ORNL-3082
UC-32 – Mathematics and Computers
TID-4500 (16th ed.)

MATHEMATICS PANEL
ANNUAL PROGRESS REPORT
FOR PERIOD ENDING DECEMBER 31, 1960

OAK RIDGE NATIONAL LABORATORY
operated by
UNION CARBIDE CORPORATION
for the
U.S. ATOMIC ENERGY COMMISSION

Printed in USA. Price $1.50. Available from the

Office of Technical Services
Department of Commerce
Washington 25, D.C.

ORNL-3082
UC-32 – Mathematics and Computers
TID-4500 (16th ed.)

Contract No. W-7405-eng-26

MATHEMATICS PANEL
ANNUAL PROGRESS REPORT
for Period Ending December 31, 1960

A. S. Householder, Director
A. C. Downing, Assistant Director

DATE ISSUED

~~APR 6 1961~~

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee
operated by
UNION CARBIDE CORPORATION
for the
U. S. ATOMIC ENERGY COMMISSION

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

MATHEMATICS PANEL PROGRESS REPORT

SUMMARY

The functions of the Mathematics Panel are partly service and partly research, and include statistical analysis and programming in addition to mathematics as such. The largest single section in this report deals with programming and programming research, primarily in the construction of a translator to Oracle language from ALGOL, the international algebraic language. The importance of this work lies first in the fact that ALGOL is the one language of general currency in which algorithms can be written for use on any machine, and hence permits the exchange and publication of algorithms throughout the computing world. But the work is also important as a research project because of the growing importance of machine-independent languages and translators which have already contributed much toward lightening the programmer's task and for which techniques are still scarcely beyond the stage of infancy.

Research in mathematics and numerical analysis has touched upon several rather distinct topics. One of these is the use of functions other than polynomials as a basis for numerical quadrature, particular attention being given to the use of trigonometric functions. These techniques will be of use in the numerical solution of differential equations. Another topic is the evaluation of functions satisfying a recursion in such a way as to optimize the stability of the computation. The study of matrices has continued, in particular with reference to the localization of characteristic roots.

The installation of some punched-card equipment will greatly facilitate the recording and expedite the analysis of biological data. A number of special problems requiring statistical analysis are described under the headings "Biology and Medicine," "Health Physics," and "Metallurgy." The three sections that follow these describe programming activities for several divisions. The most extensive of these activities is the analysis of transmission data for the determination of neutron cross sections.

In February 1961 the Oracle will complete its seventh year at ORNL, and for purposes of record, a brief summary is given of operating experience and practice.

Several training courses have been given in elementary numerical analysis and in programming, and, in addition, members of the Panel have presented a number of special lectures at ORNL and elsewhere. These are listed near the end of the report, followed by a list of publications.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

SUMMARY iii

MATHEMATICS AND NUMERICAL ANALYSIS 1

Numerical Integration and Special Functions 1

Matrices 3

PROGRAMMING AND PROGRAMMING RESEARCH 5

ALGOL Translation on Oracle 5

Introduction 5

The Oracle ALGOL Language 6

Structure of the Translator 9

Input and Output 12

Work in Progress on the Oracle ALGOL Translator 14

Magnetic-Tape Files 14

Processing of For Statements 15

Processing of Procedures 16

Syntax Checker and Other Features 19

ALGOL Algorithms 20

Bairstow-Hitchcock Iteration 20

Bessel Functions 20

Cooperative ALGOL Efforts 21

Programming Research 21

Translator Theory and Design 21

Translator-Oriented Computer Language 22

General Programming 23

Linear Regression 23

Scheduling of Drafting and Design Work 23

Evaluation of Certain Definite Integrals of Atomic Physics by Analytical Integration on a Computer 24

Adaptation of Canned Programs 25

BIOLOGY AND MEDICINE 26

Data Processing 26

Cytology and Genetics 27

Radiation Protection of Living Cells 28

Mammalian Genetics and Development 29

Mammalian Recovery 29

Radiation Immunology 29

Pathology and Physiology 30

Cell Physiology 31

HEALTH PHYSICS 31
Ecology 31
Waste Disposal 33
Dosimetry 33
METALLURGY 34
Boron Content in Fuel Plates 34
Analysis of HFIR Fuel Plates 34
REACTORS 34
Equilibrium Concentrations 34
ORR Fuel Inventory 35
Metal-Clad Fuel Elements 35
Shift of Neutron Spectra 36
Escape of Fission Products 36
Diffusion of Fission Products 36
Half-Speed Whirl 36
Heat Transfer 36
PHYSICS 37
Transmission Data for Neutron Cross Sections 37
Miscellaneous Codes 38
CHEMISTRY 42
Power Reactor Fuel Processing Plant 42
Synthetic Pilot Plant Feed Solutions 42
Physical Inventory of U^{233} Solution 42
Calibration of U^{233} Carrier 43
Radioactivation Analysis 43
Crystallography 44
ORACLE OPERATION 45
TRAINING COURSES 50
Elementary Numerical Analysis 50
Oracle Coding Course 50
ALGOL Programming 50
Biometrics Course 50
Seminars 51
LECTURES AND PAPERS 51
PUBLICATIONS 53

MATHEMATICS AND NUMERICAL ANALYSIS

W. Gautschi A. S. Householder

NUMERICAL INTEGRATION AND SPECIAL FUNCTIONS

Numerical quadrature in its simplest form amounts to evaluating a functional $\int_0^1 x(t)\, d\alpha(t)$ having a continuous distribution $d\alpha(t)$ by means of a functional having a discrete distribution. It is advantageous to consider the error also as a linear functional, namely,

$$Lx = \int_0^1 x(t)\, d\alpha(t) - \sum_{\nu=0}^{n} \alpha_\nu x(t_\nu) .$$

There are two standard methods of determining "weights" α_ν and "nodes" t_ν, which may be termed "interpolation method" and "annihilation method," respectively. Both may be defined relative to a preassigned system $\{ x_k(t) \}$ of basis functions, assumed to be a Markov system, that is, one in which each finite subsystem $x_0, \ldots, x_{m-1}$ has the property that the $(m x m)$ matrix $[x_\nu(t_\mu)]$ is nonsingular for any choice of m distinct t_μ. Functions of the form $\gamma_0 x_0(t) + \ldots + \gamma_n x_n(t)$ (γ_ν scalars) are called "polynomials of degree n" relative to $\{ x_k \}$. There exists a unique polynomial of degree n passing through $n + 1$ points if $\{ x_k \}$ is a Markov system. Given the nodes t_ν, the interpolation method integrates the polynomial passing through the points $[t_\nu, x(t_\nu)]$. The annihilation method finds L such that

$$Lx_k = 0 \qquad \text{for } k = 0, 1, \ldots, m ,$$

with m as large as possible, assuming all, some, or none of the nodes as prescribed. It can be shown that a quadrature formula is of the interpolation type if and only if it is a formula of the annihilation type with $m \geq n$.

The resulting quadrature formulas are classical, and their theory well developed, if the basis functions are the powers $x_k = t^k$. A study has been initiated of quadrature formulas of annihilation type relative to the system of circular functions

$$1, \cos (2\pi t/T), \sin (2\pi t/T), \ldots, \cos (2k\pi t/T), \sin (2k\pi t/T), \ldots ,$$

with fixed period T. Such formulas should be useful in situations where the integrand is periodic or of oscillatory type, for example, in the evaluation of Fourier coefficients. A quadrature formula is called of trigonometric order m if L annihilates at least $2m + 1$ of the first harmonics. Given $2m + 1$ distinct nodes it is shown that there exists a unique quadrature formula of trigonometric order m, provided that $T > 1$ or that $T = 1$ and that not both end points of the interval of integration are nodes simultaneously. The weights $\alpha_\nu = \alpha_\nu(T)$ are analytic functions for $|T| > 1$. If $T \to \infty$ then $\alpha_\nu(T)$ tends to the respective weight of the classical quadrature formula of algebraic order $2m$.

Quadrature formulas of trigonometric order $m > 0$ are believed to be particularly effective if they are applied repeatedly over small intervals. The following, for example, is a formula of trigonometric order $m = 1$:

$$\int_0^{2h} x(t)\,dt \approx 2h[\alpha_0 x(0) + \alpha_1 x(h) + \alpha_0 x(2h)]\,,$$

$$\alpha_0 = \left(1 - \frac{\sin u}{u}\right)\left(4\sin^2\frac{u}{2}\right)^{-1}, \quad \alpha_1 = 1 - 2\alpha_0 \qquad \left(u = \frac{2\pi h}{T}\right).$$

It is similar, but not identical, to Filon's[1] formula. The usefulness of these formulas is not restricted to definite integrals but extends equally well to indefinite integrals with oscillatory integrands.

Analogous ideas were employed some time ago[2] in connection with the numerical solution of ordinary differential equations. These studies have been continued. Specifically, formulas have been produced which adapt the well-known Adams integration formulas to differential equations with oscillatory solutions.

Definite integrals such as

$$B_n(x) = \int_{-1}^{1} t^n e^{-xt}\,dt\,, \qquad E_n(x) = \int_1^{\infty} t^{-n} e^{-xt}\,dt \qquad (n = 0, 1, 2, \ldots)$$

are basic tools in such fields as the theory of molecular structure, astrophysics, and others. Even though they satisfy simple first-order recurrence formulas, their computation presents peculiar difficulties due to numerical instability. A detailed analysis of these difficulties was started some time ago[3] and has now been brought to a conclusion.[4] As a by-product of these studies many new properties of the integrals above have come forward, for example, two inequalities valid for $0 < x < \infty$, a number of monotonicity results for quotients of such integrals, and asymptotic formulas for certain zeros.

The problem of numerical instability presents itself in an even more stringent form in connection with special functions satisfying second-order recurrence relations, such as Legendre functions, Bessel functions, functions of the parabolic cylinder, etc. A special technique to evaluate such functions from their recurrence relations, without incurring instability, was suggested several years ago by J. C. P. Miller.[5] The method has recently been further analyzed.[6] An expression has been derived for the truncation error involved, which, apart from a convergence statement, also provides

---

[1] L. N. G. Filon, "On a Quadrature Formula for Trigonometric Integrals," *Proc. Roy. Soc. Edinburgh* A49, 38–47 (1928).

[2] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959*, ORNL-2915, p 1.

[3] *Research Highlights of the National Bureau of Standards, Annual Report, Fiscal Year 1959*, p 92.

[4] W. Gautschi, "Recursive Computation of Certain Integrals," *J. Assoc. Computing Machinery* 8, 21–40 (1961).

[5] *BAAS Mathematical Tables*, vol X, *Bessel Functions*, part II, p XVII, Cambridge University Press, 1952.

[6] W. Gautschi, "Recursive Computation of the Repeated Integrals of the Error Function," *Math. of Computation* (to be published in 1961).

useful estimates when the method is employed on an automatic high-speed computer. As an illustration, the results were applied to the repeated integrals of the error function, for which conventional computing methods are known to run into heavy loss of accuracy.[7,8]

MATRICES

Close and fruitful cooperation with F. L. Bauer and others at the Institut für Angewandte Mathematik in Mainz has continued, both by correspondence and by personal contact. The personal contacts were made possible by F. L. Bauer's brief stay in Ann Arbor in the summer and by visits of A. S. Householder and A. A. Grau to Mainz in the fall. Grau was concerned primarily with programming techniques, which will be discussed elsewhere in this report.

In the area of numerical analysis, a perturbation theorem published by Bauer and Fike[9] has been shown to imply and generalize a theorem of Wielandt[10] concerning a sum of normal matrices. The generalized theorem represents a seeming generalization of the Bauer-Fike theorem, although in fact it is a consequence. It is as follows: Let A and B be normalizable matrices, which is to say that they are expressible in the form

$$A = P_A \Lambda_A P_A^{-1}, \qquad B = P_B \Lambda_B P_B^{-1},$$

where Λ_A and Λ_B are diagonal matrices. Then if λ is any characteristic root of $A + B$, and μ is any scalar, the following two relations hold:

$$\max_i |\lambda - \mu - \lambda_i(A)| \geqq \min |\lambda_i(B) - \mu| / \nu(A, B),$$

$$\min_i |\lambda - \mu - \lambda_i(A)| \leqq \max |\lambda_i(B) - \mu| \nu(A, B),$$

where

$$\nu(A, B) = \text{cond}\,(P_A^{-1} P_B)$$

and

$$\text{cond}\,(M) = \text{lub}\,(M)\,\text{lub}\,(M^{-1}).$$

The second of these inequalities has the following geometric interpretation: Let Γ be any circle (of center μ) in the complex plane enclosing all roots $\lambda_i(B)$, and let its radius be increased in the ratio $\nu(A, B)$. From this, form n (not necessarily distinct) circles by translating by each of $\lambda_i(A)$. Then the union of these translated circles contains all roots of $A + B$, within or on the boundaries. Wielandt's theorem concerned only the case $\nu(A, B) = 1$. Much of the interest in this and in the

---

[7] J. Kaye, "A Table of the First Eleven Repeated Integrals of the Error Function," *J. Math. and Phys.* 34, 119–25 (1955).

[8] M. Abramowitz, "Review 58," *Math. Tables and Other Aids to Computation* 10, 176 (1956).

[9] F. L. Bauer and C. T. Fike, "Norms and Exclusion Theorems," *Numerische Math.* 2, 137–41 (1960).

[10] H. Wielandt, "On Eigenvalues of Sums of Normal Matrices," *Pacific J. Math.* 5, 633–38 (1955).

other theorems of the inclusion and exclusion type is as a tool for making error estimates and for demonstrating convergence. This work is to be published in *Numerische Mathematik.*

Many of the methods for obtaining characteristic roots of matrices are at least facilitated if the matrix is previously reduced by a similarity transformation to a special form. In the method of Hessenberg, this special form is one having zeros everywhere below the first subdiagonal, or, equivalently, above the first superdiagonal. In the Lanczos process, zeros occur everywhere both below the first subdiagonal and above the first superdiagonal. The Lanczos form is certainly preferable, but problems of instability were involved in making the transformation. It turns out, however, that two Hessenberg transformations produce a Lanczos form, and the difficulties are thereby removed.

Many of the methods of reduction had been shown to be particular applications of a quite general method due to Krylov,[11,12] and it turns out that still another method, obtained independently by Weber and by Voetter, providing the characteristic polynomial explicitly, is also in fact an application of the Krylov method. A constructive proof of the fact had been developed when it was learned that an indirect proof had been given by Paul,[13] who was then in Munich but is now at Mainz.

In iterative methods for obtaining characteristics roots, the usual iteration converges to the root of greatest modulus, if there is a single such root, and usually then a method of deflation is applied which yields a matrix from which this root is missing but all others are the same. When the next root is of nearly equal modulus, convergence may be very slow, and when there are two roots of equal modulus (e.g. conjugate complex roots), convergence to a single one cannot take place. However, in iterating on a set of linearly independent vectors, while the vectors may not individually approach a limit, nevertheless the set approaches an invariant subspace belonging to a cluster of roots of equal or nearly equal modulus, and the rate of approach is measured by the rate of vanishing of the sequence $|\lambda_{i+1}/\lambda_i|^{\nu}$, where λ_i is a root of smallest modulus of the cluster, and λ_{i+1} a root of greatest modulus outside the cluster. It has been found that the methods of deflation can be generalized to permit simultaneous removal of all roots of the cluster. If the cluster contains i roots, there will be obtained a matrix of order i for which these are the characteristic roots, and another of order $n-i$ whose roots are the remaining roots of the original matrix. If the roots of the cluster are close in value, and not merely in modulus, the associated vectors will not be well determined individually, although their space will be well determined when $|\lambda_{i+1}/\lambda_i|$ is small. This method of deflation takes full advantage of this fact, which is otherwise not easy to deal with, and properly isolates the real seat of the difficulty, which is the separation of the vectors from one another within the well-defined subspace. The method of deflation itself is described in a paper to be published.[14]

---

[11] A. S. Householder and F. L. Bauer, "On Certain Methods for Expanding the Characteristic Polynomial," *Numerische Math.* 1, 29–37 (1958).

[12] A. N. Krylov, "O Čislennom Rešenii Uravneniya, Kotorym v Tehničeskih Voprosah Opredelyayutsya Častoty Malyh Kolebanii Material'nyh Sistem," *Izvest. Akad. Nauk S.S.S.R., Otdel. Mat. i Estestven. Nauk* 1931, 491–539.

[13] M. Paul, *Zur Kenntnis des Weber-Verfahrens,* Technische Hochschule Munchen, Diplom-Arbeit, 1957.

[14] A. S. Householder, "On Deflating Matrices," *J. Soc. Ind. Appl. Math.* (to be published in March 1961).

The application of matrix norms to the development of inclusion, exclusion, and separation theorems, discussed in previous progress reports, is work that was started here. It provides, for example, generalizations of the classical Gershgorin exclusion theorem;[15] to extensions and generalizations of the Wielandt separation and inclusion theorems;[16] and to the Bauer-Fike perturbation theorem[9] and its extension discussed above. Hill, at Mainz, has constructed a general operator by means of which it is apparently possible to recover all the Brauer exclusion theorems and others.

