAS

# TNEW

      2.60425

# HALFN

      2

# EQN

      - ( K**3*CK(-2) - 2*K**3*CK(-1) + 2*K**3*CK(0) - 2*K**3*CK(1) +

      K**3*CK(2) + K**2*CK(-2)*MU(1) + K**2*CK(-2)*MU(2) - 3*K**2*CK(-2) -

      2*K**2*CK(-1)*MU(3) + 4*K**2*CK(-1) + 2*K**2*CK(1)*MU(3) -

      4*K**2*CK(1) - K**2*CK(2)*MU(1) - K**2*CK(2)*MU(2) + 3*K**2*CK(2) +

      K*CK(-2)*MU(1)*MU(2) - K*CK(-2)*MU(1) - K*CK(-2)*MU(2) + 2*K*CK(-1) -

      2*K*CK(0)*MU(1)*MU(2) + 2*K*CK(0)*MU(1) + 2*K*CK(0)*MU(2) - 4*K*CK(0) +

      2*K*CK(1) + K*CK(2)*MU(1)*MU(2) - K*CK(2)*MU(1) - K*CK(2)*MU(2) +

      CK(-2)*MU(1)*MU(2) - 2*CK(-2)*MU(1) - 2*CK(-2)*MU(2) + 4*CK(-2) +

      2*CK(-1)*MU(3) - 4*CK(-1) - 2*CK(1)*MU(3) + 4*CK(1) - CK(2)*MU(1)*

      MU(2) + 2*CK(2)*MU(1) + 2*CK(2)*MU(2) - 4*CK(2) ) /
      ( ( 4*K ) * ( K - 1 ) * ( K + 1 ) )
# TIME IN SECONDS TO GENERATE EQN WAS
# TNEW
      1.425576E2
```

<u>Output for Problem #12</u>:

```
# DIFFEQ

      X*Y*MU(1) + DY(2)

# CONDN(1)

      YX(1)

# CONDN(2)

      YX(2)

# XSUB(1)

      -1

# XSUB(2)

      1

# SETUP TIME IN SECONDS WAS

# TNEW

      1.660141

# HALFN

      3

# EQN

      ( 8*K**3*CK(0) + K*CK(-3)*MU(1) - K*CK(-1)*MU(1) - 8*K*CK(0) -

      K*CK(1)*MU(1) + K*CK(3)*MU(1) + CK(-3)*MU(1) + CK(-1)*MU(1) -

      CK(1)*MU(1) - CK(3)*MU(1) ) / ( ( 8*K ) * ( K - 1 ) * ( K + 1 ) )

# TIME IN SECONDS TO GENERATE EQN WAS

# TNEW

      3.022622E1
```

7. Source Listing of ALTRAN Procedures

The ALTRAN implementation of the method described in this report is given in this section. There are ten ALTRAN procedures and then the last page contains sample input for Problems 6 and 9 of the preceeding section.

Index of Procedures:

```
PROCEDURE MAIN

     # MAIN PROCEDURE FOR THE CHEBYSHEV SERIES SOLUTION OF A
     #    LINEAR ODE.
     #
     # THE INPUT IS, IN THE FOLLOWING SEQUENCE:
     #    KMAX - DEGREE OF TRUNCATED CHEBYSHEV SERIES SOLUTION
     #       DESIRED;
     #    ORDER - THE ORDER OF THE DIFFERENTIAL EQUATION;
     #    DIFFEQ - THE DIFFERENTIAL EQUATION AS A MULTINOMIAL IN
     #       THE INDETERMINATES:  X, Y, DY(1), ... , DY(ORDER)
     #       WHERE X IS THE VARIABLE IN THE POLYNOMIAL COEFFI-
     #       CIENTS AND DY(I) REPRESENTS THE I-TH DERIVATIVE OF
     #       THE UNKNOWN SOLUTION FUNCTION Y;
     #    CONDN(1), ..., CONDN(ORDER) - THE ASSOCIATED CONDITIONS
     #       WRITTEN AS MULTINOMIALS IN THE INDETERMINATES
     #          YX(J) AND DYX(I,J), 1 <= J <= 5,
     #                                        1 <= I <= ORDER-1
     #       WHERE YX(J) REPRESENTS THE VALUE OF Y AT A POINT
     #       X(J) AND DYX(I,J) REPRESENTS THE VALUE OF THE I-TH
     #       DERIVATIVE OF Y AT A POINT X(J);
     #    XSUB(1), ..., XSUB(MAXSUB) - RATIONAL NUMBERS SPECIFY-
     #       ING THE VALUES OF THE POINTS X(J) MENTIONED ABOVE,
     #       WHERE MAXSUB IS THE MAXIMUM SUBSCRIPT J USED ABOVE
     #       (I.E. THE NUMBER OF DIFFERENT POINTS APPEARING IN
     #       THE ASSOCIATED CONDITIONS).
     #
     # THE OUTPUT IS AN ECHO OF THE INPUT FOLLOWED BY THE VALUES
     #    OF HALFN AND EQN WHERE EQN IS THE (LEFT-HAND SIDE OF
     #    THE) GENERAL RECURRENCE EQUATION AND HALFN IS ITS
     #    "HALF-LENGTH". THEN FOLLOWS SOME INFORMATION ABOUT THE
     #    CHOICE OF KMIN AND POSSIBLY UPDATED VALUES OF KMAX WITH
     #    RESPECT TO THE SOLUTION OF THE GENERAL RECURRENCE EQUA-
     #    TION.  FINALLY, THE COMPUTED VALUES OF THE CHEBYSHEV
     #    COEFFICIENTS ARE PRINTED OUT, AS LONG REALS IF POSSIBLE
     #    (I.E. IF THE GIVEN PROBLEM CONTAINED NO INDETERMINATES)
     #    AND OTHERWISE IN SYMBOLIC FORM.


