Acceleration Techniques for Iterated Vector and Matrix Problems*

By P. Wynn

1. Introduction. The ϵ-algorithm [1], which is closely related (see [2]) to the $e_m(S_n)$ transformation [3] and the method of summability by Stieltjes-type J-fractions [4], provides a powerful technique for transforming slowly convergent or divergent sequences (see for example [5], [6]). The quantities $\epsilon_s^{(m)}$ which satisfy the relationship

$$\epsilon_{s+1}^{(m)} = \epsilon_{s-1}^{(m+1)} + \frac{1}{\epsilon_s^{(m+1)} - \epsilon_s^{(m)}} \tag{1}$$

may be arranged in Table 1 and are seen to occur at the corners of a lozenge in this array.

The usefulness of the ϵ-algorithm lies in the fact that if the sequence

$$\epsilon_0^{(m)} = S_m \qquad m = 0, 1, \cdots . \tag{2}$$

is slowly convergent, then (in certain cases) the numerical convergence of the sequence $\epsilon_{2s}^{(0)} s = 0, 1, \cdots$ to the limit (or antilimit), with which the sequence (2) may be associated, is far more rapid.

In the application of the ϵ-algorithm so far the $\epsilon_s^{(m)}$ have been scalar quantities; it is the purpose of this paper to extend the inquiry to the cases in which the S_m are a sequence of slowly convergent arrays. In particular, the cases in which the S_m are (a) vectors (b) square matrices (c) triangular matrices will be considered.

The sums and differences of these entities are of course already well defined, but the choice of an inverse must be given some consideration. Four possibilities will be considered. They are

(1) *Primitive Inverse.* Regarding each component separately, this is equivalent to the simultaneous application of the scalar ϵ-algorithm to the components of (a), (b) and (c).

(2) *The Samelson Inverse of a Vector.* Here an extremely elegant and profound idea, due to K. Samelson, is introduced. The inverse of a vector

$$\mathbf{x} = (x_1, x_2, \cdots, x_n)$$

is taken to be

$$\mathbf{x}^{-1} = \left(\sum_{r-1}^{n} x_r \bar{x}_r\right)^{-1} (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$$

where $\bar{x}_r$ is the complex conjugate of x_r. The point $\mathbf{x}^{-1}$ is thus the inverse point of x with respect to the unit sphere in n-space.†

Received August 1, 1961.

* Communication MR of the Computation Department of the Mathematical Centre, Amsterdam.

† Dr. E. T. Goodwin has pointed out to the author that there is a strong connection between the idea of the Samelson inverse and a result ([22] p. 675, eq. (53)) of Lanczos.

TABLE 1

$\epsilon_{-1}^{(1)} = 0$	$\epsilon_0^{(0)}$	$\epsilon_1^{(0)}$	$\epsilon_2^{(0)}$	$\epsilon_s^{(0)}$	$\epsilon_{s+1}^{(0)}$	
$\epsilon_{-1}^{(2)} = 0$	$\epsilon_0^{(1)}$	$\epsilon_1^{(1)}$	$\epsilon_2^{(1)}$	$\epsilon_s^{(1)}$	$\epsilon_{s+1}^{(1)}$	
$\epsilon_{-1}^{(3)} = 0$	$\epsilon_0^{(2)}$	$\epsilon_1^{(2)}$	$\epsilon_2^{(2)}$	$\epsilon_s^{(2)}$	$\epsilon_{s+1}^{(2)}$	
	$\epsilon_0^{(3)}$					

(3 and 4) *The Normally Defined Inverses of Square and Triangular Matrices.* (1) may be applied to all cases (a), (b) and (c); (2) may be applied to (b) and (c) in two ways, first by treating the rows (or columns) separately, and second by regarding the whole matrix as one vector.

From relationship (1) there may immediately be deduced:

THEOREM 1. *If* a, $S_m \quad m = 0, 1, \cdots$ *are of the same kind,* b *is a nonzero scalar and application of the* ϵ*-algorithm relationship to the initial values* (2) *produces quantities* $\epsilon_s^{(m)}$, *then application of the same relationship to the initial values*

$$\epsilon_{-1}^{(m)*} = 0 \qquad \epsilon_0^{(m)*} = a + bS_m \qquad m = 0, 1, \cdots \tag{5}$$

produces quantities

$$\epsilon_{2s}^{(m)*} = a + b\epsilon_{2s}^{(m)}, \qquad \epsilon_{2s+1}^{(m)*} = b^{-1}\epsilon_{2s+1}^{(m)} \qquad m, s = 0, 1, \cdots \tag{6}$$

Two results of F. L. Bauer [7] relating to the use of the primitive inverse may be extended to give

THEOREM 2

If in (2)

$$\sum_{s=0}^{h} c_s S_{m+s} = b \qquad m = 0, 1, \cdots \tag{7}$$

and the roots $\lambda_1, \lambda_2, \cdots, \lambda_h$ *of the equation*

$$\sum_{s=0}^{h} c_s \lambda^s = 0 \tag{8}$$

are real and distinct, and further

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_r| > 1 > |\lambda_{r+1}| > \cdots > |\lambda_h|$$

then

$$\lim_{m\to\infty} \epsilon_{2s-1}^{(m+1)} \epsilon_{2s-1}^{(m)-1} = \lambda_s^{-1} \tag{9}$$

and

$$\lim_{m\to\infty} \epsilon_{2s-2}^{(m+1)} \epsilon_{2s-2}^{(m)-1} = \begin{matrix} \lambda_s & s = 1, 2, \cdots, r \\ 1 & s = r+1, r+2, \cdots, h. \end{matrix} \tag{10}$$

The fundamental result in the theory of the ϵ-algorithm relating to scalar quantities is

THEOREM 3. *If equation* (7) *obtains and* $\sum_{s=0}^{h} c_s \neq 0$ *then*

$$\epsilon_{2h+1}^{(s)} = \infty, \qquad \epsilon_{2h}^{(s)} = b \Big/ \left(\sum_{s=0}^{h} c_i\right) \qquad s = 0, 1, \cdots. \tag{11}$$

This theorem is also true if the S_m have a suitably defined inverse and obey a non-commutative multiplication law. (It is quite clear, that if care is taken in the positioning of b two further versions of Theorem 1 may be given in which b is non-scalar). The proof of this result, however, is one of the more advanced results in the theory of continued fractions whose elements obey a non-commutative law of multiplication.

Before this proof can be described an almost complete account of this theory must be given, and this is far too extensive to be set down in detail here. The purpose of this paper is to demonstrate that the extension of the ϵ-algorithm to non-scalar quantities has useful application in numerical analysis, and to this we now proceed.