With respect to inclusion and separation theorems, there is particular interest in obtaining the smallest possible inclusion circles about the zeros of the sequence of orthogonal polynomials that arise here, since when the generalized deflation is applied, these will be taken as the computed values of the roots in a given cluster. Radii of inclusion circles have been obtained, but it is not yet known whether these radii are the smallest possible.

---

[15] A. S. Householder, "On the Convergence of Matrix Iterations," *J. Assoc. Computing Machinery* **3**, 314–24 (1956).

[16] F. L. Bauer and A. S. Householder, "Moments and Characteristic Roots," *Numerische Math.* **2**, 42–53 (1960).

PROGRAMMING AND PROGRAMMING RESEARCH

G. J. Atta H. H. Bottenbruch L. L. Bumgarner E. C. Cooper
A. M. Culkowski M. Feliciano A. A. Grau

For the purpose of this report, the programming activities of the Mathematics Panel that are not directly connected with the programs of other divisions of the Laboratory at this time may be classed under six heads: (1) description of the present Oracle ALGOL translator, (2) work in progress on the Oracle translator, (3) algorithms, (4) cooperation in ALGOL projects of scope beyond the Laboratory, (5) programming research, and (6) general programming. The activities involve mostly some phase of ALGOL.[1,2]

ALGOL TRANSLATION ON ORACLE

Introduction

The work on a translator program was begun on the basis of the preliminary report on ALGOL,[1] formerly called the International Algebraic Language in this country; its structure was summarized in the previous Mathematics Panel Progress Report.[3]

---

[1] "Preliminary Report – International Algebraic Language," ed. by A. J. Perlis and K. Samelson, *Communs. Assoc. Computing Machinery* **1**(12), 8–22 (1958).

[2] "Report on the Algorithmic Language ALGOL 60," ed. by Peter Naur, *Communs. Assoc. Computing Machinery* **3**, 299–314 (1960).

[3] G. Atta *et al.*, *Math. Panel Progr. Rept. Sept. 1, 1958 – Dec. 31, 1959*, ORNL-2915, p 8.

The report on ALGOL 60 (ref 2) appeared in May 1960, though unofficial preliminary versions were available to the Mathematics Panel as early as March. The changes in the language introduced by the ALGOL 60 report were such that it was not possible to alter the translator to make provision for them without delaying considerably its completion. It was relatively easy to alter the translator to accommodate the transformation of the *if* and *for* statements of ALGOL 58 into the *if* and *for* clauses of ALGOL 60. The abandonment of the *do* statement (the symbol *do* has a different function in ALGOL 60) made ineffective considerable work done towards its implementation; a segment of the translator had been planned and all but coded. The necessity of processing *do* statements imposed a structure on the translator that is not now needed but which cannot be changed without a complete redesign of the translator.

At present the Oracle ALGOL translator handles quite adequately most ALGOL programs that are written. When fully completed, it will be able to handle any program in ALGOL which does not make use of blocks or some relatively obscure devices.

The Oracle ALGOL Language

A description of the part of the ALGOL language which can at present be processed by the pilot-version ALGOL translator for the Oracle follows. Considerable detail is necessary because the rules governing the version of ALGOL accepted by the Oracle translator involve not only the hardware representation, but also to some extent the structure of the language itself.

The *basic symbols* are those of the reference language of ALGOL 60. A program written in reference language will be transliterated into hardware language by the punch unit operator, subject to limitations as described below.

Expressions. Arithmetic and logical processes (in the most general sense), which the algorithmic language is primarily intended to describe, are given by arithmetic and logical expressions, respectively. Constituents of these expressions, except for certain delimiters, are numbers, variables, elementary arithmetic operators and relations, and other operators called functions. The description of both variables and functions may contain expressions. The definition of expressions is necessarily recursive.

The following are the units from which expressions are constructed:

1. (*Positive*) *numbers* N. They are represented as in ALGOL 60.

2. *Simple variables* V. Simple variables are designations for arbitrary scalar quantities, for example, numbers as in elementary arithmetic.

3. *Subscripted variables* V. Subscripted variables designate quantities which are components of multidimensional arrays.

$$I[C]\,,$$

where I is any identifier and C is of the form $E, E, \ldots, E$, a list of arithmetic expressions as defined below. Each expression E occupies one subscript position of the subscripted variable, and is called a subscript. The complete list of subscripts is enclosed in the subscript brackets [ ]. The array component referred to by a subscripted variable is specified by the actual numerical value of its subscripts.

Subscripts, however, are intrinsically integer valued, and whenever the value of a subscript expression is not integral, it is replaced by the nearest integer in the sense of proper round-off. Subscripts may not be negative quantities.

4. *Functions F*. Functions defined through function declarations are not handled. However, the standard functions of analysis will be built into the Oracle system and the following identifiers have been reserved for them: abs, sign, entier, sqrt, sin, cos, log, exp, arctan.

5. *Arithmetic expressions E*. An arithmetic expression may be (*a*) a number, a variable (other than Boolean), or a function, (*b*) any of the following: $+E_2$, $-E_2$, $E_1 + E_2$, $E_1 - E_2$, $E_1 * E_2$, E_1 / E_2, $E_2 \uparrow E_1 \downarrow$, (E_1), where E_1 is an arbitrary expression, and E_2 is an expression, the first symbol of which is neither + nor −. The parentheses ↑ ↓ denote exponentiation, where the leading expression is the base and the expression enclosed in parentheses is the exponent; $a \uparrow b \downarrow$ has the effect of $|a|^b$ since it is computed by means of the formula $e^{b \log |a|}$. The sequence of operations within one expression is from left to right, with the usual rules of precedence.

6. *A Boolean expression B* may be (*a*) a truth value, a Boolean variable, or a Boolean function, (*b*) any of the following: $(E_1 < E_2)$, $(E_1 \leqq E_2)$, $(E_1 \neq E_2)$, $(E_1 = E_2)$, $(E_1 > E_2)$, $(E_1 \geqq E_2)$, where E_1 and E_2 are arithmetic expressions, and such expressions take on the value *true* whenever the corresponding relation is satisfied for the expression involved, otherwise *false*, (*c*) if B_1 and B_2 are Boolean expressions, the following are Boolean expressions: $\neg B_1$, $B_1 \vee B_2$, $B_1 \wedge B_2$, $B_1 \equiv B_2$, (B_1). The operators $\neg$, $\vee$, $\wedge$, and $\equiv$ have the usual interpretations. Interpretation of a sequence of binary operators will be from left to right.

Statements S. They are defined recursively in the following way: Strings of one or more statements may be combined into a single (compound) statement by enclosing them within the "statement parentheses" **begin** and **end.** Single statements are separated by the statement separator ";". A statement may be made identifiable by attaching to it a label *L*, which is an identifier *I*, or an integer *G* (with the meaning of identifier). The label precedes the statement labeled, and is separated from it by the separator colon ":". Label and statement together constitute a "labeled statement." A labeled statement may not itself be labeled. The closing parenthesis **end** may be followed by the statement label (followed by a statement terminator) in order to indicate the range of the compound statement.

Assignment statements. These serve for assigning the value of an expression to a variable.

(i) $$V := E$$

where V is any variable and E an expression.

(ii) $$V := B$$

where V is any variable and B a Boolean expression.

Go to statements. The normal sequence of execution may be interrupted by the use of a *go to* statement of the form **go to** D, where for Oracle usage D must be a label.

If statements and conditional statements. The if statements are handled as in ALGOL 60:

(i) **if** B **then** S_1

(ii) **if** B **then** S_1 **else** S_2

where B is a Boolean expression and S_1 and S_2 are arbitrary statements. The following forms of conditional statements are possible:

(iii) **if** B_1 **then** S_1 **else if** B_2 **then** S_2 **else** S_3; S_4

and

(iv) **if** B_1 **then** S_1 **else if** B_2 **then** S_2 **else if** B_3 **then** S_3; S_4 .

Here B_1, B_2, and B_3 are Boolean expressions, while S_1, S_2, and S_3 are unconditional statements and S_4 is the statement following the complete conditional statement.

For statements. The form handled at present by the Oracle translator is limited to **for** $V := E_1$ **step** E_2 **until** E_3 **do** S, where V is a simple (i.e., not subscripted) variable, E_1, E_2, E_3 are expressions, and S is a statement. The group E_1 **step** E_2 **until** E_3 determines an arithmetic progression.

Stop statements. The *stop* statement, which causes the machine to halt, is of the form **stop**. Though not contained in ALGOL 60 it has been retained because we consider it useful in the Oracle.

Dummy statements. A dummy statement executes no operation. It is permitted and is sometimes useful in writing a for statement.

Compound statements. A compound statement is a sequence of statements enclosed in the statement brackets **begin end**.

Statement separator. If each of S_1 and S_2 is a statement or a compound statement and S_2 follows S_1, then the statement separator ";" must appear between the two.

Declarations. Declarations which serve to state certain facts about entities referred to within the program, and have no operational meaning, must appear at the beginning of the program. They pertain to the entire program in which they occur and their effect is not alterable by the running history of the program. They may not have a label and are always terminated by the symbol ";".

Type declarations are **integer** $a, \ldots, z$ and **boolean** $a, \ldots, z$. If desired, type declarations may be used to inform the reader of the nature of the variable. However, such declarations have no effect on the program.

An *array declaration* gives the dimensions of multidimensional arrays of quantities. Form: **array** $(I, I, \ldots, I\,[C : C'], \ldots, I\,[C ; C'] \ldots)$ where **array** is the array declarator, the I are identifiers, and the C and C' are lists of nonnegative integers separated by commas.

Comments. Comments are used to add to the program informal comments, possibly in natural language, which have no meaning whatsoever to the translator, have no effect on the program, and are intended only as additional information to the reader. A comment must be terminated with the symbol ";" except at the very end of a program.

ALGOL programs. As in ALGOL 60, an ALGOL program is a self-contained compound statement, that is, a compound statement which is not contained within another compound statement and which makes no use of other compound statements not contained within it.

Hardware language. The hardware language is an adaption of the reference language enforced by the limited number of characters on the standard input equipment of the Oracle. It has been fully described previously.[4]

Errors. Errors are divided into two classes: syntactical and mathematical.

Syntactical errors will show up during the translation process. At present, if a program written in ALGOL has syntactical errors the translation process stops without giving information which would help in checking out. For this reason an error may be hard to find without knowledge of the internal operation of both the translator and the machine. However, most errors can be detected by the operator.

Mathematical errors will occur during execution time. The operations and functions used by the object code have error stops similar to ORBIT.

Structure of the Translator

For convenience the translator is sometimes referred to as ALCOR. Segment I of ALCOR loads the source program into the machine and writes the statements and declarations in internal form on magnetic tape. This segment is divided into two parts. The first part assigns code words to characters which have an eight-bit bihex representation in Oracle alphameric code. The second part assigns code words containing other unique bihex symbols to ALGOL delimiters and identifiers. Each statement is copied sequentially on magnetic tape.

The for statements in the present version of Oracle ALGOL are preprocessed in segment II. This consists in decomposing each for statement into more elementary statement forms. The processing of the do statement of ALGOL 58 was worked out and was to be included in this segment. However, the statement was eliminated from ALGOL 60, and it therefore is not now used on the Oracle.

Segment III processes the array declarations and assigns relative addresses to numbers and simple identifiers. The array declaration yields information necessary for the translation process which is entered in tables, and the array identifier is assigned the address where this information is found. On the other hand a simple identifier is assigned the addresses which it will have in the target program. Numbers are converted to floating-point and treated as simple identifiers. The program is now in process ALGOL form and is copied on magnetic tape.

Segment IV translates this process ALGOL into Oracle compiler language from which the machine code is eventually obtained. The translation process is a sequential operation on the characters of process ALGOL. The operation on each character is divided into three steps.

---

[4]G. Atta *et al.*, *Math. Panel Progr. Rept. Sept. 1, 1958 – Dec. 31, 1959,* ORNL-2915, pp 9, 10.

The first step produces a class character which depends on the incoming character. There are six classes: variables or numbers, relations, unary operations, labels, **begin**, and any other not included above. Necessary information about the incoming character is stored in the auxiliary push-down.

In the second step the class character is paired to the currently available character in the control push-down. This is a list of class characters not previously discarded during the translation process. The pair is searched in the translation matrix, which is a table of one-word entries each of which contains a pair of characters and information directing the next set of operations. If the pair under consideration is not found in the translation matrix there is a syntactical error in the source program.

The third step follows the directions given by the translation matrix and returns to the first step. The following is a description of the building blocks. Some of the building blocks are identified by numbers, for example, 20Ψ. The symbol following this number serves as a parameter. No attempt is made to present coding techniques or difficulties, the most serious of which depend on the limited memory of the machine.

Next. The class character is assigned as follows: i, variable or number; r, relation; u, unary operation; l, label. Any other character not included above is assigned itself. An exception is the incoming character **begin** which is stored in the control push-down after which a jump is made to the beginning of this building block. Information about the incoming character needed during the translation process is stored in the auxiliary push-down. Jump to process pair.

Process pair. Combine the class character with the character in the control push-down. The next step is the table look-up.

Table look-up. Search for the pair found above in the translation matrix. If the pair is not found there is a syntactical error.

Binary operations ($11\Psi R$). When a binary operation occurs, say a o b, the translator has constructed a code in the target program which will transfer the values of a and b to consecutive temporary storage. This building block then writes in the target program the entry (with link word) to the subroutine which will perform the binary operation.

Unary operations (130θ *and* 130Ψ). Both building blocks refer to unary operations, say $u(a)$. Before this building block is used the value of the operand has been stored in a temporary cell. Both building blocks set up the entry to the subroutine which will perform the operation.

The difference between the two lies in the fact that in 130θ information about the subroutine entry is θ, whereas in the other the information is in the auxiliary push-down.

Control state substitution (19Ψ). The last character stored in the control push-down is substituted by the character indicated by Ψ.

Control state addition (20Ψ). Add Ψ to the control push-down.

Delete ($D\sigma$). Delete the current character from the control push-down.

Repeat (Rep). Jump to process pair and pair the current class character with the next character in the control push-down.

Transfer to temporary (*61*). Transfer the value of the variable in a temporary cell.

Arrays (*65, 66, and 67*). The memory location of an element of an array is computed iteratively. Given an array element, say $A[i_1, i_2, \ldots, i_n]$, the iteration consists of n steps at the end of which the memory location of the element relative to its first element is obtained. That is, the first element of the array is assumed to occupy cell 000.

Building block 65 prepares for this subscript iteration.

Building block 66 produces in the target program the entry to the subroutine which will execute a step of the iteration.

When all the subscripts of an array element are processed, then building block 67 establishes a code in the target program which computes the absolute memory location of the subscripted variable by adding the absolute memory location of the first element of the array and the relative memory location of the element.

Transfer from temporary (*68*). Set up code in the target program which will transfer the value of a variable from temporary storage to its assigned cell.

Relations (*69*). Store the relation character in auxiliary storage.

Prepare for jump address (*70*). Set up code in target program to test and jump on a Boolean variable (jump on truth value false). Store current address of target program in auxiliary push-down.

Set jump address (*71*). When a jump is set up in the target program the translator is usually ignorant of the address to which the jump is to be made. Therefore, in processing jumps the translator stores the address where the jump occurs in auxiliary storage (see building block 70). As soon as the translator has the information necessary to set the address (in the target program) to which the jump is made, building block 71 is actuated to do the necessary work.

Unconditional jump (*72*). Write code in the target program to jump unconditionally. Store current address of target program in the auxiliary push-down.

End of target program (*80*). The end of the translation process is reached when the first **begin** is deleted from the control push-down. This is the only criterion needed since the **begin-end** level has previously been tested in preprocessing. Building block 80 tests for the end of the translation process. If the end has been reached it jumps to the final steps of the program.

Label (*90*). In a go to statement where the jump is to a labeled statement the translator may have the address to which the jump is made. In this case the unconditional jump is fully coded in the target program. Otherwise the label under consideration has not been reached. The unsatisfied label table secures a new entry specifying the addresses in the target program from and to which the jump is made.

Set label address (*91*). The label used in a labeled statement appears when the target program is ready to accept the code from this labeled statement. Therefore, the address of the target program belonging to the label is known. This building block assigns the proper address to the label table.

The label table consists of two words, the first of which is the internal bihex representation of the first five letters of the label and the second is the address of the target program where the instructions corresponding to the labeled statement begin.

Stop order (*95*). Sets up a stop order in the target program.

Truth value (*110*). The strength of the if statement lies on the conditional jump. When a conditional jump is processed the target program is ready to establish whether the condition is met or not. This is done by a subroutine which will exit with the truth value of the condition stored in temporary cell. This building block sets up the entry to this subroutine.

Table 1 shows the translation matrix used in the translator.