     INTEGER  I, J, M, MAXDEG, DEGRHS, MAXSUB, HALFN, N, DEGREE, KMIN
     INTEGER  KMAX = SIREAD()
     INTEGER  ORDER = SIREAD()
     RATIONAL ARRAY (1:5)  XSUB
     REAL  TOLD, TNEW
     LONG REAL  CIREAL
     ALGEBRAIC ARRAY (0:ORDER)  POLY
     ALGEBRAIC  RPOLY, LHS, EQN, NEWEQN, RHS, RHCOEF, LCOEF, LCVAL
```

```
    ALGEBRAIC (K:20, TK(-10:10):1, CK(-10:10):1, T(0:10):1,
       PAR(1:10):1, X:8, Y:1, DY(1:IMAX(ORDER,1)):1, MU(1:5):30,
       YX(1:5):1, DYX(1:IMAX(ORDER-1,1), 1:5):1 )  DIFFEQ
           # NOTE: THE ABOVE LAYOUT SHOULD BE SUFFICIENT IF
           #        'MAX. DEG. IN X' + ORDER <= 10 .
    ALGEBRAIC ARRAY (1:IMAX(ORDER,1))  CONDN
    LONG ALGEBRAIC ARRAY  C, VAL
    ALGEBRAIC ARRAY  XLIST, RLIST, INDET
    EXTERNAL ALGEBRAIC  XK=K
    EXTERNAL ALGEBRAIC ARRAY  XTK=TK, XCK=CK, XT=T, XPAR=PAR
    EXTERNAL ALGEBRAIC ARRAY  XYX=YX, XDYX=DYX
    EXTERNAL ALGEBRAIC ARRAY  XPOWER
    LONG RATIONAL ALTRAN  LQT
    ALGEBRAIC ALTRAN  CHFORM, PINTN
    ALGEBRAIC ARRAY ALTRAN  POWERS
    LONG ALGEBRAIC ARRAY ALTRAN  SOLVER, VALUES



    # READ IN THE DIFFERENTIAL EQUATION.

    READ  DIFFEQ
    WRITE  DIFFEQ


    # READ IN THE ASSOCIATED CONDITIONS.

    CONDN(1) = DIFFEQ   # JUST DEFINES THE LAYOUT FOR  CONDN .
    DO M = 1, ORDER
       READ  CONDN(M)
       WRITE  CONDN(M)
    DOEND

    DO J = 5, 1, -1
       DO M = 1, ORDER
          IF ( DEG(CONDN(M), YX(J)) > 0 )  GO TO DEF
          DO I = 1, ORDER-1
             IF ( DEG(CONDN(M), DYX(I,J)) > 0 )  GO TO DEF
          DOEND
       DOEND
    DOEND
DEF: MAXSUB = J

    DO J = 1, MAXSUB
       READ  XSUB(J)
       WRITE  XSUB(J)
    DOEND
```

```
# PICK OFF THE POLYNOMIAL COEFFICIENTS AND DETERMINE MAXIMUM
#    DEGREE OF POLYNOMIAL TO BE CONVERTED TO CHEBYSHEV FORM.

POLY(0) = GETBLK(DIFFEQ, Y, 1)
MAXDEG = DEG(POLY(0), X)
LHS = POLY(0) * Y
DO I = 1, ORDER
   POLY(I) = GETBLK(DIFFEQ, DY(I), 1)
   MAXDEG = IMAX(MAXDEG, DEG(POLY(I), X))
   LHS = LHS  +  POLY(I) * DY(I)
DOEND
RPOLY = LHS - DIFFEQ
DEGRHS = DEG(RPOLY, X)
IF (RPOLY <> 0)  DEGRHS = DEGRHS + ORDER    # NOTE: RPOLY IS
                                  # INTEGRATED BEFORE CONVERSION.
MAXDEG = IMAX(MAXDEG, DEGRHS)


# PROCEDURE 'POWERS' COMPUTES THE CHEBYSHEV FORM OF POWERS
#    OF X.

XPOWER = POWERS(MAXDEG)


# PRINT OUT TIMING INFORMATION.

TNEW = TIME(TOLD)
WRITE  "SETUP TIME IN SECONDS WAS", TNEW


# PROCEDURE 'RECURR' COMPUTES THE GENERAL RECURRENCE EQUA-
#    TION FOR THE O.D.E., GIVEN ITS L.H.S. POLYNOMIALS.

RECURR(ORDER, POLY, X, EQN, HALFN)
WRITE  HALFN, EQN


# COMPUTE THE R.H.S OF THE SYSTEM BY INTEGRATING RPOLY AN
#    APPROPRIATE NUMBER OF TIMES (ORDER TIMES) AND THEN
#    CONVERTING TO CHEBYSHEV FORM.

XLIST = ORDER$X  # I.E. XLIST = (X,X,...,X) OF LENGTH ORDER.
RHS = CHFORM( PINTN(RPOLY,XLIST), DEGRHS, X)


# PRINT OUT TIMING INFORMATION.

TNEW = TIME(TOLD)
WRITE  "TIME IN SECONDS TO GENERATE EQN WAS", TNEW
```

```
    # THE FOLLOWING CODE SOLVES THE RECURRENCE EQUATION AND THE
    #    ASSOCIATED CONDITIONS FOR THE DESIRED CHEBYSHEV COEF-
    #    FICIENTS.


    # ENSURE THAT KMAX IS LARGE ENOUGH, AND DETERMINE THE VALUE
    #    KMIN SUCH THAT THE COEFFICIENTS C(KMIN), ..., C(KMAX)
    #    CAN BE SOLVED FOR EXPLICITLY IN THE RECURRENCE EQUATION.

    WRITE  KMAX
    IF (KMAX < DEGRHS)  DO
       KMAX  =  DEGRHS
       WRITE  "DUE TO DEGREE OF R.H.S., KMAX INCREASED TO", KMAX
    DOEND  # END OF IF-STATEMENT.

    LCOEF = GETBLK(EQN, CK(-HALFN), 1)
REDEF: KMIN = 0
    DO J = 0, KMAX
       LCVAL  =  LCOEF(K = J+HALFN)
       IF (LCVAL == 0)  KMIN = J+1
    DOEND
    WRITE  KMIN

    IF (KMAX < KMIN+HALFN)  DO
       KMAX  =  KMIN  +  HALFN
       WRITE  "DUE TO VALUE OF  KMIN + HALFN,  KMAX INCREASED TO"
       WRITE  KMAX
       GO TO REDEF  # REDEFINE KMIN.
    DOEND  # END OF IF-STATEMENT.


    # DETERMINE THE LIST OF RIGHT-HAND-SIDES INVOLVED IN SOLVING
    #    THE RECURRENCE EQUATION FOR C(KMIN), ..., C(KMAX) .

    RLIST = 0$(1)   # I.E. RLIST IS THE EMPTY LIST, INITIALLY.