2. Iterated Vector Problems.

Sequential Relaxation. Perhaps the best known schemes for producing iterated vector sequences occur in the numerical solution of linear equations. Consider, for example, the process of obtaining the sequence $(x_1^{(m)}, x_2^{(m)}, \cdots, x_n^{(m)})$ $m = 1, 2, \cdots$ from a given initial vector $(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})$ by means of the scheme

$$\begin{aligned} a_{1,1}x_1^{(m+1)} + a_{1,2}x_2^{(m)} + \cdots + a_{1,n}x_n^{(m)} &= k_1 \\ a_{2,1}x_1^{(m)} + a_{2,2}x_2^{(m+1)} + \cdots + a_{2,n}x_n^{(m)} &= k_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{n,1}x_1^{(m)} + a_{n,2}x_2^{(m)} + \cdots + a_{n,n}x_n^{(m+1)} &= k_n\,. \end{aligned} \tag{12}$$

If the primitive inverse is used it is to be expected that the conditions of Theorem 3 would apply and that the ϵ-algorithm would provide an n-step process for solving the set of equations

$$\mathbf{Ax} = \mathbf{k} \tag{13}$$

where $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{k}$ are straight-forwardly defined by inspection of the quantities occurring in (12). (This is proved [8] by decomposing the vector $\mathbf{x}^{(0)}$ in terms of the eigenvectors of $\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}$ is the main diagonal of $\mathbf{A}$). An example (Table 2, exponent in brackets), which gives the vector ϵ-scheme (primitive inverse) relating to the sequential relaxation of the set of equations

$$\begin{pmatrix} +2 & +1 & +3 & +4 \\ +1 & -3 & +1 & +5 \\ +3 & +1 & +6 & -2 \\ +4 & +5 & -2 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 4 \\ 8 \\ 6 \end{pmatrix} \tag{14}$$

TABLE 2

m	s								
	0	1	2	3	4	5	6	7	8
0	0.0								
	0.0								
	0.0								
	0.0	+2.0 (−1)							
		−7.5 (−1)							
		+7.5 (−1)							
1	+5.0	−1.666667 (−1)	−4.411765						
	−1.333333		+2.711864 (−1)						
	+1.333333		+3.168317 (−1)						
	−6.0	+9.375 (−2)	−2.16	+1.619173 (−1)					
		−1.267606 (−1)		−3.869420 (−1)					
		−2.337662 (−1)		−4.982868 (−1)					
2	+1.566667 (1)	+9.375 (−2)	+1.025802 (1)	+8.433735 (−2)	+4.065470				
	−9.222222		−3.572285		−1.340802				
	−2.944444		−3.463592		−9.252170 (−1)				
	+4.666667	−9.113924 (−2)	−1.084000 (2)	+4.330304 (−4)	−4.898484	+8.065247 (−2)			
		+5.023256 (−2)		+6.119052 (−2)		−7.528139 (−2)			
		−2.160000		−1.043340 (−1)		−2.813787 (−1)			
3	+4.694444	+8.490566 (−2)	+2.117836 (1)	+9.399904 (−2)	+1.653128 (1)	+1.533841 (−2)	−6.476506 (−1)		
	+1.068519 (1)		+8.768553 (1)		−8.668317		+1.605804		
	−3.407407		−2.977132		−6.573510		+6.325394 (−1)		
	+1.644444 (1)	−3.047404 (−2)	+1.570070	−2.147557 (−1)	−1.761132 (1)	+2.244161 (−2)	+1.243017	+4.332292 (−1)	
		+6.321951 (−2)		+5.081211 (−2)		+2.205053 (−2)		−1.032366	
		+1.640922 (−1)		−3.823916 (−1)		−1.426064 (−1)		+1.104180	
4	−2.812037 (1)	+1.767594 (−2)	+1.575188 (1)	+4.186519 (−2)	+2.074718 (1)	+6.837659 (−2)	+1.786698	−2.471836	+1.0
	+2.650309 (1)		+7.088480		−4.343692 (1)		+6.574128 (−1)		+1.0
	+2.686728		−4.807012		−2.403110		+1.434601		+1.0
	+7.301852 (1)	−7.680592 (−3)	+4.291075	−1.456719 (−2)	+2.010829 (1)	−3.029967 (−2)	+8.493496 (−1)	−8.379074 (−1)	+1.0
		+1.171190 (−2)		+3.102009 (−2)		+4.472918 (−2)		+1.886598	
		+3.064748 (−2)		+3.359876 (−2)		+1.179656 (−1)		−1.196780	
5	−1.583187 (2)	−1.553808 (−2)	−1.294576 (2)	−1.989735 (−3)	−4.281560 (1)	+1.645267 (−2)	+5.484726 (−1)	+4.166050	+1.0
	+1.118863 (2)		+5.887999 (1)		+2.950735 (1)		+1.200340		+1.0
	+3.531584 (1)		+3.340283 (2)		+0.449989		+6.739977 (−1)		+1.0
	+8.660494	+2.696816 (−2)	+1.167205 (2)	−3.025448 (−3)	+7.433117 (1)	−7.239109 (−3)	+1.090337	+1.376798	+1.0
		−7.153770 (−3)		−3.025208 (−3)		+9.402247 (−3)		−3.104928	
		+3.399518 (−2)		+3.051784 (−2)		+4.017075 (−3)		+1.870683	
6	−1.212379 (2)	−6.283961 (−3)	−1.627980 (2)	−2.558057 (−2)	−2.801389 (2)	+2.799081 (−3)	+1.270997	−6.903627	
	−2.790012 (1)		+3.010951 (2)		+1.099743 (2)		+8.792432 (−1)		
	+6.473178 (1)		+4.645171 (1)		−2.828486 (1)		+1.209712		
	−1.504748 (2)	+2.906640 (−3)	+6.489793 (1)	−1.154763 (−2)	+1.095677 (2)	−3.685574 (−3)	+9.455442 (−1)		
		−4.114212 (−3)		−8.257503 (−3)		+2.359331 (−4)			
		−2.070919 (−2)		+1.713751 (−2)		+3.792161 (−2)			
7	+2.228020 (2)	−1.640848 (−3)	−2.319818 (2)	−3.194056 (−3)	−1.529457 (2)	−6.407146 (−3)			
	−2.709600 (2)		+5.974104 (1)		+2.277123 (2)				
	+1.644404 (1)		+7.287409 (1)		+1.982883 (1)				
	−7.599158 (2)	+7.077924 (−4)	−5.789307 (2)	+1.104822 (−3)	−2.016593 (2)				
		−1.090333 (−3)		−2.304104 (−3)					
		−2.988134 (−3)		−1.714317 (−3)					
8	+1.635646 (3)	+3.884469 (−3)	+2.286721 (3)	−5.434449 (−4)					
	−1.188111 (3)		−7.641373 (2)						
	−3.182129 (2)		+8.579158 (2)						
	−5.024804 (2)	+2.243712 (−3)	−8.047707 (2)						
		+1.268304 (−3)							
		−2.137887 (−3)							
9	+2.081336 (3)	+5.763910 (−4)							
	−3.996564 (2)								
	−7.859644 (2)								
	+1.232453 (3)								

substantiates this. It will be observed that the results of Theorem 3 are not of much use in this case. But the situation is analogous to the process of determining the eigenvalues of a matrix by direct iteration, and Bodewig, to whom the matrix on the left-hand side of equation (14) is due, has pointed out [9] that this is a particularly unfavorable example.

Gauss-Seidel Relaxation. An alternative scheme for the iterative solution of equation (13) is provided by the scheme

$$\begin{aligned} a_{1,1}x_1^{(m+1)} + a_{1,2}x_2^{(m)} + \cdots + a_{1,n}x_n^{(m)} &= k_1 \\ a_{2,1}x_1^{(m+1)} + a_{2,2}x_2^{(m+1)} + \cdots + a_{2,n}x_n^{(m)} &= k_2 \\ a_{n,1}x_1^{(m+1)} + a_{n,2}x_2^{(m+1)} + \cdots + a_{n,n}x_n^{(m+1)} &= k_n . \end{aligned} \tag{15}$$

(When applied to the solution of the set (14), the results from using this scheme actually diverge even more wildly than those of (12).) Again, it is to be expected that the results of Theorem 3 apply (this time with $h = n - 1$).

The method is illustrated by the solution of

$$\begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 9 \\ 6 & 8 & 10 & 7 \\ 5 & 7 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix} \tag{16}$$

again using $\mathbf{x}_0^{(0)} = 0$. The results are shown (this time only the even-order columns of the ϵ-array are displayed) in Table 3, which refers to the use of the Samelson inverse. The results in Tables 2, 3, and 4 have been computed, as have all the results of this paper, using twelve-decimal floating point arithmetic and have been rounded off for presentation to seven decimals. It is indeed remarkable that use of the primitive inverse should serve to give reasonable estimates of three of the roots and yet be inadequate in giving the value of the fourth.