Input and Output

Oracle ALGOL contains delimiters which are used in input and output statements. The input delimiters are **read** and **read array**. The first is used to input simple variables and the second to input whole arrays. The output delimiters are **punch** and **cr**. The *punch* statement has the effect of releasing on paper tape the values of specified expressions. The *cr* statement is used to release a carriage return and its main use is to obtain a proper output format.

The building blocks associated with these statements are the following:

Read (1*C*1 and 1*C*2). These building blocks set in the target program the entry to the subroutine which will make the necessary operations.

Punch (1*A*1 and 1*A*2). Similar to the above for the output subroutine.

Cr (7*A*). Sets up code to release a carriage return.

The following is a description of the input and output conversion subroutines used in the target program.

A subroutine is activated by *read* and *read array* statements in Oracle ALGOL, which converts decimal numbers to (8, 32) floating-point numbers. The numbers to be converted are listed on paper tape in sequence and must adhere to a definite alphameric format: They are of the form $\alpha\,{}_{10}\beta$, where α is a signed or unsigned decimal number containing at most ten digits, and β is a signed or unsigned decimal integer containing at most two digits. Either α or ${}_{10}\beta$ may be omitted.

Basically, the algorithm consists of processing the number in the form $m \cdot 2^p \cdot 10^r$, where $\frac{1}{2} \leqq |m| < 1$, p is an integer such that $0 \leqq p \leqq 39$, and r is any integer. This number is converted until $r = 0$ as follows. If $r = 0$, the conversion process is terminated. If $r < 0$, m is multiplied by $\frac{8}{10}$. This has the effect of decreasing p by 3 and increasing r by 1. If $r > 0$, m is multiplied by $\frac{5}{8}$. This has the effect of increasing p by 4 and decreasing r by 1. The process is repeated until $r = 0$. If $|m|$ becomes less than $\frac{1}{2}$ in the process, it is doubled and p decreased by 1.

In this subroutine it was convenient to use the alphameric one-character read instruction 9*C*[ ]. Because of the high clutching rates, this order is potentially dangerous to the Ferranti reader. Thus, the subroutine was written in a manner to minimize the clutching rates.

Table 1. Translation Matrix

α_a \ σ_s	begin else μ	begin R	go to	r [↑ (:=	/ u + − *	if ∨ ∧ ≡ θ	L	else R	μR	rR	$+R$ $-R$	$/R$ $*R$	uR	θR	I	$\vee R$ $\wedge R$ $\equiv R$	$:=R$	$(R$	$[R$	$\uparrow R$	if R	read	punch	read array
if	$19\sigma R \mid 20\alpha$																							
go to	$19\sigma R \mid 20\alpha$																							
L	20α		20α																					
stop	$19\sigma R \mid 95$																							
I	$19\sigma R \mid 20\alpha$			$19\sigma R \mid 20\alpha$	$19\sigma R \mid 20\alpha$														20α			*Next*	20α	50
u				$19\sigma R \mid 20\alpha$	$19\sigma R \mid 20\alpha$																		20α	
(				$19\sigma R \mid 20\alpha$	$19\sigma R \mid 20\alpha$																		20α	
:							91																	
[							19α								$65 \mid 20\alpha$									
:=		20α						20α	20α						19α									
;	$19\sigma R \mid$ Process Pair	19 begin	$92 \mid D\sigma$				$90 \mid D\sigma \mid \text{Rep}$	$71 \mid \text{Rep}$	$71 \mid \text{Rep}$	$110\alpha_a \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$68 \mid \text{Rep}$				$70 \mid 19\mu$	$1C2$	$1A2$	Rep
end	$19\sigma R \mid$ Process Pair	80	$92 \mid \text{Rep}$				$90 \mid D\sigma \mid \text{Rep}$	$71 \mid \text{Rep}$	$71 \mid \text{Rep}$	$110\alpha_a \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$68 \mid \text{Rep}$					$1C2$	$1A2$	Rep
else	$19\sigma R \mid$ Process Pair	20α	$92 \mid \text{Rep}$				$90 \mid D\sigma \mid \text{Rep}$		$72 \mid 19\alpha$	$110\alpha_a \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$68 \mid \text{Rep}$					$1C2$	$1A2$	Rep
)										$110\alpha_a \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$		$D\sigma$						
+				*Next*						20α	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$		20α	20α	20α	20α			20α	
−				$19\sigma R \mid 20\theta$						20α	$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$		20α	20α	20α	20α			20α	
*										20α	20α	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$		20α	20α	20α	20α			20α	
/										20α	20α	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$		20α	20α	20α	20α			20α	
↑										20α	20α	20α	20α	20α	$61 \mid \text{Rep}$		20α	20α		20α			20α	
↓											$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$					$11\sigma \mid D\sigma$				
]											$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$				$66 \mid 67 \mid D\sigma$					
r											$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$			69						
,											$11\sigma \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	$130\alpha_a \mid \text{Rep}$	$130\theta \mid \text{Rep}$	$61 \mid \text{Rep}$				66			681	*Next*	*Next*
∧															$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	20α	20α			20α			
∨															$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	20α	20α			20α			
≡															$61 \mid \text{Rep}$	$11\sigma \mid \text{Rep}$	20α	20α			20α			
read array	$19\sigma R \mid 20\alpha$																							
read	$1C1 \mid 19\sigma R \mid 20\alpha$																							
punch	$1A1 \mid 19\sigma R \mid 20\alpha$																							
cr	$7A \mid 19\sigma R$																							

For output, the translator writes into the target program an entry to a subroutine that converts numbers in internal representation to the format used by the ALGOL system on the output equipment.

As above, the method used in the conversion consists of manipulating numbers of the form $m \cdot 2^p \cdot 10^r$. If $p = 0$, conversion to decimal form is completed. If $p < 0$ or if $p > 0$, m is multiplied by $^{10}/_8$ or $^{16}/_{10}$ respectively and corresponding adjustments made in p and r. If $|m|$ becomes greater than or equal to 1, it is halved and p increased by 1.

WORK IN PROGRESS ON THE ORACLE ALGOL TRANSLATOR

Magnetic-Tape Files

The ALGOL 60 language was designed to simplify formulation of computation processes to be executed on electronic computers. Emphasis in the design of the language was put on ease of formulation rather than on efficient use of the facilities of a computer such as magnetic tapes. To ensure more efficient use of the Oracle, provisions have been made in the Oracle-ALGOL language to use as auxiliary storage the Oracle magnetic tapes.

A new declaration has been added to the language which declares that one or more arrays are to be stored on one of the magnetic tapes during the execution of the program. The declaration begins with the new delimiter **tape** followed by an integer i $(0 \leq i \leq 3)$; a normal ALGOL *array* declaration follows. Only one tape declaration is given for each tape, and these declarations are required to be at the very beginning of each program. Elements of tape arrays may be used in the same way as elements of normal arrays. However, an efficient program will be achieved only if the tape array elements are used sequentially.

The manner in which a magnetic tape is used may be specified by the following three new statements which have been added to the language:

1. **tapestate** i := **read only**
2. **tapestate** i := **write only**
3. **tapestate** i := **read write**

During the execution of the target program one block of each tape which is declared will be represented in high-speed storage. If an element of a tape array is referred to, a subroutine first determines whether that block of tape in which the element lies is represented in high-speed storage (internal storage). If it is represented, the required operation is executed without any tape motion. If it is not represented, the following action takes place depending on the state of the tape:

1. In the *read only* state the block in which the referenced element lies is read into high-speed storage.

2. In the *write only* state the block just represented in high-speed storage is written on tape, and the tape is positioned at the block in which the referenced element lies.

3. In the *read write* state, the block just represented in high-speed storage is written on tape, and the block in which the referenced element lies is read into high-speed storage.

In the implementation of the above specifications, a numerical address was associated with each location of each tape. The addresses used for tape 0 range from 2^{32} to $2^{33} - 1$; those for tape 1 range from 2^{33} to $2^{32} + 2^{33} - 1$; those for tape 2 range from $2^{32} + 2^{33}$ to $2^{34} - 1$; and those for tape 3 range from 2^{34} to $2^{35} - 1$. This means that the thirty-second, thirty-third, and thirty-fourth bits of the binary representation of an address determine the tape number.

One part of the implementation is the construction of a program which scans the tape array declaration and builds up tables according to the conventions used for ordinary arrays.

The major part of the implementation is the construction of a subroutine called "tape-positioner." This subroutine, when supplied with a tape address, controls the necessary tape motions and replaces the tape address by a high-speed storage address. Tape manipulation is minimized by using both forward and backward tape instructions.

The "tape positioner" is in an advanced state of preparation. The programming and coding is completed.

The techniques devised for use of the Oracle tapes in an ALGOL translator, although they may not be the ultimate solution to the problem of efficient use of auxiliary storage, are applicable to a wide class of computing machines.

Processing of For Statements

Work is almost complete on a modification of the Oracle-ALGOL translator which will process the complete for statement of ALGOL 60. It is an ALGOL-to-ALGOL translation in a preprocessing stage prior to the main translation into machine code, as was the ALGOL 58 for statement translation, which, however, was restricted to the most frequently occurring case of the for statement.

When an ALGOL program containing a for statement is processed by the for statement translator, the resulting program is still in ALGOL, but with the addition of three internal delimiters. These are **SJ** (subroutine jump), **SSE** (set subroutine exit), and **return**. Let us consider the following example:

for $v := E,\ E1$ **step** $E2$ **until** $E3$ **do** Σ ,

where Σ is any ALGOL statement and the E's are arithmetic expressions. After processing by the for statement translator, this statement will be replaced by the following set of statements:

```
     begin v := E;
     SJ M1;
     v := E1;
L1:  if (v − E3) * E2 ≦ 0 then
     begin SJ M1;
     v := v + E2;
     go to L1 end;
     go to L2;
M1:  SSE L3;
     Σ;
L3:  return;
L2:  ; end
```

The labels appearing here are created internally and in such a way as to avoid confusion with labels in the original program.

The for statement processor is coded and debugged. The main segment of the translator must be modified to accommodate the new internal delimiters. Completion is expected in a matter of weeks.

Processing of Procedures

The ALGOL translator as it stands now cannot translate programs which contain procedures. The ability to translate procedures will be added to the translator.

Procedure declarations and procedure statements will be replaced in a preprocessing phase by statements and declarations which can be handled by the already existing parts of the translator. Some additions to these parts must be made, however, in order to allow efficient translation. These additions include:

1. handling of dummy arrays (see below),
2. setting up an order **go to** *L* in a location assigned to a variable,
3. execution of an order **go to** *a* where *a* is a variable set previously in accordance with 2,
4. transmission of control information for one array to the locations reserved for the control information of some other array (normally a dummy array).

The procedure processor makes use also of the statements **SJ** *L*, **SSE** *L*, and **return** which were described in the section "**Processing of For Statements.**"

Certain features of the ALGOL 60 procedures had to be sacrificed in order to keep the changes in the already existing parts to a minimum. Thus the Oracle procedures will have the following restrictions:

1. No functions are permissible.
2. No procedure *P*1 which is a parameter of another procedure *P* may have a procedure as a parameter.
3. No procedures may be defined within procedures.
4. "Dynamic parameter expressions" are not allowed.
5. Local arrays may not have variable dimensions.

We do think that these restrictions, though numerous and seemingly serious, will rarely matter in the "average program." Most of the programs published in the *Communications of the Association for Computing Machinery* so far can be handled by our procedure processor without any changes. It is usually fairly simple to change an ALGOL 60 procedure which does make use of these features into a procedure acceptable to the Oracle translator. In particular, the following remarks apply to the individual restrictions:

1. A function is changed into a procedure with one more parameter, the "result" of the function evaluation.
2. No such case has been encountered yet.
3. Nested definition of procedures may be replaced by "parallel" definition; the common local variables must be made global.

4. Dynamic parameter expressions must be changed into procedures.
5. Local arrays with variable dimensions must be made parameters, and consequently declared outside the procedure.

Although it is "usually" fairly simple to make these changes, there are cases (hopefully infrequent) where this is complicated, and this is just the reason why these features were not incorporated into the translator. We insist on specifications for the parameters of a procedure. In addition to the ALGOL 60 specifications we allow the specification **output**. Addition of the specification permits the translator to produce a more efficient program. If one of the parameters is specified to be an array, the number of parameter-positions of this array must be indicated. If it is specified to be a procedure, its parameters must also be specified.

Example: **procedure** *A* (*b*, *c*, *L*, *d*, *f*); **real** *b*; **integer output** *c*;
label *L*; **array** *d*[,]; **procedure** *f* (**real**, **array** [ ], **label**);
begin

.
.
.

procedure body

.
.
.

end

The following main steps are necessary for the elimination of procedures.

Translation of the procedure body:

1. Assign definite "entities" to the formal parameters of each procedure and create internal identifiers for these entities. Create declarations for these entities if this is necessary.
2. Change the names of all parameters of the procedure in accordance with 1.
3. Change the names of all local quantities used in the procedure so that they are different from all other names used or created elsewhere in the program.
4. Create a local label *L* and add the two statements **SSE** *L* and *L*: **return** at the beginning and end of the procedure body.

Translation of the procedure call:

5. Write statements which assign the input parameters.
6. Write the statement "*subroutine jump* to the first statement of the procedure."
7. Write statements which assign the outputs of the procedure to the corresponding variables of the procedure call.

Table 2 exhibits many of the necessary actions. The program assumes an array $a[0]$, $a[1]$, ..., $a[10]$ to be in memory. It reads a sequence of numbers x and computes $y = l(x)$ by linear interpolation in the table $a[0]$ to $a[10]$.

Table 2. Example of Necessary Actions for Elimination of Procedures

Program With Procedures	Program Without Procedures	Remarks
begin real x, y; **array** a[0:10]	**begin real** x, y; **array** a[0:10]	No changes
procedure interpolate (p, y)	*go to* M;	The name of the procedure becomes a label; the statement "Set subroutine exit" is inserted. The procedure is skipped by the statement *go to* M which transfers control to the first statement of the original program.
value p; **real** p, y;	interpolate: SSE L;	
	real p, q;	
begin integer x;	**begin integer** z;	The variable x is changed to z. The parameter y is changed to q to avoid collision with the y used outside
x := ent[p];	z := ent[p]	
if x < 0 **then** y := a[0]	**if** z < 0 **then** q := a[0]	
else if x > 10 **then** y := a[10]	**else if** z > 10 **then** q := a[10]	
else y := a[x] + (a[x + 1] − a[x]) * (p − x);	**else** q := a[z] + (a[z + 1] − a[z]) * (p − z)	
end interpolate;	L : **return**	
read x; **punch** x;	M : **read** x; **punch** x;	No change
interpolate (x, y);	p := x; SJ interpolate; y := q	Translation of procedure call
punch y;	**punch** y;	No change
interpolate (x − 1, y);	p := x − 1, SJ interpolate; y := q	Translation of procedure call
punch y; **cr**;	**punch** y; **cr**	No change
stop	**stop**	
end	**end**	

The following entities are created for the different types of parameters:

Type of Parameter	Entity Created for This Parameter
simple variable	simple variable
label	simple variable
array	"dummy" array, that is, an array for the control information of which storage space is reserved. No storage space for the elements of such a dummy array will be reserved. At the time of the procedure call the control information of the actual array will be placed into the locations reserved for the control information of the dummy array
procedure	1. a simple variable which contains, at the time of execution of the procedure, a jump order to the first order of the actual procedure 2. entities corresponding to the parameters of the procedure in accordance with this list

Specifications for the Oracle-ALGOL procedures, method of translation, and translation matrix with subroutines are described in a forthcoming report.[5] Coding has not yet started.

Syntax Checker and Other Features

One of the most useful features of an algebraic translator is its ability to recognize syntactical errors in the source program and to print these out in an understandable language. The translation process can be designed so that all syntactical errors are found. However, it seems advantageous to make a separate syntax check pass through the program. This relieves the main translation process of the burden of detecting and monitoring errors and of resuming the scan of the program after they have been found. Furthermore, the syntax check can be performed as the very first pass of the program, when the program is still in original form, so that all the information desirable for intelligible error monitoring is still available. There is some overlap between the functions of a syntax checker and those of a translator, and this might suggest that a separate syntax check pass is wasteful. We therefore give, as an additional justification for a separate syntax checker, some of the constituents of a translator which are not required in a syntax checker:

1. Optimal or "good" use of the facilities of a machine in the target program, which even in unsophisticated translators accounts for at least half of the instructions in the translator program.
2. Recognition of hierarchy in arithmetic expression.
3. Assignment of storage space for variables, arrays, and programs.
4. Change of identifiers corresponding to the assignment in 3.
5. Conversion of constants and assignment of storage space to constants.

Output of the syntax checker will be on the cathode ray tube; it will be in reference language. Some new ALGOL symbols have been added to the character plot routine for the Oracle curve plotter.

---

[5]See ORNL CF-61-3-1 (1961) (to be published).

Work was started on a syntax checker for the existing part of the translator, but was postponed pending incorporation of some of the additions now under preparation.

