be taken into account in any practical example where some upper bound for the error should be estimated.

HERBERT E. SALZER

Ordnance Corps
Diamond Ordnance Fuze Laboratories
Washington, D. C.

NOTE BY REFEREE

The function $F(p)$ is subject to certain restrictions because it is a Laplace transform. In order for $F(p)$ to be the Laplace transform of the function $f(t)$ given by (1), it is sufficient that $F(p)$ have the form (cf. G. DOETSCH [5]):

$$F(p) = a/p + F_1(p)/p^{1+\delta},$$

where $\delta > 0$, $a$ is a constant, and $F_1(p)$ is analytic and bounded in the half plane $\mathrm{Re}(p) > c$. We assume that this condition is satisfied. Whether this condition is also sufficient for the convergence of the $n$-point quadrature formula to the true value of $f(t)$ in (1), when $n$ tends to infinity, has not been determined. The author makes use here of the fact that the convergence occurs whenever $F(p)$ is a polynomial in $1/p$ without a constant term; in fact, the quadrature is exact for polynomials of degree not greater than $2n$. G. SZEGÖ [10] has shown that under quite general conditions a Gauss-Jacobi type quadrature formula which converges for polynomials also converges for a much wider class of functions. Unfortunately his theorems do not seem to apply directly to the present case because the integral (1) involves a complex valued weight function which is not of bounded variation.

1. H. S. CARSLAW & J. C. JAEGER, *Operational Methods in Applied Mathematics*, 2nd edition, Oxford University Press, 1949, p. 75.
2. H. E. SALZER & R. ZUCKER, "Table of the Zeros and Weight Factors of the First Fifteen Laguerre Polynomials," Amer. Math. Soc., *Bull.*, v. 55, 1949, p. 1004–1012.
3. G. SZEGÖ, *Orthogonal Polynomials*, Amer. Math. Soc., *Colloquium Pub.*, v. 23, 1939, p. 46-47.
4. H. L. KRALL & O. FRINK, "A New Class of Orthogonal Polynomials: The Bessel Polynomials," Amer. Math. Soc., *Trans.*, v. 65, 1, 1949, p. 100–115.
5. G. DOETSCH, *Theorie und Anwendung der Laplace-Transformation*, Springer, Berlin, 1937, p. 128.
6. The shift in notation from $(n+1)$ to $n$ in $A_i^{(n)}$ will cause no confusion after the $A_i$'s have been computed and are ready for use in (6).
7. It was called to the author's attention by H. L. KRALL that $P_n(x) \equiv (-1)^n y_n(x, 1, -1)$ where $y_n(x, a, b)$ are "generalized Bessel polynomials" (see [4]).
8. G. SZEGÖ, *op. cit.*, p. 41–42.
9. Formula (14) holds for $n = 2$ if we define $P_0(x) \equiv 1$.
10. G. SZEGÖ, *op. cit.*, p. 341–342.

# On the Improvement of the Solutions to a Set of Simultaneous Linear Equations using the ILLIAC

The basic method used for solving simultaneous linear equations on the University of Illinois' electronic digital computer, the ILLIAC, has already been described in detail by WHEELER and NASH [1]. The routine currently in use on the ILLIAC, programmed by Wheeler [2], makes use of the method of elimination to solve the set of $n$ simultaneous linear equations

$$\sum_{j=0}^{n-1} a_{ij}x_j + a_{in} = 0 \quad i = 0, 1, 2, \cdots, n-1 \tag{1}$$

in a manner very similar to that used by a human solving such a system.