    DO DEGREE = KMIN+HALFN, DEGRHS
       RHCOEF = GETBLK(RHS, T(DEGREE), 1)
       RLIST = (RLIST, RHCOEF)
    DOEND
```

```
     # PROCEDURE 'SOLVER' SOLVES THE RECURRENCE EQUATION FOR THE
     #    COEFFICIENTS C(KMIN) TO C(KMAX) IN TERMS OF THE PARA-
     #    METERS PAR(1), ..., PAR(HALFN).  THE RECURRENCE EQUA-
     #    TION MUST BE IN STANDARD FORM WITH THE INDETERMINATES
     #    INDEXED FROM 0 TO N.

     N = 2*HALFN
     NEWEQN  =  EQN(K = K+HALFN)

     INDET = 1$( CK(-HALFN) )
     DO I = 1, N
        INDET = (INDET, CK(-HALFN+I))
     DOEND

SOLV: C = SOLVER(NEWEQN, N, 1$INDET, K, RLIST, KMIN, KMAX, PAR, HALFN)


     # INTRODUCE FURTHER PARAMETERS FOR UNDEFINED COEFFICIENTS.

     DO I = 0, KMIN-1
        C(I)  =  PAR(HALFN + I + 1)
     DOEND


     # PRINT OUT TIMING INFORMATION.

     TNEW = TIME(TOLD)
     WRITE  "TIME IN SECONDS TO COMPUTE PARAMETRIC SOLUTION WAS"
     WRITE  TNEW


     # DETERMINE THE VALUES OF THE PARAMETERS WHICH SATISFY THE
     #    ASSOCIATED CONDITIONS AND THE "SPECIAL" FORMS OF THE
     #    RECURRENCE EQUATION, IF ANY.

     VAL = VALUES(CONDN, ORDER, XSUB, MAXSUB, EQN, HALFN, HALFN+KMIN,
                                          RHS, C, KMAX, REPEAT)


     # SUBSTITUTE THE VALUES FOR THE PARAMETERS IN ARRAY C.

     DO I = 0, KMAX
        DO J = 1, HALFN+KMIN
           C(I)  =  C(I)( PAR(J) = VAL(J) )
        DOEND
     DOEND
```

```
       # WRITE OUT THE VALUES OF C, AS REALS IF POSSIBLE.

       DO I = 0, KMAX
          C(I) = LQT(C(I), ASIS)  # "LONG RATIONAL TEST".
          CIREAL = LR( C(I) )     # "LONG REAL CONVERSION".
          WRITE  I, CIREAL
          GO TO CONT
  ASIS: WRITE  C(I)    # IF C(I) NOT RATIONAL, WRITE OUT AS IS.
 CONT: DOEND


       # PRINT OUT FINAL TIMING INFORMATION.

       TNEW = TIME(TOLD)
       WRITE  "TIME IN SECONDS TO COMPUTE VALUES FOR THE PARAMETERS WAS"
       WRITE  TNEW
       TNEW = TIME()
       WRITE  "TOTAL ELAPSED TIME IN SECONDS WAS", TNEW



       GO TO STOP


       # IF LINEAR SYSTEM WAS SINGULAR IN PROCEDURE 'VALUES',
       #    REPEAT WITH KMAX INCREMENTED BY ONE.

 REPEAT: KMAX = KMAX + 1
       WRITE  "PARAMETER EQUATIONS SINGULAR -- KMAX INCREASED TO"
       WRITE  KMAX
       IF (KMAX <= 20)  GO TO SOLV
          ELSE  WRITE "TERMINATION DUE TO SIZE OF KMAX"


 STOP: END  # END OF PROCEDURE MAIN.
```

```
PROCEDURE POWERS(MAXDEG)
  INTEGER VALUE  MAXDEG

  # PROCEDURE TO COMPUTE THE CHEBYSHEV FORM OF THE POWERS X**K
  #    FOR 0 <= K <= MAXDEG.  THE K-TH CHEBYSHEV POLYNOMIAL IS
  #    REPRESENTED BY THE INDETERMINATE XT(K), WHERE XT IS AN
  #    EXTERNAL ALGEBRAIC ARRAY OF INDETERMINATES.  THE VALUE
  #    RETURNED IS AN ARRAY DIMENSIONED 0 TO MAXDEG CONTAINING
  #    THE APPROPRIATE CHEBYSHEV FORMS.  THE FOLLOWING DECLAR-
  #    ATION MUST APPEAR IN THE CALLING PROCEDURE:
  #        ALGEBRAIC ARRAY ALTRAN POWERS .


  INTEGER  K, HALFK, KMOD2, J
  INTEGER ARRAY  KCHOOS, TEMP
  ALGEBRAIC ARRAY (0:MAXDEG)  A
  EXTERNAL ALGEBRAIC ARRAY  XT


  A(0) = XT(0)
  IF (MAXDEG > 0)  A(1) = XT(1)
  IF (MAXDEG > 1)  A(2) = (XT(2)+XT(0))/2
  KCHOOS = 1$(1)

  DO K = 3, MAXDEG

    KMOD2 = IMOD(K, 2, HALFK)

    # UPDATE ARRAY KCHOOS SO THAT KCHOOS(J) = "K CHOOSE J",
    #    FOR J = 1, ..., K/2 .

    IF (KMOD2 == 1)  TEMP = KCHOOS
      ELSE  TEMP = (KCHOOS, KCHOOS(HALFK-1))
    KCHOOS = HALFK$(0)  # CREATES ARRAY OF DESIRED LENGTH.
    KCHOOS(1) = K
    DO J = 2, HALFK
      KCHOOS(J) = TEMP(J) + TEMP(J-1)
    DOEND


    # COMPUTE CHEBYSHEV FORM OF X**K INTO ARRAY ELEMENT A(K).

    A(K) = XT(K)
    DO J = 1, HALFK-1
      A(K) = A(K)  +  KCHOOS(J) * XT(K - 2*J)
    DOEND
    IF (KMOD2==1) A(K) = (A(K) + KCHOOS(HALFK)*XT(1))/2**(K-1)
      ELSE  A(K) = (A(K) + 1/2*KCHOOS(HALFK)*XT(0))/2**(K-1)
  DOEND

  RETURN( A )

END  # END OF PROCEDURE POWERS.
```

```
PROCEDURE RECURR(ORDER, POLY, X, EQN, HALFN)
   INTEGER VALUE  ORDER
   ALGEBRAIC ARRAY VALUE  POLY
   ALGEBRAIC VALUE  X
   ALGEBRAIC  EQN
   INTEGER  HALFN