Any translator in small and intermediate size machines requires provision for segmentation. This allows a long target program to be stored in magnetic tape and called into fast memory in segments one at a time. A technique similar to that used in ORBIT is being used to code this feature.

Finally, *switches* are being arranged by the use of techniques of pseudo-addressing parallel to those used in handling arrays.

ALGOL ALGORITHMS

Bairstow-Hitchcock Iteration

An ALGOL procedure was designed utilizing most of the features of the program for solving polynomial equations by Bairstow-Hitchcock iteration available for the Oracle.[6] This was published in the literature[7] and was certified by another installation.[8]

Bessel Functions

This study was motivated by the appearance in the literature[9] of two ALGOL algorithms for computing I_n, the modified Bessel function of the first kind. One algorithm made use of a convergent series and the other made use of an asymptotic series. It was desirable to determine conditions on n and x which would indicate the more appropriate method to use in computing $I_n(x)$ in a given situation.

Computations carried out on the Oracle yielded two significant facts. First, the amount of computation necessary to satisfy a given accuracy criterion is often considerably less with the asymptotic series, while on the other hand use of the asymptotic series in some cases does not yield meaningful results. This failure of the asymptotic series is due in some instances to the small size of $|x|$ and in others to overwhelming round-off error.

An examination of the asymptotic series led to the relation $x \geq \max(10, \frac{1}{8} n^2)$ as a basis for preferring the asymptotic series, and this was confirmed by the computations over the range of values of n and x investigated.

A report is available[10] which includes ALGOL procedures for use of the series.

---

[6] G. Atta *et al.*, *Math. Panel Progr. Rept. Sept. 1, 1958 – Dec. 31, 1959*, ORNL-2915, p 13.

[7] A. A. Grau, "Solution of Polynomial Equation by the Bairstow-Hitchcock Method," ALGOL Algorithm No. 3, *Communs. Assoc. Computing Machinery* 3, 74–75 (1960).

[8] H. C. Thacher, "Certification," *Communs. Assoc. Computing Machinery* 3, 354 (1960).

[9] Dorethea S. Clarke, "Bessel Function I. Series Expansion" and "Bessel Function I. Asymptotic Expansion," *Communs. Assoc. Computing Machinery* 3, 240 (1960).

[10] L. L. Bumgarner, *Use of Convergent and Asymptotic Series for Computation of the Modified Bessel Function of the First Kind*, ORNL CF-60-12-47 (Dec. 12, 1960).

COOPERATIVE ALGOL EFFORTS

The benefits derived from the Oracle ALGOL translator go beyond having a convenient programming language for the Oracle. The main benefit lies in the ability to exchange programs with other installations with different machines. ALGOL is not yet universally accepted as the language in which to express computation processes and will probably undergo further changes before it will be accepted. Constant cooperation among computer users is necessary to keep the efforts towards a "final" universal language from diverging. The Mathematics Panel is involved in the following cooperative efforts:

1. ALGOL Maintenance Committee. This committee is set up to discuss interpretations of the ALGOL 60 language, to prevent hasty changes, and to make suggestions for the orderly adding of new statements or features to the language.

2. SHARE ALGOL Committee. The Mathematics Panel is writing a syntax checker for the SHARE-ALGOL translator.

3. Model Translator. During a visit to Mainz, Germany, A. A. Grau held discussions with F. L. Bauer, K. Samelson, and M. Paul of the Institut der Angewandte Mathematik, Johannes Gutenberg Universität, directed toward the description, for publication, of a translator for ALGOL 60 in machine-independent terms. This translator will benefit from the practical experience obtained in devising working translators at the two institutions and combine the best aspects of slightly diverging theories. The theory of translation is best described in terms of recursive subroutines based on syntactic structures with anticipatory features as developed at ORNL, while the fact that ALGOL is essentially a bracket-structure language permits the use of a particular type of switching program within the translator that permits greater speed. The formal translator program will incorporate the latter, developed at the Institut.

While the effort will be to incorporate the full ALGOL language, insofar as it is clear and unambiguous, there are features permitted by the report which will in all probability be seldom used but complicate greatly the translation process or the target program. In footnotes to the general translator it will be pointed out how reasonable restrictions in the language will greatly simplify both the translator and the target program.

PROGRAMMING RESEARCH

The efforts devoted to the completion and improvement in the existing ALGOL translator for the Oracle have been discussed above. Here are discussed plans and theory which it is not now contemplated will be used in a translator for the Oracle. Everything referred to in this section has been formulated in machine-independent terms and therefore can be applied to any machine.

Translator Theory and Design

The greatest machine-dependent problem that is encountered in devising a translator for the Oracle for any algebraic language is its limited memory (2048 words). This makes it impossible to adapt easily specifications for a translator designed elsewhere, even when formulated in a machine-independent fashion. For example, the existing Oracle ALGOL translator is like the translator used at the Institut der Angewandte Mathematik at Mainz, Germany, based on principles developed by Bauer and Samelson.[11] However, machine-independent plans for the latter could not be used for the Oracle, and the design was drawn up from basic principles, though the method of description in terms of symbol, number, and address cellars, building blocks, and translation matrix were retained. It turns out that it will be difficult, if not impossible, to implement the complete ALGOL 60 language for the Oracle without writing a complete new translator based on extensive refinement and revisions of the basic techniques.

Considerations of this kind have led to a re-examination of the Bauer and Samelson techniques in an effort to introduce as much simplification into the process as possible. This has resulted in a formulation which replaces their symbol cellar by a control push-down list whose elements are determined not by incoming symbols which cannot be processed immediately, but by the syntactic elements of the language. The translator is a set of recursive subroutines, that is, subroutines one of which may slave another including itself on a new level without loss of information on the old, where each subroutine is based on the syntactic skeletons defining an ALGOL concept. States of the translator correspond to important points in these skeletons. Adding a state to the control push-down, and deleting the last state, correspond to entry to or exit from the corresponding subroutine. The nesting of subroutines corresponds to the complete control push-down list. The auxiliary push-down list, which is a generalization of the address push-down of Bauer and Samelson, is a convenient device for storing systematically the information needed by the various subroutines.

The revised formulation of the process permits the use of an anticipatory device dictated by syntax. States are added to the control push-down when it is known from the rules governing the language that corresponding linguistic structures must follow. A final simplification is the separation of the control function of the binary and unary operations from their strictly individual operational meanings by a suitable choice of internal representation.

Specifications[12] for a translator have been drawn up in terms of this formulation. These have not been used as the basis for an actual translator, but they would be used for any new translator for any machine which the Mathematics Panel would undertake. An example of the simplification which is made possible is furnished by the fact that the matrix or main switch needed for processing the Boolean and arithmetic expressions of ALGOL contains 22 entries compared to a previous list of several dozen.

Translator-Oriented Computer Language

The problem of translating mechanically an algebraic language such as ALGOL into a machine language induces the realization that translation would be facilitated by a suitable choice of order structure on a machine. It is also apparent that such a choice of language likewise facilitates

---

[11] F. L. Bauer and K. Samelson, "Sequential Formula Translation," *Communs. Assoc. Computing Machinery* 3, 76–82 (1960).

[12] A. A. Grau, *The Structure of an ALGOL Translator*, ORNL-3054 (Jan. 23, 1961).

hand coding, since the mechanical translation of ALGOL differs in only minor respects from the conversion effected by a human translator (coder) in formulating in the language of a machine a problem stated in mathematical terms.

The translation of ALGOL based on push-down techniques leads to the design of a natural translator-oriented computer language.[13] It may be used with existing machines as an intermediate language, which can then be converted into machine language by using the methods of symbolic assembly programs. The possibility of incorporating some or all of the features in the hardware order structure remain to be investigated.

GENERAL PROGRAMMING

Linear Regression

A. M. Culkowski

Two additions were made to the Linear Regression and Correlation Code II. It is now possible to have an identifying label for each case run and to solve an equation involving only one coefficient.

Another version of this code is being prepared which, when completed, will provide a complete response surface analysis including the computation of two analysis of variance tables, eigenvalues, and eigenvectors as well as the regression analysis available at present.

Input and output has been changed to alphameric form and the values per treatment of the dependent variable may be given in groups with more than one observation per group.

The means are still given for the observation or design matrix and, in addition, the average response for each response matrix.

If one so chooses, the design matrix may be weighted, through input request, by the standard weight formula

$$\frac{1.0}{\left[\frac{\Sigma y^2 - (\Sigma y)^2}{n - 1}\right]} .$$

Any other weighting may be done by formula.

Upon completion, this code will be available as a canned program. At present the program is about 65% complete.

Scheduling of Drafting and Design Work

H. H. Bottenbruch

The scheduling of drafting and design in the Engineering and Mechanical Division at ORNL is presently done by the leaders of the different groups. The scheduling is a troublesome process,

[13] A. A. Grau, *A Translator-Oriented Intermediate Computer Language*, ORNL CF-60-9-66 (Sept. 21, 1960).

and it is very hard to get a schedule which is "smooth" in the sense that very important jobs are processed as quickly as possible and that there is little idle time. Idle time may arise because a member of a group must wait for the completion of certain parts of a project before he can start his part.

It was investigated whether a computer can design a schedule, thereby taking over the troublesome work from the group leaders, and at the same time come out with a better schedule.

There is not yet any positive answer to the question. Scheduling processes have been studied in the literature,[14,15] but there is no algorithm (other than exhaustive searching, which is prohibitive even in simple cases because of the vast number of possible different schedules) which will yield the "best schedule" or which will even recognize an arbitrarily given schedule as "sufficiently good." Scheduling processes have been programmed for electronic computers. These programs try a large number of possible schedules, the schedules being selected at random. By applying some rules of thumb one can reduce the sample to include only the supposedly more interesting cases.

Evaluation of Certain Definite Integrals of Atomic Physics by Analytical Integration on a Computer

H. H. Bottenbruch

The determination of properties of nuclei by the cluster theory as developed by Wildermuth and others requires the evaluation of integrals of the type

$$\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} P(x_1, x_2, \ldots, x_n) \cdot \exp[-Q(x_n, \ldots, x_m)]\, dx_1\, dx_2 \ldots dx_m \, .$$

Here P is a polynomial in $x_1, x_2, \ldots, x_n$; Q is a positive definite quadratic form in the variables $x_1, x_2, \ldots, x_m$; and $m > n$. Fairly interesting cases have values of n from 3 to 10 and values of m from 10 to 40; the degree of the polynomial is usually between 6 and 12.

Numerical integration is impossible because of the large number of grid points required. The integration may be performed analytically, but for hand computations this process is tedious, especially so since even for simple nuclei a large number of integrals of this type must be evaluated. The analytical integration can be performed by a computer. The integral is represented in a convenient way by storing the coefficients of the quadratic form Q and the terms of the polynomial. Integrating with respect to the variables $x_{n+1}, \ldots, x_m$ yields an integral of the same general structure, but with different coefficients in the quadratic form. These new coefficients can be computed from the original coefficients by a simple algorithm. The polynomial is treated as a constant.

---

[14] J. Heller, *Combinatorial, Probabilistic and Statistical Aspects of an M × J Scheduling Problem*, NYO-2540 (Feb. 1, 1959).

[15] R. Bellman, "Some Mathematical Aspects of Scheduling Theory," *J. Soc. Ind. Appl. Math.* 4 (1956).

Integrating with respect to the variable x_n gives an integral of the same form in the variables $x_1, x_2, \ldots, x_{n-1}$; the new quadratic form and the new polynomial can be determined from the previous ones by an algorithm which in principle is simple, but actually requires a fairly long program for the de- and encoding of the form in which the polynomials are stored. There are several ways in which one can store a polynomial in several variables with numerically given coefficients. One can store the coefficients in consecutive storage locations by arranging the terms in lexicographical order, reserving one location for each term of the polynomial which contains the coefficient of that term. This method is wasteful of storage space if many terms have coefficients equal to zero. It also requires time-consuming subroutines for the determination of the subscript value of a given term as well as for the determination of the exponent structure[16] of a term which is stored at a certain location. These operations, however, occur quite frequently in the integration routine.

One can also represent each term by giving explicitly its exponent structure in addition to the coefficient. Thus, the term $3.1 \cdot x_1^3 \cdot x_2^4 \cdot x_3^5 \cdot x_6^8$ would be represented by the numbers 3.1, 3, 4, 5, 0, 0, 8. Since the exponents are small positive integers, one storage location is usually sufficient to store the exponent structure of one term. It is sufficient to store only those terms with coefficients different from zero. Some estimates on computing and search time necessary in the integration routine suggested that it is best to store all the terms – also those with zero coefficients – and to arrange these terms in some fixed (lexicographical) order. An ALGOL program for the evaluation of these integrals has been written. Time and storage space estimates derived from this program indicate that a large number of interesting nuclei can be treated in this way. The more complicated nuclei, however, are beyond the scope of this technique.

Adaptation of Canned Programs

A. H. Culkowski J. H. Durfee D. A. Griffin J. J. Rayburn G. H. Stakes

Canned programs which require some coding and at least some knowledge of the program are used routinely for the problems of various divisions. The first four applications listed below made use of the nonlinear curve fitting code for the IBM 704, the fifth used an Oracle matrix multiplication code, and the last two applied the Oracle regression code.

1. From observed data for the concentrations of uranium diffusing into the aluminum cladding of a fuel plate, an attempt was made to determine the coefficient of diffusivity of uranium in aluminum and the maximum concentration of uranium in the fuel.

2. The determination of multipliers and exponents was the first step in an estimation procedure for the representation of acetylene final partial pressure in the equations for the C_2H_2-C_6H_6-radon system. Only one of the two terms was well determined by this method.

3. An empirical fit of the adsorption of gas on activated charcoal as a function of temperature and pressure was attempted. A simplification in the function had to be made before an adequate fit was obtained.

---

[16] The exponent structure of a term $f \cdot x_1^{l_1} \cdot x_2^{l_2} \ldots x_n^{l_n}$ is the sequence of numbers $l_1, l_2, \ldots, l_n$.

4. A study was made of the characteristics of silicon solid state counters by fitting data representing alpha particle fine structure of Cm^{244} to the sum of two Gaussians. The computed parameters were compared to known values.

5. To determine the radial distribution of radiation intensity in an axially symmetric plasma, an analytical solution which had been determined was evaluated by computing matrix elements and performing a matrix multiplication.

6. Fitting the observed intensity distribution of light emitted by an atom permitted the determination of the half-intensity width of Doppler-broadened spectrum lines.

7. In the calculation of spectrum line intensity from the photographic density of a line image, sequential use of the regression code was made.

BIOLOGY AND MEDICINE

M. A. Kastenbaum   J. E. Parham

Data Processing

Data emanating from studies carried on under the Biology and Medicine contract have been, until recently, analyzed with the aid of no more than desk calculators. During the past year a determined effort has been made to program many of these problems for analysis by the various computers available at ORNL, the Y-12 Plant, and ORGDP. The effectiveness of this endeavor is directly dependent on the ease with which the data can be prepared for processing by high-speed computers. To this end, proposals were made early in 1960 for the installation of a small data-processing unit in the Biology Division. By midyear one person on the supervisory level had been employed, and two IBM machines (type 026 card punch; type 056 card verifier) had been received on a rental basis. The data-processing unit was placed in operation soon afterward when one key-punch operator was hired.

In addition to the card punch and card verifier, the unit has purchased two each of five different types of control panels (IBM types 519, 085, 535, 608, and 407). These panels are being used in the processing of data on the IBM machines at the Y-12 Plant's Tabulating Service Department. Requisitions have been made to procure on a rental basis an IBM type 083 sorter, a type 085 collator, and a type 519 document originating machine.

At present, data from three sources are being punched on IBM cards. The first of these sources is the specific-locus-mutation experiments in the Mammalian Genetics and Development Section. In these experiments male mice are irradiated and subsequently mated with one or more females. The progeny are then studied for various specific genetic effects. The data from one of these experiments have been recorded on approximately 40,000 IBM cards. Control panels for the IBM 608 calculator have been wired to compute several statistics of interest from the data appearing on the cards.

A second major source of data is the animal autopsy records of the Pathology and Physiology Section. In these experiments mice of several strains are subjected to various doses of radiation at different intensities for the purpose of studying the pathological effects of irradiation on mammals. Data from approximately 40 experiments have been recorded on about 10,000 IBM cards. Tabulations and other forms of processing have been done on the equipment of the Y-12 Tabulating Service Department.

Finally, information from the new abstracting journal, *Statistical Abstracts*, has been punched on about 1000 cards.

At least four other senior investigators in the Biology Division have expressed an interest in having their data put on IBM cards. The nature and volume of this work is still not fully appreciated. However, if the enthusiasm of these investigators is indicative of the volume of material to be processed, then the capacity of the data-processing unit will be strained beyond its limits before very long.

Cytology and Genetics

The analysis of mutation processes in paramecia subjected to various doses of radiation and to different chemical mutagens has continued on a regular basis. Crude data representing the frequencies of individuals in various mutant and normal categories are prepared by processing on an IBM 610 computer. The computer has been programmed to convert the data by a variance-stabilizing transformation and to analyze the transformed values.

