STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

MR 58

Note on a Converging Factor for a Certain Continued Fraction

P. Wynn

1963

Numerische Mathematik 5, 332—352 (1963)

Note on a Converging Factor for a Certain Continued Fraction*

By

P. Wynn

* Communication MR 58 of the Computation Department of the Mathematical Centre, Amsterdam.

1. Introduction

At the present time considerable interest is being taken in the computation of functions of a complex variable. For this purpose continued fractions have shown themselves to be very useful. This paper concerns itself with a device for accelerating the numerical convergence of a certain class of continued fractions. The method used is of considerable theoretical interest in itself.

In order to indicate how the formulae developed may be used, a complete ALGOL programme is given. This programme may be used to derive the numerical results which are given and (should the reader be sufficiently interested) to carry out further numerical experiments.

2.

In [1] the concept of a converging factor for a continued fraction was introduced. This computational device consisted in essence of the replacement of the tail

$$u_n = \frac{a_n}{b_n+}\,\frac{c_n}{d_n+}\cdots\frac{y_n}{z_n+}\,\frac{a_{n+1}}{b_{n+1}+}\,\frac{c_{n+1}}{d_{n+1}+}\cdots \tag{1}$$

of the continued fraction (the form of whose coefficients, apart from the first three, is periodic; the functions $a_n, b_n, c_n, d_n, \ldots, y_n, z_n$ being $2p$ in number)

$$C = \xi_0 + \frac{\xi_1}{\xi_2+}\,\frac{a_1}{b_1+}\,\frac{c_1}{d_1+}\cdots\frac{y_1}{z_1+}\,\frac{a_2}{b_2+}\,\frac{c_2}{d_2+}\cdots \tag{2}$$

by a series approximation of the form

$$u_n = \sum_{s=-k}^{+\infty} \alpha_s\, n^{-s}. \tag{3}$$

In the cases considered $a_n, b_n, \ldots, y_n, z_n$ were rational functions of their suffix, and the coefficients α_s $(s=-k, -k+1, \ldots)$ were determined recursively from the difference equation

$$u_n = \frac{a_n}{b_n+}\,\frac{c_n}{d_n+}\cdots\frac{y_n}{z_n+u_{n+1}}. \tag{4}$$

3.

The process may be illustrated by the following example:

$$\frac{\pi}{2} = 1 + \frac{1}{1+}\,\frac{1.2}{1+}\cdots\frac{n(n+1)}{1+}\cdots. \tag{5}$$

Here the converging factor

$$u_n = \frac{n(n+1)}{1+}\,\frac{(n+1)(n+2)}{1+}\cdots \tag{6}$$

satisfies the difference equation

$$u_n(1+u_{n+1}) = n^2 + n. \tag{7}$$

Inspection of this equation reveals that in the notation of equation (3), $k=1$, and that there are two possible values for α_{-1}, namely

$$\alpha_{-1}^{(1)} = 1, \qquad \alpha_{-1}^{(2)} = -1. \tag{8}$$

Subsequent coefficients $\alpha_s^{(r)}$ ($s=0, 1, \ldots; r=1, 2$) are determined recursively from equation (7). First, we derive the expansion of u_{n+1} in inverse powers of n

$$\begin{aligned} u_{n+1} &= \sum_{s=-1}^{+\infty} \alpha_s (n+1)^{-s} \\ &= \alpha_{-1} n + (\alpha_{-1} + \alpha_0) + \sum_{s=1}^{+\infty} \alpha_s \sum_{k=0}^{\infty} \binom{-s}{k} n^{-k-s} \\ &= \alpha_{-1} n + (\alpha_{-1} + \alpha_0) + \sum_{s=1}^{\infty} \Delta^s \alpha_1 n^{-s} \end{aligned} \tag{9}$$

and write

$$u_{n+1} + 1 = \sum_{s=-1}^{+\infty} \beta_s n^{-s} \tag{10}$$

where

$$\beta_{-1} = \alpha_{-1}, \qquad \beta_0 = \alpha_{-1} + \alpha_0 + 1, \qquad \beta_s = \Delta^{s-1}\alpha_1 \qquad (s=1, 2, \ldots). \tag{11}$$

Equation (7) then asserts that

$$\beta_0 \alpha_{-1} + \beta_{-1} \alpha_0 = 1 \tag{12}$$

and thereafter

$$\sum_{h=-1}^{s} \alpha_h \beta_{s-h} = 0 \qquad (s=1, 2, \ldots). \tag{13}$$

Assuming that the quantities $\alpha_{-1}, \alpha_0, \alpha_1, \ldots, \alpha_{s-1}$ are known, equation (13) contains two unknown quantities, α_s and $\Delta^{s-1}\alpha_1$. However $\Delta^{s-1}\alpha_1$ may easily be expressed as the sum of α_s and further known quantities; equation (13) may thus easily be rearranged to give α_s, and used recursively. The mechanics of this process are not completely trivial, but the important thing at this stage is to observe that there is no difficulty in principle in constructing the sets of coefficients $\alpha_s^{(r)}$.

4.

The various convergents C_r $(r=-1, 0, 1, \ldots)$ of the continued fraction (2) may be evaluated by writing (2) as

$$C = b_0' + \frac{a_1'}{b_1'+} \frac{a_2'}{b_2'+} \cdots \frac{a_n'}{b_n'+} \cdots \tag{14}$$

in which

$$b_0' = \xi_0, \quad a_1' = \xi_1, \quad b_1' = \xi_2, \quad a_2' = a_1, \quad b_2' = b_1, \ldots, \tag{15}$$

and evaluating the sequences A_r, B_r, $(r=-1, 0, 1, \ldots)$ given by

$$A_{-1} = 1, \qquad A_0 = \xi_0, \qquad A_r = b_r' A_{r-1} + a_r' A_{r-2}, \tag{16}$$

$$B_{-1} = 0, \qquad B_0 = 1, \qquad B_r = b_r' B_{r-1} + a_r' B_{r-2} \qquad (r = 1, 2, \ldots) \tag{17}$$

when

$$C_r = A_r / B_r, \qquad (r = -1, 0, 1, \ldots) \tag{18}$$

and in particular

$$C_{p(n-1)+2} = \frac{b_n A_{p(n-1)+1} + a_n A_{p(n-1)}}{b_n B_{p(n-1)+1} + a_n B_{p(n-1)}} \qquad (n = 1, 2, \ldots). \tag{19}$$

The converging factor is made use of to construct the quantity

$$C^{(r)}_{p(n-1)+2} = \frac{A_{p(n-1)+1} + u_n^{(r)} A_{p(n-1)}}{B_{p(n-1)+1} + u_n^{(r)} B_{p(n-1)}}. \tag{20}$$

In favorable cases (and the continued fraction (5) provided one such) the numerical convergence of the series (3) was rapid for both sets of coefficients $\alpha_s^{(r)}$ $(s=-1, 0, 1, \ldots; r=0, 1)$, and $C^{(1)}_{p(n-1)+2}$ was a considerably better approximation to C than was $C_{p(n-1)+2}$.

