Coding of a Laplace Boundary Value Problem for the UNIVAC

General Explanation of the UNIVAC System.—The UNIVAC [1] (Universal Automatic Computer) system includes a high-speed electronic digital computer and certain auxiliary devices. This system, which deals with numerical data in decimal form, and which also can handle alphabetic characters, has been designed as a general-purpose tool for scientific and commercial use. A salient characteristic of the system is its flexibility. In its design, particular attention was given to the needs of the Census Bureau, where sorting and collating of information play a predominant role; and a thorough investigation has demonstrated the suitability of the system for such applications.[2] However, the UNIVAC system is not limited to statistical applications and will be useful in performing complicated numerical computations underlying a vast body of scientific research. It is the purpose of this paper to give an example of the application of this system to the solution of the type of problem just mentioned.

All information to be used by the UNIVAC, which is the central computing unit of the system, is first recorded on magnetic tape. Such tape is prepared on a Unityper, which resembles a standard typewriter. One or more tapes are prepared; one usually contains instructions—another may carry numerical data peculiar to the functions needed in the course of the work. All tapes to be used in a problem are then put on input-output readers controlled by the UNIVAC. Some of the tapes are used to record output data; others may be used for temporary storage of data needed at various intermediate steps of the calculations. Recording of new data on a tape automatically erases any previous record in the interval of the tape being used.

When a problem is completed, either the final results are printed on a directly-connected typewriter or they are put on one or more of the tapes. The directly-connected typewriter is usually used for results comprising a very small amount of data. If the results are on tapes, they are inserted into a Uniprinter. The Uniprinter prints the results while another problem is being solved on the UNIVAC. Proper instructions are inserted in the output tapes to control the Uniprinter for tabs, decimal points, spaces, printing of alphabetic headings, etc. With this feature, the results can be arranged in almost any desired form.

The internal memory of the UNIVAC contains 1000 memory locations numbered from 0000 to 0999. Each memory location accommodates 12D digits. A memory location is usually filled by a signed 11-digit number or two instructions. An instruction is usually a letter (which occupies two digit spaces) followed by four digits, designating the memory location, e.g., B0101. A 12-digit group is referred to as a "word." When information is read from a memory location "m," this information still remains in "m" until new information is sent for replacing the old.

The tape record is partitioned into fixed lengths called "blocks," each of which contains 60 words. The first block is read into the first 60 memory locations in .072 second when a start button is pressed. Then the computer automatically goes to memory location 0000 for its first instruction. After the first instruction is executed, the next instruction comes from the succeeding memory location, unless a transfer instruction is given. Such a transfer interrupts this sequence and causes the next instruction to be drawn from a new memory location designated by the transfer command. Thereafter instructions are taken in sequence from successive locations until a new transfer is encountered.

All arithmetic operations are carried out in the Accumulator, A, of the UNIVAC and the associated registers and circuits. It should be noted that when a word is read into one of these associated registers, the previous contents of that register are erased. An abbreviated list from the UNIVAC instruction code,[1] including the description of only those instructions which are to be used in the solution of the Laplace boundary value problem treated later in this paper is appended below. In this list of instructions, m is a number designating a memory location; letters preceding m are operation symbols. In the description of each instruction, letters denote registers of the UNIVAC. The contents of a register are indicated by parentheses. Thus (m) signifies "word stored in m," (L) signifies "word stored in L," etc. The key mnemonic words are italicized in the description of the operation.

In denoting the different registers, A signifies accumulator. When cleared, this register contains all decimal zeros, even in the sign position. It has capacity for an 11-digit number and sign; a negative number is registered in it in the absolute-value form with a negative sign. The X register is a 60-pulse delay line with an extra 5-pulse delay which can be switched in for shifting purposes. It receives numbers which are to be sent to the accumulator and provides for the checking of the sign to effect algebraic addition of the number to the contents of the accumulator. The L register is a one-word register which contains the multiplicand during the multiplication process. It also holds the word for the T and Q comparison processes. The I register holds one block of words which has been read in from magnetic tape.