Rejoining of radiation-induced chromosome breaks in the nucleus has been observed in a pattern which may be explained by the existence of only a limited number of sites within the nucleus where chromosomes are close enough to rejoin if broken. For *Tradescantia* microspores, calculations based on a proposed model have indicated that the number of sites is only 4, whereas in *Vicia* it is 2. Using these numbers of sites it has been possible to characterize the observed distribution of chromosome exchanges as binomial in many cases. However, a number of experiments have yielded results which deviate substantially, not only from the binomial but also from other distributions such as the Poisson, negative binomial, Poisson-binomial, and double Poisson; as a result, serious doubt has been cast on the original model. Studies are proceeding on construction of a model which might better fit the biological material.

Mutants induced by exposure of extracellular phage T4 to ethyl methanesulfonate are found in mixed clones; that is, the treated phage individuals which have mutant descendants also have nonmutant descendants. Several underlying mechanisms of induced mutation that could generate such mixed clones have been considered. A mathematical model for mutagenesis based on certain simplifying assumptions was developed, and the results have been compared with observed laboratory data.

Pretreatment with AET in experiments with *Neurospora crassa* has failed to give any substantial protection against the mutagenic and lethal effects of x rays. However, it had been noted that the AET molecule could be rearranged according to the pH of the solution. It was therefore of interest to study the influence of pH during AET pretreatment. Survival curves were fitted to the data from several experiments, and comparisons were made on the effect of increased x-ray dose on survival for several levels of pH.

The effect of P^{32} on *Amoeba proteus* is to introduce a number of lethals in successive generations as this one-celled animal divides. Inasmuch as lethals are not observed during the first few generations, it has been conjectured that the P^{32} impairs some mechanism within the chromosome of the parent amoeba but not sufficiently to cause lethality. However, in the process of division, the impairment is duplicated and may be passed on equally to both sisters or preferentially to one sister. By this process some of the descendants in succeeding generations will acquire sufficient damage to cause death. The sensitive mechanism within the chromosome is assumed to take the form of elementary fibers or strands, one of which is damaged by the P^{32} in the parent amoeba. The total number of such strands is unknown. However, on the assumption that there are m sensitive strands in each chromosome, most or all of which must be damaged to result in death, it is not difficult to calculate the probability of finding a lethal in the nth generation, given that one of the m strands is damaged in the parent amoeba. The probabilities are generated by applying the theory of finite Markov processes in the presence of various initial conditions. Numerical results have been produced by the Oracle for the cases $m = 4, 8, 16$ and $n = 0, 1, 2, \ldots, 100$. The results of the computations have been compared with those reported earlier[1] as well as with data which are being collected at present in the Biology Division.

One effect of x radiation on male *Drosophila melanogaster* is to alter the sex ratio (of males to females) in the offspring. Reduction in the number of hatched eggs is attributable to both gametic and zygotic deaths. To determine the relative proportions of deaths which are gametic and zygotic, an experiment was proposed in which two types of males, attached $\hat{XY}/Y$ and attached $\hat{XY}/0$, are irradiated and mated with the same type of female. The different types of hatched offspring are then counted for the various doses of x rays used. Several mathematical models which incorporate parameters relating to gametic and zygotic deaths have been proposed and tested against the observed data. As yet none of these models has adequately described the observed frequencies. Studies of new models are being undertaken.

Radiation Protection of Living Cells

The adequacy of the multihit survival model was tested for data collected in several experiments with *Escherichia coli*. In all instances it was of interest to estimate the "extrapolation number" and to determine the goodness of fit of the model at the high dose values. The importance of a *weighted* regression analysis in securing an adequate fit was pointed up in all cases. Unless an appropriate adjustment is made at the high dose (low survival) levels, the values at the lower doses are unduly heavily weighted. This usually leads to a very poor fit at the high

[1] H. Friedrich-Freska and F. Kaudewitz, "Letale Spatfolgen nach Einbau von P32 in Amoeba proteus und ihre Deutung durch genetische Untereinheiten," *Z. Naturforsch.* 8b, 343–55 (1953).

dose points. In general, the weights are taken to be inversely proportional to the variance of the survival fraction.

Mammalian Genetics and Development

Progeny of male mice irradiated at different dose rates are known to have varying incidences of mutation at certain specific loci. Data from experiments in which Cs^{137} or Co^{60} had been applied at rates of 10 and 90 r/week were analyzed by weighted linear regression techniques. The object of the analysis was to determine whether the change in mutation rate with increased total dose is constant for the two different dose rates.

It has been suggested that the sex ratio in progeny of irradiated mice is different from that of nonirradiated mice. Experiments in which males are irradiated with 300 r of x rays and then allowed to mate with nonirradiated females have yielded data which can be compared with simultaneous control experiments. Chi-square tests for homogeneity of the data in the various experiments and normal deviate tests on the transformed ratios were employed in the analysis of these data.

The effect of x rays on human cells in culture is to produce a variety of aberrations which may be observed in the nucleus. The frequency of these aberrations has been demonstrated to increase linearly or quadratically with increased dose. Weighted linear and quadratic regression models were fitted to data involving the frequencies of deletions, isochromatids, interstitial deletions, and exchanges, and the results were compared with earlier, reported studies.

Mammalian Recovery

The shapes of x-ray survival curves of a number of human cell lines in tissue culture were investigated with the object of reappraising the estimates of the "extrapolation number" of sensitive targets reported in earlier studies. Several types of experiments were performed; in each experiment 40 control dishes and 20 experimental dishes were observed at each dose point. Colonies of 50 or more cells were scored as survivors after ten days of incubation. The results of the experiments were fitted to the standard multitarget model and estimates of the slopes and the "extrapolation number" were derived. Where the data failed to fit the standard model a more elaborate multiprocess survival curve was tried. All the fits were made by the method of least squares, in which the parameters are estimated by a graphical minimization of the sum of squares of deviations of the observed points from the prescribed model.

Radiation Immunology

It has been shown that, in the nonlethal x-ray dose range, the primary antibody response of intact mice is depressed but the secondary antibody response is not. In the midlethal dose range both responses are depressed. With heavily x-rayed isologous mice used for in vivo cultures of a known number of spleen cells from donors exposed to various doses of total body x radiation, the relative radiosensitivity of primary and secondary antibody-forming cells was determined.

The analytic procedures involved regression analyses in which the titer was fitted as a function of viable-spleen-cell dose. These data were shown to fit simple linear functions on logarithmic scales. Moreover, the analysis showed that, for varying doses of x rays, the fitted lines had slopes which did not differ significantly in the range of interest. It was therefore reasonable to estimate a single average slope for all the data. The fact that the lines for primary and secondary responses at all x-ray doses were parallel made it meaningful to use the difference in the intercepts as a measure of the depression of activity with increased x-ray dose. By the same token the relative antibody-forming activity could be estimated with exact confidence limits for the various x-ray doses. Finally, these estimates of the relative antibody-forming activity when plotted against dose of x rays were found to fit the standard multitarget curve.

An in vivo model based on the rejection (or functional inactivation) of spleen cells transferred into homologous mice, specifically preimmunized and exposed to 800 r of x rays, has been considered for the purpose of describing a quantitative assay system for the study of hematopoietic transplantation antigens. In this model the fate of the transferred spleen cells can be followed by their ability to produce agglutinins against an unrelated antigen (sheep rbc). To find the optimal experimental conditions, estimates were made of the rate of appearance of agglutinins in circulations in isologous and homologous situations. In addition, the relationship between survival of spleen cells in homologous recipients and dose of x rays (with 900 r, 100% of the transferred cells survive, whereas, with 500 r, almost all of them are rejected) was studied, and the RD_{50} (i.e., the dose of antigen capable of stimulating the recipients to reject 50% of the transferred cells) was determined. Confidence limits were computed for all the point estimates as well as for all the regression lines.

Pathology and Physiology

A long-term study has been proposed to define more precisely the shape of the dose-response curve in the range 10–400 r of acute radiation, to measure the effect of 10 r of acute gamma radiation, and to compare the dose response for acute (> 5 r/min) and chronic (< 10 r/day) gamma radiation. Experiments are to be conducted with large numbers of male and female RF and LAF_1 mice. The primary variables by which response will be measured are life shortening, myeloid leukemias, and ovarian tumors.

Intensive statistical analyses have been carried out on data from previous experiments to determine the minimum sample sizes necessary to detect dose effects at all dose levels. Of primary importance, however, were the low dose levels. All sample sizes were estimated only after consideration was given to the two types of statistical errors which could be made in the interpretation of the results. A type I error would be committed by stating at the end of the study that a detectable dose effect exists when, in fact, no such effect exists; a type II error would be committed by stating that no detectable dose effect exists when, in fact, such an effect does exist. The importance of errors of both types is immediately apparent in light of all the debates in recent years concerning the shape of the dose-response curve in the low-dose range.

The presently proposed sampling plan is a two-stage plan which is designed so that the second stage will not be necessary if the first stage gives satisfactory results. In this way, the total sample size, which involves many scores of thousands of mice, might be substantially reduced.

Another phase of this long-term study includes the serial sacrifice of animals exposed at various doses. The optimal number of animals to be sacrificed and the optimal times of sacrifice are dependent on the death rate of the strain and the rate at which radiation is applied. To this end the shapes of mortality-intensity curves for varying doses in previous experiments have been carefully studied. A considerable amount of preliminary study remains before the final plan of the experiment can be completed.

The effect of radiation on mean survival times, leukemia incidence, and tumor incidence was studied in RF female mice. Two groups of female mice received whole-body doses of 100 and 300 r of x rays, and three groups received doses of 300 r to the lower, middle, or upper parts of their bodies. The data were so analyzed as to compare the irradiated groups among themselves and with a group of control animals. A similar study of the effects of splenectomy and Sr^{90} on RF male mice was carried out with the same objectives in mind.

Nitrogen mustard derivatives are reported to be similar to x rays in their biological action. The LD_{50} doses of two drugs, methyl-bis-(β-choroethyl)amine (HN2) and triethylenemelamine (TEM), were computed for data on ten-week-old $(101 \times C_3H)F_1$ and RF mice and compared with the LD_{50} dose of x rays to determine the relative sensitivities of different tissues and organs to these agents. A similar study was done with nitrogen mustard on RF female mice to demonstrate the damaging effect of this drug on the brain tissue of normal mice.

Cell Physiology

One of the striking effects of subzero temperatures on yeast cells is the decrease in survival that occurs between −10 and −30°C when the water surrounding the cells is frozen. The functional relationship of survival to temperature and the physical state of the medium has been investigated. In addition, studies were carried out to determine some of the manifestations of injuries in cells rendered nonviable by low-temperature exposure. Observed changes in size, volume, and density, of frozen and thawed cells, were analyzed by analysis-of-variance techniques.

HEALTH PHYSICS

D. A. Gardiner M. A. Kastenbaum N. B. Alexander H. P. Carter

Ecology

Clinch River Studies. – At several stations along the Clinch River, experiments were conducted to attempt to determine the pattern of distribution of sediments in a plane across the stream. The experimental design chosen was the composite design in two dimensions (breadth and depth), and, wherever possible, the scale of the design was selected so as to make it rotatable. The responses measured in samples taken at the design points were: amount of suspended sediment and amount of total solids, both suspended and dissolved.

The designs called for samples to be taken at nine points in an imaginary plane perpendicular to the line of flow, and at almost all stations it was shown that real differences existed in amount of sediment from point to point. The attempt to describe the distribution of sediment by a full quadratic equation in the two factors, breadth and depth, was in general disappointing. Only in a small number of cases was the quadratic equation adjudged adequate to describe the differences in sediment existing from point to point.

The experiment showed, however, that "grab samples" (i.e., single samples taken at a random position at a station) are inadequate for estimating the amount of sediment in the stream. Experimentation will continue, using designs more elaborate than the two-dimensional composite.

White Oak Lake Bed Studies. – Among the objectives of the studies of small mammals residing on the White Oak Lake bed is the estimation of the population of various species. Different trapping procedures such as a capture-tag-release-recapture design may be used to estimate the number of animals of a particular species in a designated area. Various mathematical models have been proposed in the literature to describe the relationship between (y) the proportion of a sample which has been previously marked and recaptured and (x) the total number of animals that have been marked and returned to the population. The simplest model is a linear relationship in which y is directly proportional to x, with a proportionality factor given by the inverse of the population size. The failure of this model to give reasonable results in some cases has prompted many investigators to propose alternative models. Recently, an attempt was made to fit some of the White Oak Lake bed data to an alternative model which relates y to x as a power function. Least-squares estimates of the parameters were computed, and tests of hypotheses were performed. In general, the tests indicated that for the White Oak Lake bed data, population estimates based on the linear model were not significantly different from those based on the more complicated power function.

In order to study the variation in mycelium growth at different depths, data were collected by counting the growth on a number of slides which had been embedded in a homogeneous soil environment. Each count represented the number of times the threadlike filaments of fungus crossed the parallel lines of the microscopic field under observation. On the assumption that such counts are Poisson distributed, each observation was transformed to a new value equal to 0.5 plus the square root of the count. Alternatively, the counts were assumed to follow a Gaussian distribution, in which case no transformation was necessary. Both the transformed and the untransformed data were then analyzed by analysis-of-variance techniques to determine whether the number of mycelial filaments differed according to the depth.

Nonparametric methods of analysis were used in interpreting the data collected in the tissue analysis program of the Health Physics Division. These methods included the estimation of rank correlations, the two-sample sign test, and confidence-limit estimation on the median.

Waste Disposal

Waste-Water Treatment. – Following the successful application of response-surface methods for optimizing the efficiency of the lime-soda softening method of treating low-level liquid wastes reported in the preceding annual report,[1] a program of experimentation was initiated for the purpose of finding optimum points of operation for a treatment process utilizing the addition of sodium phosphate to the treatment mixture. The treatment variables in this study are amount of sodium phosphate, amount of excess soda ash, amount of clay, and proportion of stoichiometric requirement for lime. Also investigated were the effects of two clay types (Grundite and vermiculite) and the difference in effect caused by adding the phosphate at two different times during the treatment process.

The experimental designs chosen for the first stage of the program were four-dimensional simplices randomly oriented around the center of the region of the treatment variables. Definite evidence of better Cs^{137} removals was obtained from the use of Grundite clay with phosphate added at 11.5 min after the treatment process was begun. (Sr^{90} removals were unaffected by clay type and time of addition of phosphate.)

At one of the design points, removal of 98.8% of Sr^{90} was demonstrated, with an accompanying Cs^{137} removal of about 80%. Experimental designs are being planned for the neighborhood of the best Sr^{90} removals in an effort to improve the removal of Cs^{137}.

Dosimetry

Neutron-Dose Calculations. – Earlier estimates of neutron dose in the body were obtained by use of a slab model for the body. This is probably adequate to determine the maximum dose in the body for most neutron energies. However, dosimetry has advanced to the point where midline dose, average dose, and other concepts are of increasing importance. In particular, sodium activation in the blood is of major importance for the estimation of dose from neutrons, and this occurs throughout the body. Thus, better calculations, using a finite elliptical cylinder as phantom, are under way. A computer code is planned in order to permit easy modification to other phantoms so that it can also be used to obtain more accurate dosimetry of animal experiments. A similar code is under development for gamma-ray-dose calculations.

Nuclear-Recoil-Energy Distribution in a Spherical Fast-Neutron Dosimeter. – If a plane beam of neutrons of a given energy is incident upon a spherical dosimeter having a sensitive inner volume and a specified wall thickness, nuclear recoils may occur throughout the dosimeter volume. Some recoils occurring within the inner volume deposit only a fraction of their energy in the gas region because they escape into the wall, while recoils occurring in the wall may enter the gas, again depositing a fractional part of their energy within the gas. The information needed is the number of those recoils which deposit the energy in the gas in specified energy levels. Because of the fairly low neutron-interaction probability in a dosimeter, a Monte Carlo calculation of the problem will force the neutron histories to remain in the instrument until the mth collision. Since the contribution from first collisions

[1] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959,* ORNL-2915, pp 24–25.

in the dosimeter is expected to be quite important, these collisions are forced to occur uniformly throughout the volume of the dosimeter, and compensation is made by appropriately weighting a factor carried along with the neutron history. This problem has been flow-charted and programmed for automatic computation to the point where compilation and the securing of cross sections are the next steps.

METALLURGY

D. A. Gardiner

Boron Content in Fuel Plates

Metallurgists responsible for estimating the boron content of fuel plates fabricated for the core of the Army SM-1 reactor requested consultatory assistance in evaluating the boron content of sampled fuel plates. Standard analysis-of-variance techniques and confidence-interval estimates were recommended. The results of the evaluation are to be released in an ORNL report.