   # PROCEDURE TO COMPUTE THE GENERAL RECURRENCE EQUATION FOR A
   #    LINEAR ODE WITH POLYNOMIAL COEFFICIENTS.
   #
   # INPUT PARAMETERS:
   #    ORDER - THE ORDER OF THE DIFFERENTIAL EQUATION;
   #    POLY - ARRAY OF LEFT-HAND SIDE POLYNOMIALS SUCH THAT
   #       POLY(ORDER) IS THE COEFFICIENT OF THE HIGHEST-ORDER
   #       DERIVATIVE, ETC.;
   #    X - THE NAME OF THE INDETERMINATE IN THE POLYNOMIALS.
   #
   # OUTPUT PARAMETERS:
   #    EQN - THE LEFT SIDE OF THE GENERAL RECURRENCE EQUATION;
   #    HALFN - THE "HALF-LENGTH" OF THE GENERAL RECURRENCE
   #       EQUATION.
   #
   # EXTERNAL VARIABLES:
   #    XK - THE NAME OF THE INDETERMINATE WHICH WILL APPEAR IN
   #       THE COEFFICIENTS OF THE GENERAL RECURRENCE EQUATION;
   #    XTK - THE ARRAY OF INDETERMINATES SUCH THAT XTK(J) IS
   #       USED TO REPRESENT THE CHEBYSHEV POLYNOMIAL OF DEGREE
   #       K+J, WHERE K IS AN INDETERMINATE;
   #    XCK - THE ARRAY OF INDETERMINATES SUCH THAT XCK(J) WILL
   #       APPEAR IN THE GENERAL RECURRENCE EQUATION REPRESENT-
   #       ING C(K+J) WHERE K IS AN INDETERMINATE.
   #
   # PROCEDURES REQUIRED:
   #    CHFORM; PRODTK; INTEGR; DIFFN(A SYSTEM PROCEDURE).


   INTEGER  I, J, M, DEGREE, HALF, SIGN
   INTEGER ARRAY ( 1:IMAX(ORDER,1), 1:IMAX(ORDER,1) )  COMB
   ALGEBRAIC  P, FACTOR, TERM, COEF
   ALGEBRAIC ARRAY  XLIST
   EXTERNAL ALGEBRAIC  XK
   EXTERNAL ALGEBRAIC ARRAY  XTK, XCK
   ALGEBRAIC ALTRAN  CHFORM, PRODTK, INTEGR, DIFFN
```

```
# SET UP BINOMIAL COEFFICIENTS IN ORDER-BY-ORDER ARRAY COMB.

DO I = 1, ORDER
   COMB(I,1) = I
   DO J = 2, I-1
      COMB(I,J) = COMB(I-1,J-1) + COMB(I-1,J)
   DOEND
   COMB(I,I) = 1
DOEND


# THE FOLLOWING CODE CONVERTS THE L.H.S. OF THE O.D.E. TO
#    INTEGRATED FORM:
#       P0*Y + INTEGRAL(P1*Y) + INTEGRAL(INTEGRAL(P2*Y)) +
#         . . . + INTEGRAL(INTEGRAL(...(INTEGRAL(PN*Y))...)
#    WHERE N=ORDER.  EACH PASS THROUGH THE M-LOOP DETERMINES
#    THE APPROPRIATE POLYNOMIAL PM AND CONVERTS IT TO CHEBY-
#    SHEV FORM.
# THEN THE SUBSTITUTION
#       Y = 1/2 * C(0)*T(0) + SUM( C(K)*T(K) )
#    (WHERE THE SUM IS OVER K = 1, 2, 3, ... ) MAKES THE
#    L.H.S. OF THE O.D.E. AN INFINITE SUM WITH GENERAL TERM
#                C(K) * FACTOR ,
#    WHERE
#       FACTOR = P0*T(K) + INTEGRAL(P1*T(K)) + . . .
#         + INTEGRAL(INTEGRAL(...(INTEGRAL( PN*T(K) ))...) .
#    EACH PASS THROUGH THE LOOP ADDS ONE TERM INTO FACTOR .


DEGREE = DEG( POLY(ORDER), X)
P = CHFORM( POLY(ORDER), DEGREE, X)
FACTOR = PRODTK(P, DEGREE)
HALFN = DEGREE

DO M = 1, ORDER

   # DETERMINE THE M-TH POLYNOMIAL IN THE INTEGRATED FORM.

   P = POLY(ORDER-M) - (ORDER-M+1)*DIFF( POLY(ORDER-M+1), X)
   SIGN = -1
   XLIST = 1$X   # I.E. XLIST = (X) INITIALLY.
   DO I = M-2, 0, -1
      SIGN = - SIGN
      XLIST = (XLIST, X)
      P  =  P  +  SIGN * COMB(ORDER-I, M-I) *
                            DIFFN( POLY(ORDER-I), XLIST)
   DOEND
```

```
      # CONVERT THE POLYNOMIAL TO CHEBYSHEV FORM.

      DEGREE = DEG(P, X)
      P = CHFORM(P, DEGREE, X)


      # ADD THE APPROPRIATE TERM INTO FACTOR .

      TERM = PRODTK(P, DEGREE)
      HALF = DEGREE
      DO I = 1, M
         TERM = INTEGR(TERM, HALF)
         HALF = HALF + 1
      DOEND

      FACTOR = FACTOR + TERM
      HALFN = IMAX(HALFN, HALF)

   DOEND



   # IN EACH TERM OF THE EXPRESSION
   #              C(K) * FACTOR     (SEE COMMENT ABOVE)
   #    DO A CHANGE OF INDEX SO THAT THE COEFFICIENT OF T(K) IS
   #    OBTAINED; THIS YIELDS THE GENERAL RECURRENCE EQUATION.

   EQN = 0
   DO J = -HALFN, HALFN
      COEF = GETBLK(FACTOR, XTK(J), 1)
      COEF = COEF(XK=XK-J)
      EQN = EQN + COEF*XCK(-J)
   DOEND


END  # END OF PROCEDURE RECURR.
```

```
PROCEDURE CHFORM(P, DEGREE, X)
   INTEGER VALUE  DEGREE
   ALGEBRAIC VALUE  P, X

   # PROCEDURE TO COMPUTE THE CHEBYSHEV FORM OF A POLYNOMIAL P
   #    OF GIVEN DEGREE IN THE INDETERMINATE X.  IT IS ASSUMED
   #    THAT THE EXTERNAL ARRAY XPOWER HAS BEEN INITIALIZED TO
   #    CONTAIN THE CHEBYSHEV FORM OF THE POWER X**K AS ITS
   #    ELEMENT XPOWER(K), FOR 0 <= K <= DEGREE.  THE FOLLOWING
   #    DECLARATION MUST APPEAR IN THE CALLING PROCEDURE:
   #          ALGEBRAIC ALTRAN CHFORM .