Some of the estimates (taking each root in isolation) produced by use of the primitive inverse are better than those produced by use of the Samelson inverse, but if a least squares indication of the error in the estimates is used, then use of the Samelson inverse is superior. This appears generally to be true.

The numerical results displayed in Tables 3 and 4 are not particularly impressive, but this is hardly to be expected since the Wilson matrix [10] (on the left-hand side of equation (16)) is well known to be ill-conditioned.

The foregoing results have served to bring certain points into focus, in particular that the user need not be unduly frightened by the rapid divergence of the sequence to be transformed, but in the event use of the ϵ-algorithm is to be recommended neither as a method for the direct solution of a set of linear equations nor as a technique for estimating the eigenvalues of a matrix.

Iterated vector sequences occur very frequently in optimization and approximation problems, but perhaps they occur most naturally in numerical analysis in the iterative solution by digital methods of equations in which the solution is a function of a single variable. More specifically, reference is being made to the solution of

$$\alpha\{y(x), x\} = \beta\{y(x), x\} \tag{17}$$

TABLE 3

m	s				
0	0				
	0				
	0				
	0				
1	+4.6	+3.810690			
	−2.0 (−2)	−1.766784 (−2)			
	+5.56 (−1)	+1.150607			
	+3.136 (−1)	+1.007187			
2	+3.647200	+2.262368	+2.276624		
	−1.736000 (−2)	+2.060921 (−2)	+2.016264 (−1)		
	+8.433280 (−1)	+1.091095	+6.618651 (−1)		
	+5.295568 (−1)	+1.050041	+1.198757		
3	+3.082754	+2.276609	+2.333036	+2.329996	
	−3.279680 (−3)	−5.669567 (−2)	+2.044079 (−1)	+9.996350 (−1)	
	+9.763705 (−1)	+1.047934	+6.628969 (−1)	+1.664600	
	+6.821856 (−1)	+1.085579	+1.198153	+9.997397 (−1)	
4	+2.750761	+2.287822	+2.329996	+9.999154 (−1)	+1.000277
	+1.584073 (−2)	−2.504813 (−1)	+2.072094 (−1)	+1.000656	+1.000115
	+1.022904	+1.022770	+6.639249 (−1)	+7.758344 (−1)	+9.039898 (−1)
	+7.929176 (−1)	+1.114529	+1.197555	+1.000214	+9.999536 (−1)
5	+2.557421	+2.296513	+2.327167	+1.000277	+1.000127
	+3.644006 (−2)	+1.032570	+2.100183 (−1)	+9.995052 (−1)	+9.999178 (−1)
	+1.022769	+1.022904	+6.649406 (−1)	+3.329484 (−1)	+1.224386
	+8.752889 (−1)	+1.137530	+1.196962	+9.996347 (−1)	+9.999873 (−1)
6	+2.446372	+2.303113	+2.324582	+1.000019	
	+5.662204	+3.382326 (−1)	+2.128077 (−1)	+1.000149	
	+9.991194 (−1)	+1.077096	+6.659566 (−1)	+8.660018 (−1)	
	+9.379713 (−1)	+1.155232	+1.196377	+1.000531	
7	+2.383815	+2.307990	+2.322279		
	+7.545437 (−2)	+2.610966 (−1)	+2.155269 (−1)		
	+9.651736	+1.389942	+6.669650 (−1)		
	+9.866184 (−1)	+1.168357	+1.195802		
8	+2.349537	+2.311459			
	+9.255223	+2.362798 (−1)			
	+9.282793 (−1)	−4.587012 (−1)			
	+1.024993	+1.177629			
9	+2.331498				
	+1.078324 (−1)				
	+8.923411 (−1)				
	+1.055661				

where α and β operate on $y(x)$ and x, by constructing the sequence $y_r(x) r = 0, 1, \cdots$ by means of the equation

$$\alpha\{y_{r+1}(x), x\} = \beta\{y_r(x), x\}. \tag{18}$$

If x ranges (or may be transformed from another variable so as to do so) from a to $a + nh$ then the iterated vectors become

$$(y_r(a), y_r(a + h), \cdots, y_r(a + nh)) \qquad r = 0, 1, \cdots$$

where h is a suitable interval. Examples of the application of the ϵ-algorithm in such cases now follow.

TABLE 4

m	s				
0	0				
	0				
	0				
	0				
1	+4.6	+3.780995			
	−2.0 (−2)	−1.748738 (−2)			
	+5.56 (−1)	+7.377991 (−2)			
	+3.136 (−1)	+4.537392 (−1)			
2	+3.647200	+2.290460	+2.310052		
	−1.736000 (−2)	+5.900807 (−2)	+7.761601 (−2)		
	+8.433280 (−1)	+1.036824	+9.780429 (−1)		
	+5.295568 (−1)	+9.803224 (−1)	+1.010201		
3	+3.082754	+2.310449	+2.327842	+2.323830	
	−3.279680 (−3)	+7.759300 (−2)	+2.011186 (−1)	+2.033882 (−1)	
	+9.763705 (−1)	+9.781162 (−1)	+6.644644 (−1)	+6.657814 (−1)	
	+6.821856 (−1)	+1.010156	+1.197278	+1.196510	
4	+2.750761	+2.329024	+2.323830	+1.000010	+1.000196
	+1.584073 (−2)	+9.340374 (−2)	+2.033883 (−1)	+1.000647	+9.995082 (−1)
	+1.022904	+9.287211 (−1)	+6.657814 (−1)	+9.994166 (−1)	+1.000566
	+7.929176 (−1)	+1.034256	+1.196510	+1.000180	+9.996452 (−1)
5	+2.557421	+2.347263	+2.319847	+1.000196	+1.000110
	+3.644006 (−2)	+1.071077 (−1)	+2.056189 (−1)	+9.995082 (−1)	+9.999540 (−1)
	+1.022769	+8.863379 (−1)	+6.671455 (−1)	+1.000566	+1.000010
	+8.752889 (−1)	+1.053689	+1.195713	+9.996452 (−1)	+1.000030
6	+2.446372	+2.366192	+2.315893	+1.000048	
	+5.662204 (−2)	+1.195974 (−1)	+2.078090 (−1)	+1.000149	
	+9.991194 (−1)	+8.482916 (−1)	+6.685656 (−1)	+9.995584 (−1)	
	+9.379713 (−1)	+1.069723	+1.194878	+1.000543	
7	+2.383815	+2.386159	+2.311967		
	+7.545437 (−2)	+1.323159 (−1)	+2.099600 (−1)		
	+9.651736 (−1)	+8.110905 (−1)	+6.700522 (−1)		
	+9.866184 (−1)	+1.084318	+1.193997		
8	+2.349537	+2.404774			
	+9.255223 (−2)	+1.475526 (−1)			
	+9.282793 (−1)	+7.706173 (−1)			
	+1.024993	+1.100770			
9	+2.331498				
	+1.078324 (−1)				
	+8.923411 (−1)				
	+1.055661				

Integral Equations: *Fredholm Integral Equations of the Second Kind.* An attempt to solve the equation

$$\int_a^b k(x,t)f(t)\,dt = g(x) + f(x) \tag{19}$$

may be made by setting up the classical iterative scheme

$$f_{r+1}(x) = -g(x) + \int_a^b k(x,t)f_r(t)\,dt. \tag{20}$$

An example of this technique is provided by the work of E. R. Love [11], who shows

that the equation

$$f(x) = 1 - \int_{-1}^{1} \frac{a}{\pi\{a^2 + (x - t)^2\}} f(t)\, dt \tag{21}$$

occurs in potential theory. This leads to the recursion among the sequence of continuous functions $f_r(x)$ $r = 0, 1, \cdots ; -1 \leqq x \leqq 1$

$$f_{r+1}(x) = 1 - \int_{-1}^{1} \frac{a}{\pi\{a^2 + (x - t)^2\}} f_r(t)\, dt. \tag{22}$$