Analysis of HFIR Fuel Plates

The analysis of variance was used to obtain estimates of components of variance in the fabrication of fuel plates for the High Flux Isotope Reactor. The data consisted of 28 measurements, representing deviations from a predetermined involute form, on each of several fuel plates of three varieties. The object of the analysis was to estimate the basic components of variability in order to better understand the fabrication process and in order to determine the probability of meeting acceptance standards. Work is still proceeding on the latter phases of this project.

REACTORS

N. B. Alexander   H. P. Carter   D. K. Cavin   D. A. Gardiner
M. T. Harkrider   M. J. Mader   D. J. Wehe

Several calculations relative to reactors and associated technology were made by the Mathematics Panel. Almost all the work reported in this section was done on the IBM 704.

Equilibrium Concentrations

The program previously reported[1] for computing equilibrium concentrations of reactors has been modified and expanded into a program now called ERC-5 (ref 2). The ERC-5 program computes the

---

[1] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959*, ORNL-2915, p 27.

[2] *ERC-5 Program for Computing the Equilibrium States of Two-Region Thorium Breeder Reactors*, ORNL CF-60-10-87 (Oct. 20, 1960).

equilibrium states of two-region, fluid or solid fuel, thermal breeder reactors. Input consists of operating conditions (power, processing rates, volume of fuel and fertile streams, etc.), initial concentrations, and reaction-rate coefficients estimated by means of 34-group, multiregion diffusion calculations using the GNU program. Provision was made to treat cases where the fuel stream passes through end blankets, and where part of the fertile stream is passed through the core. In the case of solid-fuel fertile streams, a continuous bi-directional loading was assumed. Output consists of (*a*) equilibrium concentrations of fuel, fertile, and fission product isotopes; (*b*) distribution of neutron absorptions in the various isotopes; (*c*) distribution of fissions between fuel and fertile streams; (*d*) inventories; and (*e*) production rates.

Neutron Losses to Pa^{233}

The blanket of the aqueous homogeneous reactor is divided into three regions, and each region away from the core contains a smaller neutron flux. Neutron losses to Pa^{233} can be kept low by shifting the material in the inner high-flux blanket region to the outer regions for decay to U^{233}. A code called PLSB-1 has been written; it computes the ratio of neutron losses to Pa^{233}, when the blanket is shifted batchwise, to the neutron losses to Pa^{233}, when the blanket is mixed continuously.

ORR Fuel Inventory

A code is being written which computes ORR fuel inventory. The problem is in two sections: first, the calculation of the burnup of U^{235} after each unloading of the ORR, and second, the calculation of a new core loading from available fuel elements.

The problem makes necessary the keeping of a complete and readily accessible inventory of the fuel elements available for use. This inventory is currently being kept up to date and stored on the Oracle magnetic tape along with the program. The program is able to make any changes in the inventory as specified in a key word and automatically makes all changes necessary each time a core loading is removed from the reactor and the U^{235} burnup is calculated. All calculations are in fixed point for maximum speed. The program is designed for ease of operation and simplicity of input so that personnel from Reactor Operations may, with only limited knowledge of the machine, prepare the input and run the program. This section has been debugged and is currently being used.

The next section of the code will select elements for the next core loading of the reactor, taking into account certain elements which must be used and others which must not be used. A usable average value is computed using the four control rods. Then, other elements are selected by a matching method which minimizes errors progressively. The selection method attempts to use older elements if possible. This section should be ready for use in the near future.

Metal-Clad Fuel Elements

The computer program[3] which was written to estimate the mechanical life of metal-clad fuel elements has been used extensively with only minor revisions. The program is to be rewritten for the IBM 7090 in the near future.

---

[3] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959,* ORNL-2915, p 26.

Shift of Neutron Spectra

The program to determine the shift of neutron spectra in going from the middle of the moderator volume to the surface of the fuel element in a one-dimensional heterogeneous reactor[4] was temporarily abandoned. The program was successfully run for a few cases but it was discovered that the theoretical basis for the work was unsatisfactory for the other cases of interest.

Escape of Fission Products

A program which is in the final checkout stage has been written to determine the escape of fission products from an unshielded fuel element consisting of a homogeneous admixture of fissile material and a moderator such as graphite or beryllium oxide. The program computes the escape of the various fission products as a function of time, both in terms of fraction released and number of nuclides escaping per square centimeter of the fuel element. It also computes the total gamma radiation dose rate of the various fission products which escape from the fuel element.

Diffusion of Fission Products

In an experimental study of the diffusion of fission products in porous matrices, data were obtained on the rates of diffusion of alkali and rare-earth oxides from a depleting source enclosed in a hollow cylinder. The data indicated mass transfer combined with chemical reactions with the matrix. This process can be described by a partial differential equation with appropriate boundary conditions. Numerical solutions were obtained for this equation after converting it to finite difference form.

Half-Speed Whirl

The theoretical study of the instability phenomena in gas bearings known as "half-speed whirl" requires a knowledge of the values of three bearing parameters, namely, bearing load, eccentricity, and attitude angle. The values for these parameters are functions of the bearing geometry, lubricant viscosity, journal rotational speed, and ambient pressure. The values are obtained from the solution of the nonlinear partial differential Reynolds bearing equation. This equation has been set up for numerical solution on a computer, and solutions are being obtained for various bearings of interest. The computer solutions are used to design gas-bearing tests as well as bearings for hermetically sealed compressors for use with reactors and in other applications where normal lubrication is not permissible.

Heat Transfer

A subcommittee of the Atomic Energy Commission's Sodium Components Development Program Working Group is responsible for drafting a proposal for a shell-side heat transfer study. The study is intended to develop basic heat transfer information and relationships for the shell side of heat exchangers. The geometric variables chosen for study were: direction of flow over tubes, pattern

---

[4] *Ibid.*, p 27.

of tube array, tube diameter, and tube spacing. Operational variables were mass flow rate in tubes, mass flow rate in the shell, inlet temperature to tubes, and inlet temperature to the shell.

In cooperation with statisticians of North American Aviation, Inc., an experimental program incorporating the geometric and operational variables was devised and submitted for approval. The experimental design developed was one which would allow for analysis of the data by multiple-regression techniques along the lines of response-surface methods.

PHYSICS

N. B. Alexander
D. Arnurius
G. J. Atta
S. E. Atta
N. A. Betz
H. H. Bottenbruch
H. P. Carter
D. K. Cavin
A. H. Culkowski
N. M. Dismuke
A. C. Downing
J. H. Durfee
M. B. Emmett
R. Faudree
R. D. Freeman
W. Gautschi
D. A. Griffin
M. T. Harkrider
J. G. LaTorre
J. J. Rayburn
G. H. Stakes
J. Wehe
M. C. Ward

Transmission Data for Neutron Cross Sections

Transmission curves obtained with the "fast chopper" exhibit dips corresponding to resonances in the total cross section of the sample material. The experimentally observed shape of a resonance differs from the true shape because of distortions introduced by finite instrument resolution and by the Doppler broadening resulting from the thermal motion of the atoms in the sample. The corrections required to account for these effects greatly complicate the relationship between the transmission and the total cross section. Hence the analysis is carried out directly on the measured transmission curve.

It was assumed that the cross section is represented by the sum of single-level Breit-Wigner formulas, with interference between resonance and potential scattering but no interference between resonances; that the Doppler broadening is represented by a Gaussian function; and that the instrument resolution is represented by a Gaussian function. The instrument drift was fitted to a second-degree polynomial by the method of least squares.

The distortion to the transmission curve becomes more serious for resonances at higher energies and for those having smaller total widths. In the high-energy region the shape of the dip in the transmission curve is determined almost completely by the resolution function and the sample thickness. The method for analyzing resonances in this region is based on the area of the transmission dips. The accuracy of the area analysis method is reduced to determining only the neutron widths of the resonances for assumed total widths.

In the low-energy regions of good resolution, the problem was to find an adequate and practical method of analysis of the nonlinear function to determine the energies, the total widths, and the neutron widths of the resonances. Several methods of approach were studied or considered.

The trial-and-error method consisted of choosing trial values of the Breit-Wigner parameters and then computing the effect of the resolution and Doppler broadening on the curve. Sets of such computed curves were compared with the observed curve, and the best fit was selected.

Another method considered was to separate the resolution function from the transmission curve by the use of Fourier integral transforms and, hence, determine the Breit-Wigner parameters directly from the cross-section curve.

The least-squares method for solving such a nonlinear problem used the Gauss method of reducing the problem to one to which linear methods could be applied. The nonlinear function was linearized with respect to each of its parameters by expanding into a truncated Taylor series. New estimates for the parameters were obtained by using the initial estimates of the parameters to evaluate the coefficients in the expansion. The process was repeated until some convergence criterion was satisfied.

The difficulty connected with any approach to the problem lay in the unreasonably large amount of computer time needed to evaluate so many double integrals. However, this difficulty was mitigated by representing the Breit-Wigner cross-section integral in terms of the real and imaginary parts of the complex probability integral, which could be evaluated very accurately with a small amount of computer time by the use of tables and recurrence formulas.

An area analysis program and a least-squares shape analysis program have been written for the IBM 704 computer. A considerable amount of difficulty has been caused by the computational error arising from the use of recurrence formulas in these programs.

For the area analysis program, the transmission is analyzed in groups, with no more than 12 resonances in any group. The shape analysis program solves for the parameters, total widths, neutron widths, and resonance energies for as many as four resonances at once by the least-squares method.

Preliminary processing of raw transmission and capture cross-section[1,2] data is carried out on the Oracle. An output alternative is to punch the processed data on paper tape and to transcribe to cards for 704 input to the transmission analysis codes. Some changes and additions have been made to the processing code.

Miscellaneous Codes

In connection with a study of capture cross-section measurements an Oracle code was written for computing the average activation cross section for a spectrum of neutrons as a function of boron thickness.

A program for computing resonance parameters from low-energy neutron cross sections of fissionable nuclei[3] was revised so that it could be used with the 704 monitor and so that the code could be

---

[1] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959,* ORNL-2915, p 27.

[2] R. C. Block, G. G. Slaughter, and J. A. Harvey, *Phys. Div. Ann. Progr. Rept. Feb. 10, 1960,* ORNL-2910, pp 35, 41.

[3] N. J. Pattenden and J. A. Harvey, *Phys. Div. Ann. Progr. Rept. Feb. 10, 1960,* ORNL-2910, p 42.

placed on a reserve tape as a permanent record. To write the code (which was in sections and on cards) on a reserved tape, a SHARE distributed program was modified and used. A complete description of the program and its use is available.

Nuclear structure investigation[4] by Coulomb excitation experiments makes use of a set of integrals over the range of the ion going into the thick target. In the integrand are two functions available in tabular form. One of the functions, the Coulomb excitation function, is independent of the nuclear properties of the ion beam and target; the other function, the stopping power, depends on both ion charge and target material. To carry out the integration an attempt was made to fit the two functions with an elementary function representation. For light nuclei it was possible to fit the stopping power, but a table with interpolation was used to evaluate the excitation function. For heavy ions both functions were evaluated by interpolation from a table.

A study of the Auger process in several inert gases and possibly in other elements is being made by means of a Monte Carlo calculation. The charge distribution following the removal of a K or L electron, the various modes by which an Auger process can occur, and the probability of occurrence of a particular mode are some of the questions of interest. An Oracle code for argon was written; for krypton and possibly xenon and other elements, a 7090 code will be used. The need for a fast machine with a large store increases with the complexity of the atom.

A code for computing gamma-ray spectra was written to calculate, without broadening for resolution, the pulse-height distribution of light pulses due to gamma-ray interactions in a liquid-scintillator tank of variable dimensions. The code computes by a Monte Carlo technique the pulse-height distributions due to several coincident gamma rays of varying energy, including summing effects. Input data are pulse-height distributions of single gamma rays which were computed by C. Zerby of the Neutron Physics Division. The purpose of the code is to obtain a better understanding of observed gamma-ray pulse-height distributions in radiative neutron capture experiments.

To calculate ion trajectories in electrostatic fields a new program for the 704 has been written and partially debugged. Original formulation of path trajectories has been changed since a previous report,[5] and the changes have been incorporated in the new program. Potential distribution is still computed with an Oracle code, and transfer of these potential values is then made from Oracle magnetic tape to Oracle paper tape to IBM cards.

In the Brueckner theory of a many-fermion system it is of current interest to investigate reaction-matrix singularities and bound-state solutions of the Bethe-Goldstone wave equation for an interacting pair in a medium. It has been found possible to reduce this problem for a one-dimensional system with two-particle interaction $v(x) = -g/(c^2 + x^2)$ to a differential equation plus boundary condition.

An IBM code has been written to compute the positive-energy bound-state, negative-energy bound-state, and continuum solutions of the wave equation from a Runge-Kutta integration of the differential equation. The integration is carried out to the accuracy specified by means of an input parameter.

---

[4]F. K. McGowan and P. H. Stelson, *Phys. Rev.* **109**, 901 (1958).

[5]*Phys. Div. Ann. Progr. Rept. Feb. 10, 1960*, ORNL-2910, pp 74–87.

The computation of the neutron-stripping cross section, the scattering and polarization of spin-one particles, and the study of scattering in terms of an optical model with spin-orbit coupling[6] will require several segments and program modifications. A 704 program has been written by the Electronuclear Research Division with assistance from the Physics Division and the Mathematics Panel. In one segment of the code, several alternative neutron bound-state functions have been programmed. A solution proportional to an exponential function, the confluent hypergeometric function, and an iterative method for calculating the eigenvalue and corresponding wave function were coded. The entire program will be transferred to the 7090.

The code written for the phenomenological nuclear-potential study is still modified and used as new experimental data are examined.

Inelastic scattering of polarized relativistic electrons by atoms has been calculated in a modified first Born approximation. In order to evaluate the matrix elements, the Thomas-Fermi model of the atom has been used. The results show that the asymmetry coming from the inelastic scattering makes a negligible change in the asymmetry to be attributed to the total, inelastic-plus-Mott, scattering. G. Felsner and M. E. Rose are preparing a paper on the calculations and results, to be published in *Il Nuovo Cimento.*

In the application of the spherical harmonics method to neutron transport theory[7] a set of integrals has been given for (1) the distance of the extrapolated end point, (2) the linear extrapolation distance, (3) the density at the boundary, (4) the current density at the boundary, and (5) related normalization factors.

These integrals are all improper at the lower limit, having the form

$$\int_0 f(x)\,\frac{dx}{x(\ln x)^2}\;,$$

where $f(x)$ is continuous at $x = 0$. This particular singularity makes the integrals extremely difficult to evaluate numerically while retaining more than four significant digits. The loss of significance is greater for smaller values of the multiplication constant.

These integrals have been evaluated for multiplication constants between 0.99 and 0.45 with both positive and negative values for the constant of anisotropy. The tabulated results are to be published in a forthcoming ORNL report. It is hoped that future study of the weight function $x/(\ln x)^2$ will permit the extension of the table to values of the multiplication constant between 0.45 and 0.05.

For a study of penetration of gamma rays by the Neutron Physics Division a Monte Carlo program has been initiated. The code will be fairly general in that various source geometries, various media, and various boundaries can be incorporated. Of primary interest is the study of

---

[6]G. R. Satchler, *Phys. Div. Ann. Progr. Rept. Feb. 10, 1960,* ORNL-2910, pp 4–5.

[7]W. Kofink, *Studies of the Spherical Harmonics Method in Neutron Transport Theory: I,* ORNL-2334 (May 11, 1957); *II,* ORNL-2358 (July 11, 1957); *III,* ORNL-2850 (Oct. 21, 1959); *IV,* ORNL-2901 (Feb. 8, 1960).

deep penetration, especially with the use of conditional Monte Carlo.[8] At present, subroutines are being rewritten to conform to FORTRAN and FAP for use on the IBM 7090.

The calculation of the saturated flux of certain foils was coded for the same division. A counter was set up for the foils, and from this counter the flux, the mean, and the standard deviation of the fluxes per foil were computed.

Computation of the axial and radial field components[9] was programmed on the 704 for any general configuration of coaxial coils or coil pair.

A second part of this program permits the optimization of the currents in the individual coils to obtain the best least-squares approximation to a uniform field at some specified distance from the axis. This program was used to verify coil configuration designs of DCX-2.

The method chosen was based upon the integrals developed in a report.[10] Since the primary interest was in accuracy, the code was designed to use 4-, 8-, or 16-point Gaussian integration as needed for the double integrations. By comparing results it was found that four-point Gaussian for coils distant from the point of interest and eight-point Gaussian elsewhere gave accuracy of at least six significant digits.

Computations were made to determine the quality of focus for particles a considerable distance outside the midplane in an inhomogeneous magnetic field. The field is invariant with respect to the azimuthal coordinate ϕ and possesses mirror symmetry with respect to the plane $\theta = \pi/2$. The solution of the equations of motion of particles moving in this field is to be used to determine the focal characteristics of a proposed electromagnetic separator using an inhomogeneous field.

The initial positions, initial velocities, and the equations of motion of an ion of charge c and mass m were given. These equations were normalized by using a dimensionless coordinate system, and the Runge-Nystrom method was used in solving the resulting three second-order differential equations in which first derivatives were present.