In brief, the procedure used is as follows:

a) The augmented matrix

$$a_{ij} \qquad \begin{matrix} i = 0, 1, 2, \cdots, n-1 \\ j = 0, 1, 2, \cdots, n \end{matrix} \tag{2}$$

is scaled (by reducing each row by an appropriate factor) and punched as decimals with up to 12 places, row by row, on the input tape; each row is followed by an identifying symbol.

b) The computer reads this matrix a row at a time, converts the numbers to binary form, and eliminates an additional variable from each additional row by using the previous rows which are stored in the memory of the machine. When this process is completed the matrix (2) is stored in ILLIAC as the triangular matrix

$$c_{ij} \qquad \begin{matrix} i = 0, 1, 2, \cdots, n-1 \\ j = 0, 1, 2, \cdots, n \end{matrix} \tag{3}$$

with $c_{ij} = 0$ for $0 \leq j < i \leq n - 1$.

c) The solutions are obtained by a process of back substitution. Due to the fact that ILLIAC can represent numbers only in the range $-1$ to 1, provision must be made for automatic scaling of the results. If bars indicate the ILLIAC representation of a number ($-1 \leq \bar{x} < 1$), and if $\bar{x}_n = 10^{-p}$ represents a scale factor, then the scaled solutions are formed during the back substitution by successively applying

$$\bar{x}_i = -\sum_{j=i+1}^{n} \frac{c_{ij}\bar{x}_j}{c_{ii}} \quad i = n-1, n-2, \cdots, 1, 0. \tag{4}$$

The true solutions are given in terms of the scaled solutions by

$$x_i = \frac{\bar{x}_i}{\bar{x}_n} = \frac{\bar{x}_i}{10^{-p}} \quad i = 0, 1, 2, \cdots, n-1. \tag{5}$$

The program first attempts to carry out the back substitution with $p = 1$. If at any stage the sum in (4) exceeds scale, $p$ is increased by unity and the process is repeated. After the back substitution has been successfully completed the solutions $\bar{x}_i$ are printed in columnar form followed by $\bar{x}_n$ which serves to locate the decimal point in the column. Thus the machine actually operates on the set of equations

$$\sum_{j=0}^{n-1} a_{ij}\bar{x}_j + a_{in}\bar{x}_n = 0 \quad i = 0, 1, 2, \cdots, n-1 \tag{6}$$

whose solutions can be forced to remain within scale by proper adjustment of $\bar{x}_n$.

This method is extremely straightforward and is in fact quite similar to that which would be used to solve such a set of equations "by hand". The chief utility of such a process applied to an automatic computing machine is that it minimizes the space necessary to store the matrix of coefficients in the equations. Any method that demands storage of the entire matrix $a_{ij}$ requires $n(n+1) \sim n^2$ memory locations. This elimination method uses the input tape as auxiliary storage and forms the reduced triangular matrix $c_{ij}$ as this tape is read in and thus requires only $\frac{1}{2}n(n+3) \sim \frac{1}{2}n^2$ memory locations.

If the set of equations being solved is poorly conditioned in having a nearly

singular determinant,

$$\text{Det } (a_{ij}) \text{ small} \quad i, j = 0, 1, 2, \cdots, n-1, \tag{7}$$

the above outlined process can yield solutions which are in error compared to the true solutions. Such errors can be aggravated by rounding-off in the many arithmetical steps of the process and by the truncation errors due to the finite register length of 39 binary digits. In order to obtain an improved set of solutions in such circumstances an iterative application of the method of residues has been programmed [3] for the ILLIAC. If $x'_i$ indicate the solutions to the equations (1) obtained by the ILLIAC and if corrections $\epsilon_i$ are defined by

$$x_i \equiv x'_i + \epsilon_i \tag{8}$$

where the $x_i$ are the actual solutions of (1), then the corrections can be found from

$$\sum_{j=0}^{n-1} a_{ij}\epsilon_j + \delta_i = 0 \quad i = 0, 1, 2, \cdots, n-1 \tag{9}$$

where $\delta_i$ are the residues defined by

$$\delta_i \equiv \sum_{j=0}^{n-1} a_{ij}x'_j + a_{in} \quad i = 0, 1, 2, \cdots, n-1. \tag{10}$$

Just as the original solutions were in error due to the finite register length and the rounding-off errors in the arithmetic processes so are the $\epsilon_i$ in error. It could not be expected that a single application of equations (9) to find the corrections $\epsilon_i$ would yield the correct solutions to (1) by the use of (8).