Using the integration formulas

$$\int_a^{a+nh} f(t)\, dt = \tfrac{1}{2} f_0 + f_1 + f_2 + \cdots + f_{n-1} + \tfrac{1}{2} f_n + C \tag{23}$$

where

$$C = (-\tfrac{1}{12}\Delta + \tfrac{1}{24}\Delta^2 - \tfrac{19}{720}\Delta^3 + \cdots)(f_n - f_0) \tag{24}$$

or

$$C = \tfrac{1}{12}(\Delta f_0 - \nabla f_n) - \tfrac{1}{24}(\Delta^2 f_0 + \nabla^2 f_n) + \tfrac{19}{1120}(\Delta^3 f_0 - \nabla^3 f_n) - \cdots \tag{25}$$

(22) is transformed into an iteration scheme among the vectors

$$(f_r(-1), f_r(-1 + h), \cdots, f_r(1 - h), f_r(1)) \qquad r = 0, 1, \cdots .$$

Table 5 shows the results of this scheme and the effects of accelerating it (Samelson inverse), when $a = 1.0$ and $h = 0.25$.

Due to the symmetry of the interval of integration and the fact that the kernel is a function of $x - y$, the solution vector is of course symmetric about the origin, and for this reason the values of the vectors for negative argument are not shown. The transformed results check to four decimals those derived by Fox and Goodwin [12] by use of the more normal method of approximating (21) by a set of linear equations. (These authors also recommend, with the support of an example, the acceleration of the iterative scheme (22) by means of Aitkens' δ^2 process [13], which corresponds in the present notation to the use of the sequence $\epsilon_2^{(s)}$ $s = 0, 1, \cdots$.)

Volterra Integral Equations. The equations now being considered, are of the form

$$\int_a^x k(x, t) f(t)\, dt = g(x) + f(x) \tag{26}$$

and lead to the iterative scheme

$$f_{r+1}(x) = -g(x) + \int_a^x k(x, t) f_r(t)\, dt. \tag{27}$$

An example of such a scheme is provided by the integral equation

$$\int_0^x \frac{-2}{(x - t + 2)^2} f(t) = f(x) - \frac{1}{(x + 2)^2} \tag{28}$$

of Friedlander [14] occurring in the theory of parabolic reflectors. This leads to the

TABLE 5

m	s			
	0	2	4	
0	1.0			
	1.0			
	1.0			
	1.0			
	1.0			
1	0.49999	0.65702		
	0.50990	0.66353		
	0.53961	0.68323		
	0.58731	0.71524		
	0.64757	0.75623		
				x
2	0.72889	0.65743	0.65741	0.00
	0.73367	0.66382	0.66381	0.25
	0.74833	0.68320	0.68320	0.50
	0.77271	0.71489	0.71490	0.75
	0.80465	0.75568	0.75570	1.00
3	0.62499	0.65741		
	0.63212	0.66381		
	0.65366	0.68320		
	0.68868	0.71490		
	0.73350	0.75570		
4	0.67212			
	0.67818			
	0.69660			
	0.72679			
	0.76577			

recursion

$$f_{r+1}(x) = (x+2)^{-2} - 2\int_0^x (x-t+2)^{-2} f_r(t)\, dt. \tag{29}$$

Again, by the use of formula (23) this is transformed into an iterated vector scheme. Table 6 gives the results of this scheme and the accelerated results (primitive inverse). The integration is carried out over the range of $0 \leqq x \leqq 1.75$ at an interval in x of 0.25. After a further two iterations of (29), the original sequence would have attained the accuracy of the transformed results shown and, in view of the simplicity of (29) and (23), this means that this example is not a particularly striking one; however use of the acceleration technique here is instructive, and if the original equation had been more complicated (as in practice such equations normally are) it would have been useful.

The results may be checked by use of the power series expansion

$$f(x) = \tau_0 - \tfrac{1}{2}x\tau_1 + (\tfrac{1}{2}x)^2\tau_2 - \cdots \tag{30}$$

where

$$\tau_0 = 1,$$

$$\tau_n = \frac{1}{4}(n+1) + \tau_0 + \frac{\tau_1}{\binom{n}{1}} + \frac{\tau_2}{\binom{n}{2}} + \cdots + \frac{\tau_{n-1}}{\binom{n}{n-1}}, \quad n = 1, 2, \cdots \tag{31}$$

derived by Fox and Goodwin [12] directly from the integral equation (28). For

TABLE 6

m	s			
	0	2	4	
0	0.25			
	0.197 531			
	0.160 000			
	0.132 231			
	0.111 111			
	0.094 675			
	0.081 633			
1	0.25	0.25		
	0.172 960	0.174 396		
	0.120 665	0.124 994		
	0.084 293	0.091 740		
	0.058 491	0.068 767		
	0.039 921	0.052 535		
	0.026 386	0.040 828		
				Correct
2	0.25	0.25	0.25	0.25
	0.174 485	0.174 427	0.174 425	0.174 424
	0.125 530	0.125 161	0.125 154	0.125 152
	0.093 110	0.092 145	0.092 119	0.092 118
	0.071 260	0.069 466	0.069 408	0.069 411
	0.056 310	0.053 549	0.053 446	0.053 444
	0.045 939	0.042 144	0.041 987	0.041 980
3	0.25	0.25		
	0.174 424	0.174 425		
	0.125 131	0.125 153		
	0.092 026	0.092 116		
	0.069 172	0.069 401		
	0.052 989	0.053 430		
	0.041 231	0.041 957		
4	0.25			
	0.174 425			
	0.125 154			
	0.092 125			
	0.069 429			
	0.053 498			
	0.042 090			

large values of x this series is unsuitable for direct use, but may be transformed into a continued fraction (again using the ϵ-algorithm), which converges quite rapidly.

Ordinary Differential Equations. The operational equation (17) may also refer to a differential equation. For example, the differential equation

$$y'' + f(x, y) = 0 \tag{32}$$

may be transcribed [15] as

$$\begin{aligned} &y(x + h) + \phi\{x + h, y(x + h)\} = \\ &\quad 2y(x) - 10\phi\{x, y(x)\} - y(x - h) - \phi\{x - h, y(x - h)\} + \Delta\{y(x)\} \end{aligned} \tag{33}$$

where

$$\phi\{x, y(x)\} = \frac{h^2}{12} f(x, y), \tag{34}$$

h is the interval of tabulation, and

$$\Delta = 0\{\delta^6 y(x)\}. \tag{35}$$

Neglecting Δ, the following iteration scheme may be derived

$$\begin{aligned} &y_{r+1}(x + h) - 2y_{r+1}(x) + y_{r+1}(x - h) \\ &\quad = -\phi\{x + h, y_r(x + h)\} - 10\phi\{x, y_r(x)\} - \phi\{x - h, y_r(x - h)\}. \end{aligned} \tag{36}$$

This may be solved for $y_{r+1}(x + h)$ to give a forward recursion scheme using values of $y(x)$ computed independently when $x = a$, $a + h$, or used as the basis for a relaxation scheme using a two-point boundary condition giving values of $y(x)$ at $x = a$ and $x = a + (n + 1)h$. The latter has been carried out for the equation [15]

$$y'' - \frac{x^2 + 1/4}{x^2} y - \frac{\pi^2}{y^3} = 0 \tag{37}$$

which is satisfied by

$$y = \pi\{x(J_0(x)^2 + Y_0(x)^2)/2\}^{1/2} \tag{38}$$

for $a = 5.0$, $h = 0.5$, $n = 5$. The results are displayed together with the accelerated results (Samelson inverse) in Table 7. $y_0(x)$ is derived from equation (36) by replacing the right-hand side by zero.