5.

Use of the converging factor $u_n^{(2)}$ brought to life a ghost function with which the continued fraction (1) may be associated. A number of conjectures regarding this function were made in the original treatment. Here we do not pursue this matter further, other than allowing for its investigation in the ALGOL programme.

6.

In the original treatment the converging factor was applied to a number of continued fraction expansions with varying degrees of success until the following expansion:

$$\frac{z^{-1}\,{}_2F_0(a+1, b+1; -z^{-1})}{{}_2F_0(a, b; -z^{-1})} = \frac{1}{z+a+b+1-} \frac{(a+1)(b+1)}{z+a+b+3-} \frac{(a+2)(b+2)}{z+a+b+5-} \cdots \tag{21}$$

was encountered.

Proceeding as in the above example we write

$$u_n = \frac{(a+n)(b+n)}{z+a+b+2n+1-} \frac{(a+n+1)(b+n+1)}{z+a+b+2n+3-} \cdots \tag{22}$$

and derive immediately the difference equation

$$u_n\{2n + z + a + b + 1 - u_{n+1}\} = ab + (a+b)n + n^2. \tag{23}$$

Substituting the series

$$u_n = \sum_{s=-1}^{+\infty} \alpha_s n^{-s}, \qquad u_{n+1} = \alpha_{-1} n + \alpha_{-1} + \alpha_0 + \sum_{s=1}^{\infty} \Delta^{s-1} \alpha_1 n^{-s} \tag{24}$$

into equation (23) we derive from the coefficients of n^2

$$\alpha_{-1}(2 - \alpha_{-1}) = 1. \tag{25}$$

Thus $\alpha_{-1} = 1$, and there appears to be only converging factor. From the coefficients of n in (23) we have

$$\alpha_{-1}(z + a + b + 1 - \alpha_0 - \alpha_{-1}) + \alpha_0(2 - \alpha_{-1}) = a + b \tag{26}$$

and this reduces to $z = 0$, which may very well not be so, and in any case does not serve to determine α_0.

This formal difficulty was overcome by writing

$$z = c(n + h) \tag{27}$$

where

$$c = e^{i \arg(z)}. \tag{28}$$

It is a substitution which is frequently encountered in work on converging factors associated with certain asymptotic series and making it was a natural step to take.

Equation (23) now envolves to the form

$$u_n\{n(2 + c) + ch + a + b + 1 - u_{n+1}\} = ab + (a + b)n + n^2 \tag{29}$$

and we obtain

$$\alpha_{-1}^{(1)} = \tfrac{1}{2}(2 + c - \eta), \qquad \alpha_{-1}^{(2)} = \tfrac{1}{2}(2 + c + \eta) \tag{30}$$

where

$$\eta = \sqrt{c(4 + c)}. \tag{31}$$

Thereafter there is no difficulty in determining further coefficients α_s $(s = 0, 1, \ldots)$ from equation (29). Numerical experiments on a somewhat modest scale served to show that some improvement in the numerical convergence of expansion (22) could be effected.

7.

In fact a subtle blunder has been made. Subsequent to the substitution (27) the converging factor is not a function of n alone but of n and h. Equation (29) is incorrect. After (27) we must write

$$u_n(h) = \frac{(a+n)(b+n)}{ch + a + b + 1 + (2+c)n -}\;\frac{(a+n+1)(b+n+1)}{ch + a + b + 3 + (2+c)n -}\cdots \tag{32}$$

and obtain

$$u_n(h)\{ch + a + b + 1 + (2 + c)n - u_{n+1}(h - 1)\} = ab + (a + b)n + n^2. \tag{33}$$

Now it transpires that a converging factor may be derived on the basis of (33) if we assume that

$$u_n = \sum_{s=-1}^{+\infty} \alpha_s(h)\, n^{-s} \tag{34}$$

where the $\alpha_s(h)$ $(s = -1, 0, 1, \ldots)$ are not constants, but polynomials of degree $s+1$ in h, or

$$\alpha_s(h) = \sum_{k=0}^{s+1} a_{s,k}\, h^k. \tag{35}$$

8.

By equating corresponding powers of n in equation (33) we obtain an expression for $\alpha_r(h)$; by equating corresponding powers of h in this expression we obtain the coefficients $a_{r,s}$ $(s=0, 1, \ldots, r+1)$.

Let us enquire a little more closely into how this is done. We first dismiss the functions $\alpha_{-1}(h)$ and $\alpha_0(h)$. Equating coefficients of n^2 in (33) we have

$$\alpha_{-1}(h)\{2+c-\alpha_{-1}(h-1)\}=1 \tag{36}$$

or, writing

$$\eta = \sqrt{c(4+c)} \tag{37}$$

and confining our attention to the converging factor $u_n^{(1)}$,

$$\alpha_{-1}(h) = (2+c-\eta)/2. \tag{38}$$

Equating coefficients of n in (33) we have

$$\alpha_{-1}\{ch+a+b+1-\alpha_0(h-1)\}+\alpha_0(h)\{2+c-\alpha_{-1}\}=a+b. \tag{39}$$

(Since $\alpha_{-1}(h)$ is a constant, nothing is lost by referring to it as α_{-1}.) If

$$\alpha_0(h) = a_{0,1}\, h + a_{0,0} \tag{40}$$

then

$$\alpha_0(h-1) = a_{0,1}\, h + a_{0,0} - a_{0,1}. \tag{41}$$

Accordingly, from the coefficients of h in (39), we have

$$\alpha_{-1}(c-a_{0,1})+a_{0,1}(2+c-\alpha_{-1})=0 \tag{42}$$

or, with (38),

$$a_{0,1} = -\alpha_{-1}\, c/\eta. \tag{43}$$

From the constant term, we derive

$$a_{0,0}=\{a+b-1-\alpha_{-1}(a+b-1-c-a_{0,1})\}/\eta. \tag{44}$$

To set up a scheme for deriving the further coefficients $a_{r,s}$ $(r=1, 2, \ldots;\ s=0, 1, \ldots, r+1)$ we return to equation (39) and observe that

$$ch+a+b+1+(2+c)n-u_{n+1}(h-1)=\sum_{s=-1}^{+\infty}\beta_s(h)\, n^{-s} \tag{45}$$

where

$$\beta_{-1}(h)=2+c-\alpha_{-1}, \tag{46}$$

$$\beta_0(h)=(c-a_{0,1})\,h+a+b+1+a_{0,1}, \tag{47}$$

and

$$\beta_s(h) = -\Delta^{s-1}\alpha_1(h-1) \tag{48}$$

(the differences, of course, are taken with respect to the suffix of the polynomial, not with respect to h).