Hence, it has proved useful to apply the method of residues several times and also artificially to increase the numerical size of a residue contained in the machine by multiplying the residue by a factor $B = 2^s$ in order to gain numerical accuracy in the solution of (9). Let superscripts zero designate the solutions obtained by the conventional solution of the equation (1) as outlined in the above equations (3)–(5). Then if $x_i^{(k)}$ represent the solutions at the $k^{\text{th}}$ stage of the iterative process, the residues at this stage are defined by:

$$\delta_i^{(k)}x_n^{(0)} \equiv \sum_{j=0}^{n-1} a_{ij}x_j^{(k)} + a_{in}x_n^{(0)} \quad i = 0, 1, 2, \cdots, n-1, \tag{11}$$

where ILLIAC representations of the numbers are implied even though the bars have been dropped. The scale factor $x_n^{(0)}$ as determined by the first solution of equations (6) is maintained throughout the process. The corrections $\epsilon_i^{(k)}$ at the $k^{\text{th}}$ stage are determined by dividing the result of (11) by $x_n^{(0)}$ to form $\delta_i$ and solving the equations:

$$\sum_{j=0}^{n-1} a_{ij}x_j^{(k)} + B\delta_i^{(k)} = 0 \quad i = 0, 1, 2, \cdots, n-1. \tag{12}$$

The sum in (11) is formed by double precision continued multiplication which is a facility readily available on the ILLIAC. Hence the $\delta_i^{(k)}$ are capable of accepting a multiplication by the scaling up factor $B$ and still retain a register of significance.

The equations (12) are identical in form to equations (1) and can be solved in the same fashion. The $(k+1)^{\text{th}}$ approximation to the solutions are:

$$x_i^{(k+1)} = x_i^{(k)} + B^{-1}\epsilon_i^{(k)} \frac{x_n^{(0)}}{x_n^{(k)}} \quad i = 0, 1, 2, \cdots, n-1 \tag{13}$$

where $x^{(k)}$ is the automatic scale factor supplied by ILLIAC in the solution of (12) which guarantees that each iterate is expressed in the same scale.

To use Library Routine 100 the scaled coefficients of the augmented matrix are placed upon the input tape just as described in the use of Library Routine 51. To produce the $x_i^{(0)}$ the internal operation of the routine is identical to steps (a)–(c) outlined above. On each subsequent reading of the input tape the routine computes the residues (11) as each row of the input tape is read, solves the equations (12), forms the improved solutions (13), and prints them. At the $k^{\text{th}}$ stage the factor $B$ is set to be

$$B = 2^{6+4k} \quad k = 1, 2, \cdots. \tag{14}$$

If at any stage, multiplication by such a factor would cause the expression on the left hand side of equation (11) to exceed scale, the machine stops, replaces $B$ by $B' = B2^{-2}$ and puts itself into such a condition that the reading of the coefficient tape for the $k^{\text{th}}$ stage can be repeated. If the equations are so badly conditioned that equation (11) will accept multiplication by no $B > 1$ without exceeding scale, this program is to no avail; the machine stops after producing a characteristic indicating signal. If the problem has been removed from the machine after the $k^{\text{th}}$ approximations to the solutions have been obtained it is unnecessary to repeat the process from the beginning if at a later time further iterations are desired. Provisions have been made so that the program, after being placed in ILLIAC, will first read in the $x_i^{(k)}$ from some previous iteration and store them in such a way that the condition of the machine is the same as if the iterative process had been continuing and had reached the $k^{\text{th}}$ stage. Subsequent iterations can hence be resumed from this stage.

This method of annihilation of residues was chosen and used in ILLIAC to minimize storage requirements. By making the arithmetic procedures during subsequent iterations as nearly like those used on the first solution, the length of the operating routine stored in the memory has been reduced since most of it serves double duty. Again, only a triangular matrix need be stored at each iteration stage. This economy of memory space has however been achieved by the sacrifice of some speed since the tape of coefficients must be read on each iteration and the triangular matrix must be reformed at each iteration. The total tape reading time is at most $\frac{1}{30}kn^2$ sec. where $k$ is the number of iterations necessary; the total computation time is $\frac{1}{500}kn^3$ sec., which is essentially dominated by the $n^3$ multiplications necessary to form the reduced triangular matrix. This program was designed to use only high speed storage so that even with the increased efficiency of storage yielded by this procedure $n$ is still limited to moderate values so that the above times are not excessive in spite of the $k$ repetitions.