   INTEGER  K
   ALGEBRAIC  COEF, NEWP
   EXTERNAL ALGEBRAIC ARRAY  XPOWER


   NEWP = 0
   DO K = 0, DEGREE
      COEF = GETBLK(P, X, K)
      NEWP = NEWP + COEF*XPOWER(K)
   DOEND

   RETURN(NEWP)

END  # END OF PROCEDURE CHFORM.
```

```
PROCEDURE PRODTK(P, DEGREE)
   INTEGER VALUE  DEGREE
   ALGEBRAIC VALUE  P

   # PROCEDURE TO MULTIPLY A POLYNOMIAL P OF GIVEN DEGREE, AS-
   #    SUMED TO BE IN CHEBYSHEV FORM, BY THE CHEBYSHEV POLYNO-
   #    MIAL OF DEGREE K, WHERE K IS AN INDETERMINATE.  IN P,
   #    THE CHEBYSHEV POLYNOMIAL OF DEGREE J IS REPRESENTED BY
   #    XT(J) WHERE XT IS AN EXTERNAL ARRAY.  THE REPRESENTA-
   #    TION OF THE CHEBYSHEV POLYNOMIAL OF DEGREE K+J IN THE
   #    RETURNED POLYNOMIAL, WHERE K IS AN INDETERMINATE, IS
   #    XTK(J) WHERE XTK IS AN EXTERNAL ARRAY.  THE FOLLOWING
   #    DECLARATION MUST APPEAR IN THE CALLING PROCEDURE:
   #          ALGEBRAIC ALTRAN PRODTK .


   INTEGER  J
   ALGEBRAIC  COEF, NEWP
   EXTERNAL ALGEBRAIC ARRAY  XTK, XT


   NEWP = 0
   DO J = 0, DEGREE
      COEF = GETBLK(P, XT(J), 1)
      NEWP = NEWP  +  COEF * (XTK(J)+XTK(-J))/2
   DOEND

   RETURN(NEWP)

END  # END OF PROCEDURE PRODTK .
```

```
PROCEDURE INTEGR(P, HALFN)
   INTEGER VALUE  HALFN
   ALGEBRAIC VALUE  P

   # PROCEDURE TO INTEGRATE A POLYNOMIAL P WHICH IS ASSUMED TO
   #    BE IN CHEBYSHEV FORM, MORE EXPLICITLY IN THE FORM
   #          V(0) * XTK(-H) + . . . + V(2*H) * XTK(H)
   #    WHERE H = HALFN, XTK IS AN EXTERNAL ARRAY SUCH THAT
   #    XTK(J) REPRESENTS THE CHEBYSHEV POLYNOMIAL OF DEGREE
   #    K+J WITH K AN INDETERMINATE, AND V(0), ..., V(2*H) ARE
   #    RATIONAL EXPRESSIONS IN THE INDETERMINATE SPECIFIED BY
   #    THE EXTERNAL VARIABLE XK.  THE FOLLOWING DECLARATION
   #    MUST APPEAR IN THE CALLING PROCEDURE:
   #          ALGEBRAIC ALTRAN INTEGR .


   INTEGER  J
   ALGEBRAIC  NEWP
   EXTERNAL ALGEBRAIC  XK
   EXTERNAL ALGEBRAIC ARRAY  XTK


   NEWP = 0
   DO J = -HALFN, HALFN
      NEWP = NEWP + GETBLK(P, XTK(J), 1) * ( XTK(J+1)/(XK+J+1) -
                                         XTK(J-1)/(XK+J-1) )/2
   DOEND

   RETURN(NEWP)

END  # END OF PROCEDURE INTEGR .
```

```
PROCEDURE SOLVER(EQN, N, ZK, K, RLIST, KMIN, KMAX, PAR, NPAR)
   INTEGER VALUE  N, KMIN, KMAX, NPAR
   ALGEBRAIC VALUE  EQN, K
   ALGEBRAIC ARRAY VALUE  RLIST, PAR
   ALGEBRAIC ARRAY (0:N) VALUE  ZK

   # PROCEDURE TO SOLVE A RECURRENCE EQUATION FOR AN NPAR-
   #    PARAMETER SOLUTION.
   #
   # INPUT PARAMETERS:
   #    EQN - THE LEFT SIDE OF THE RECURRENCE EQUATION;
   #    N - INDICATES THAT EQN CONTAINS N+1 TERMS;
   #    ZK - ARRAY OF THE N+1 INDETERMINATES APPEARING IN EQN
   #       REPRESENTING THE DEPENDENT VARIABLE Z -- I.E. EQN IS
   #       OF THE FORM
   #         U0 * ZK(0) + U1 * ZK(1) + . . . + UN * ZK(N)
   #       WHERE THE UI ARE RATIONAL EXPRESSIONS IN THE INDE-
   #       TERMINATE K AND ZK(I) REPRESENTS Z(K+I);
   #    K - THE NAME OF THE INDETERMINATE IN EQN;
   #    RLIST - LIST OF RIGHT-HAND SIDES OF THE RECURRENCE EQU-
   #       ATION CORRESPONDING TO THE CASES K = KMIN, ..., KMAX
   #       OF EQN(THE RIGHT-HAND SIDE IS ZERO BEYOND THE NUMBER
   #       OF ELEMENTS IN RLIST);
   #    KMIN, KMAX - INDICATE THE RANGE OF SUBSCRIPTS FOR WHICH
   #       THE SOLUTION IS TO BE COMPUTED -- I.E. THE SOLUTION
   #       Z(KMIN), ..., Z(KMAX) IS DESIRED;
   #    PAR - NAMES OF THE PARAMETERS TO BE USED;
   #    NPAR - NUMBER OF PARAMETERS DESIRED IN THE SOLUTION.
   #
   # OUTPUT:
   #     THE VALUE RETURNED IS AN ARRAY DIMENSIONED FROM 0 TO
   # KMAX CONTAINING THE DESIRED NPAR-PARAMETER SOLUTION.  IF
   # KMIN > 0 THEN THE FIRST KMIN ELEMENTS OF THE ARRAY ARE
   # ARBITRARILY SET TO ZERO.
   #
   # ASSUMPTIONS:
   #    IT IS ASSUMED THAT KMIN >= 0, KMAX >= KMIN+NPAR, AND
   # THAT KMAX >= KMIN+LENGTH-1 WHERE LENGTH IS THE LENGTH OF
   # THE ARRAY RLIST.
   #
   #    THE FOLLOWING DECLARATION MUST APPEAR IN THE CALLING
   # PROCEDURE:
   #          ALGEBRAIC ARRAY ALTRAN SOLVER .
```