The coefficient of n^{-r} in equation (39) then gives

$$\sum_{s=-1}^{r-1}\alpha_s(h)\,\beta_{r-s}(h)=0. \tag{49}$$

Equation (49) will be used to determine the polynomials $\alpha_{r+1}(h)$ recursively; we make two remarks concerning it. The first is that each of the products on the left hand side is a polynomial in h of degree $r+1$. The second is that if we have already determined $\alpha_s(h)$ $(s=-1, 0, 1, \ldots, r-1)$ then equation (49) contains two unknown functions $\alpha_r(h)$ and $\beta_r(h)$ $\left(\equiv -\Delta^{r-1}\alpha_1(h-1)\right)$ but, as we shall see, there is a relationship between these functions involving quantities which have already been determined. In principle, then, a process exists by means of which $\alpha_r(h)$ may be determined from equation (49).

Bearing in mind that we wish to mechanise this process, let us inquire a little more deeply into the requirements of equation (49).

We must firstly be able to form the polynomials $\beta_s(h)$, defined by equation (48), by means of differencing. Suppose that we have a two dimensional array $d_{n,s} \cdot d_{0,s}$ contains the coefficients of the successive powers of h $(s=0, 1, \ldots, r-1)$ in $-\alpha_{r-2}(h-1)$, $d_{1,s}$ those in $-\Delta\alpha_{r-3}(h-1)$, $d_{2,s}$ those in $-\Delta^2\alpha_{r-4}(h-1)$, and finally $d_{r-3,s}$ those in $-\Delta^{r-3}\alpha_1(h-1)$. We now arrive with the coefficients $b\, r_s$ $(s=0, 1, \ldots, r)$ in $-\alpha_{r-1}(h-1)$ and replace in succsion $d_{0,s}$ $(s=0, 1, \ldots, r)$ by the coefficients of the successive powers of h in $-\alpha_{r-1}(h-1)$, $d_{1,s}$ by those in $-\Delta\alpha_{r-2}(h-1)$, $d_{2,s}$ by those in $-\Delta^2\alpha_{r-3}(h-1)$, and finally $d_{r-2,s}$ by those in $-\Delta^{r-2}\alpha_1(h-1)$.

Now let us write equation (49) in the form

$$\alpha_{-1}\beta_r(h) + \alpha_r(h)\,\beta_{-1} = -\sum_{s=0}^{r-1}\beta_s(h)\,\alpha_{r-s-1}(h) \tag{50}$$

on the left hand of which stand two unknown functions $\alpha_r(h)$ and $\beta_r(h) = -\Delta^{r-1}\alpha_1(h-1)$. But there is of course a very simple relationship between these functions. It is

$$\begin{aligned}\alpha_r(h-1) &= \alpha_{r-1}(h-1) + \Delta\alpha_{r-2}(h-1) + \\ &+ \Delta^2\alpha_{r-3}(h-1) + \cdots + \Delta^{r-2}\alpha_1(h-1) + \Delta^{r-1}\alpha_1(h-1).\end{aligned} \tag{51}$$

The function $\Delta^{r-1}(h-1)$ may thus be eliminated from equation (50), which in its modified form contains apparently two unknown functions $\alpha_r(h)$ and $\alpha_r(h-1)$, but in essence of course, only one. We note in passing that equation (51) involves a process of summation through a line of backward differences. Reference to the previous paragraph shows that we have just formed this line of differences. We may then, with some economy, perform the processes of formation and summation at the same time.

We wish to evaluate the polynomial on the right hand side of (50). If $r-s-1>s$ then

$$\begin{aligned}\beta_s(h)\,\alpha_{r-s-1}(h) &= \sum_{u=0}^{s+1} h^u \sum_{v=0}^{u} b_{s,v}\, a_{r-s-1,u-v} + \\ &+ \sum_{u=s+2}^{r-s} h^u \sum_{v=0}^{s+1} b_{s,v}\, a_{r-s-1,u-v} + \\ &+ \sum_{u=r-s+1}^{r+1} h^u \sum_{v=u}^{r+1} b_{s,v}\, a_{r-s-1,u-v}.\end{aligned} \tag{52}$$

If $r-s-1<s$, then $\beta_s(h)\,\alpha_{r-s-1}(h)$ may also be expressed as three sums as in (52). Finally the case $r-s-1=s$ may be considered, and the right hand side

of (50) evaluated by summing these expressions from $s=0$ to $s=r-1$. But we are caused to split up the product $\beta_s(h)\,\alpha_{r-s-1}(h)$ into three components as in (52) solely to take into account the fact that $b_{s,v}$ is undefined for $v>s+1$ and that $a_{r-s-1,u-v}$ is undefined for $v>u$. Instead of writing down a number of differing formulae we can far more simply say that the coefficient of h^u in the expression $\sum_{s=0}^{r-1}\beta_s(h)\,\alpha_{r-s-1}(h)$ is the sum from $s=0$ to $r-1$ of all scalar products of the form $\sum_{v=0}^{r+1} b_{s,v}\,a_{r-s-1,u-v}$ provided that $u\geqq v$ and $u-v\leqq r-s$.

We have now reached the stage where equation (49) has evolved to the form

$$(2+c-\alpha_{-1})\,\alpha_r(h)-\alpha_{-1}\,\alpha_r(h-1)=-\sum_{s=0}^{r+1}\sigma_s\,h^s. \tag{53}$$

The right hand side of this equation is composed partly of terms obtained by summing through a line of backward differences as in equation (51) and partly from the addition of cross products as in equation (50). But as is easily verified

$$\alpha_r(h-1)=\sum_{s=0}^{r+1}h^s\sum_{u=s}^{r+1}(-1)^{u-s}\binom{u}{s}a_{r,u} \tag{54}$$

that is, the coefficient of h^s involves the quantities $\alpha_{r,s},\alpha_{r,s+1},\ldots,\alpha_{r,r+1}$. Thus, if we examine the coefficients of h^s in (54) in the order $s=r+1, r, r-1,\ldots,0$, we find that $a_{r,s}$ may always be expressed in terms of quantities which have previously been determined. More concisely

$$(2+c-\alpha_{-1})\,a_{r,s}-\alpha_{-1}\sum_{u=s}^{r+1}(-1)^{u-s}\binom{u}{s}a_{r,u}=-\sigma_s\,(s=r+1\,(-1)\,0) \tag{55}$$

leads to

$$a_{r,s}=\left\{\alpha_{-1}\sum_{u=s+1}^{r+1}(-1)^{u-s}\binom{u}{s}a_{r,u}-\sigma_s\right\}\Big/\eta\quad(s=r+1(-1)\,0). \tag{56}$$

At the same time that we determine $a_{r,s}$ we may easily evaluate the coefficients in $-\alpha_r(h-1)$ and thus we return to the formation of the differences to obtain $\beta_r(h)$, the summation of these differences to eliminate $\Delta^r\alpha_1(h)$, and so on.

