MATHEMATICS OF COMPUTATION, VOLUME 26, NUMBER 120, OCTOBER 1972

Note on Backward Recurrence Algorithms

By F. W. J. Olver and D. J. Sookne

Abstract. An algorithm is given for the computation of the recessive solution of a second-order linear difference equation, based upon a combination of algorithms due to J.C.P. Miller and F.W.J. Olver. A special feature is automatic and rigorous control of truncation error.

The method is illustrated by application to the well-used example of the Bessel functions $J_r(x)$.

1. Introduction and Summary.

Let

$$a_r y_{r-1} - b_r y_r + c_r y_{r+1} = 0 \qquad (r = 1, 2, \cdots) \tag{1}$$

be a given difference equation in which the coefficients a_r and c_r do not vanish. Suppose that the equation has a pair of solutions f_r and g_r such that $f_r/g_r \to 0$ as $r \to \infty$. Then f_r is said to be a *recessive* (or *subdominant* or *distinguished*) solution of the difference equation at $r = \infty$, and g_r is said to be *dominant*. The recessive solution is unique, apart from a constant factor. The dominant solution is not unique, however, since any constant multiple of f_r may be added to g_r without affecting the asymptotic form of g_r.

Computation of f_r from (1) by forward recurrence is usually impractical owing to strong instability. On the other hand, backward application of (1) provides a stable way of computing f_r (but not g_r), since rounding errors grow no faster than the wanted solution, as a rule.* In the next section, we describe briefly two published algorithms which enable f_r to be computed without the need for accurate starting values at high values of r.

In Section 3, certain difficulties in the implementation of the algorithms are described, and in the next section, it is shown how these difficulties can be overcome by combining the algorithms.

In Section 5, the well-used Bessel function example is considered. A computing routine is described in which the truncation error is bounded rigorously, without loss of efficiency. The method is compared with methods of earlier writers.

The concluding section, Section 6, gives proofs of certain results used in earlier sections.

2. The Two Algorithms.

The first algorithm, which we shall refer to as *Algorithm* I, is due to J. C. P. Miller [2] and has been used extensively in the computation

Received August 23, 1971.

AMS 1970 *subject classifications.* Primary 65Q05; Secondary 15A06, 33A40, 39A10, 40A25, 65D20.

Key words and phrases. Bessel functions, difference equations, error bounds, FORTRAN, Miller algorithm, recursion.

* Full discussions of this aspect and related matters are given by Gautschi in [1].

of special functions and in other contexts. It proceeds as follows. For a suitably chosen large integer N, a "trial" solution $y_r^{(N)}$ of (1) is generated recursively for $r = N, N-1, \cdots, 0$, beginning with $y_N^{(N)} = 0$ and $y_{N-1}^{(N)} = 1$. Then f_r is found by multiplying the $y_r^{(N)}$ by a normalizing factor λ_N. For example, if the value of f_0 is given then $\lambda_N = f_0/y_0^{(N)}$. More generally, if f_r satisfies a condition of the form

$$m_0 f_0 + m_1 f_1 + m_2 f_2 + \cdots = 1, \tag{2}$$

with given coefficients m_r, then

$$\lambda_N = 1/(m_0 y_0^{(N)} + m_1 y_1^{(N)} + \cdots + m_{N-1} y_{N-1}^{(N)}).$$

The value of N can be estimated from the asymptotic form of f_r for large r. But often such information is unavailable, in which event N is assigned arbitrarily. The adequacy of the guess is tested by repetition of the algorithm with a higher value and comparing results. If agreement is inadequate, then additional higher values must be tried.

Algorithm II was proposed by one of the present writers [3] primarily for the solution of inhomogeneous difference equations of the second order.** It is, in part, a forward recurrence procedure. Beginning with $p_0 = 0$ and $p_1 = 1$, we compute a solution p_r of (1) for $r = 1, 2, \cdots$. Also computed is a sequence $\{e_r\}$ defined by

$$e_0 = f_0, \qquad e_r = a_r e_{r-1}/c_r \qquad (r \geqq 1).$$

Here, f_0 is either the given value of the wanted solution or an arbitrary value; in the latter event a final normalization has to be effected by use of a relation of the form (2), as in Algorithm I. Computation of p_r and e_r is terminated automatically at a certain value of r, which we denote by $N + 1$. The determination of N is described below. The required approximation $f_r^{(N)}$ to the wanted solution f_r is then generated according to the equations $f_N^{(N)} = 0$ and

$$p_{r+1} f_r^{(N)} - p_r f_{r+1}^{(N)} = e_r \qquad (r = N-1, N-2, \cdots, 0). \tag{3}$$

This algorithm is based on the solution of the simultaneous set of equations (1) for $r = 1, 2, \cdots, N-1$, with the conditions $y_0 = f_0$ and $y_N = 0$. Computation of p_r and e_r represents the elimination stage, and (3) is the process of back-substitution.