The value of $y(5.0)$ has been computed with the assistance of the power series expansions

$$\begin{aligned} J_0(x) &= \sum_{r=0}^{\infty} \frac{(-1)^r}{(r!)^2} \left(\frac{x}{2}\right)^{2r} \\ Y_0(x) &= 2\left\{\sum_{r=0}^{\infty} \frac{(-1)^r}{(r!)^2} \left(\frac{x}{2}\right)^{2r} \left(\log\left(\frac{x}{2}\right) - \psi(r)\right)\right\}/\pi. \end{aligned} \tag{39}$$

and the value of $y(8.5)$ by use of the formula

$$y(x) = \{\pi(P_0(x)^2 + Q_0(x)^2)\}^{1/2} \tag{40}$$

where

$$P_0(x) \sim 1 - \frac{1^2 3^2}{2!(8x)^2} + \frac{1^2 3^2 5^2 7^2}{4!(8x)^4} - \frac{1^2 3^2 5^2 7^2 9^2 11^2}{6!(8x)^6} + \cdots$$

$$Q_0(x) \sim \frac{-1^2}{1!(8x)} + \frac{1^2 3^2 5^2}{3!(8x)^3} - \frac{1^2 3^2 5^2 7^2 9^2}{5!(8x)^5} + \cdots \tag{41}$$

(In the event these asymptotic series are not of much use for the argument given, but thirteen terms of the corresponding continued fraction expansion gives twelve-decimal accuracy).

Inspection of Table 7 reveals that both the original and the transformed results are better at the end than at the beginning of the range of x; this merely reflects

TABLE 7

m	s			
	0	2	4	
0	1.76865			
	1.76904			
	1.76942			
	1.76980			
	1.77019			
	1.77057			
1	1.76953	1.76880		
	1.77066	1.76931		
	1.77144	1.76976		
	1.77180	1.77014		
	1.77177	1.77045		
	1.77144	1.77071		
				Correct
2	1.76509	1.76880	1.76880	1.76897
	1.76264	1.76932	1.76931	1.76950
	1.76145	1.76977	1.76976	1.76992
	1.76184	1.77014	1.77014	1.77026
	1.76380	1.77045	1.77045	1.77054
	1.76703	1.77071	1.77071	1.77076
3	1.78737	1.76888		
	1.80279	1.76948		
	1.81149	1.76998		
	1.81184	1.77035		
	1.80386	1.77062		
	1.78924	1.77080		
4	1.67863			
	1.60708			
	1.56771			
	1.56816			
	1.60839			
	1.68069			

the fact that in (35) the differences increase as the origin is approached. For an alternative iterative scheme based upon a forward recursion see [15].

In equation (33) Δ is, of course, the difference correction of Fox and Goodwin's method VII [16]. The divergence of successive iterates is however so wild that it is futile to apply this technique. The variation in $y_r(x)$ with r is due more to the variation of $\phi\{x, y_r(x)\}$ with r than to the variation of the difference correction.

A case in which the variation in the iterates is due solely to the variation in the difference correction is provided by the equation [17]

$$y'' - (z^{-4} - 2z^{-3})y + z^{-2} = 0 \tag{42}$$

which is satisfied by the function

$$T(z) \sim 1!z^2 + 2!z^3 + 3!z^4 + \cdots . \tag{43}$$

Equation (42) serves as a basis for the recursion scheme

$$\begin{aligned}&(1 - h^2f(z + h)/12)y_{r+1}(z + h)\\ &- (2 + 5h^2f(z)/12)y_{r+1}(z) + (1 - h^2f(z - h)/12)y_{r+1}(z - h)\\ &\quad = -h^2\{5(6z^2)^{-1} + (12(z + h)^2)^{-1} + (12(z - h)^2)^{-1}\} - \Delta\{y_r(z)\}\end{aligned} \tag{44}$$

where

$$f(z) = 2z^{-3} - z^{-4}$$

and

$$\Delta\{y(z)\} \equiv \{\delta_h^6/240 - 13\delta_h^8/15120 + 17\delta_h^{10}/10080 - \delta_h^{12}/29700\}y(z). \tag{45}$$

Values of $T(z)$ for small negative values of z may be computed by using the continued fraction expansion

$$T(-z) = \frac{z^2}{1+}\,\frac{2z}{1+}\,\frac{1z}{1+}\,\frac{3z}{1+}\,\frac{2z}{1+}\,\frac{4z}{1+}\,\frac{3z}{1+}\,\frac{\left[\frac{r}{2}\right]z}{1+}\,\frac{\left(\left[\frac{r}{2}\right]+1\right)z}{1+} \tag{46}$$

while for larger values of z the power series expansion

$$T(-z) = x^{-1} + e^x\left\{\gamma + \log(x) - \sum_{n=1}\frac{(-1)^{n-1}x^n}{n(n!)}\right\}, \qquad x = z^{-1} \tag{47}$$

is available.

Using these expansions to provide boundary conditions at $z = -0.5$ and $z = -1.2$, and further function values outside these points to assist in the computation of (45), (44) has been used with $h = 0.1$ to produce a vector sequence, which has been accelerated using the primitive inverse (see Table 8).

It is, of course, a feature of the iterated solution of two-point boundary condition problems that the iterates are available as vectors, but if forward recursion from one point is used, each function value may be iterated and corrected before proceeding to the next, and this is of course to be preferred.

3. Iterated Matrix Problems. As in the case of iterated vector problems, matrix sequences occur most naturally in numerical analysis in the digital solution of

TABLE 8

m	s		
	0	2	4
0	0.18526 42708 0.23554 25892 0.28896 68980 0.34511 58040 0.40365 25004 0.46430 26898		
1	0.18526 44914 0.23554 28499 0.28896 71560 0.34511 60219 0.40365 26582 0.46430 27789	0.18526 44677 0.23554 28488 0.28896 71465 0.34511 60156 0.40365 26541 0.46430 27704	
2	0.18526 44649 0.23554 28488 0.28896 71462 0.34511 60154 0.40365 26540 0.46430 27695	0.18526 44705 0.23554 28510 0.28896 71490 0.34511 60161 0.40365 26545 0.46430 27714	0.18526 44702 0.23554 28473 0.28896 71489 0.34511 60164 0.40365 26543 0.46430 27714
3	0.18526 44720 0.23554 28465 0.28896 71501 0.34511 60162 0.40365 26545 0.46430 27719	0.18526 44702 0.23554 28473 0.28896 71489 0.34511 60164 0.40365 26543 0.46430 27714	
4	0.18526 44696 0.23554 28477 0.28896 71483 0.34511 60164 0.40365 26542 0.46430 27713		

iterative problems involving functions of two variables. A most fruitful source of such problems lies in the theory of partial differential equations, and examples of their treatment will now be given.