9.

We have now shown how the converging factor $u_n(h)$ may be expressed formally as the sum of a series. But it is a matter of numerical experience that in many cases a continued fraction which may in a certain sense be associated with a given power series converges far more rapidly than the series. We would be well advised therefore, to transform the series for $u_n(h)$ into such a continued fraction. This may conveniently be done by application of the ε-algorithm [2]. The theory of this algorithm has adequately been described elsewhere [3]; it will suffice here to state that if from the initial values

$$\varepsilon_0^{(0)}=0,\qquad \varepsilon_0^{(m)}=\sum_{s=-1}^{m-2}\alpha_s(h)\,n^{-s},\qquad (m=1,2,\ldots), \tag{57}$$

$$\varepsilon_1^{(m)}=n^{m-1}\{\alpha_{m-1}(h)\}^{-1}\qquad (m=0,1,\ldots) \tag{58}$$

further quantities $\varepsilon_s^{(m)}$ $(m=0, 1, \ldots; s=2, 3, \ldots)$ are constructed by means of the relationship

$$\varepsilon_s^{(m)} = \varepsilon_{s-2}^{(m+1)} + \frac{1}{\varepsilon_{s-1}^{(m+1)} - \varepsilon_{s-1}^{(m)}} \tag{59}$$

then the quantities $\varepsilon_{2s}^{(m)}$ are convergents of certain continued fractions, and as such provide better estimates of the formal sum of the series whose partial sums are given by (57). The quantities $\varepsilon_s^{(m)}$ may be displayed in the array

$$\begin{array}{cccc}
\varepsilon_0^{(0)} & & & \\
 & \varepsilon_1^{(0)} & & \\
\varepsilon_0^{(1)} & & \varepsilon_2^{(0)} & \\
 & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} \\
\varepsilon_0^{(2)} & & \varepsilon_2^{(1)} & \vdots \\
 & \varepsilon_1^{(2)} & \vdots & \\
\varepsilon_0^{(3)} & \vdots & & \\
\vdots & & &
\end{array}$$

and it can be seen that the quantities in (59) occur at the vertices of a lozenge in this array. The various members of this array are most economically (with regard to storage space) computed by retaining a vector l which at a given stage contains the following quantities: $l_0 \equiv \varepsilon_0^{(m)}, l_1 \equiv \varepsilon_1^{(m-1)}, l_2 \equiv \varepsilon_2^{(m-2)}, \ldots, l_m \equiv \varepsilon_m^{(0)}$. We arrive with a new partial sum $\varepsilon_0^{(m+1)}$ and replace in succession l_0 by $\varepsilon_0^{(m+1)}$, l_1 by $\varepsilon_1^{(m)}, \ldots$, and add $l_{m+1} \equiv \varepsilon_{m+1}^{(0)}$. The formation of these quantities is carried out by means of (59) and uses one working space and two auxiliary storage locations.

10.

The Converging Factor $u_n^{(2)}$. All the preceding working which relates to the construction of a converging factor refers to that converging factor which may, by judicious numerical experimentation, be identified with $u_n^{(1)}$. The converging factor $u_n^{(2)}$ may be constructed in precisely the same way by changing the definition of η from that given by equation (31) to

$$\eta = -\sqrt{c(c+4)}. \tag{60}$$

11.

An ALGOL Programme. A programme for constructing the converging factor $\left(\text{either as a series } \sum_{r=-1}^{rmax} \alpha_r(h)\, n^{-r} \text{ or as a continued fraction derived from this series}\right)$ and applying it will now be given.

Before doing so it is necessary to make a few remarks. The algorithmic language ALGOL [1] in which this programme is written, does not immediately cater for arithmetic operations with complex numbers. It is therefore necessary to construct an arsenal of procedures for doing this and to devise a convention which governs their use. We therefore stipulate that all complex numbers are to be represented by arrays containing at least two members. There is an integer i which is defined globally throughout the block in which the complex arithmetic takes place, and all complex numbers (e.g. z, br_s) may be recognized throughout the programme by virtue of the fact that they contain the index i (e.g. $z[i]$, $br[i, s]$). i takes two values, zero corresponding to the real part (e.g. $\mathrm{Re}(z) \equiv z[0]$, $\mathrm{Re}(br_s) \equiv br[0, s]$) and unity corresponding to the imaginary part. The integer i may not therefore (except in circumstances which are difficult to envisage) be used for any other purpose.

Referring to the ALGOL programme there is a procedure *eq* (*one, other*) which carries out an instruction analogous to the operation *one* := *other* for real numbers. Similarly *seqeq* (*third, second, first*) carries out an assignment similar to *third* := *second* := *first*. (The procedure *eq* (*one, other*) may also be used if *other* is an expression of the form, for example, $a \times x[i] + b \times y[i] + \cdots$ in which $a, b, \ldots$ are real numbers*). The procedure *cm* (*res, one, other*) carries out an assignment similar to *res* := *one* × *other*, and *cd* (*res, one, other*) one similar to *res* := *one*/*other*. It is however necessary to ensure that numbers which occur in the arithmetic as real numbers are treated as such (i.e. with their imaginary parts put equal to zero), and for this purpose the procedure *real* (*variable*) is used**. The function of further procedures such as *mod* (*it*), *arg* (*it*), *comp sqrt* (*res, it*) is obvious. Further details are to be found in [5].

We are thus in a position to carry out the required arithmetic. Now however, there is the difficulty that the coefficients ($r = 0, 1, \ldots, r\max$; $s = 0, 1, \ldots, r+1$), which must be retained throughout the computation, are members of a triangular array, and such arrays are not defined in ALGOL. This may be overcome by constructing a mapping function (the integer procedure *mf* (*m*1, *m*2) which maps the $\alpha_{r,s}$, $b_{r,s}$ onto a linear array (of complex numbers)).