The value of N is found in the following way. We have from [3, Section 5]

$$f_r - f_r^{(N)} = E_N p_r, \tag{4}$$

where

$$E_N = \frac{e_N}{p_N p_{N+1}} + \frac{e_{N+1}}{p_{N+1} p_{N+2}} + \frac{e_{N+2}}{p_{N+2} p_{N+3}} + \cdots. \tag{5}$$

Generally, this series converges fairly rapidly and the sum is of the same order of magnitude as the first term. Suppose, for example, that the final solution is required to D decimal places. Then the size of the function $e_N p_r/(p_N p_{N+1})$ is examined as the computations proceed. As soon as this test function falls below $\frac{1}{2} \times 10^{-D}$ for all values of r in the range of interest, the corresponding value of N is accepted.

In cases where the series (5) does not converge rapidly, the actual value of E_N

** A similar method has been described in [4], written apparently without knowledge of [3].

can be found *a posteriori* by computing values of p_r beyond $r = N + 1$ and summing (5) numerically. Multiplication of E_N by the various p_r then enables the truncation errors (4) to be estimated reliably.

3. Implementation. The principal difficulty associated with Algorithm I is the estimation of N. Computing time is wasted if either the asymptotic estimate or the initial guess is much too low or much too high. Another difficulty is a slight uncertainty associated with the acceptance criterion: mere numerical agreement of solutions computed with two different values of N does not guarantee their accuracy.

Neither of these problems attends Algorithm II. The optimum N is determined automatically, and the expansion (5) is available to bound truncation errors in the final solution. In consequence, although Algorithm II entails the more complicated computing procedure, the fact that only one application is needed may make it faster in practice.

There is a difficulty, however, in constructing fully satisfactory computing programs based on Algorithm II stemming from possible loss in accuracy in the formation of the sequence p_r. Since p_r is a linear combination of f_r and g_r, it increases ultimately in proportion to g_r.*** Initially, however, p_r may behave more like a multiple of f_r, in which event precision is lost by cancellation. Whether this affects the accuracy of the final solution depends on the normalization condition being used. There *is* a final loss if the condition prescribes the value of f_0, but not, as a rule, with a more general condition of the form (2); compare [3, Section 7].

4. Combined Algorithm. Although none of the drawbacks mentioned in Section 3 is catastrophic in practice, they can be overcome altogether—without sacrifice of speed—by judicious combination of Algorithms I and II. The modification applies when the dominant solution g_r tends to infinity with r in such a way that ultimately the absolute value of g_r is monotonic.

First, an integer M is chosen large enough to insure that if the sequence p_r is computed from (1), beginning with $p_M = 0$ and $p_{M+1} = 1$, then its members are nondecreasing in absolute value. Thus, cancellation is completely precluded. The selection of M is discussed below.

Second, Algorithm II is applied to compute, for $r \geqq M + 1$, the recessive solution which satisfies the condition $y_M = 1$.

Third, the values of y_r for $r = M - 1, M - 2, \cdots, 0$ are computed by backward application of (1). As in the case of Algorithm I, this is a stable procedure.

Finally, the wanted solution f_r is found by multiplying the y_r by a normalizing constant determined from (2).

The exact choice of M is not critical, and an acceptable value can often be determined by application of the following result:

LEMMA 1. *If* $|b_r| \geqq |a_r| + |c_r|$ *when* $r \geqq M + 1$, *and* p_r *is the solution of* (1) *satisfying* $p_M = 0$ *and* $p_{M+1} = 1$, *then*

$$|p_r| \geqq |p_{r-1}| \qquad (r \geqq M + 1). \tag{6}$$

This is proved in Section 6.

*** Even in a case in which, initially, p_r is *exactly* a multiple of f_r, it behaves eventually as a multiple of g_r owing to the introduction of rounding errors.

In practice, the procedure lends itself to two improvements, as follows.