Boundary Value Problems. Two examples of the acceleration of the iterative solution of boundary value problems will be given. They are both provided by the equation

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} - \frac{2\partial^2\phi}{\partial x \partial y} = 0. \tag{48}$$

This, with the appropriate boundary conditions, has the solution

$$\phi = x/(x + y).$$

In both cases the boundary is the square whose vertices in the x, y plane are at

(49) $1, 1 + (n+1)h;\quad 1 + (n+1)h, 1 + (n+1)h;\quad 1 + (n+1)h, 1;\quad 1, 1.$

The derivatives in (48) have been approximated by means of the well-known schemes [18]

(50) $$\boxed{\begin{matrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{matrix}}\ \phi_{x,y} \doteq 6h^2\left\{\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right\},\qquad \boxed{\begin{matrix} -1 & & 1 \\ & 0 & \\ 1 & & -1 \end{matrix}}\ \phi_{x,y} \doteq 4h^2\frac{\partial^2\phi}{\partial x\partial y}.$$

In the first example equation (48) has been rearranged to give the iteration scheme

(51) $$\frac{\partial^2\phi_{r+1}}{\partial x^2} + \frac{\partial^2\phi_{r+1}}{\partial y^2} = \frac{2\partial^2\phi_r}{\partial x\partial y}.$$

Use of (50) gives a system of linear equations for the n^2 function values at the lattice points interior to the square (49). The right-hand sides of these equations are estimated by the use of (50) with modification of the appropriate equations by means of the boundary conditions. The first member of the sequence $\phi_0(x, y)$ is obtained by replacing the right-hand side of equation (51) by zero. The results of this iterative scheme (when $h = 0.25$ and $n = 4$) and subsequent acceleration using a matrix inverse are displayed in Table 9. (It is neither desirable nor feasible to display the complete matrices in this case. Instead the scalar norm $n^{-2}\sum_{i,j} |\phi_r(x, y) - x/(x + y)|$ has been given. This is so for the other tables in this section).

Equation (48) has also been used to derive the scheme

(52) $$\frac{\partial^2\phi_{r+1}}{\partial x\partial y} = \frac{1}{2}\left\{\frac{\partial^2\phi_r}{\partial x^2} + \frac{\partial^2\phi_r}{\partial y^2}\right\}.$$

The treatment here is precisely as in the preceding case. (It is perhaps of interest to remark that it is easy to construct the inverse of the matrix occurring in the resulting set of linear equations; this means that multiplication by this inverse matrix may be carried out analytically, and there is no need, as in the preceding case, to solve the set of equations numerically (see [23])).

The results of the iteration procedure and subsequent acceleration using the Samelson inverse vector-wise, when $h = 0.1$, $n = 4$, are shown in Table 10. The behavior of the iterated solutions is most odd in this example. The surfaces flap

TABLE 9

m	s		
	0	2	4
0	0.71 (−3)		
1	0.18 (−3)	0.93 (−4)	
2	0.65 (−4)	0.13 (−4)	0.61 (−5)
3	0.37 (−4)	0.70 (−5)	
4	0.11 (−4)		

TABLE 10

m	s					
	0	2	4	6	8	10
0	.62 (−3)					
1	.77 (−2)	.41 (−3)				
2	.15 (0)	.24 (−2)	.40 (−3)			
3	.29 (+1)	.12 (−1)	.13 (−2)	.53 (−4)		
4	.56 (+2)	.96 (−1)	.25 (−1)	.11 (−3)	.90 (−5)	
5	.11 (+4)	.52 (0)	.56 (−1)	.21 (−3)	.21 (−4)	.34(−5)
6	.21 (+5)	.40 (+1)	.13 (+2)	.70 (−3)	.12 (−3)	
7	.40 (+6)	.23 (+2)	.23 (+1)	.19 (−2)		
8	.77 (+7)	.17 (+3)	.37 (+2)			
9	.15 (+9)	.99 (+3)				
10	.28 (+10)					

about the line $x = y$, $\phi = 0$ with increasing amplitude. This feature is preserved during the accelerations.

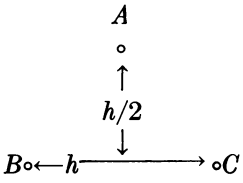
Initial Value Problems. Here the equation which has been selected for treatment is

$$\phi_{xx} - \phi_{yy} = 4\phi^2. \tag{53}$$

Initial values of ϕ, $\partial\phi/\partial x$ and $\partial\phi/\partial y$ have been given at the points $1, 0; 1 + h, 0; \cdots ; 1 + (n + 1)h, 0$. The solution to this equation with appropriate initial values is

$$\phi = (x^2 - y^2)^{-1}. \tag{54}$$

If A, B and C are three neighboring points and BA and CA are characteristics of equations (53)



then the numerical integration of (35) proceeds according to the schemes (see e.g., (19))

$$\begin{aligned} U_A &= (U_C + V_C - U_B - V_B)/2 - h(M_C - M_B)/8 \\ V_A &= (U_C + V_C - U_B + V_B)/2 - h(2M_A + M_C + M_B)/8 \\ \phi_A &= (\phi_B + \phi_C)/2 + h(U_B + V_B + V_C - U_C + 2V_A)/8 \end{aligned} \tag{55}$$

where

$$U \equiv \frac{\partial\phi}{\partial x}, \qquad V \equiv \frac{\partial\phi}{\partial y} \quad \text{and} \quad M \equiv 4\phi^2.$$

Thus with the boundary conditions stated, a sequence of triangular arrays are produced. The first member of the sequence was obtained by replacing the right-

hand side of (53) by zero. The results of this process together with the accelerated results (primitive inverse) when $h = 0.25$, $n = 12$ are shown in Table 11.

This example has been chosen chiefly to show that the method proposed is feasible. It would seem more economical to use here a special feature of two-dimensional initial value problems, namely, that successive iterates are available as vectors rather than as matrices as in the case of boundary value problems. This implies a reduction in storage space for the acceleration technique; also that if the iteration itself converges rapidly at any stage then this advantage is preserved and the acceleration technique held in the reserve to be used when necessary.

4. Note on Programming. When the quantities involved are scalar the ϵ-algorithm may be put into effect in the following way. Passing from left to right in Table 1, each column is displaced downward half a row with respect to its left-hand neighbor; the ϵ-array is thus transformed into half a square array (the e-array, say). The quantities $\epsilon_s^{(m)}$ $m = 0, 1, \cdots, m' - s$ have now become $e_{i,j}$ $j = s$ $i = j, j+1, \cdots, m'$. Assuming that the initial sequence $S_m = \epsilon_0^{(m)} = e_0^{(m)}$ $m = 0, 1, \cdots, m'$ has been inserted, the construction of the e-array then runs

$$\text{for } j := 1(1)m' \text{ do for } i := j(1)m' \text{ do}$$

$$\text{if } j = 1 \text{ then } e_{i,1} := (e_{i,0} - e_{i-1,0})^{-1}$$

$$\text{else } e_{i,j} := e_{i-1,j-2} + (e_{i,j-1} - e_{i-1,j-1})^{-1}.$$

This implies a prodigal use of the storage space but the "program" is quite simple.

When the quantities involved are vectors or matrices the implied wastage of storage space in the preceding procedure may be unacceptable. In that event it is only necessary to retain a linear array during the formation of the ϵ-scheme.

This line of vectors or matrices corresponds to what would in a normal difference table be referred to as a line of backward differences; it contains the quantities $\epsilon_0^{(m)}, \epsilon_1^{(m-1)}, \epsilon_2^{(m-2)}, \cdots, \epsilon_m^{(0)}$. After computing a new line these quantities are sequentially replaced by the quantities $\epsilon_0^{(m+1)}, \epsilon_1^{(m)}, \epsilon_2^{(m-1)}, \cdots, \epsilon_{m+1}^{(0)}$, incidentally increasing the length of the line by one unit. It is necessary to retain two extra blocks of intermediate storage space during the computation which proceeds as

TABLE 11

m	s			
	0	2	4	6
0	.54 (−1)			
1	.11 (−1)	.13 (−2)		
2	.99 (−3)	.78 (−4)	.97 (−4)	
3	.17 (−3)	.98 (−4)	.97 (−4)	.97(−4)
4	.93 (−4)	.97 (−4)	.97 (−4)	
5	.97 (−4)	.97 (−4)		
6	.97 (−4)			

follows:

compute $\epsilon_0^{(0)}$; * $l_0 = \epsilon_0^{(0)}$; $m := 1$;

EPSALG: compute $\epsilon_0^{(m)}$; $s := 0$; $b := \epsilon_0^{(m)}$;

EPSLOOP: recip := inverse of $(b - l_s)$;

‡ if $s \neq 0$ then begin *recip* := *recip* + l_{s-1} ; $l_{s-1} := c$ end;

$c := b$; $b :=$ recip; $\overset{*}{*}$ $s := s + 1$; if $m - 1 \geqq s$ then go to EPSLOOP;

$l_{m-1} := c$; $l_m := b$; †$m := m + 1$; if $m_{\max} \geqq m$ then go to EPSALG.