A mapping function of a somewhat similar form is encountered in the evaluation of the initial numerators and denominators of the continued fraction

$$\frac{1}{z+a+b+1-}\;\frac{(a+1)(b+1)}{z+a+b+3-}\;\frac{(a+2)(b+2)}{z+a+b+5-}\cdots. \tag{61}$$

In the notation of equations (16) and (17) the numerators (let us call them $A_{0,s}$) and denominators ($A_{1,s}$) satisfy the recursions

$$A_{j,s+1} = (z + a + b + 2s + 1)\,A_{j,s} - (a+s)(b+s)\,A_{j,s-1} \qquad (j = 0, 1;\; s = 1, 2, \ldots). \tag{62}$$

But in each case we require storage space for two complex numbers (since when $A_{j,s+1}$ has been computed, $A_{j,s-1}$ is no longer required and $A_{j,s+1}$ may be written where $A_{j,s-1}$ previously stood). But we should like the programme to be as *übersichtlich* as possible, and we therefore introduce the two integers S and *Sdash*; and when s is even these take on the values 0, 1, and 1, 0, otherwise.

A remark should also be made concerning the summation of the series. The programme as it stands continues to add in terms of the series $\alpha_r(h)\, n^{-r}$ until such time as

$$|\alpha_{r+1}(h)\, n^{-r-1}| > |\alpha_r(h)\, n^{-r}| \quad \text{and} \quad |\alpha_{r+2}(h)\, n^{-r-2}| > |\alpha_{r+1}(h)\, n^{-r-1}| \tag{63}$$

* This remark applies with equal force to the inputs to all the complex arithmetic procedures.

** The distinction between the real of the ALGOL report and the real of this paper is precisely the same as that between the titles *wirklicher Geheimrat* and *Geheimrat*.

(if this occurs before $r = rmax$ is reached) when it stops. But the decision as to the point at which the terms of a series are of no further use, is largely a matter concerning the users nerves, and the reader may not be in sympathy with this convention.

Finally it will be remembered that only the even columns of the ε-array are of interest in the transformation of the converging factor series. As these are produced they are mapped onto a display vector ($di[i, ms]$), and afterwards fished out and printed in an array which corresponds to table 1 with the columns of odd order missing.

With these remarks in mind and the comments to guide him the following ALGOL programme may be read without difficulty.

It reads as data $a, b, \varrho, \vartheta, \pi$, and si ($+1$ for the converging factor $u_n^{(1)}$ and -1 for the converging factor $u_n^{(2)}$), and immediately prints out $a, b, \varrho, \vartheta/\pi, si, h$ and n. It then computes the coefficients $\alpha_{r,s}$ ($r = -1, 0, 1, \ldots, r\max$; $s = 0, 1, \ldots, r+1$). To indicate the numerical behaviour of the polynomials $\alpha_r(h)$ and that of the terms of the series $\sum_{r=-1} \alpha_r(h)\, n^{-r}$, it continues to print out* the rows

$$\mathrm{Re}(\alpha_r(h)),\quad \mathrm{Im}(\alpha(h)),\quad |\alpha_r(h)|,\quad \mathrm{Re}(\alpha_r(h)\, n^{-r}),\quad \mathrm{Im}(\alpha_r(h)\, n^{-r}),\quad |\alpha_r(h)\, n^{-r}|$$

for $r = -1, 0, 1, \ldots$ until either condition (63) is satisfied or $rmax$ is reached. It then prints the numerical sum of the converging factor series (truncated if necessary), the n^{th} convergent C_n of (21), and the modified convergent C'_n obtained by application of the converging factor. It then prints out the even order ε-array for the converging factor (two triangular arrays, in the event, the real and imaginary parts being separated) and the two triangular arrays (again the real and imaginary parts have been separated) which correspond to the application of the transformed converging factor to the continued fraction (21).

Converging factor for continued fractions:

```
begin
    comment This programme uses the following computer oriented procedures
    (procedures the bodies of which must be written in code):
    procedure NLCR, which executes a carriage return.
    procedure TAB, which the moves the carriage to the next tabulator stop.
    real procedure read, which reads a number from the tape and advances the
                    tape to the next number.
    procedure print(x): prints the value of the variable x;

    integer rmax;
    rmax := read;

    begin
        real a, b, multiple of pi, rho, h, theta, power of n, factor, sign of sqrt;
        integer i, r, s, n, j, twormax, rs, col, S, Sdash, u, v, ncr, r1, sanfang;
        boolean still converging, display converging factor alone;
```