```
INTEGER  I, J, NZEROS, KVAL, KLAST, LENGTH, L
ALGEBRAIC  U0, ZK0
ALGEBRAIC ARRAY  UNLIST
LONG ALGEBRAIC  ZKVAL
LONG ALGEBRAIC ARRAY  GENSOL, SOLN, VALIST
LONG ALGEBRAIC ARRAY (0:KMAX)  Z


# SOLVE RECURRENCE FOR Z(K).

U0 = GETBLK(EQN, ZK(0), 1)
ZK0  =  ZK(0)  -  EQN / U0


# 'UNLIST' IS THE LIST OF UNKNOWNS IN ZK0.

UNLIST = 1$( ZK(1) )
DO L = 2, N
   UNLIST = (UNLIST, ZK(L))
DOEND


# COMPUTE NPAR INDEPENDENT SOLUTIONS AND FORM THEIR LINEAR
#    COMBINATION INTO GENSOL.

GENSOL = 0
DO I = 1, NPAR

   SOLN = 1$(1)
   NZEROS = N

   DO KVAL = KMAX-I, KMIN, -1

      NZEROS = NZEROS - 1
      IF (NZEROS >= 0)  VALIST = (SOLN, NZEROS$0)
         ELSE DO
               VALIST = 1$( SOLN(1) )
               DO J = 2, N
                  VALIST = (VALIST, SOLN(J))
               DOEND
              DOEND  # END OF ELSE-CLAUSE.

      ZKVAL  =  ZK0(K = KVAL)(UNLIST = VALIST)
      SOLN = (ZKVAL, SOLN)

   DOEND

   GENSOL  =  GENSOL  +  PAR(I) * (SOLN, (I-1)$0)

DOEND
```

```
    # COMPUTE A PARTICULAR SOLUTION AND ADD IT INTO GENSOL.

    LENGTH = DBINFO(RLIST)(0,1)

    IF (LENGTH > 0)  DO

       KLAST  =  KMIN + LENGTH - 1
       SOLN   =  1$( RLIST(LENGTH) / U0(K=KLAST) )
       NZEROS = N;  L = LENGTH

       DO KVAL = KLAST-1, KMIN, -1

          NZEROS = NZEROS - 1
          IF (NZEROS >= 0)  VALIST = (SOLN, NZEROS$0)
             ELSE DO
                   VALIST = 1$( SOLN(1) )
                   DO J = 2, N
                      VALIST = (VALIST, SOLN(J))
                   DOEND
                  DOEND  # END OF ELSE-CLAUSE.

          ZKVAL  =  ZK0(K = KVAL)(UNLIST = VALIST)
          L  =  L - 1
          ZKVAL  =  ZKVAL  +  RLIST(L) / U0(K=KVAL)
          SOLN = (ZKVAL, SOLN)

       DOEND

       GENSOL  =  GENSOL  +  (SOLN, (KMAX-KLAST)$0)

    DOEND  # END OF IF-STATEMENT.


    # RETURN 'GENSOL' IN POSITIONS KMIN TO KMAX OF ARRAY Z,
    #    WHICH IS DIMENSIONED FROM 0 TO KMAX.

    Z = 1$(KMIN$0, GENSOL)

    RETURN( Z )

END  # END OF PROCEDURE SOLVER .
```

```
PROCEDURE VALUES(CONDN, ORDER, XSUB, MAXSUB, EQN, HALFN, NEQNS,
                                        RHS, C, KMAX, LAB)
  INTEGER VALUE  ORDER, MAXSUB, HALFN, NEQNS, KMAX
  RATIONAL ARRAY VALUE  XSUB
  ALGEBRAIC VALUE  EQN, RHS
  ALGEBRAIC ARRAY VALUE  CONDN
  LONG ALGEBRAIC ARRAY VALUE  C
  LABEL  LAB

  # PROCEDURE TO DETERMINE VALUES FOR THE PARAMETERS APPEARING
  #   IN THE GENERAL SOLUTION OF THE RECURRENCE EQUATION BY
  #   INVOKING THE INITIAL EQUATIONS, WHICH ARE DETERMINED BY
  #   THE ASSOCIATED CONDITIONS OF THE DIFFERENTIAL EQUATION
  #   AND ALSO BY THE SPECIAL CASES OF THE GENERAL RECURRENCE
  #   EQUATION.
  #
  # INPUT PARAMETERS:
  #   CONDN - AN ARRAY OF THE ASSOCIATED CONDITIONS OF THE
  #      DIFFERENTIAL EQUATION;
  #   ORDER - THE ORDER OF THE DIFFERENTIAL EQUATION;
  #   XSUB - AN ARRAY CONTAINING THE POINTS OF EVALUATION
  #      USED IN THE INITIAL CONDITIONS;
  #   MAXSUB - THE LENGTH OF ARRAY XSUB;
  #   EQN - THE GENERAL RECURRENCE EQUATION;
  #   HALFN - THE "HALF-LENGTH" OF EQN;
  #   NEQNS - THE NUMBER OF INITIAL EQUATIONS TO BE USED,
  #      WHICH MUST CORRESPOND TO THE NUMBER OF PARAMETERS IN
  #      THE GENERAL SOLUTION (THE FIRST ORDER EQUATIONS COME
  #      FROM THE ASSOCIATED CONDITIONS OF THE DIFFERENTIAL
  #      EQUATION AND THE REST ARE SPECIAL CASES OF THE GENE-
  #      RAL RECURRENCE EQUATION);
  #   RHS - THE RIGHT-HAND SIDE OF THE INTEGRATED FORM OF THE
  #      DIFFERENTIAL EQUATION, IN CHEBYSHEV FORM;
  #   C - AN ARRAY DIMENSIONED FROM 0 TO KMAX CONTAINING
  #      THE PARAMETRIC SOLUTION OF THE GENERAL RECURRENCE
  #      EQUATION;
  #   KMAX - THE DEGREE OF THE SOLUTION BEING OBTAINED;
  #   LAB - LABEL VARIABLE TO WHICH THE PROCEDURE WILL RETURN
  #      IF THE LINEAR EQUATIONS DEFINING THE PARAMETERS ARE
  #      SINGULAR.
  #
  # OUTPUT:
  #   THE VALUE RETURNED IS AN ARRAY DIMENSIONED FROM 1 TO
  # NEQNS CONTAINING THE COMPUTED VALUES OF THE PARAMETERS.
  #
```