The above-mentioned list of instructions to be used in the subsequent discussion is as follows:

Instruction	Explanation of Instruction
Am	*Add* (m) to (A), result in A; (m) also left in X.
Bm	Clear (A), then put (m) in A; (m) also left in X.
Cm	Put (A) in m, *clear* A.
Hm	Put (A) in m without clearing A (i.e., *hold* (A) in A).
Km	Put (A) in L, clear A, disregard m.
Lm	Put (m) in L; (m) also left in X.
Mm	*Multiply* (m) and (L), rounding off the product to 11 digits and adding it to (A), result in A.
Qm	Transfer control to m if (A) = (L).
Sm	*Subtract* (m) from (A), result in A; −(m) also left in X.
Tm	Test to see if (A) is greater than (L); if so, transfer control to m.
Um	*Unconditional* transfer of control to m.
Xm	Add (X) to (A), result in A, disregard m; (X) unaltered.
00m	Pass on to next order without doing any arithmetic operation; disregard m.

Instruction	Explanation of Instruction
	Shift Orders
Δnm	*Shift* all digits of A, *including the sign,* n digits to the *left,* dropping the n left-hand digits; n ranges from 0 to 9; disregard m.
.nm	*Shift* all digits of A, *including the sign,* n digits to the *right,* dropping the n right-hand digits; disregard m.
	Tape Orders for 1 to 9 Tapes
1nm	*Read* one block of data (60 words) from tape n and store in I, tape running in *forward* direction; disregard m.
3nm	*Transfer data* (60 words) previously stored in I to 60 consecutive memory locations, beginning with m, where m is an integral multiple of 20; then *read* one block of data (60 words) from tape n and store in I, tape running in a *forward* direction.
5nm	*Write* 60 consecutive words, starting with m, where m is an integral multiple of 20, on tape n, tape moving in a *forward* direction.
4.m	Stop machine operations and produce a signal; disregard m.

Underlying Mathematical Considerations in the Numerical Solution of a Laplace Boundary Value Problem.—For purposes of simplicity, a two-dimensional potential problem will be considered. The iterative method used here for the solution of the plane potential problem is a finite-difference method originally proposed by LIEBMANN.[3] The UNIVAC is well adapted for the solution of all sorts of problems where iterative procedures are effective, and not only the plane potential problem, but many others of a similar nature are easily dealt with. When an automatic procedure has once been set up for a single equation, any number of iterations may be made according to the same routine; and, although the computing time will increase in direct proportion to the number of iterations to a good approximation, no further human effort is required. An important simplification in the instructions for a single iterative cycle can be made because of the fact that for all interior points of the lattice exactly the same sort of operations must be carried out. Consequently, after the coding applicable to one lattice point has been worked out, this coding may be automatically altered within the computer so as to apply to the next lattice point. The necessary alterations are systematic, and consequently the routine which accomplishes such alterations can easily be generalized so that all of the required alterations throughout the iterative cycle are accomplished properly.

The LAPLACE equation,

$$\partial^2 W/\partial x^2 + \partial^2 W/\partial y^2 = 0,$$

together with the values of $W(x, y)$ on a closed boundary in the xy plane, determines the function W at all points on the interior of the boundary. Here it will be assumed that the boundary is a simple rectangle with sides parallel to the x and y coordinate axes. The finite-difference solution deals only with values of W at discrete and equally spaced points, which will be called lattice points. These lattice points are, for convenience, serially numbered in a systematic way. Let the serial number or index be called j. Then the rectangular array of points may be arranged in q columns and p

rows. In terms of p and q, the index j for each lattice point can be exhibited in the following array:

$$\begin{array}{llllll}
1 & 2 & . & . & q-1 & q \\
q+1 & q+2 & . & . & 2q-1 & 2q \\
. & . & . & . & . & . \\
. & . & . & . & . & . \\
(p-1)q+1 & (p-1)q+2 & . & . & pq-1 & pq
\end{array}$$

The lattice points lying on the boundary are included in this array and appear in the first and last rows and the first and last columns. At these points the value of the function W is specified in advance. Actually, no use is made in the computations of the corner points ($j = 1$, q, $(p - 1)q + 1$, and pq), but these points are included in the array so as not to disturb the systematic character of the enumeration.