* The author is the guest of a non-profit making organisation.

```
array aux0, aux1, aux2, z, c, eta, am1, sum, converging factor [0 : 1],
aux3 [0:1, 0:1], alpha [0:1, 1:((rmax + 4) × (rmax + 1)) ÷ 2],
beta [0:1, 1:((rmax + 3) × rmax) ÷ 2], d [0:rmax − 2, 0:rmax, 0:1],
sigma [0:1, 0:rmax + 1], A [0:1, 0:1, 0:1], l [0:rmax + 2, 0:1],
alphar, termr [−2:0, 0:1], modtermr [−2:0],
di [0:1, 1:((rmax + 2) × (rmax + 6)) ÷ 4, 0:1];
procedure eq (one, other); real one, other;
comment serves to execute "one := other" with complex numbers and
        uses, as do the following procedures, the implicit parameter i;
for i := 0, 1 do one := other;

procedure seqeq (third, second, first);
real third, second, first;
comment serves to execute "third := second := first" for complex
        numbers;
for i := 0, 1 do third := second := first;
procedure cm (res, one, other); real res, one, other;
comment serves to execute "res := one × other" with complex
        numbers;
begin
    real Reone, Imone, Reother, Imother;
    i := 0;
    Reone := one; Reother := other;
    i := 1;
    Imone := one; Imother := other;
    res := Reone × Imother + Imone × Reother;
    i := 0;
    res := Reone × Reother − Imone × Imother
end cm;

procedure cd (res, one, other);
real res, one, other;
comment serves to execute "res := one/other" for complex numbers;
begin
    real Reone, Imone, Reother, Imother, denom;
    i := 0;
    Reone := one; Reother := other;
    i := 1;
    Imone := one; Imother := other;
    denom := Reother × Reother + Imother × Imother;
    res := (Imone × Reother − Reone × Imother)/denom;
    i := 0;
    res := (Reone × Reother + Imone × Imother)/denom
end cd;

real procedure real (variable);
real variable;
```

```
real := (if i = 0 then variable else 0);
real procedure mod (it);
real it;
comment serves to compute the modulus of a complex number it;
begin
      real Reit, Imit;
      i := 0; Reit := it;
      i := 1; Imit := it;
      mod := sqrt (Reit × Reit + Imit × Imit)
end mod;

procedure polar form (res, r, theta);
real r, theta, res;
comment serves to compute real and imaginary part of a complex
        number res, given the modulus r and argument theta;
begin
      real r1, theta1;
      r1 := r; theta1 := theta;
      i := 0;
      res := r1 × cos (theta1);
      i := 1;
      res := r1 × sin (theta1)
end polar form;

procedure comprecip (res, it);
real res, it;
comment serves to compute the reciprocal res of a complex number it;
begin
      real Reit, Imit, denom;
      i := 0;
      Reit := it;
      i := 1;
      Imit := it;
      denom := Reit × Reit + Imit × Imit;
      res := − Imit/denom;
      i := 0; res := Reit/denom
end comprecip;

real procedure arg (it);
      real it;
comment serves to compute the argument of a complex number it;
begin
      real Reit, Imit;
      i := 0;
      Reit := it;
      i := 1;
      Imit := it;
```

```
        arg := (if Reit > 0 then arctan (Imit/Reit) else
                if Imit = 0 then 3.14159265358979 else
                sign (Imit) × 1.5707963267949 − arctan (Reit/Imit))
end arg;

procedure compsqrt (res, it);
real res, it;
comment serves to compute the square root res of a complex number it;
polar form (res, sqrt (mod (it)), 0.5 × arg (it));

procedure compprint (it);
real it;
comment prints a complex number it;
for i := 0, 1 do print (it);

procedure druck (it);
real it;
comment prints the complex number it and its modulus;
begin
      compprint (it);
      print (mod (it))
end druck;

boolean procedure even (integer);
integer integer;
comment the value of even is true if integer is even,
                                         false if integer is odd;
even := (integer = 2 × entier (integer/2));

procedure cma (res, one, other, it);
real res, one, other, it;
comment serves to execute "res := one × other + it" for complex
            numbers;
begin
      array aux4 [0 : 1];
      cm (aux4 [i], one, other);
      eq (res, aux4 [i] + it)
end cma;

procedure convfac (res, un);
real res, un;
comment serves to execute
res := (− un × A [0, S] + A [0, Sdash])/(− un × A [1, S] + A [1, Sdash])
A [j, S] being the complex number given by the array A [j, S, i];
begin
      for j := 0, 1 do
      cma (aux3 [j, i], − un, A [j, S, i], A [j, Sdash, i]);
      cd (res, aux3 [0, i], aux3 [1, i])
end convfac;
```

```
            procedure add in backward difference;
            cma (sigma [i, s], − am1 [i], aux1 [i], sigma [i, s]);

    procedure sum and display converging factor;
    begin
        NLCR;
        druck (alphar [− 2, i]);
        druck (termr [− 2, i]);
        eq (converging factor [i], converging factor [i] + termr [− 2, i]);
        for s := − 2, − 1 do
        begin
            eq (alphar [s, i], alphar [s + 1, i]);
            eq (termr [s, i], termr [s + 1, i]);
            modtermr [s] := modtermr [s + 1]
        end s
    end sum and display converging factor;

    procedure NT;
    begin
        NLCR; NLCR;
        TAB; TAB; TAB
    end NT;

    integer procedure mf (m1, m2);
    value m1; integer m1, m2;
    mf := ((m1 + 1) × (m1 + 2)) ÷ 2 + m2;
Introduction:
    a := read; b := read; rho := read;
    multiple of pi := read; factor := read;
    sign of sqrt := read; col := read;
    n := entier (rho/factor);
    h := rho/factor − n;
    NLCR;
    print (a); print (b);
    print (rho); print (multiple of pi);
    NLCR;
    print (factor); print (n);
    print (h); print (sign of sqrt);
    theta := multiple of pi × 3.14159 26535 89793;
    polar form (z [i], rho, theta);
    polar form (c [i], factor, theta);
Prepare application of converging factor:
    seqeq (A [0, 0, i], A [1, 1, i], real (1));
    eq (A [1, 0, i], z [i] + real (a + b + 1));
    eq (A [0, 1, i], 0);
    S := 1; Sdash := 0;
    for s := 1 step 1 until n − 1 do
```

```
begin
        for j := 0, 1 do
        cma(A[j, S, i], z[i] + real(a + b + 2×s + 1), A[j, Sdash, i],
                                        − (a + s)×(b + s)×A[j, S, i]);
        Sdash := S; S := 1 − Sdash
end s;
for j := 0, 1 do
cma(aux3[j, i], z[i] + real(a + b + 2×n + 1), A[j, Sdash, i],
                                        − (a + n)×(b + n)×A[j, S, i]);
Print nth convergent:
cd(aux1[i], aux3[0, i], aux3[1, i]);
NLCR; NLCR;
druck(aux1[i]);
comment Computation of eta, am1 (i.e. alpha[−1]),
alpha[1 and 2], beta[1 and 2];
cm(aux1[i], c[i], c[i] + real(4));
compsqrt(aux1[i], aux1[i]);
eq(eta[i], sign of sqrt×aux1[i]);
eq(am1[i], (c[i] + real(2) − eta[i])/2);
cm(aux1[i], − am1[i], c[i]);
cd(alpha[i, 2], aux1[i], eta[i]);
cma(aux1[i], − am1[i], real(a + b − 1) − c[i] + alpha[i, 2], real(a + b − 1));
cd(alpha[i, 1], aux1[i], eta[i]);
eq(beta[i, 1], alpha[i, 2] − alpha[i, 1] − am1[i] + real(a + b + 1));
eq(beta[i, 2], c[i] − alpha[i, 2]);
seqeq(sigma[i, 0], sigma[i, 1], 0);
for r := 1 step 1 until rmax do
begin
        eq(sigma[i, r + 1], 0);

Form cross products and accumulate:
        for s := 0 step 1 until r − 1 do
        for u := 0 step 1 until r + 1 do
        for v := 0 step 1 until s + 1 do
        if (u ≧ v) ∧ (u − v ≦ r − s) then
        cma(sigma[i, u], beta[i, mf(s, v)], alpha[i, mf(r − s − 1, u − v)],
                                                        sigma[i, u]);
        comment Determination of a[r, s] and b[r, s];
        for s := r + 1 step − 1 until 0 do
        begin
                eq(sum[i], 0);
                ncr := 1;
                for u := s + 1 step 1 until r + 1 do
                begin
                        ncr := − (ncr×u) ÷ (u − s);
                        eq(sum[i], sum[i] + ncr×alpha[i, mf(r, u)])
                end;
```

```
            cma (sigma [i, s], am1 [i], sum [i], − sigma [i, s]);
            if s = 0 ∧ r = 1 then
                    eq (sigma [i, s], sigma [i, s] + real (a × b));
            cd (alpha [i, mf (r, s)], sigma [i, s], eta [i]);
            if r ≠ rmax then
Differencing and adding through line of backward differences:
            begin
                for u := 0 step 1 until r − 1 do
                begin
                    if u = 0 then
                    begin
                        eq (aux1 [i], − alpha [i, mf (r, s)] − sum [i]);
                        eq (sigma [i, s], 0)
                    end
                else
                    begin
                        eq (aux0 [i], aux1 [i] − (if s ≠ r + 1 then
                                              d [u − 1, s, i] else 0));
                        eq (d [u − 1, s, i], aux1 [i]);
                        add in backward difference;
                        eq (aux1 [i], aux0 [i])
                    end;
                    if u = r − 1 then
                    begin
                        seqeq (d [u, s, i], beta [i, mf (r, s) ], aux1 [i]);
                        add in backward difference
                    end
                end u
            end Differencing and adding
        end s
    end r;

Computation of converging factor:
    still converging := true ;
    seqeq (l [0, i], converging factor [i], 0);
    power of n := 1/n;
    twormax := 2 × rmax;
    for r := − 1 step 1 until rmax do
    begin
        r1 := (if r > 0 then 0 else r − 1);
        if r = − 1 then eq (alphar [r1, i], am1 [i]) else
        begin
            eq (alphar [r1, i], 0);
            for s := r + 1 step − 1 until 0 do
            eq (alphar [r1, i], alpha [i, mf (r, s)] + h × alphar [r1, i])
        end;
```

```
        eq (termr [r1, i], alphar [r1, i]/power of n);
        modtermr [r1] := mod (termr [r1, i]);

Add in converging factor term if series still converging:
        if r ≧ 1 ∧ still converging then
        begin
                if modtermr [−2] > modtermr [−1] ∧
                                        modtermr [−1] > modtermr [0] then
                sum and display converging factor else
                still converging := false
        end;

Application of epsilon algorithm to converging factor series:
        eq (aux1 [i], termr [r1, i] + l [0, i]);
        for s := 0 step 1 until r + 1 do
        begin
                comprecip (aux0 [i], (if s = 0 then termr [r1, i] else
                                                        aux1 [i] − l [s, i]));
                if s ≠ 0 then
                begin
                        eq (aux0 [i], aux0 [i] + l [s − 1, i]);
                        eq (l [s − 1, i], aux2 [i])
                end;
                eq (aux2 [i], aux1 [i]);
                eq (aux1 [i], aux0 [i]);
                if even (s) then
                begin
                        rs := (s × (twormax + 4 − s)) ÷ 4 + r + 2;
                        eq (di [0, rs, i], aux2 [i]);
                        convfac (di [1, rs, i], aux2 [i])
                end;
                if s = r + 1 ∧ even (r) then
                begin
                        rs := ((r + 2) × (twormax − r + 6)) ÷ 4;
                        eq (di [0, rs, i], aux1 [i]);
                        convfac (di [1, rs, i], aux1 [i])
                end
        end s;
        eq (l [r + 1, i], aux2 [i]);
        eq (l [r + 2, i], aux1 [i]);
        power of n := n × power of n
end r;
if still converging ∧ modtermr [−1] ≦ modtermr [0] then
begin
        sum and display converging factor;
        sum and display converging factor
end;
```

```
Print converging factor and modified convergent:
    NT;
    druck (converging factor [i]);
    convfac (aux0 [i], converging factor [i]);
    NT;
    druck (aux0 [i]);

Display application of epsilon algorithm to converging factor and the corresponding
modified convergents:
    display converging factor alone := true;
Triangular display:
    for i := 0, 1 do
    begin
        for sanfang := 0 step 2×col until rmax + 2 do
        begin
            NLCR;
            for r := 1 step 1 until rmax + 2 − sanfang ÷ 2 do
            begin
                NLCR;
                for s := sanfang step 2 until
                                sanfang + 2 × (col − 1) do
                begin
                    if s ÷ 2 ≦ r ∧ r ≦ rmax + 2 − s ÷ 2 then
                    begin
                        rs := (s × (twormax + 6 − s)) ÷ 4 + r;
                        print (di [if display converging factor
                                alone then 0 else 1, rs, i]);
                    end
                end s
            end r
        end sanfang
    end i;
    if display converging factor alone then
    begin
        display converging factor alone := false;
        goto Triangular display
    end
  end
end Converging factor for continued fractions
```