This method would be undesirable if enough storage were available in order to allow $n$ to become large (order of 100, say) especially if that storage were rela-

tively slow (for example, a drum). In such a case the entire matrix should be read and stored or processed but once (reading time proportional to $n^2$ not $kn^2$) and some iteration scheme requiring only $n^2$ multiplications per iteration used (computation time proportional to $kn^2$ not $kn^3$). This gain in computation speed is not quite so favorable as it may appear due to the greatly increased access time to the slower memory.

The ILLIAC's memory is sufficient to accommodate a system of 39 equations when used with Routine 51. The additional length of Routine 100 restricts to 37 the number of equations that it can handle. With 37 equations the operation time of Routine 100 is about 4 minutes per iteration. For fewer equations the operation time is very roughly proportional to the square of the number of equations.

The utility of this residue annihilating routine may be illustrated by an example comparing the results obtained with Routines 51 and 100 for sets of equations purposely constructed to be poorly condition to various degrees.

The set of equations:

$$(15) \qquad (1 - 10^{-q}) \begin{aligned} x_1 - x_2 + x_3 - x_4 + 3 &= 0 \\ x_1 - x_2 + x_3 - x_4 + 2 &= 0 \\ x_1 - x_2 \qquad\quad + 1 &= 0 \\ x_1 \qquad\quad - x_4 + 3 &= 0 \end{aligned}$$

has been constructed to have

$$(16) \qquad \begin{aligned} |\mathrm{Det}\ (a_{ij})\ 1 &= 10^{-q} \\ x_1 &= 10^q \\ x_2 &= 10^q + 1 \\ x_3 &= 10^q + 2 \\ x_4 &= 10^q + 3. \end{aligned}$$

These equations were solved to eleven decimal digits for $1 \leq q \leq 9$ by both Routines 51 and 100 as indicated in parentheses with the results:

$q = 1$

| $x_i$(51) | $x_i^{(0)}$(100) | $x_i^{(1)}$(100) |
|---|---|---|
| 10.000 000 05 | 10.000 000 004 | 10.000 000 000 |
| 11.000 000 02 | 11.000 000 000 | 11.000 000 000 |
| 12.000 000 06 | 12.000 000 004 | 12.000 000 000 |
| 13.000 000 04 | 13.000 000 002 | 13.000 000 000 |

$q = 2$

| $x_i$(51) | $x_i^{(0)}$(100) | $x_i^{(1)}$(100) |
|---|---|---|
| 100.000 000 20 | 100.000 000 20 | 100.000 000 00 |
| 101.000 000 16 | 101.000 000 16 | 101.000 000 00 |
| 102.000 000 20 | 102.000 000 20 | 102.000 000 00 |
| 103.000 000 18 | 103.000 000 18 | 103.000 000 00 |

$q = 3$

| $x_i$(51) | $x_i^{(0)}$(100) | $x_i^{(1)}$(100) |
|---|---|---|
| 1 000.000 000 1 | 1 000.000 000 2 | 0 999.999 981 8 |
| 1 000.999 999 8 | 1 000.999 999 8 | 1 000.999 981 8 |
| 1 002.000 000 2 | 1 002.000 000 2 | 1 001.999 981 7 |
| 1 003.000 000 0 | 1 003.000 000 0 | 1 002.999 981 8 |

$q = 4$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ | $x_i^{(2)}(100)$ |
|---|---|---|---|
| 10 000.001 820 | 10 000.001 821 | 10 000.000 000 | 10 000.000 000 |
| 10 001.001 817 | 10 001.001 817 | 10 001.000 000 | 10 001.000 000 |
| 10 002.001 821 | 10 002.001 821 | 10 001.999 999 | 10 002.000 000 |
| 10 003.001 819 | 10 003.001 819 | 10 002.999 999 | 10 003.000 000 |