The value of the finite-difference solution W at any point will be denoted by $W(j)$. The value of $W(j)$ at an interior point is related to the values at the adjacent points by the following equation:

$$W(j) = \tfrac{1}{4}[W(j-1) + W(j+1) + W(j-q) + W(j+q)].$$

The functional values in the above equation are those which exactly satisfy the difference equation which is to be solved in lieu of the LAPLACE differential equation; they provide an approximation to the solution of the original equation. By writing out the $(p - 2)(q - 2)$ linear equations derived in this way for all interior points and solving these by any appropriate method, one could, of course, obtain the solution in a direct manner, but with a great deal of labor. In the Liebmann process, successive approximations to the correct solutions are obtained by a suitable modification of the above equation. Let $W(j)_i$ be the ith approximation to $W(j)$. Then the next approximation is obtained from the equation:

$$W(j)_{i+1} = \tfrac{1}{4}[W(j-1)_i + W(j+1)_i + W(j-q)_i + W(j+q)_i].$$

Actually, it is more convenient in forming the next approximation for any given point to utilize the best approximation so far obtained for the neighboring points; therefore the following equation is the one actually used:

$$W(j)_{i+1} = \tfrac{1}{4}[W(j-1)_{i+1} + W(j+1)_i + W(j-q)_{i+1} + W(j+q)_i].$$

It will be noted that in this latter equation use is made of the $(i + 1)$th approximation, which is already available for points $(j - 1)$ and $(j - q)$. This assumes that the computations are carried out so as to proceed systematically from the lowest value of j to the highest throughout the lattice. In most cases, somewhat quicker convergence will be found when this formula is used than if the preceding formula is used.

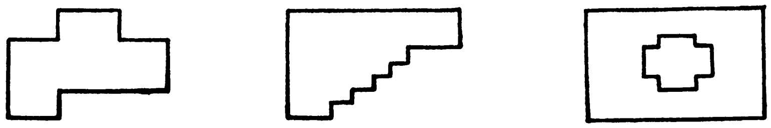
Any desired test of convergence may be used, since the UNIVAC is capable of carrying out any desired comparison process. The test which has been incorporated in the program presented here is merely an example of one such method. At each point and for each iteration, the absolute value

$$|W(j)_{i+1} - W(j)_i| = |\delta|$$

is formed. The sum of these absolute values over all interior points for a single iteration is used in testing convergence. Iteration is continued until this sum is diminished to some satisfactory preassigned value known as the maximum allowable error. (This maximum allowable error should not be taken as zero, since it is possible that because of round-off errors there will be no iterative cycle for which the above sum will reduce to zero; but instead this sum will ultimately vary in the periodic manner with a small amplitude.)

If the higher differences of the function W are not negligible, then the solution to the difference equation will differ appreciably from the solution to LAPLACE'S differential equation. The discrepancy between the two depends, of course, on the fineness of the lattice. Having the solution to the difference equation, one may cause the computer to calculate the appropriate differences in order to estimate whether a finer lattice should be used. Another possible procedure is known as the "deferred approach to the limit." In this case, several solutions of the same problem with different values of p and q are carried out, and from these one may infer the solution for the differential equation by extrapolation. These procedures are not incorporated in the present coding but could easily be handled by the UNIVAC. They are mentioned here only to emphasize the fact that the convergence tests discussed above have to do only with the solution to the difference equation and that avoidance of truncation error is a separate problem.

In the coded routine which follows, a simple rectangular boundary has been assumed. It should be pointed out, however, that modification of this routine to handle somewhat more general boundaries is not difficult. So long as the domain is bounded by vertical or horizontal lines connecting equally spaced lattice points, no particular difficulty is encountered. As examples of such domains the following illustrations are given. (Obviously curvilinear boundaries may be approximated by such rectangular boundaries.)



The same systematic serial numbering of lattice points would be used as in the case of the simple rectangular boundary. In order to make use of the same iteration equation and corresponding coded routine, this numbering should be such that the points which are the neighbors of point j are always $(j - 1)$, $(j + 1)$, $(j - q)$, and $(j + q)$. This is achieved by setting up a simple rectangular lattice which covers the actual domain of interest. Then, just as the corner points of the rectangular lattice are ignored in the simple problem, all other points which lie outside of the domain of interest will also be ignored. This means that in the actual routine for the more complicated cases an additional set of constants must be supplied as part of the initial data of the problem. These constants specify those values of j which mark off the beginning and end of each row in the lattice.