Numerical results

Some numerical results which have been produced by means of the preceding ALGOL programme are summarized in the following tables which relate to the application of the converging factor $u_n^{(1)}$ to the continued fraction (21) when $a=b=0$, $|c|=1.0$, and $z=3.5\,e^{i3\pi/4}$ (i.e. $n=3$, $h=0.5$).

Firstly, we have

$$C_3 = -0.152029526 - i0.280947592.$$

Table 1 gives the values of the coefficients $\alpha_r(h)$ and the terms $\alpha_r(h)\,n^{-r}$, the numerical sum of the converging factor series, and the modified convergent $C_n^{(1)}$ to be obtained by its use.

Table 1

r	$\mathrm{Re}\{\alpha_r(h)\}$	$\mathrm{Im}\{\alpha_r(h)\}$	$\lvert\alpha_r(h)\rvert$	$\mathrm{Re}\{\alpha_r(h)\,n^{-r}\}$	$\mathrm{Im}\{\alpha_r(h)\,n^{-r}\}$	$\lvert\alpha_r(h)\,n^{-r}\rvert$
−1	+0.386752	−0.526531	0.653308	+1.160255	−1.579592	1.959925
0	−0.255823	+0.152993	.298081	−0.255823	+0.152993	0.298081
1	+0.118291	−0.005702	.118428	+0.039430	−0.001901	.039476
2	−0.022164	−0.037616	.043660	−0.002463	−0.004180	.004851
3	−0.030233	+0.033586	.045189	−0.001120	+0.001243	.001674
4	+0.039731	+0.000033	.039731	+0.000491	+0.000001	0.000491
			$u_3^{(1)}$	+0.940770	−1.431436	
			$C_3^{(1)}$	−0.150410854	−i0.279886159	

Table 2 gives the real and imaginary parts respectively of these modified convergents which are to be derived by applying the ε-algorithm to the converging factor series, and using the members of the resulting even column ε-array as approximations to the converging factor

Table 2

m \ s	0	2	4	6
1	−0.150792787	−0.150415488		
2	.150339170	.150408649	−0.150410661	
3	.150417729	.150411452	.150410739	−0.150410704
4	.150411824	.150410882	−0.150410067	
5	.150409872	−0.150410650		
6	−0.150410854			

m \ s	0	2	4	6
1	−0.279527494	−0.279880032		
2	.279903761	.279888879	−0.279886109	
3	.279891318	.279884459	.279885802	−0.279885921
4	.279883523	.279886629	−0.279886032	
5	.279886271	−0.279886604		
6	−0.279886158			