(i) Suppose that the solution y_r is required in floating-point form. That is, y_r is assumed to be needed to S significant figures for the range $(0, L)$, both S and L being given. From [3, (5.03)], we have

$$y_r = p_r \sum_{s=r}^{\infty} \frac{e_s}{p_s p_{s+1}}. \tag{7}$$

As in the case of (5), it is reasonable, for large r, to approximate this expansion by its first term. Then referring to (4), we see that the relative error of $y_r^{(N)}$ is approximately

$$(e_N/p_N p_{N+1})(p_r p_{r+1}/e_r). \tag{8}$$

The value of $N\,(>L)$ is found by insuring that this quantity is bounded by $\frac{1}{2} \times 10^{-S}$ for all values of r in (M, L). The back-substitution begins with $y_N = 0$ and $y_{N-1} = e_{N-1}/p_N$, but, instead of continuing by application of (3), we revert to the original difference equation (1), generating y_r from $r = N - 2$ through $r = M$ down to $r = 0$. This is allowable, since errors in the backward recursion grow no faster than the wanted solution.

This refinement simplifies the programming because the formula used for y_r is the same in the ranges $(0, M)$ and (M, L). Moreover, in generating the sequence p_r, only current values need be stored. In this form, *the role of Algorithm* II *may be regarded simply as a procedure for finding the optimum N for use with Algorithm* I.

Perhaps, it should be noted that this refinement applies only to homogeneous difference equations as a rule; in the case of an inhomogeneous equation, both forward *and* backward recursion may be unstable.

(ii) In [5], it is shown how to construct lower bounds for the $|p_r|$, and thence an upper bound for E_N. By using these bounds, it is possible to choose N in such a way that the truncation error $E_N p_r$ falls below the specified tolerance in f_r throughout the given range of values of r. In other words, *all* terms in the expansion (5) are automatically taken into account in determining N, and not merely the first term.

Both improvements (i) and (ii) are incorporated in the example given in the next section.

5. Example. Consider the recurrence relation

$$y_{r-1} - (2r/x)y_r + y_{r+1} = 0 \tag{9}$$

satisfied by the Bessel functions $J_r(x)$ and $Y_r(x)$, and suppose that x is real and positive. It is well known that for fixed x and varying r the behavior of the Bessel functions is quite different in the ranges $0 \leqq r < x$ and $x < r < \infty$. In the former range, the functions oscillate with slowly changing amplitude, whereas in the latter range, $Y_r(x)$ tends rapidly to $-\infty$ and $J_r(x)$ tends rapidly to zero. Let us suppose that $J_r(x)$ is required to fixed-point accuracy for $r \leqq x$ and floating-point accuracy for $r > x$. More precisely, if $[x]$ denotes the integer part of x then $J_r(x)$ is required to D decimal places for $0 \leqq r \leqq [x]$, and S significant figures for $[x] \leqq r \leqq L$. Here, S and L are prescribed, and D is the number of decimal places in $J_{[x]}(x)$ corresponding

to S significant figures in the same quantity. Since $|J_r(x)|$ is bounded by unity,† it follows that $D \geqq S$.

A sequence $p_M, p_{M+1}, \cdots$ is computed by forward application of (9), beginning with $p_M = 0$ and $p_{M+1} = 1$. In the notation of Lemma 1, we have $|b_r| = 2r/x$ and $|a_r| + |c_r| = 2$. Hence, with $M = [x]$, we can be sure that $p_M, p_{M+1}, \cdots$ is a non-decreasing sequence. Since all the quantities e_r in this example are unity, computation of p_r is terminated at $r = N$, where $N\,(>L)$ is the least odd integer for which

$$p_N p_{N+1} > (2 \times 10^S) p_L p_{L+1}; \tag{10}$$

compare (8). Values of y_r for $r = N, N-1, \cdots, 0$ are found by backward application of (9), beginning with $y_N = 0$ and $y_{N-1} = 1/p_N$. Finally, the desired $J_r(x)$ are obtained by multiplying the y_r by the factor $1/(y_0 + 2y_2 + 2y_4 + \cdots + 2y_{N-1})$, since††

$$J_0(x) + 2J_2(x) + 2J_4(x) + \cdots = 1.$$

Because $y_M = 1$, this normalizing factor equals $J_M(x)$, approximately, and therefore cannot exceed unity in absolute value.

The actual truncation error in each y_r for the range (M, L) could be found from (4) and (5). Instead, however, we adopt the suggestion made in Section 4(ii) and increase N to $\bar{N}$, say, to *guarantee* that $\frac{1}{2} \times 10^{-S}$ is an upper bound for the relative truncation errors in $y_M, y_{M+1}, \cdots, y_L$. This depends on the following result, which is a refinement of Theorem 2 of [5] in the present case.