```
# EXTERNAL VARIABLES:
#    XK - THE NAME OF THE INDETERMINATE WHICH APPEARS IN THE
#       GENERAL RECURRENCE EQUATION AS THE INDEPENDENT VARI-
#       ABLE;
#    XCK - THE ARRAY OF INDETERMINATES SUCH THAT XCK(J) REP-
#       RESENTS THE TERM C(K+J) IN THE GENERAL RECURRENCE
#       EQUATION;
#    XT - THE ARRAY OF INDETERMINATES SUCH THAT XT(J) REPRE-
#       SENTS THE CHEBYSHEV POLYNOMIAL OF DEGREE J IN THE
#       POLYNOMIAL RHS;
#    XPAR - THE NAMES OF THE PARAMETERS APPEARING IN THE
#       GENERAL SOLUTION C;
#    XYX - THE ARRAY OF INDETERMINATES APPEARING IN THE AS-
#       SOCIATED CONDITIONS (I.E. IN THE ARRAY CONDN) REPRE-
#       SENTING THE Y-VALUES AT VARIOUS POINTS;
#    XDYX - THE ARRAY OF INDETERMINATES APPEARING IN THE AS-
#       SOCIATED CONDITIONS (I.E. IN THE ARRAY CONDN) REPRE-
#       SENTING THE DERIVATIVES OF Y AT VARIOUS POINTS.
#
# PROCEDURES REQUIRED:
#    EVAL; DERIV; ASOLVE AND LIT (TWO SYSTEM PROCEDURES).
#
# ASSUMPTIONS:
#    IT IS ASSUMED THAT KMAX >= HALFN AND KMAX >= ORDER-1
# (THE FORMER SUBSUMES THE LATTER SINCE HALFN >= ORDER).
#
#    THE FOLLOWING DECLARATION MUST APPEAR IN THE CALLING
# PROCEDURE:
#          ALGEBRAIC ARRAY ALTRAN VALUES .


INTEGER  I, J, M, L, DEGREE, NDIFF, KVALUE
ALGEBRAIC  RHCOEF
LONG ALGEBRAIC  LHS, NUM, DEN, COEF
LONG ALGEBRAIC ARRAY (1:NEQNS)  ROW, B, SOLN
LONG ALGEBRAIC ARRAY (1:NEQNS, 1:NEQNS)  A
ALGEBRAIC ARRAY  UNLIST
LONG ALGEBRAIC ARRAY  VALIST, CDIFF
EXTERNAL ALGEBRAIC  XK
EXTERNAL ALGEBRAIC ARRAY  XYX, XDYX, XCK, XT, XPAR
LONG INTEGER ALTRAN  LIT
LONG ALGEBRAIC ALTRAN  EVAL
LONG ALGEBRAIC ARRAY ALTRAN  DERIV, ASOLVE
```

```
     # BUILD UP LIST OF SUBSTITUTIONS FOR ALL Y-VALUES APPEARING
     #    IN THE INITIAL CONDITIONS.

     DO J = 1, MAXSUB
        DO M = 1, ORDER
           IF ( DEG(CONDN(M), XYX(J)) > 0 )  DO
              UNLIST = ( UNLIST, XYX(J) )
              VALIST = ( VALIST, EVAL(C, KMAX, XSUB(J)) )
              GO TO OUT
           DOEND  # END OF IF-STATEMENT.
        DOEND
OUT: DOEND

     # BUILD UP LIST OF SUBSTITUTIONS FOR ALL DERIVATIVE VALUES
     #    APPEARING IN THE INITIAL CONDITIONS.

     CDIFF = C;  DEGREE = KMAX

     DO I = 1, ORDER-1
        DO J = 1, MAXSUB
           DO M = 1, ORDER

              IF ( DEG(CONDN(M), XDYX(I,J)) > 0 )  DO

                 # "CDIFF" SHOULD REPRESENT THE I-TH DERIVATIVE
                 # OF Y. IF DEGREE = KMAX-I THEN IT DOES; OTHER-
                 # WISE NDIFF DIFFERENTIATIONS OF THE SERIES ARE
                 # REQUIRED.

                 NDIFF = DEGREE - (KMAX-I)
                 DO L = 1, NDIFF
                    CDIFF = DERIV(CDIFF, DEGREE)
                    DEGREE = DEGREE - 1
                 DOEND

                 UNLIST = ( UNLIST, XDYX(I,J) )
                 VALIST  = ( VALIST, EVAL(CDIFF, DEGREE, XSUB(J)) )
                 GO TO ESCAPE

              DOEND  # END OF IF-STATEMENT.

           DOEND
ESCAPE: DOEND
     DOEND
```

```
# THE FIRST "ORDER" ROWS OF THE LINEAR SYSTEM ARE DETERMINED
#    BY THE INITIAL CONDITIONS.

DO M = 1, ORDER
   ROW(M)  =  CONDN(M)(UNLIST = VALIST)
DOEND


# THE REMAINING ROWS OF THE LINEAR SYSTEM ARE DETERMINED BY
#    THE SPECIAL CASES OF THE RECURRENCE EQUATION FOR VALUES
#    OF K (IF ANY) IN THE RANGE ORDER <= K <= NEQNS-1 .

# FIRST, SET UP LIST OF UNKNOWNS APPEARING IN EQN .

UNLIST = 0$UNLIST   # I.E. MAKE UNLIST THE EMPTY LIST.
DO J = -HALFN, HALFN
   UNLIST = ( UNLIST, XCK(J) )
DOEND

# FOR EACH ROW, SET UP LIST OF SUBSTITUTIONS AND SUBSTITUTE.

DO KVALUE = ORDER, NEQNS-1

   VALIST = 0$VALIST  # I.E. MAKE VALIST THE EMPTY LIST.