Measurements of the infrared absorption spectra of polybutadiene before and after irradiation showed peaks characteristic of each of the unsaturated species. Required are the concentrations of three isomeric unsaturated groups in polybutadiene rubber before and after exposure to radiation. The infrared absorption was then measured for various concentrations of hydrocarbon standards for each of the unsaturated groups. An expression for the infrared absorption of each characteristic peak as a function of concentration was computed by the least-squares method of curve fitting.

Since each unsaturated group shows slight infrared absorption at the peak characteristic of one or more of the other groups, concentrations could not be calculated from the individual absorption-calculation expressions. Instead, a set of simultaneous equations was formulated allowing for this interference of one species at the peak characteristic of another group. This set of equations for three unknowns, containing terms to the third power, was solved by an iterative method for two polybutadiene samples.

---

[8] S. K. Penny and C. D. Zerby, "Examination of the Range of Applicability of Conditional Monte Carlo to Deep-Penetration Problems," *Neutron Phys. Div. Ann. Progr. Rept. Sept. 1, 1960*, ORNL-3016, p 209.

[9] *Math. Panel Ann. Progr. Rept. Dec. 31, 1959*, ORNL-2915, p 31.

[10] N. B. Alexander and A. C. Downing, *Tables for a Semi-Infinite Circular Current Sheet*, ORNL-2828 (Sept. 29, 1959).

CHEMISTRY

D. A. Gardiner M. M. Goff D. J. Wehe

Power Reactor Fuel Processing Plant

After a temporary shutdown of the Power Reactor Fuel Processing Plant in late 1959 a material balance of plutonium became of urgent interest. During the processing of spent fuel in the S-240 and H-240 programs strange and temporarily unexplainable data began to appear in routine reports, with the consequence that considerable uncertainty was attached to estimates of feed, product, and losses. A material balance calculation was performed incorporating estimates of precision of the various data entering the calculation.

The estimates of precision involved the propagation of errors in many complicated forms and necessitated in some cases the development of correction equations to obtain better estimates. The results of the calculation were reported in a letter to the Director of the Chemical Technology Division.[1]

Synthetic Pilot Plant Feed Solutions

An experiment designed by workers in the Analytical Chemistry Division yielded data on performance of analysts in three laboratories. The analyses of plutonium content of synthetic samples formed a split-plot experiment, and the analyses of uranium content formed a split-split-plot experiment.

The data were analyzed to obtain estimates of bias and precision for each of the laboratories. Due to the type of experiment used to obtain the data, precision estimates were obtainable only in approximate form and sometimes were biased estimates. Exact tests of uniformity of individual analysts and of uniformity within laboratories were possible and were performed.

Physical Inventory of U^{233} Solution

An estimate of the U^{233} content of solutions processed in the Power Reactor Fuel Processing Plant during the later months of 1959 was calculated from data on 13 batches of solution. Estimates of error involving the use of approximate propagation-of-error formulas were also calculated.

A new and different method of estimating calibration curves for two process tanks was tried, the properties of which are not yet fully assessed. The model for this method may be expressed as

$$Y_t = \alpha + \beta x_t + \epsilon_t$$

in which Y_t represents the liquid level read from a recorder after the tth increment of solution is added to the tank, x_t represents the volume of solution in the tank after the tth addition, and ϵ_t is a random error given by

$$\epsilon_t = \delta_t + \rho\delta_{t-1} + \rho^2\delta_{t-2} + \ldots + \rho^{t-1}\delta_1 .$$

---

[1] ORNL CF-60-3-53 (Mar. 21, 1960).

In the formula above, δ_i ($i = 1, 2, \ldots, t$) is a random error independent of δ_j ($i \neq j$), and ρ is a measure linking prior errors with the current error.

In estimating α and β it was also assumed that ϵ_t was proportional to $x_t - x_{t-1}$.

Calibration of U^{233} Carrier

A special vessel fabricated to carry U^{233} solutions from the ORNL site was used to examine two different methods of calibration. In the first method, which is used more commonly and universally with larger vessels, carefully measured increments of solution are added to the vessel, and liquid-level measurements are observed. The increments accumulate in the vessel until it is filled. In the other method, carefully measured volumes were poured into the vessel, the liquid-level measment was observed, and the vessel was then emptied before another volume of solution was added. The several volumes chosen were put into the carrier in a random order.

From each set of data collected a linear calibration curve of the form $Y = a + bx$ was estimated, using ordinary least-squares procedures. The residual mean squares were also calculated and could not be judged significantly different at the 10% level of significance. However, the residual mean square for the cumulative method was smaller than that for the random method by a factor of 3.25.

Radioactivation Analysis

A recent innovation in radioactivation analysis is to speed up and somewhat automate the analysis by programming digital computers to analyze gamma-ray spectra as they are received from a pulse-height analyzer.

The problem of analysis, in the large, is to first detect photopeaks in the presence of Compton effects and other noiselike processes, second to identify the isotopes producing the radiation, and finally to determine the contribution of each isotope's activity to the observed photopeaks. This last step in analysis is sometimes accomplished by a series of successive subtractions of radioactivity (after the isotopes have been identified) and is sometimes called "spectrum stripping."

Two recently proposed procedures[2,3] for accomplishing the above have been thoroughly studied and found somewhat less than satisfactory in regard to the very first part of the problem, that is, the detecting of true photopeaks in the presence of Compton effects and random processes. The complex gamma-ray spectrum may be looked upon as an empirical distribution of known functional form (a Gaussian form superimposed upon a rectangular distribution, perhaps) but with unknown parameters. The specific problem here has been and still is to find a test, with known probability properties, to locate true Gaussian photopeaks among other disturbances.

The literature of statistics has been and continues to be diligently reviewed with the above goals in mind.

---

[2] W. E. Kuykendall *et al.*, *Computer Techniques for Radioactivation Analysis*, Office of Isotopes Development, USAEC, Texas Engineering Experiment Station, Texas A and M College System, TEES-2565-1 (May 1, 1960).

[3] J. W. Nostrand, Jr., *et al.*, *704 Program Report, The Reduction of Gamma Spectral Data from a 100-Channel Pulse Height Analyzer*, Grumman Aircraft Engineering Corporation, Bethpage, N.Y., RM-175 (April 1960).

New programs have been written for the Oracle in anticipation of the installation of the automatic neutron diffractometer at the Oak Ridge Research Reactor. The equipment reads a control paper tape in which is punched the commands for setting the dials of the orienting equipment. When the orienter receives a command to "take data," a punch records the counts as the spectrometer scans through the reflection. This output must be processed to extract the intensity of the reflection. After a crystal has been oriented on the spectrometer, a control program prepares the control tape of instructions, a printout of the reflections to be scanned, and a data-identifying tape to be used with the processing code. The processing program determines the intensity of each reflection and reports this with identification. It is planned that these data will in the future be written on the magnetic tape which is compatible with the 704 so that the data desired for other programs may be easily extracted. A by-product of the two programs was one which gives a printed listing of the data tape.

When the orienting equipment is in operation, much of the data produced will be three-dimensional in nature. Therefore a 704 program for calculating the absorption correction coefficient of a crystal was written to handle both two-dimensional and three-dimensional data. The program is similar to one written for the Oracle.[4] The report on this program is nearly complete.

Projects in the future include codes which will ease the transition from one type of calculation to another.

Crystallography

The programming efforts in this section consist of the revision of existing programs necessitated by a change in computer equipment or standards. Work was done on four programs: ORXLS (ref 5), A Crystallographic Least Squares Refinement Program for the IBM 704; ORXFE (ref 6), A crystallographic Function and Error Program for the IBM 704; MIFR1 (ref 7), Two- and Three-Dimensional Crystallographic Fourier Summation Program for the IBM 704; and FØUR PLØT (ref 8), a contour plotting program. Some modifications have been effected to make the programs compatible with the monitor system for the 704. Further modifications are now in process to make the programs compatible with the IBM 7090 and its monitor.

---

[4]W. R. Busing and H. A. Levy, *Acta Cryst.* 10, 180 (1957).

[5]W. R. Busing and H. A. Levy, *ORXLS, A Crystallographic Least Squares Refinement Program for the IBM 704*, ORNL CF-59-4-37 (April 1959).

[6]W. R. Busing and H. A. Levy, *ORXFE, A Crystallographic Function and Error Program for the IBM 704*, ORNL CF-59-12-3 (December 1959).

[7]W. G. Sly and D. P. Shoemaker, *MIFR1, Two- and Three-Dimensional Crystallographic Fourier Summation Program for the IBM 704*, Massachusetts Institute of Technology, Cambridge, Mass.

[8]A. Zalkin, *FØUR PLØT*, Lawrence Radiation Laboratory, Livermore, Calif.

ORACLE OPERATION

E. C. Long E. McDaniel S. O. Smith G. H. Stakes

Machine operations have been maintained on a three-shift five-day-week schedule. The good operating time and the operating ratio (good computing time/scheduled computing time) are as follows:

	Monthly Computing Time (hr)	Operating Ratio (%)
High	461	98.3
Average	350	95.2
Low	199	89.5

Mathematics Panel personnel used 12.1% of the machine time.

Figures 1 and 2 indicate the hours of machine time used by various divisions during calendar year 1960 on the Oracle and the IBM 704 respectively. The length of the bar represents the total time and is the sum of Mathematics Panel programmed time and time programmed by other personnel.

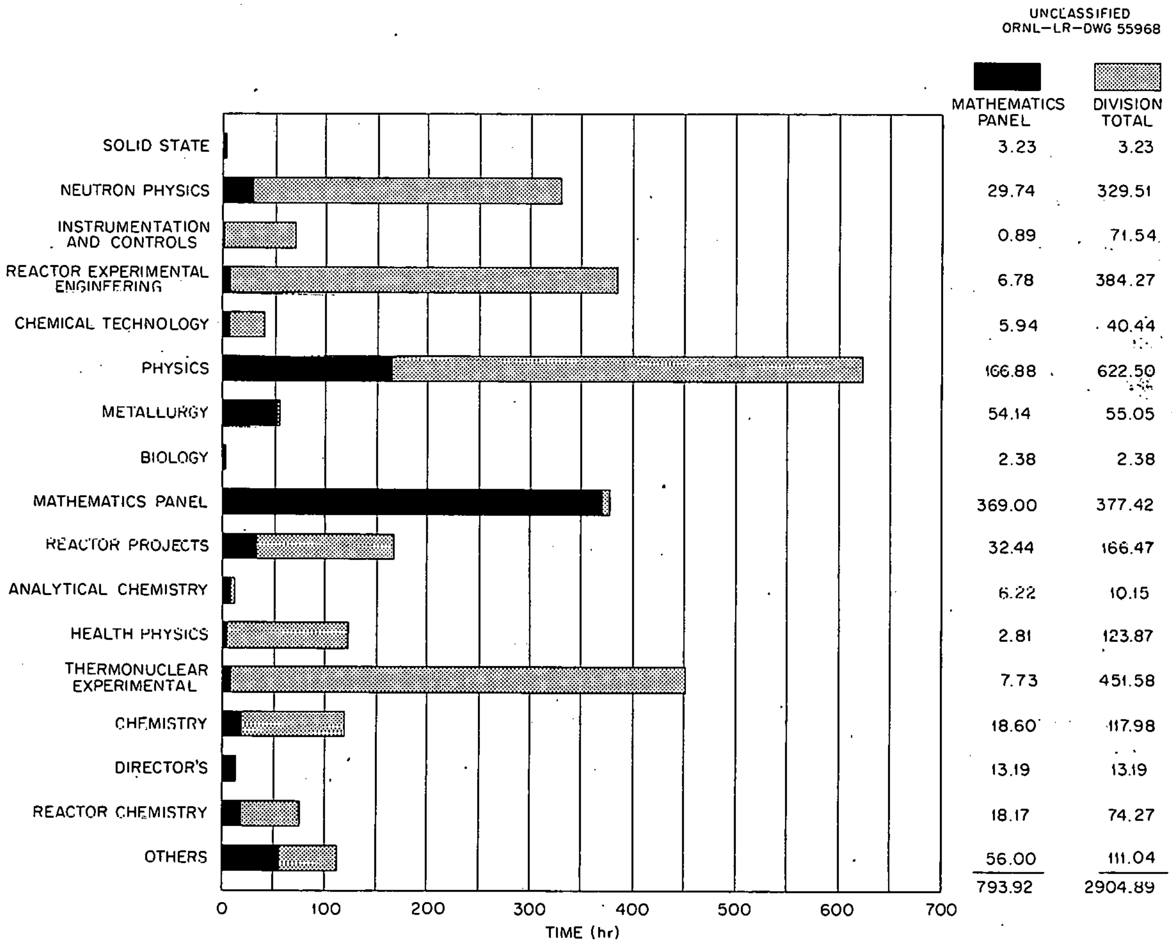

Fig. 1. Oracle Time Used by Divisions During 1960.

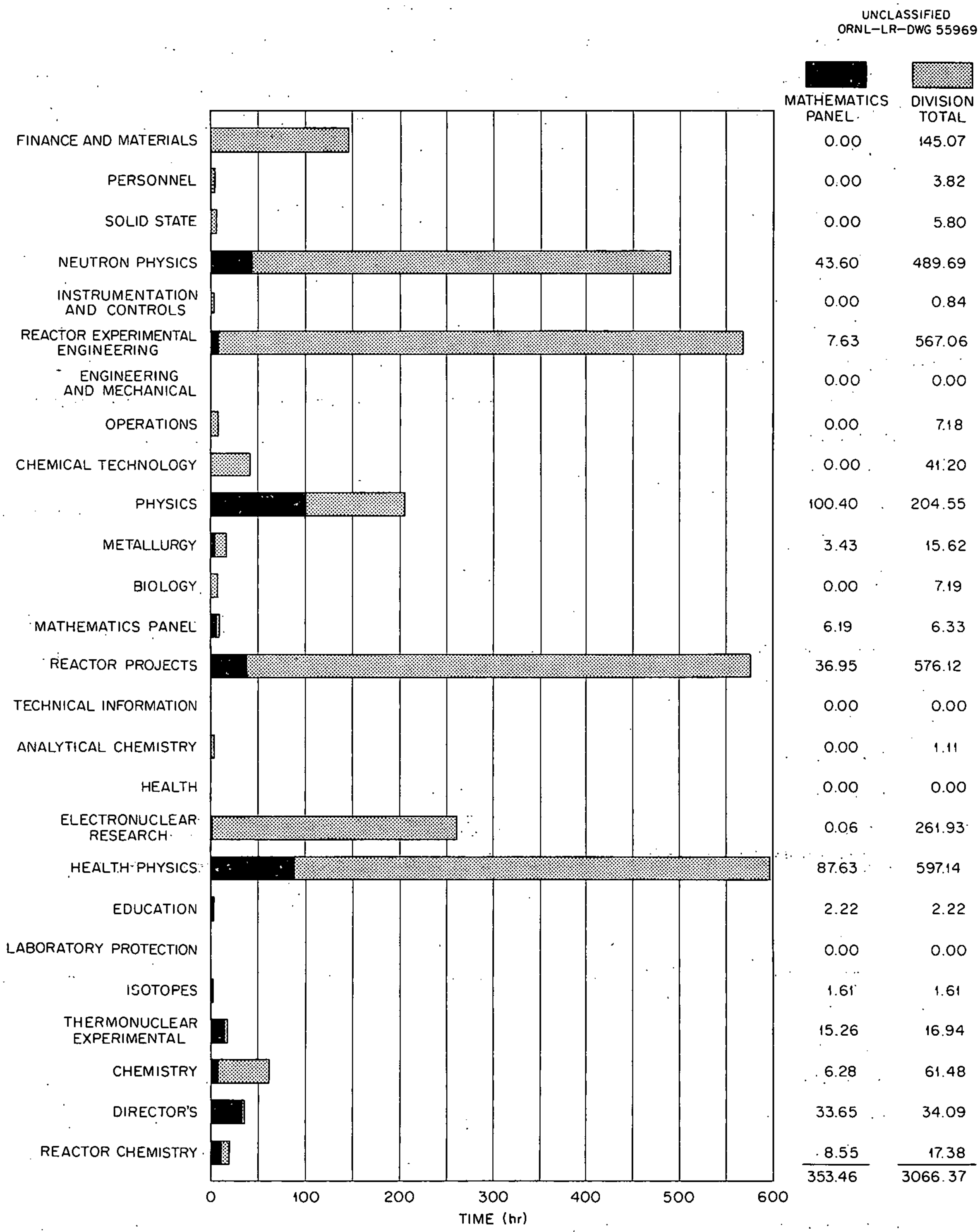

Fig. 2. IBM 704 Computer Time Used by Divisions During 1960.