The correct value of the continued fraction in question is

$$-0.150410705 - i0.279885923.$$

In order to illustrate the effect of $\arg(z)$ upon the numerical behaviour of the converging factor certain figures are given in Table 3. These relate to the case $a=b=0$, $|c|=1.0$, $|z|=3.5$, i.e. $n=3$, $h=0.5$. The value of $\arg(z)$ is given in the first column. The second and third columns contain the moduli of $\alpha_{-1}(h)$

and $\alpha_4(h)$ respectively; the fourth and fifth columns contain $|C_3|$ and $|C_3^{(1)}|$ respectively ($C_3^{(1)}$ has been computed on the assumption that $u_n^{(1)}$ may be approximated by the partial sum $\sum_{r=-1}^{4} \alpha_r(h)\, n^{-r}$); the sixth column contains the value of $C_n^{(1)}$ which has been computed by applying the ε-algorithm to the initial values $\varepsilon_0^{(m)} = \sum_{r=-1}^{m-2} \alpha_r(h)\, n^{-r}$ $(m=0, 1, \ldots, 6)$ and using $\varepsilon_6^{(0)}$ as an approximation to $u_n^{(1)}$; the seventh column contains the modulus of the correct result

Table 3

$\arg(z)$	$\lvert\alpha_{-1}(h)\rvert$	$\lvert\alpha_4(h)\rvert$	$\lvert C_3\rvert$	$\lvert C_3^{(1)}\rvert$ (series)	$C_3^{(1)}$ (ε-alg.)	correct
0	0.381 966	0.017 307	0.230 803 934	0.230 819 326	0.230 819 332	0.230 819 332
$\pi/4$	0.403 861	0.012 924	0.238 593 791	0.238 569 606	0.238 569 603	0.238 569 603
$\pi/2$	0.480 533	0.046 751	0.264 186 360	0.264 289 222	0.264 289 208	0.264 289 208
$3\pi/4$	0.653 308	0.039 731	0.319 444 080	0.317 741 541	0.317 741 260	0.317 741 263
π	1.0	0.125 210	0.355 963 303	0.431 104 196	0.431 077 928	0.431 077 657

(Note: When $z=-x$, and x is real and positive, then $C_n^{(1)}$ is an approximation to $e^{-x}\left\{\gamma+\ln(x)+\sum_{n=1}^{\infty}\frac{x^n}{n\,(n!)}+i\pi\right\}$, as one would expect.)

It will be seen that both the rate of convergence of the converging factor series and the degree of improvement which may be effected by application of the ε-algorithm, are substantially independent of $\arg(z)$. The variation in the relative accuracy of the transformed convergent $C_3^{(1)}$ is mainly influenced by the relative accuracy of the convergent C_3, i.e. by the convergence behaviour of the continued fraction itself.

It will be recalled that in the relationship $z=c(n+h)$, the choice of $|c|$ was arbitrary, but that thereafter all other parameters were fixed. In the preceding numerical examples $|c|$ was taken to be 1.0 for simplicity. The effect of $|c|$ upon the numerical behaviour of the converging factor is illustrated in Table 4, which refers to the case $a=b=0$, $z=3.0$.

Table 4

c	n	$\lvert\alpha_{-1}(h)\rvert$	$\lvert\alpha_4(h)\rvert$	$\lvert\alpha_{-1}(h)\, n\rvert$	$\lvert\alpha_4(h)\, n^{-4}\rvert$	$\lvert C_n\rvert$	$\lvert C_n^{(1)}\rvert$
0.5	5	0.5	0.131 923	2.5	0.000 211	0.262 081 881	0.262 083 740 038
1.0	2	0.381 966	0.076 374	0.763 932	0.004 773	0.261 904 762	0.262 079 998 123
2.0	1	0.267 949	0.007 856	0.267 949	0.007 856	0.260 869 565	0.261 877 638 010
						correct	0.262 083 740 038

The value of $C_n^{(1)}$ has been computed by using as an approximation

$$u_n^{(1)} = \sum_{r=-1}^{4} \alpha_r(h)\, n^{-r}.$$

It can be seen that the magnitude of $|\alpha_r(h)|$ decreases more rapidly, the larger $|c|$ becomes. However, the fact that a small value of $|c|$ implies a relatively large value of n, means that the converging factor series converges more rapidly for such values of c. Furthermore, the large value of n implies that C_n itself is more accurate. Thus, in conclusion, a value of c for which $|c|$ is small is to be preferred.

For the sake of completeness we give some numerical details relating to a case in which both a and b are not zero, namely that in which $a=0.0$, $b=-0.5$, $z=5.0e^{i\pi/2}$, $|c|=1.0$, (i.e. $n=4$, $h=1.0$).

Here

$$C_4=0.01793\,70833 \quad -i0.19523\,243250.$$

Table 5 gives even order ε-arrays corresponding to those in Table 2.

Table 5

m \ s	0	2	4
1	+0.01793 50965 3	+0.01793 67363 0	
2	.01793 71180 1	.01793 69166 0	+0.01793 69173 0
3	.01793 69043 5	.01793 69173 4	+0.01793 69170 9
4	.01793 69166 8	+0.01793 69169 9	
5	+0.01793 69172 3		

m \ s	0	2	4
1	−0.19523 10888 0	−0.19523 10575 7	
2	.19523 08984 1	.19523 10526 6	−0.19523 10546 6
3	.19523 10865 6	.19523 10546 4	−0.19523 10542 3
4	.19523 10500 5	−0.19523 10541 3	
5	−0.19523 10545 5		

The correct value of the continued fraction in question is

$$0.01793\,69171\,0-i0.19523\,10542\,2.$$

Acknowledgement. The numerical results of this paper were produced on the X 1 computer in Amsterdam using an ALGOL translator constructed by E. W. DIJKSTRA and J. A. ZONNEVELD.

References

[1] WYNN, P.: Converging Factors for Continued Fractions. Num. Math. **1**, 272 (1959).
[2] — On a Device for Computing the $e_m(S_n)$ Transformation. MTAC **10**, 91 (1956).
[3] — The Rational Approximation of Functions which are Formally Defined by a Power Series Expansion. Maths. of Comp. **14**, 147 (1960).
[4] BACKUS, J. W., F. L. BAUER, J. GREEN, C. KATZ, J. MCCARTHY, P. NAUR (editor), A. J. PERLIS, H. RUTISHAUSER, K. SAMELSON, B. VAUQUOIS, J. H. WEGSTEIN, A. VAN WIJNGAARDEN and M. WOODGER: Report on the Algorithmic language ALGOL 60. Num. Math. **2**, 106 (1960).
[5] WYNN, P.: An Arsenal of ALGOL Procedures for Complex Arithmetic. BIT **2**, 232 (1962).

Stichting Mathematisch Centrum
2e Boerhaavestraat 49
Amsterdam

(Received February 5, 1963)