LEMMA 2. *Let $x > 0$, $M = [x]$, and the sequence $p_M, p_{M+1}, \cdots$ be defined by $p_M = 0$, $p_{M+1} = 1$, and*

$$p_{r+1} = (2r/x)p_r - p_{r-1} \qquad (r \geqq M + 1).$$

Then

$$p_{s+1}/p_s \geqq \min(p_{r+1}/p_r, \lambda_r) \qquad (s \geqq r \geqq M),$$

where λ_r is the largest zero of the quadratic $\lambda^2 - 2(r+1)x^{-1}\lambda + 1$.

The proof of this result is given in Section 6. To apply the lemma, write

$$\kappa_r = p_{r+1}/p_r, \qquad \rho_r = \min(\kappa_r, \lambda_r), \tag{11}$$

and let N be the least integer for which (10) is satisfied. The value of ρ_N is found by testing the sign of

$$\Delta_N \equiv p_{N+1}^2 - 2(N+1)x^{-1}p_{N+1}p_N + p_N^2.$$

If $\Delta_N \leqq 0$, then $\rho_N = p_{N+1}/p_N$, otherwise

$$\rho_N = \lambda_N = (N+1)/x + \{((N+1)/x)^2 - 1\}^{1/2}.$$

Having obtained ρ_N, we compute $\bar{N}$ as the least integer exceeding N for which

$$p_{\bar{N}}^2 > (2 \times 10^S) p_L p_{L+1} \rho_N / (\rho_N^2 - 1). \tag{12}$$

† [6, Section 2.5].

†† [6, Section 2.22].

To verify that $\bar{N}$ has the desired property when defined in this way, we have, from (5) and (11),

$$E_{\bar{N}} = \frac{1}{p_{\bar{N}}^2}\left(\frac{1}{\kappa_{\bar{N}}} + \frac{1}{\kappa_{\bar{N}}^2\kappa_{\bar{N}+1}} + \frac{1}{\kappa_{\bar{N}}^2\kappa_{\bar{N}+1}^2\kappa_{\bar{N}+2}} + \cdots\right).$$

Use of Lemma 2 gives

$$E_{\bar{N}} \leqq \frac{1}{p_{\bar{N}}^2}\left(\frac{1}{\rho_N} + \frac{1}{\rho_N^3} + \frac{1}{\rho_N^5} + \cdots\right) = \frac{\rho_N}{p_{\bar{N}}^2(\rho_N^2 - 1)}. \tag{13}$$

Since y_r is at least $1/p_{r+1}$ (compare (7)), the truncation error of $y_r^{(N)}$ relative to the magnitude of y_r is bounded by $E_{\bar{N}}p_rp_{r+1}$. If $M \leqq r \leqq L$, then from (12), (13), and Lemma 1, we see that

$$E_{\bar{N}}p_rp_{r+1} \leqq E_{\bar{N}}p_Lp_{L+1} \leqq \rho_Np_Lp_{L+1}/p_{\bar{N}}^2(\rho_N^2 - 1) \leqq \tfrac{1}{2} \times 10^{-S},$$

as required.

Remarks (i). In an unpublished paper [7], Kahan proposed the following method for estimating N. Starting with $y_{\lfloor x \rfloor} = 0$ and $y_{\lfloor x \rfloor + 1} = \beta$, where β is an arbitrary positive number (though small in practice) the sequence y_r is computed by forward application of (9). Then, for fixed-point computation to D decimals, N is the least integer for which $y_{N+1} \geqq (2 \times 10^D)\beta$. FORTRAN programs based on this criterion have been constructed [8]. In spirit, this procedure is the same as in the present paper, but lacks the precise control of error.

(ii) Asymptotic estimates of an acceptable N are given in [1, Section 5]. These are somewhat complicated to reproduce in full, but it is to be noted that they give a value exceeding $\frac{1}{2}ex$, where e is the base of natural logarithms. For large x, this is a considerable overestimate. For example, with $L = x = 1024$ and $S = 19$, the criterion of the present section gives $\bar{N} - x = 130$, compared with $\frac{1}{2}ex - x = 368$.

(iii) Numerical tables of the optimum N for floating-point accuracy (S significant figures) are given in [9] for the ranges $x \leqq 100$ and $S \leqq 30$. Apart from the fact that the range of x is somewhat restrictive, tables of this kind are cumbersome to incorporate in a flexible computing program.

(iv) FORTRAN programs for this example, together with extensions to complex x, are being prepared.