In any numerical work, and particularly in extensive calculations, attention must be given to the magnitudes of the numbers which occur at various points of the work. With the UNIVAC, facilities are provided which make it unnecessary for anyone to estimate what magnitudes are likely to occur or to provide in advance the proper scale factors to prevent numbers from running out of bounds. When numbers exceeding unity may possibly occur, the operator can insert as a part of his problem a subroutine which will automatically introduce appropriate scale factors and carry on the computation correctly. Also subroutines can be used in such a way as to achieve what is known as a "floating decimal point," making it entirely unnecessary for the operator to give any attention to the magnitudes of numbers. In this case all numbers are put in the form $A(10^s)$, where A is a number whose magnitude is less than unity and s is an integer which may range between exceedingly wide limits. The pair of numbers A and s are then used by the computer in conjunction with subroutines which always cause them to be interpreted as $A(10^s)$.

In the plane potential problem presented here, there is no need to resort to a "floating decimal" process nor to make use of the special facilities for accumulator overflow, since a single scale factor applied to the original input data will automatically insure that no number greater than unity will ever occur during the course of the computation. This scale factor is conveniently chosen as an integral power of 10. After applying such a scale factor to all boundary values, no boundary value should have a magnitude greater than .25. For example, if the largest boundary value for a given problem is 625 before scaling, then one should use a scale factor of 10^{-4} so that the boundary value as presented to the computer becomes .0625. Consequently, the sum of four function values which must be formulated during the iterative process will never exceed unity, and since such sums are multiplied by .25 to obtain a new function value, no number exceeding unity will ever occur in the computation.

It is thus apparent that a fixed decimal point is satisfactory for this problem, once such a scale factor has been introduced, and that the introduction of this scale factor is so simple as to be trivial. After convergence has been obtained and the resulting function values are read out, nothing more than a shift in decimal point is required to introduce into these results the compensating scale-factor to make them applicable to the original problem.

Preparation of a Boundary Value Problem for Solution by the UNIVAC System.—In the preparation of a problem for solution by the UNIVAC system, use is made of a typical coding sheet having three columns: the first shows memory locations; the second contains the first half of a word or one instruction; and the third contains the second half of a word or another instruction. It is convenient to stagger the halfwords so as to allow notes to be entered at the right of each instruction.

In the following programming routine, parentheses around an order (e.g., the orders stored in 0001) denote the fact that this order is continually being modified. Also, it should be noted that, at the beginning of the routine, the initial start button is pressed, the first block of orders are read from tape 1 into the I tank and thence to memory cells 0000 to 0060, and the first order stored in cell 0000 is automatically executed.

The actual program for the problem is as follows:

p number of rows
q number of columns
i number of the iteration

```
                                 Explanation of Instructions

0000   110000              One block of words from tape 1 to I register.
                 320060    Contents of I register transferred to memory cells 60–119
                           inclusive.
0001   (320100          ⎫
                 A 0001)⎪
0002   L 0060           ⎪
                 Q 0005 ⎬  Boundary values and initial internal values placed in
0003   A 0061           ⎪  memory locations 100–999 inclusive.
                 C 0001 ⎪
0004   000000           ⎪
                 U 0001 ⎭
0005   (B 0101             W(2)_i placed in A.
                 A 0130)   W(q + 1)_i added to (A).
0006   (A 0132             W(q + 3)_i added to (A).
                 A 0161)   W(2q + 2)_i added to (A).
0007   K 0000              4W(q + 2)_{i+1} placed in L; A cleared.
                 M 0044    W(q + 2)_{i+1} placed in A.
0008   H 0059              (A) stored in 0059; also held in A.
                (S 0131)   W(q + 2)_{i+1} − W(q + 2)_i = δ placed in A.
0009   Δ10000              δ shifted one digit to the left; left-hand digit dropped.
                 .10000    δ shifted one digit to the right.
0010   A 0058              Σ|δ| placed in A.
                 C 0058    Σ|δ| cleared to 0058; A cleared.
0011   A 0059              W(q + 2)_{i+1} placed in A.
                (C 0131)   W(q + 2)_{i+1} cleared to 0131; replaces W(q + 2)_i.
0012   A 0005              (B0101 A0130) placed in A.
                 A 0045    (B0102 A0131) placed in A; order to be stored in 0005
                           modified; (0045) in X.
0013   C 0005              (B0102 A0131) cleared to 0005; A cleared.
                 X 0000    (0045) now in A.
0014   A 0006              (A0133 A0162) placed in A; order to be stored in 0006 now
                           modified.
                 C 0006    (A0133 A0162) placed in 0006; A cleared.
0015   A 0008
                 A 0046
0016   C 0008
                 X 0000
0017   A 0011
                 C 0011
0018   A 0046
                 A 0056    1 added to count of averages.
0019   L 0057              If the count of averages = the count denoting end of row
                 Q 0021    (n(q − 2)), control transferred to 0021.

0020   C 0056              Count cleared to 0056.
                 U 0005    Return to instruction sequence for computing averages
                           (W(j)_i values) of next row.
```

```
                                  Explanation of Instructions

0021  L 0054             If iteration has been completed (i.e., values of W(j)ᵢ for
                Q 0030   784 values of j completed), control transferred to 0030.
0022  A 0053
                C 0057 ⎫
0023  A 0005           ⎬ q − 2 added to count in 0057.
                A 0047 ⎭ 3 added to labels at row end.
0024  C 0005
                X 0000 ⎫
0025  A 0006           ⎪
                C 0006 ⎪
0026  A 0008           ⎪
                A 0048 ⎬ 3 added to labels at row end.
0027  C 0008           ⎪
                X 0000 ⎪
0028  A 0011           ⎪
                C 0011 ⎭
0029  000000
                U 0005   Return to subroutine beginning in 0005.

0030  B 0055             Maximum allowable error placed in A.
                L 0058   Σ|δ| placed in L.
0031  000000             If error is within limits, control transferred.
                T 0039

0032  B 0049           ⎫ If error is greater than or equal to max. allowable error,
                C 0005 ⎪ averaging continued.
0033  A 0050           ⎪
                C 0006 ⎬ Original orders reset in 0005, 0006, 0008, 0011.
0034  A 0051           ⎪
                C 0008 ⎪
0035  A 0052           ⎪
                C 0011 ⎭
0036  C 0056             "No. of averages" count set to zero.
                C 0058   Σ|δ| set to zero.
0037  A 0053             (q − 2) = 28 placed in A.
                C 0057   (q − 2) = 28 cleared to 0057.
0038  00000
                U 0005

0039  (530100)         ⎫
                L 0062 ⎪
0040  B 0039           ⎪
                Q 0043 ⎪
0041  A 0061           ⎬ Final values inserted in tape 3.
                C 0039 ⎪
0042  000000           ⎪
                U 0039 ⎪
0043  000000           ⎪
                4.0000 ⎭ Machine stopped.

                                  Explanation of Storage

0044  025000    000000   To divide the sum of the values of the four surrounding
                         points by four.
```

			Explanation of Storage
0045	000001	000001	To alter orders so the same process can be carried out for the next point to the right.
0046	000000	000001	
0047	000003	000003	
0048	000000	000003	
0049	B 0101	A 0130	To put the initial orders in the associated memory positions in preparation for the execution of the next iteration.
0050	A 0132	A 0161	
0051	H 0059	S 0131	
0052	A 0059	C 0131	
0053	000000	000028	Contains $q - 2$ which is used to determine when a row is completed. (Constant)
0054	000000	000784	Contains $(p - 2)$ $(q - 2)$ for determining the completion of an iteration. (Constant)
0055			Contains the maximum allowable error against which $\Sigma\lvert\delta\rvert$ is checked.
0056	000000	000000	Contains count for number of averages which is checked against 0057 and 0054 to determine the end of the row and end of iteration, range from 0–784. (Variable)
0057	000000	000028	Contains $n(q - 2)$ to determine the end of a row. (Variable)
0058	000000	000000	To store the sum of the absolute difference between the values at points for successive iterations. (Variable)
0059			Used for temporary storage.
0060	320940	A 0001	Used in tape orders.
0061	000060	000000	
0062	530940	L 0062	

Time Required for Solution of the Problem.—The following formula enables one to calculate the number of seconds required for the UNIVAC solution of the problem if the values of p and q are specified and if the number of iterations is known or can be estimated:

Time required in seconds

$$= 2.3 + x[(p - 2)([q - 2].01071 + .005075)] + x(.000945) + (x - 1)(.004235),$$

where x is the number of iterations, (e.g.: $p = 30, q = 30, t = 2.3 + 8.6x$ seconds). For p and $q = 30$, we see that each iteration requires 8.6 seconds. There is also a period of 2.3 seconds associated with input and output, which time is independent of the number of iterations so long as the storage requirements do not exceed the internal memory capacity. This applies, then, to all values of p and q whenever the product pq does not exceed 900.