$q = 5$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ |
|---|---|---|
| 100 000.181 91 | 100 000.181 92 | 100 000.000 00 |
| 100 001.181 88 | 100 001.181 88 | 100 001.000 00 |
| 100 002.181 92 | 100 002.181 92 | 100 002.000 00 |
| 100 003.181 90 | 100 003.181 90 | 100 003.000 00 |

$q = 6$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ | $x_i^{(2)}(100)$ |
|---|---|---|---|
| 1 000 018.190 2 | 1 000 018.190 3 | 0 999 999.999 7 | 1 000 000.000 0 |
| 1 000 019.189 9 | 1 000 019.189 9 | 1 000 000.999 7 | 1 000 001.000 0 |
| 1 000 020.190 3 | 1 000 020.190 3 | 1 000 001.999 7 | 1 000 002.000 0 |
| 1 000 021.190 1 | 1 000 021.190 1 | 1 000 002.999 7 | 1 000 003.000 0 |

$q = 7$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ | $x_i^{(2)}(100)$ |
|---|---|---|---|
| 10 001 819.176 | 10 001 819.176 | 09 999 999.669 | 10 000 000.000 |
| 10 001 820.172 | 10 001 820.173 | 10 000 000.669 | 10 000 001.000 |
| 10 001 821.176 | 10 001 821.176 | 10 000 001.669 | 10 000 002.000 |
| 10 001 822.174 | 10 001 822.175 | 10 000 002.669 | 10 000 003.000 |

$q = 8$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ | $x_i^{(2)}(100)$ | $x_i^{(3)}(100)$ |
|---|---|---|---|---|
| 100 182 149.38 | 100 182 149.38 | 099 999 668.22 | 100 000 000.60 | 100 000 000.00 |
| 100 182 150.35 | 100 182 150.35 | 099 999 669.22 | 100 000 001.61 | 100 000 001.00 |
| 100 182 151.39 | 100 182 151.38 | 099 999 670.22 | 100 000 002.61 | 100 000 002.00 |
| 100 182 152.36 | 100 182 152.37 | 099 999 671.22 | 100 000 003.61 | 100 000 003.00 |

$q = 9$

| $x_i(51)$ | $x_i^{(0)}(100)$ | $x_i^{(1)}(100)$ |
|---|---|---|
| 1 000 000 000.3 | 1 037 735 849.4 | 0 999 301 188.0 |
| 1 000 000 001.0 | 1 037 735 850.1 | 0 999 301 189.0 |
| 1 000 000 001.8 | 1 037 735 851.9 | 0 999 301 190.0 |
| 1 000 000 003.0 | 1 037 735 852.1 | 0 999 301 191.0 |

| $x_i^{(2)}(100)$ | $x_i^{(3)}(100)$ |
|---|---|
| 1 000 026 370.3 | 0 999 999 511.6 |
| 1 000 026 371.2 | 0 999 999 512.6 |
| 1 000 026 372.3 | 0 999 999 513.6 |
| 1 000 026 373.2 | 0 999 999 514.6 |

| $x_i^{(4)}(100)$ | $x_i^{(5)}(100)$ |
|---|---|
| 1 000 000 018.4 | 0 999 999 999.3 |
| 1 000 000 019.4 | 1 000 000 000.3 |
| 1 000 000 020.4 | 1 000 000 001.3 |
| 1 000 000 021.4 | 1 000 000 002.3 |

| $x_i^{(6)}(100)$ |
|---|
| 1 000 000 000.0 |
| 1 000 000 001.0 |
| 1 000 000 002.0 |
| 1 000 000 003.0 |

Several remarks about these results are in order. It is not to be expected that the zeroth order solutions of Routine 100 should agree with the solutions of Routine 51 since the conversion routine which reads the matrix into ILLIAC is slightly different in the two programs. The one in Routine 51 is accurate to $\pm 2^{-39}$, that in Routine 100 to $\pm 2^{-40}$. It is to be noted that as the equations become more poorly conditioned a greater number of iterations are necessary before Routine 100 converges to the correct answer.