6. Proofs of Lemmas.

LEMMA 1. If (6) holds for a certain value of r—as is the case when $r = M + 1$—then

$$|p_{r+1}| = \left|\frac{b_rp_r - a_rp_{r-1}}{c_r}\right| \geqq \frac{(|b_r| - |a_r|)\,|p_r|}{|c_r|} \geqq |p_r|.$$

LEMMA 2. By use of Lemma 1, it is seen that p_r is positive for $r \geqq M + 1$. From the definitions (11) and the given conditions, we have

$$\kappa_{r+1} = 2(r + 1)/x - 1/\kappa_r, \tag{14}$$

and

$$\lambda_r = 2(r + 1)/x - 1/\lambda_r. \tag{15}$$

(i) Suppose first that $\kappa_r \geqq \lambda_r$. Then (14) and (15) imply that $\kappa_{r+1} \geqq \lambda_r$. Next,

$$\kappa_{r+2} = \frac{2(r+2)}{x} - \frac{1}{\kappa_{r+1}} \geqq \frac{2(r+1)}{x} - \frac{1}{\kappa_{r+1}} \geqq \lambda_r.$$

Continuation of this argument shows that $\kappa_{r+s} \geqq \lambda_r$ for $s = 0, 1, \cdots$, as required.

(ii) Alternatively, suppose that $\kappa_r < \lambda_r$. Because $\kappa_r \geqq 1$ (Lemma 1), it follows that κ_r separates the two λ-zeros of the function

$$\lambda - 2(r+1)/x + 1/\lambda.$$

Hence

$$\kappa_r - 2(r+1)/x + 1/\kappa_r < 0,$$

and therefore, from (14), $\kappa_r < \kappa_{r+1}$.

Next, either $\kappa_{r+1} \geqq \lambda_{r+1}$ or $\kappa_{r+1} < \lambda_{r+1}$. As in (i) the first alternative implies that $\kappa_{r+s} \geqq \lambda_{r+1}$ ($s \geqq 1$). Since λ_r is an increasing function of r this gives $\kappa_{r+s} > \lambda_r$ and thence $\kappa_{r+s} \geqq \kappa_r$, $s = 0, 1, \cdots$, as required. For the second alternative, we reason as in the preceding paragraph that $\kappa_{r+1} < \kappa_{r+2}$ and then examine the alternatives $\kappa_{r+2} \geqq \lambda_{r+2}$ and $\kappa_{r+2} < \lambda_{r+2}$. And so on.

Acknowledgments. The writers are pleased to acknowledge helpful discussions with their colleague Dr. H. J. Oser. The work was funded in part by the Center for Computer Sciences and Technology at N.B.S.

Institute for Fluid Dynamics and Applied Mathematics
University of Maryland
College Park, Maryland 20742

Mathematical Analysis Section, IBS
National Bureau of Standards
Washington, D. C. 20234

1. W. GAUTSCHI, "Computational aspects of three-term recurrence relations," *SIAM Rev.*, v. 9, 1967, pp. 24–82. MR **34** #3927.

2. BRITISH ASSOCIATION FOR THE ADVANCEMENT OF SCIENCE, "Bessel functions. Part II," *Mathematical Tables,* v. 10, Cambridge University Press, Cambridge, 1952.

3. F. W. J. OLVER, "Numerical solution of second-order linear difference equations," *J. Res. Nat. Bur. Standards Sect. B*, v. 71, 1967, pp. 111–129. MR **36** #4841.

4. J. G. WILLS, "On the use of recursion relations in the numerical evaluation of spherical Bessel functions and Coulomb functions," *J. Computational Phys.,* v. 8, 1971, pp. 162–166.

5. F. W. J. OLVER, "Bounds for the solutions of second-order linear difference equations," *J. Res. Nat. Bur. Standards Sect. B,* v. 71, 1967, pp. 161–166. MR **37** #4981

6. G. N. WATSON, *A Treatise on the Theory of Bessel Functions,* 2nd ed., Cambridge Univ. Press, Cambridge, 1944.

7. W. KAHAN, "Note on bounds for generating Bessel functions by recurrence." (Unpublished.)

8. D. JORDAN, *Argonne National Laboratory Library Routine*, ANL C370S—BESJY, October 1967.

9. S. MAKINOUCHI, "Note on the recurrence techniques for the calculation of Bessel functions $J_\nu(x)$," *Tech. Rep. Osaka Univ.*, v. 15, 1965, pp. 185–201. MR **33** #6813.