There are a number of comments which should be made. First, the value of m in this program is the value of m in $\epsilon_0^{(m)}$ which is at the beginning of the line *l*. Second, the meaning of the := sign depends upon the nature of the quantities involved. Thus, in the case of n-length vectors "$c := b$" should be interpreted as "for $i := 1(1)n$ *do* $c_i := b_i$". Third, if the quantities $\epsilon_0^{(m+1)} - \epsilon_0^{(m)}$ are being computed, rather than $\epsilon_0^{(m)}$, then it may improve stability to substitute

EPSALG: compute $(\epsilon_0^{(m+1)} - \epsilon_0^{(m)})$; $b := l_0 + (\epsilon_0^{(m+1)} - \epsilon_0^{(m)})$;

recip := inverse of $(\epsilon_0^{(m+1)} - \epsilon_0^{(m)})$;

set s to zero, and enter at ‡, rather than at EPSLOOP. Fourth, in any provisional inquiry it is useful to sample in some way the quantities $\epsilon_s^{(m)}$ with even suffices (as in Tables 9 to 11). This can be done by inserting the following instructions at the points * and $\overset{*}{*}$:

* $display_0 := norm\ (\epsilon_0^{(0)})$;

$\overset{*}{*}$ if $s = 0$ mod (2) then $display_{s(2m_{\max}-s)/4+m} := norm\ (c)$;

if $s = m - 1$ and $m = 0$ mod (2) then $display_{m(2m_{\max}-m+4)/4} := norm\ (b)$;

This maps scalar norms of the quantities $\epsilon_s^{(m)}$ with even suffix upon a linear array $display_i$ $i = 0(1)([m_{\max}/2] + 1)^2$. They may be extracted after completion of the ϵ-scheme computation by means of the following program:

for *sanfang* := $0(2col)m_{\max}$ do begin New Line Carriage Return;

for $m := 0(1)m_{\max} -$ *sanfang* do begin New Line Carriage Return;

for $s :=$ *sanfang* (2) *sanfang* + 2*col*-2 do if $[s/2] \leqq m \leqq m_{\max} - [s/2]$

then print $(display_{s(2m_{\max}+2-s)/4+m})$ end m end *sanfang*;

This prints out a table having the format of Tables 9 to 11 in strips of *col* columns which can subsequently be glued into position. If a non-scalar sample (of $n + 1$ quantities, say) is required, then the insertions at * and $\overset{*}{*}$ must suitably be ex-

tended and the printing out must be done as follows:

for $s := 0(2)\ m_{\max}$ do begin New Line Carriage Return;

for $m := 0(1)\ m_{\max} - s$ do for $i := 0(1)\ n$ do begin

if $i = 0 \bmod(col)$ then New Line Carriage Return;

print $(sample_{i,s(2m_{\max}+2-s)/4+m})$ end i end s.

Fifth, reference must be made to criteria for terminating the ϵ-process. (In the above program it has been assumed that $m_{\max}$ is known.) In the case of the scalar ϵ-process, the author's experience with a large number of examples has been that if the ϵ-process converges to a limit L and $| \epsilon_m^{(0)} - \epsilon_{m-2}^{(2)} | \leqq \delta$ for some even m then $| L - \epsilon_m^{(0)} | \leqq \delta$. This is a purely empirical test and one can easily construct examples in which it breaks down (indeed, one has a singly infinite number of degrees of freedom with which to do so). Nevertheless, if desired, this test may be inserted at the point † and carried out as follows † if $m = 0 \bmod(2)$, then if $| b - l_{m-2} | \leqq \delta$, go to EXIT.

At the point in the program labelled EXIT the required limit is, of course, b.

It is essential to stress that this type of estimation must be applied with discretion in the case of vector and matrix sequences.

An example (Table 12) will make this clear. This refers to the iteration of the scheme (44) using the first term of the difference correction and its subsequent acceleration (primitive inverse) when $h = 0.5$. To save space the successive estimates of $T(1.5)$ have been extracted and shown in a scalar ϵ-scheme. The same sort of behavior is shown by all components of the iterated vectors.

Judging from the table, one might be excused for thinking that the last column contains the required result correct to 11 decimals; but this is not at all true (the agreement is good only to 4 decimals). It is essential to obtain an independent estimate of how good the results are likely to be (in most cases this will be provided by examining remainder terms needed in the finite difference formulas. This applies in all cases of this paper). The terms

$$-\delta_{0.5}^{6} y_r(1.5)/240 \qquad \text{for} \qquad r = 0(1)3 \text{ are}$$

$-0.0001\ 2325\ 120$, $-0.0001\ 1909\ 511$, $-0.0001\ 1925\ 940$, $-0.0001\ 4925\ 181$, indeed provide slightly pessimistic estimates of the truncation errors in the original sequence and in the final transformed result.

TABLE 12

m	s		
	0	2	4
0	0.7239 5826 349		
1	0.7240 5913 419	0.7240 5713 559	
2	0.7240 5709 519	0.7240 5716 053	0.7240 5716 006
3	0.7240 5716 270	0.7240 5716 005	
4	0.7240 5715 994		

The point to be made here is that the original sequence undoubtedly converges to some limit. The ϵ-algorithm in this case is a convergence-preserving device. However, in view of the large interval of tabulation, this limit and the transformed result are not very interesting.

5. Conclusion. It is perhaps of interest to comment on reasons for preferring the use of one inverse in the ϵ-algorithm to any other. The consistently good numerical agreement between results produced by using the various inverses in any one case leads the author to conjecture that, within a certain limit, the rate of convergence of the ϵ-process is independent of the inverse. This being so, the choice of an inverse will depend on relative complexity and on numerical stability. On both counts the matrix inverses can be discarded. This leaves the choice between the primitive and Samelson inverses, and the author would suggest the latter for the following reason. If in any application of the ϵ-algorithm $\epsilon_{s-2}^{(m+1)} = \epsilon_{s-2}^{(m+2)} = \alpha$ then the following situation obtains:

$$\begin{array}{ccccc} & & \epsilon_{s-1}^{(m)} & & \\ & (\alpha)\epsilon_{s-2}^{(m+1)} & & (\alpha)\epsilon_s^{(m)} & \\ \epsilon_{s-3}^{(m+2)} & & (\infty)\epsilon_{s-1}^{(m+1)} & & (?)\epsilon_{s+1}^{(m)} \\ & (\alpha)\epsilon_{s-2}^{(m+2)} & & (\alpha)_s{}^{(m+1)} & \\ & & \epsilon_{s-1}^{(m+2)} & & \end{array} \tag{56}$$

the value of $\epsilon_{s+1}^{(m)}$ computed by means of the ϵ-algorithm becomes indeterminate. This is by no means an exceptional occurrence. It can be overcome by elimination and one obtains the singular rules

$$\epsilon_{s+1}^{(m)} = \epsilon_{s-1}^{(m+2)} + \epsilon_{s-1}^{(m)} - \epsilon_{s-3}^{(m+2)} \tag{57}$$

when

$$\epsilon_{s-2}^{(m+1)} = \epsilon_{s-2}^{(m+2)}$$

and

$$\epsilon_{s+1}^{(m)} = a(1 + a\epsilon_{s-1}^{(m+1)-1})^{-1} \tag{58}$$

where

$$a = \epsilon_{s-1}^{(m+2)}(1 - \epsilon_{s-1}^{(m+2)}\epsilon_{s-1}^{(m+1)-1})^{-1} + \epsilon_{s-1}^{(m)}(1 - \epsilon_{s-1}^{(m)}\epsilon_{s-1}^{(m+1)-1})^{-1} \tag{59}$$

should $\epsilon_{s-2}^{(m+1)}$ and $\epsilon_{s-2}^{(m+2)}$ be nearly equal. But this complicates the program a great deal. (The misfortune just described is of course recognized as a state of chronic instability, for example see Table 3.) At least, when using the Samelson inverse the probability of encountering this misfortune is reduced by a factor of n, and this is good enough reason for preferring it.