The deterioration of the Routine 100 solution for $q = 3$ upon iteration and the already excellent Routine 51 solution for $q = 9$ demand especial attention. Both of these effects are traceable to the rounding off of the scaling factors to 39 binary digits in the machine. For the case $q$, a scaling factor $10^{-q-1}$ is used by both Routine 51 and Routine 100 for the first iteration in accordance with equation (5). In addition Routine 100 uses the scaling factor again on each subsequent iteration to form the residues and to reduce the resultant corrections to the uniform scale of the zeroth order solutions. The machine representation of the scaling factor was examined for each of the above cases. For $q = 3$, the scaling factor $10^{-4}$ must be rounded down by six parts in the 11th sexadecimal digit when the number is rounded to ILLIAC's register size of 10 sexadecimal digits. This is the largest rounding-off error encountered in the sequence. This accounts for the deterioration of Routine 100's first iterate since the correction is multiplied by this scaling factor before being added to the zeroth order solution. For $q = 9$, the scaling factor $10^{-10}$ when represented to 10 sexadecimal digits is also exact to 11 sexadecimal digits, thus accounting for the already excellent Routine 51 solution. Such effects would argue for the use of binary scaling factors during the carrying out of the back substitution. However, if the answers were presented with such factors they would not be immediately useful; if the answers were reconverted with the appropriate decimal scaling factors before presentation not only would additional memory space be required for such reconversion but the imprecision of these decimal scale factors would be introduced.

Recent changes in the ILLIAC have resulted in an increased efficiency in the order alteration portions of a program. Various routines of the Program Library are being rewritten to utilize this increased efficiency. When Routine 100 is rewritten, sufficient memory space will be gained so that extra orders can be added to the program which will allow the residues to be computed more accu-

rately. This should speed the convergence and should eliminate most of the effect of imprecision in the scaling factor.

JAMES N. SNYDER

University of Illinois
Urbana, Illinois

1. D. J. WHEELER & J. P. NASH, "Digital and Analogue Computers and Computing Methods." Symposium at the 18th Applied Mechanics Division Conference of the American Society of Mechanical Engineers, University of Minnesota, June 18–20, 1953.
2. D. J. WHEELER, *The Automatic Linear Equation Solver*, University of Illinois Computer Library Routine No. 51.
3. J. N. SNYDER, *The Complete Linear Equation Solver*, University of Illinois Computer Library Routine No. 100.

# The Use of Iterative Methods for finding the Latent Roots and Vectors of Matrices

In a recent note in *MTAC* E. BODEWIG [1] presented what he claimed was "a practical refutation of the iteration method for the algebraic eigenproblem." In my opinion this note gave an entirely misleading impression of the value of the iterative method. Moreover an example was chosen as the basis of this refutation which so far from serving the purpose for which it was used, is in fact quite well suited to the iterative method provided it is used in a flexible manner. The iterative method, supplemented by a number of simple devices for accelerating convergence, has been used very effectively on the Pilot ACE to find the latent roots and vectors of a very large number of matrices, symmetric and unsymmetric, real and complex, up to orders as high as 60. Very high accuracy has been achieved even when many or all of the latent vectors of a matrix have been wanted. The details of the method used have been described in a recent paper [2] by WILKINSON, but since that paper was written a magnetic drum store has been added to the Pilot ACE and the speed of iteration has thereby been considerably increased particularly for the larger matrices. The addition of the drum has also led to modifications in the details of the programme which have made it much more satisfactory to use. For this reason a general description of the programme is given below. The note includes an assessment of the value of iteration and concludes with the results achieved with it on Bodewig's example.

An understanding of the iterative programme will be aided by a description of the one or two facilities provided on the Pilot ACE, which are employed therein. There is a register, called the input register, which stores one of the standard words of 32 binary digits, into which a number may be inserted manually by means of 32 keys. The number thus inserted is displayed on a set of 32 lights and this number may be changed at any time during computation by the operator. The input register is addressable in the same way as all the other storage registers and the machine has access in 32 microseconds to the number held there, but it cannot send a word *to* the input register. There is a second register, the output register, the contents of which are also displayed on a set of 32 lights. The machine may send a number to the output register in 32 microseconds, but it cannot read the number stored in it. The machine is also equipped with a monitoring device on which are displayed the contents of 32 consecutive storage registers. The