   IF (KVALUE == 0)  VALIST = (HALFN$0, C(0)/2)
      ELSE  DO J = HALFN, KVALUE, -1
               VALIST = ( VALIST, C(J-KVALUE) )
            DOEND

   DO J = -IMIN(KVALUE-1,HALFN), HALFN
      IF  (KVALUE+J <= KMAX)  VALIST = ( VALIST, C(KVALUE+J) )
         ELSE  VALIST = (VALIST, 0)
   DOEND

   LHS   =  EQN(XK = KVALUE)(UNLIST = VALIST)
   RHCOEF = GETBLK(RHS, XT(KVALUE), 1)
   ROW(KVALUE+1)  =  LHS - RHCOEF

DOEND


# REMOVE THE DENOMINATOR FROM EACH ROW, CHECKING FIRST THAT
#    IT IS A CONSTANT, SINCE THE LINEAR SYSTEM IS
#          ROW(I) = 0, I = 1, ..., NEQNS .
# THIS WILL MAKE THE LINEAR SYSTEM SOLVER MORE EFFICIENT.

DO I = 1, NEQNS
   DEN = ADEN(ROW(I), NUM)
   DEN = LIT(DEN, OMIT)   # "LONG INTEGER TEST"
   ROW(I) = NUM    # ASSIGN THE NUMERATOR TO ROW(I).
OMIT:DOEND   # IF DEN NOT AN INTEGER, ROW(I) IS LEFT UNCHANGED.
```

```
        # ASSIGN COEFFICIENT MATRIX OF LINEAR SYSTEM TO "A" AND
        #    RIGHT-HAND SIDE TO "B".

        DO I = 1, NEQNS

           LHS = 0
           DO J = 1, NEQNS
              COEF = GETBLK(ROW(I), XPAR(J), 1)
              A(I,J) = COEF
              LHS = LHS + COEF*XPAR(J)
           DOEND

           B(I) = LHS - ROW(I)

        DOEND


        # SOLVE THE LINEAR SYSTEM.

        SOLN = ASOLVE(A, B, SING)
        RETURN( SOLN )

        # IF LINEAR SYSTEM WAS SINGULAR, RETURN TO THE LABEL "LAB".

SING: RETURN  LAB


    END  # END OF PROCEDURE VALUES.
```

```
PROCEDURE EVAL(C, DEGREE, XVALUE)
   LONG ALGEBRAIC ARRAY VALUE  C
   INTEGER VALUE  DEGREE
   RATIONAL VALUE  XVALUE

   # PROCEDURE TO EVALUATE A TRUNCATED CHEBYSHEV SERIES C OF
   #    GIVEN DEGREE AT THE GIVEN POINT XVALUE.  THE FOLLOWING
   #    DECLARATION MUST APPEAR IN THE CALLING PROCEDURE:
   #          ALGEBRAIC ALTRAN EVAL .


   INTEGER  I
   RATIONAL  TWOX
   LONG ALGEBRAIC  BOLD, TEMP, B


   TWOX = 2*XVALUE
   BOLD = 0;  TEMP = 0
   B = C(DEGREE)

   DO I = DEGREE-1, 0, -1
      BOLD = TEMP
      TEMP = B
      B = TWOX*B - BOLD + C(I)
   DOEND

   RETURN( (B-BOLD)/2 )

END  # END OF PROCEDURE EVAL .
```

```
PROCEDURE DERIV(C, DEGREE)
   LONG ALGEBRAIC ARRAY VALUE  C
   INTEGER VALUE   DEGREE

   # PROCEDURE TO DIFFERENTIATE A TRUNCATED CHEBYSHEV SERIES C
   #    OF GIVEN DEGREE.  THE FOLLOWING DECLARATION MUST APPEAR
   #    IN THE CALLING PROCEDURE:
   #          ALGEBRAIC ARRAY ALTRAN DERIV .


   INTEGER  I
   LONG ALGEBRAIC ARRAY (0:DEGREE-1)  CNEW


   CNEW(DEGREE-1)  =  2 * DEGREE * C(DEGREE)
   CNEW(DEGREE-2)  =  2 * (DEGREE-1) * C(DEGREE-1)

   DO I = DEGREE-2, 1, -1
      CNEW(I-1)  =  2*I*C(I) + CNEW(I+1)
   DOEND


   RETURN( CNEW )

END  # END OF PROCEDURE DERIV.
```

Sample Input for Problem 6:

```
10
2
(1 + X**2) * DY(2)  -  DY(1)  +  X * Y  -  (2 - X**2)
YX(1)  -  1
DYX(1,1)  +  2 * YX(2)  -  1/2 * YX(3)
0
1
-1
```

Sample Input for Problem 9:

```
10
4
DY(4)  -  Y
YX(1)
YX(2)  -  1
DYX(2,1)
DYX(3,3)  -  DYX(1,2)
0
1
-1
```

References

[1] C.W. Clenshaw, The numerical solution of linear differential equations in Chebyshev series, Proc. Camb. Phil. Soc., 53 (1957), pp. 134-149.

[2] P.M. Dew and R.E. Scraton, Chebyshev methods for the numerical solution of parabolic partial differential equations in two and three space variables, J. Inst. Maths. Applics., 16 (1975), pp. 121-131.

[3] L. Fox, Chebyshev methods for ordinary differential equations, Comput. J., 4 (1962), pp. 318-331.

[4] L. Fox and I.B. Parker, Chebyshev Polynomials in Numerical Analysis, Oxford Univ. Press, London, 1968.

[5] W.B. Gragg and G.D. Johnson, The Laurent-Padé table, Proc. IFIP Congress '74, pp. 632-637.

[6] D. Knibb, The numerical solution of parabolic partial differential equations using the method of Lanczos, J. Inst. Maths. Applics., 11 (1973), pp. 181-190.

[7] C. Lanczos, Applied Analysis, Prentice Hall, New York, 1957.

[8] S. Lewanowicz, Solution of a first-order nonlinear differential equation in Chebyshev series, Zastosowania Matematyki, 15 (1976), pp. 251-268.

[9] H.J. Norton, The iterative solution of nonlinear ordinary differential equations in Chebyshev series, Comput. J., 7 (1964), pp. 76-85.

[10] M.J.D. Powell, On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria, Comp. J., 9 (1967), pp. 404-407.