The reader will have noted that the examples in this paper are both small and simple. They were chosen as such in order to simplify the exposition, and the various studies given have been selected in order to indicate the varieties of numerical behavior which can occur as a result of applying an acceleration technique, rather than to display the technique in a favorable light. Integro-differential equations have not been treated, but it is unlikely that they would have introduced new points. A more serious consequence of the simplification has been that only square and triangular two-dimensional arrays have been dealt with, whereas in practice

much more varied arrays are likely to occur. But the Samelson inverse can quite clearly be applied to such arrays so that no generality has been lost. This also applies to arrays of higher dimensions.

It is appropriate at this juncture to mention that the author's experience in a large number of practical cases has been that the accelerated results were never actually worse than the original iterated sequence, though whether the gain in accuracy is worth the extra computation involved depends upon the example being considered.

Many standard iterative processes of numerical analysis are of course already rapidly convergent. In particular, it seems (although the author is unaware of any general theorem to this effect) that if the partial differential equation

$$\alpha\{\phi(x, y), x, y\} = \beta\{\phi(x, y), x, y\} \tag{60}$$

is iterated according to the scheme

$$\alpha\{\phi_{r+1}(x, y), x, y\} = \beta\{\phi_r(x, y), x, y\} \tag{61}$$

and the equation

$$\alpha\{\phi(x, y), x, y\} = 0 \tag{62}$$

and (60) are of the same type (e.g., both elliptic) then the resulting scheme converges, and it is only when equations (60) and (62) do not conform that difficulty may arise. This is exemplified by Table 11 and may be illustrated further by means of the examples

$$\phi_{xx} + \phi_{yy} = 2\{\phi_x^2 - \phi_y^2\} \tag{63}$$

$$\phi_{xy} = \phi_x^2 - \phi_y^2 \tag{64}$$

which both have the same solution as (48), viz., $x/(x + y)$. This phenomenon also occurs in the method of Rayleigh and Jansen [21] for the iterative solution of the equation of potential flow.

Acceleration techniques may have interesting repercussions upon the practical treatment of partial differential equations, for suppose that (60) is a hyperbolic partial differential equation with curved characteristics, while (62) has linear characteristics. The evaluation of the right-hand side of (60) along linear characteristics will not, in general, present much difficulty and avoids the usual circumstantial interpolation. If equation (60) is nonlinear some sort of iteration is usual. The normal objection to such a procedure is that it does not converge, but this, as Table 11 for example indicates, is no longer a critical point.

However, it would be most desirable for further work to be done on the convergence behavior of iterative processes and acceleration techniques. The latter is, of course, an interesting, useful, and largely unexplored domain of research. It is also desirable that further work should be done on the exploitation of these techniques and this paper was written in the hope that the subject would be thrown open to a larger forum of experimentation.

6. Acknowledgement. The results displayed in Tables 2 to 4 were produced by the author using the XI computer in Amsterdam, and those displayed in Tables

5 to 12 were produced by the author with the aid of the ALGOL compiler constructed by E. W. Dijkstra and J. Zonneveld for this computer. The author is grateful to Prof. Dr. K. Samelson for describing his vector inverse to him.

Mathematisch Centrum
2e Boerhaavestraat 49,
Amsterdam-O.

1. P. WYNN, "On a device for computing the $e_m(S_n)$ transformation," MTAC, v. 10, 1956, p. 91.
2. P. WYNN, "The rational approximation of functions which are formally defined by a power series expansion," *Math. Comp.*, v. 14, 1960, p. 147.
3. D. SHANKS, "Non-linear transformations of divergent and slowly convergent sequences," *J. Math. Phys.*, v. 34, 1955, p. 1.
4. T. J. STIELTJES, "Sur la réduction en fraction continue d'une serie précédant suivant les puissances descendants d'une variable," *Ann. Fac. Sci. Univ. Toulouse*, v. 3, 1889, p. 1.
5. P. WYNN, "A comparison between the numerical performances of the Euler transformation and the Epsilon algorithms," *Chiffres*, v. 4, 1961, p. 23.
6. P. WYNN, "On repeated application of the ϵ-algorithm," *Chiffres*, v. 4, 1961, p. 19.
7. F. L. BAUER, *Connections between the q-d algorithm of Rutishauser and the ϵ-algorithm of Wynn*, A technical report prepared under the sponsorship of the Deutsche Forschungsgemeinschaft, Project No. BA:106, Nov. 1957.
8. J. R. SCHMIDT, "On the numerical solution of linear simultaneous equations by an iterative method," *Phil. Mag. Ser.* 7, v. 32, 1941, p. 362.
9. E. BODEWIG, "A practical refutation of the iteration method for the algebraic eigenproblem," MTAC, v. 8, 1945, p. 237.
10. J. MORRIS, "An escalator process for the solution of linear simultaneous equations," *Phil. Mag. Ser.* 7, v. 37, 1946, p. 106.
11. E. R. LOVE, "The electrostatic field of two equal circular coaxial conducting discs," *Quart. J. Mech. Appl. Math.*, v. 2, 1949, p. 428.
12. L. FOX & E. T. GOODWIN, "The numerical solution of non-singular integral equations," *Philos. Trans. Roy. Soc. London. Ser. A*, v. 245, 1953, p. 501–534.
13. A. C. AITKEN, "On Bernoulli's numerical solution of algebraic equations," *Proc. Roy. Soc. Edinburgh*, v. 46, 1926, p. 287.
14. F. G. FRIEDLANDER, "The reflection of sound pulses by convex parabolic reflectors," *Proc. Cambridge Philos. Soc.*, v. 37, 1941, p. 134.
15. C. W. CLENSHAW & F. W. J. OLVER, "Solution of differential equations by recurrence relations," *MTAC*, v. 5, 1951, p. 34.
16. L. FOX & E. T. GOODWIN, "Some new methods for the numerical integration of ordinary differential equations," *Proc. Cambridge Philos. Soc.*, v. 45, 1949, p. 373.
17. L. FOX & J. C. P. MILLER, "Table making for large arguments, the exponential integral," *MTAC*, v. 5, 1951, p. 163.
18. R. A. BUCKINGHAM, *Numerical Methods*, Pitman, London, 1957, p. 504–505.
19. D. R. HARTREE, *Numerical Analysis*, Oxford, 1952, p. 238.
20. P. WYNN, "Singular rules for certain non-linear algorithms," to appear.
21. R. SAUER, *Einführung in die theoretische Gasdynamik*, Springer, Berlin, 1951.
22. C. LANCZOS, "Linear systems in self-adjoint form," *Amer. Math. Monthly*, v. 65, 1958, p. 665.
23. P. WYNN, "On the solution of a certain boundary-value problem," *Nord. Tid. Inf. Behand.*, v. 2, 1962, p. 61.