Estimation of the number of iterations required is not always easy, and no attempt will be made here to discuss this except to point out that this depends not only upon p, q, and the maximum allowable error, but also upon the boundary conditions.

When, to obtain the proper fineness of lattice, p and q must be chosen so large that their product exceeds 900, the external magnetic tape memory has to be used during the course of the problem solution to accommodate some of the information. Since the UNIVAC can be equipped with as many as 12 magnetic tapes, all under its automatic control, and each of these can store approximately one million decimal digits, there is ample capacity for handling problems for which hundreds of thousands of lattice points are used. There is no essential difficulty in programming problems of this magnitude. The programming is particularly easy when either p or q, whichever is smaller, is small enough so that three rows (or columns) of the lattice can be held in the internal memory at one time; that is, problems for which either p or q is less than 300 are easily programmed. If both p and q exceed 300, somewhat more elaborate programming must be used, and more time will have to be allowed for transfer of data to and from tapes.

The formula which has been given here to be used in estimating time of solution is strictly applicable only when the product pq is less than 900. However, it will be observed that the computation time per iteration is almost four times as large as the time required for input and output. (The high external-internal transfer rate of 10 000 decimal digits per second has been provided so that limitations from this cause normally do not occur.) It may be expected that the additional programming required to accomplish the transfers to and from magnetic tape for these larger problems will add slightly to the operating time of the computer, but the above considerations would indicate that one could take $\frac{1}{100}pq$ as the approximate time in seconds for each iteration of a large problem. Since the number of iterations required depends in a complex way upon various factors, including the boundary conditions, and is strongly dependent upon pq, there is no need to have a more exact estimate of the iteration time for a large problem.

Summary.—The solution of the plane potential problem using the Liebmann method of finite differences has been formulated in terms of the instruction code for the UNIVAC computer. Explicit coding has been given only for the simple rectangular boundary, but the nature of the modifications required for more general boundaries has been discussed. Time estimates for each iteration have been made, and other topics such as convergence tests, truncation errors, and scale factors have been considered. Problems with hundreds of thousands of lattice points can be handled automatically by the UNIVAC system using coded routines only slightly more complicated than the one presented.

FRANCES E. SNYDER & HUBERT M. LIVINGSTON

E-MCC

EDITORIAL NOTE: It is believed that there is wide interest in the question of instructing high-speed electronic computers now under design to perform the sequences of operations pertinent to selected problems. This article, submitted by members of the staff of a company engaged in the development and construction of electronic digital computers, is considered to be a useful introduction to the use of the instruction code for the computer therein discussed. It should perhaps be pointed out here, however, that the tone of the article is not to be construed to mean that UNIVAC systems have a history of successful operation, rather as an indication of the familiarity of the authors with the design features of the proposed machine and their evaluation of its potential utility. In fact, the construction of the first UNIVAC has not yet been completed.

[1] The UNIVAC trade mark and Instruction Code were in 1948 copyrighted by the Eckert-Mauchly Computer Corporation (E-MCC), Philadelphia, Pa.

[2] The UNIVAC system was designed by the E-MCC under contract with the NBS, supported by the Bureau of Census (BC). The investigation of the suitability of the system was carried out jointly by the NBS and the E-MCC together with the help of interested persons in the BC.

[3] H. LIEBMANN, "Die ausgenährte Ermittelung harmonischer Funktionen und konformer Abbildungen (nach Ideen von Boltzmann and Jacobi)," Akad. d. Wissen., Munich, *Berichte*, 1918, p. 385–416. The Liebmann method is actually a specialization of the GAUSS-SEIDEL iterative process written up by G. SHORTLEY & R. WELLER, "The numerical solution of Laplace's equation," *Jn. Appl. Physics*, v. 9, 1938, p. 334–348.