The Oracle-IBM liaison magnetic tape unit is scheduled for installation during the early part of calendar year 1961. This unit of the Oracle will permit reading and writing IBM-type tapes on the Oracle in IBM binary coded decimal mode at a density of 200 characters per inch. All reading and writing will be by records. Reading and writing in unedited Oracle mode will also be available. In addition to allowing communication with other machines, this tape will also serve as a fifth tape unit for the Oracle.

Orders for the tape unit include, in addition to reading and writing in either IBM or Oracle mode, the following: rewind to load point, hunt forward or backward one record, and write end of file.

Figure 3 indicates how the Oracle time was used during the year. Scheduled computing time consists of production and debugging time, as well as idle time unavailable (I_u), lost time (L), repair time (R), and test time (T). Idle time unavailable is machine outage due to power failure or other trouble not due to malfunction of the machine. Engineering time is composed of maintenance time (M) only.

During the seven years of Oracle operation an efficient system of machine maintenance has been developed. Each week, $15\frac{3}{4}$ hr are reserved to provide for preventive maintenance on the Oracle. This maintenance consists largely of the following:

1. Tubes (approximately 150 each week) are replaced. These replacements are scheduled in a manner to ensure that no tube remains in the machine longer than 6000 hr. There are certain exceptions to this rule. For example, in the Clear and Driver chassis, the 3C33's will not normally last 6000 hr. Hence, these chassis are replaced on a separate schedule.
2. Diagnostic test routines are run to determine the condition of the various units.
3. Magnetic heads and transports are cleaned three times each day.
4. Voltage and current readings on all the units are logged once each week.
5. A performance check on the electrostatic memory is made, and the read-around errors at a ratio of 400 are logged once each day.
6. Records are kept of the number of blocks of wide magnetic tape used each day.
7. All amplifier gains in both the fast and auxiliary memory are made and logged once a month.
8. Error cards are logged and filed.
9. The $3\frac{1}{2}$-hr period on Monday morning is used to make repairs that the shift technician has been unable to do the preceding week. For example, the replacement of a tape transport requires a minimum of 2 hr, with two to three men working, assuming that no undue difficulties are encountered.

Various pulses that are subject to drift, such as the output from the Ferranti reader, are checked and, if need be, adjusted.

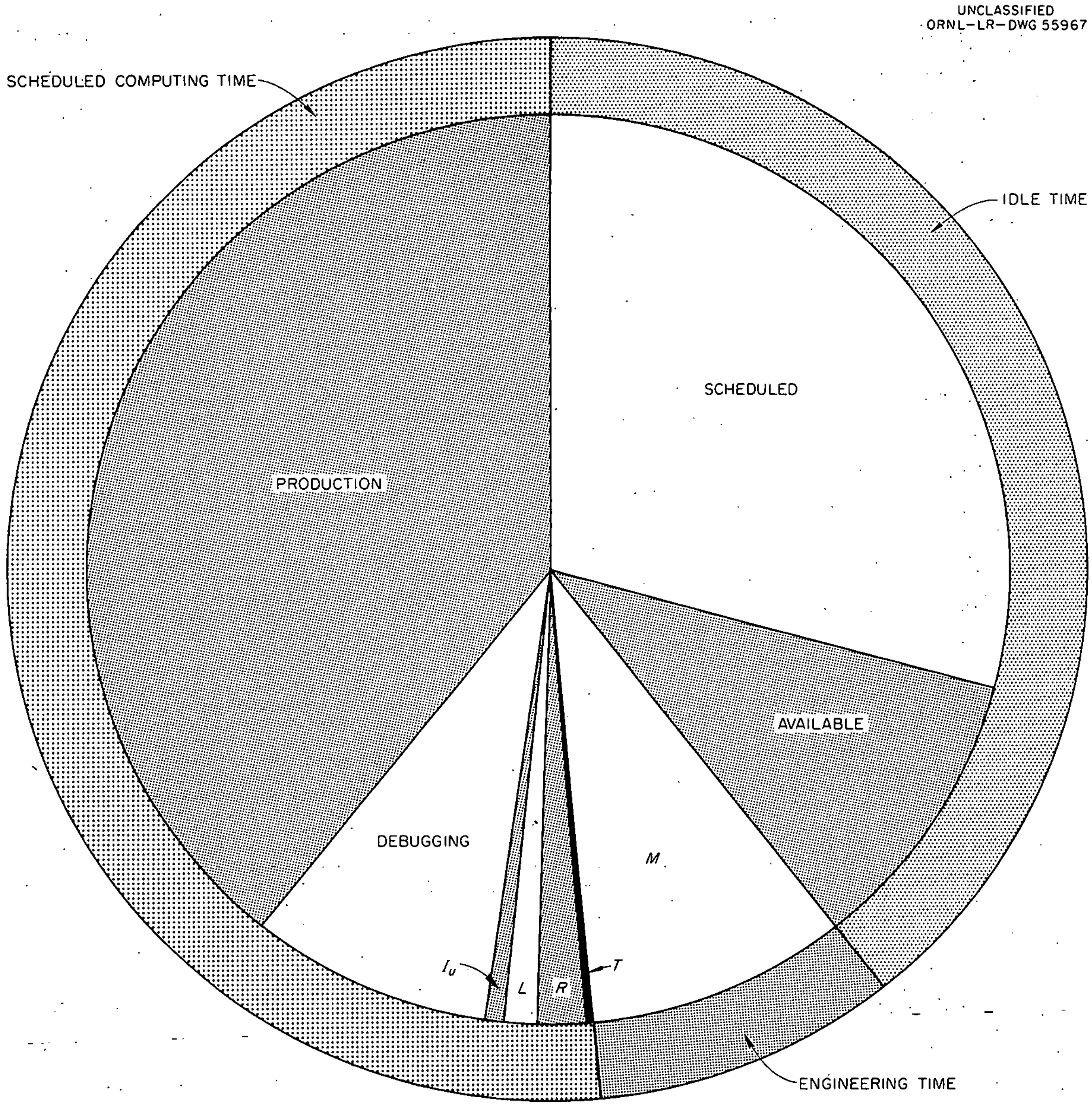

Fig. 3. Use of Oracle Time During 1960.

More extensive tests of the electrostatic and auxiliary memories are possible because of the extra time and manpower that are available at this time. These tests include checking the auxiliary memory with half the amplifiers disabled, shifting the memory pattern (this prolongs the life of the CR tubes), and checking the memory clock pulse train.

10. Complaints of machine malfunctions of a random or intermittent nature are investigated under conditions designed to make the malfunction a recurrent one. If this is possible, it facilitates the correction considerably.

As is well known, the great majority of machine malfunction is a result of vacuum tube failure. Some of these failures are very easily located, while others are extremely difficult to find. For example, if there is an intermittently failing tube in the arithmetic unit, which contains nearly 2000 tubes, the time necessary to find it may vary from a few minutes to several hours. Therefore, it is imperative to be extremely careful of the quality of tubes installed in this unit. Only premium tubes are purchased, and these are subjected to stringent tests. Thus, the rejection percentage is probably somewhat higher than normal. Out of a representative batch of 2125 new tubes tested, 101 were unacceptable. Even though this 4.75% rejection seems rather high, it is a considerable improvement over the first year of operation of the Oracle, when the rate on some types of new tubes was as high as 20 to 25%. Data on all tube failures are kept as follows:

1. tube type,
2. manufacturer's name,
3. location in machine where used (if not new),
4. type of failure, and
5. number of hours in use.

Analysis of these data indicated that certain tube types of a specific manufacturer were more reliable than the same type made by any other manufacturer. By specifying the manufacturer, the rejection ratio has been reduced.

In order to utilize the limited time allotted to maintenance to the greatest advantage, it is necessary to make all the preparation possible beforehand. As an example, the method used for routine tube changes will be explained: Replicas made of wood of all the chassis in the machine are used to store tube replacements. Each chassis replica has the chassis name printed on the front, and each hole is labeled in matrix notation to identify the tube type and location in that chassis. If a chassis is scheduled for change during a maintenance period of a certain date, the tubes are tested on the preceding shift. Mistakes of inserting the wrong type tube in a socket are kept to a minimum by this method of labeling. The tubes that are removed are checked and are discarded if they do not meet specifications. If acceptable, they are marked with a red dot and replaced in the chassis replica. New tubes are then checked and used to replace the ones thrown away. It is seen that each tube that is inserted in the machine receives at least two tests.

A compilation of tests made on 3385 tubes removed revealed that 26.45% or 1280 tubes were unacceptable. The reason for this high percentage is that from 60 to 90% of the tubes had seen service ranging from 6000 to 24,000 hr. This is not to say that these tubes would cause malfunction if left in the machine, but the probability of marginal operation is too great to warrant further use.

Since October 1956, records that have been kept of the number of blocks of magnetic tape that have been read, hunted, or written by the auxiliary memory show this figure to be 225,303,674 blocks. A rough approximation indicates that this operation consumed about 16% of the available computing time.

From 8000 to 10,000 passes over tapes of the type being used is about the limit that could be expected. Preliminary tests of a new tape that has been checked suggest that 20,000 or more passes may be realized.

TRAINING COURSES

Elementary Numerical Analysis

A series of nine 1-hr lectures in basic numerical analysis for beginning programmers was given. This series will be repeated as demand warrants. Lecture notes and lists of recommended reading matter will be made available. Lectures are listed below.

Rounding Errors	A. C. Downing	October 1960
Zeros of Functions (two lectures)	A. C. Downing	October 1960
Elementary Functions (two lectures)	W. Gautschi	October and November 1960
Linear Equations – Curve Fitting (two lectures)	N. M. Dismuke	November 1960
Ordinary Differential Equations (Primarily Runge-Kutta)	W. Gautschi	November 1960
Eigenvalues and Eigenvectors	A. A. Grau	December 1960

Oracle Coding Course

E. C. Long

Basic Oracle coding was presented in ten 4-hr lectures.

ALGOL Programming

H. Bottenbruch A. A. Grau

In the field of applied programming, three series of coding courses were presented. ALGOL programming was presented twice in two three-lecture series. The lectures did not presuppose previous acquaintance with the Oracle and were of an elementary nature.

Biometrics Course

M. A. Kastenbaum

During July and August 1960, a six-week course in elementary biometrical methods was offered to members of the Biology and the Health Physics Division. Approximately 50 members of these divisions participated.

Seminars

The following topics were presented at Mathematics Panel Seminars:

H. Bottenbruch

"Linear Programming," Parts I, II, III, February 1960.

"Structure and Use of ALGOL 60," Parts I, II, September 1960.

H. P. Carter

"Introduction to the IBM 7090 for IBM 704 Programmers," December 1960.

A. A. Grau

"The ALGOL Language and Its Use on the Oracle," series of three lectures, January–February 1960.

D. E. Arnurius, S. E. Atta, H. Bottenbruch, A. H. Culkowski, and J. A. Harvey

"The Problem of Processing Data from the Fast Chopper Time-of-Flight Neutron Spectrometer," February 1960.

"Analysis of Data to Determine Resonance Parameters," February 1960.

The following topic was presented at a Mathematics Panel Statistics Seminar:

Richard B. Will

"A Mathematical Model for Quantitative Analysis with a Multichannel Analyzer – An Example of Spectrum Stripping," August 1960.

LECTURES AND PAPERS

Bottenbruch, H. H., "Motivations for the International Algebraic Language ALGOL," Digital Computer Laboratory, University of Illinois, Urbana, March 1960.

Bottenbruch, H. H., "Operative Mathematik: A Proposal for the Foundation of Mathematics and Logic by P. Lorenzen," University of Tennessee, Knoxville, May 1960.

Bottenbruch, H. H., "ALGOL Translation by Recursive Procedures," 2nd Meeting of ALGOL Workers' Group, University of Pennsylvania, Philadelphia, May 1960.

Bottenbruch, H. H., "The Impact of ALGOL on the Logical Design of Computers," Meeting of AEC Computer Group, Berkeley, California, June 1960.

Bottenbruch, H. H., "Structure, Use, and Translation of ALGOL 60," series of lectures, Digital Computer Laboratory, University of Illinois, Urbana, July 1960.

Carter, H. P., "Modular Lattices Induced by Finite Groups," University of Tennessee, Knoxville, May 1960.

Downing, A. C., "The Design of Fixed Point Iterations," North Carolina State College, Raleigh, October 1960.

Downing, A. C., "Some Pitfalls of Numerical Analysis," North Carolina State College, Raleigh, October 1960.

Gardiner, D. A., "Least Squares vs 'Simplified' Regression," Tennessee Section, American Society for Quality Control, Oak Ridge, February 1960.

Gardiner, D. A., "Planning of Experiments," Northeast Tennessee and Tennessee Sections, American Society for Quality Control, University of Tennessee, Knoxville, March 1960.

Gardiner, D. A., "The Mathematics Course for Business," Southeastern Section, American Accounting Association, University of Tennessee, Knoxville, April 1960.

Gardiner, D. A., "Calibration with Correlated Errors," Southern Methodist University, Dallas, May 1960.

Gardiner, D. A., "Response Surface Techniques," 14th Annual Convention, American Society for Quality Control, San Francisco, May 1960.

Gardiner, D. A., "Statistics at Oak Ridge National Laboratory," Summer Session in Statistics, Southern Regional Education Board, University of Florida, Gainesville, June 1960.

Gardiner, D. A., "Linear Regression with Correlated Elements," Summer Session in Statistics, Southern Regional Education Board, University of Florida, Gainesville, June 1960.

Grau, A. A., "ALGOL and ALGOL Translation," series of lectures, Argonne National Laboratory, Chicago, February 17–18, 1960.

Grau, A. A., "Recursive Procedures in ALGOL Translation," 2nd Meeting of ALGOL Workers' Group, University of Pennsylvania, Philadelphia, May 1960.

Grau, A. A., "Recursive Processes and ALGOL Translation," ACM Committee on Languages Compiler Symposium, National Bureau of Standards, Washington, D.C., November 17, 1960.

Householder, A. S., "Moments and Characteristic Roots," University of Tennessee, Knoxville, January 1960.

Householder, A. S., "Moments and Characteristic Roots," Argonne National Laboratory, Chicago, January 1960.

Householder, A. S., "Moments and Characteristic Roots," Meeting of American Mathematical Society, Chicago, January 1960.

Householder, A. S., "On the Kantorovich Inequality," Meeting of Mid-Southeast Chapter of ACM, Atlanta, April 1960.

Householder, A. S., "Inclusion and Exclusion Theorems for Characteristic Roots," General Atomic Division, General Dynamics Corporation, San Diego, June 1960.

Householder, A. S., "Errors and Convergence in Solving Matrix Problems," Engineering Summer Conferences, University of Michigan, Ann Arbor, June 1960.

Householder, A. S., "Deflation of Matrices," Eidgenossische Technische Hochschule, Zurich, Switzerland, September 1960.

Householder, A. S., "Deflation of Matrices," National Physical Laboratory, Teddington, England, October 1960.

Kastenbaum, M. A., "The Separation of Molecular Compounds by Countercurrent Dialysis: A Stochastic Process," Department of Statistics, Michigan State University, East Lansing, January 1960.

Kastenbaum, M. A., "Exact Partitioning of Chi-Square in Contingency Tables," Department of Public Health Statistics, University of Michigan, Ann Arbor, January 1960.

Kastenbaum, M. A., "Countercurrent Dialysis: A Stochastic Process," University of North Carolina, Chapel Hill, March 1960.

Kastenbaum, M. A., "Partitioning Chi-Square," invited address, North Carolina Chapter, American Statistical Association, Chapel Hill, March 1960.

Kastenbaum, M. A., "Countercurrent Dialysis: A Stochastic Process," Annual Meeting, American Statistical Association, Palo Alto, California, August 1960.

PUBLICATIONS

Atta, G. J., *An Empirical Study of the Sampling Distributions of Some Disease Incidence Estimates*, ORNL-2957 (July 5, 1960).

Bumgarner, L. L., *Use of Convergent and Asymptotic Series for Computation of the Modified Bessel Function of the First Kind*, ORNL CF-60-12-47 (Dec. 12, 1960).

Downing, A. C., *On the Convergence of Steady State Multiregion Diffusion Calculations*, ORNL-2961 (Aug. 9, 1960).

Grau, A. A., "Solution of Polynomial Equations by Bairstow-Hitchcock Method," ALGOL Algorithm No. 3, *Communs. Assoc. Computing Machinery* **3**, 74 (1960).

Grau, A. A., *The Structure of an Algol Translator*, ORNL-3054 (Jan. 23, 1961).

Grau, A. A., *Natural Translator Oriented Language*, ORNL CF-60-9-66 (Sept. 16, 1960).

Householder, A. S. (with F. L. Bauer), "Moments and Characteristic Roots," *Numerische Math.* **2**, 42–53 (1960).

Householder, A. S., and F. L. Bauer, "On Certain Iterative Methods for Solving Linear Systems," *Numerische Math.* **2**, 55–59 (1960).

Householder, A. S. (with F. L. Bauer), "Some Inequalities Involving the Euclidean Condition of a Matrix," *Numerische Math.* **2**, 308–11 (1960).

Kastenbaum, M. A., "The Separation of Molecular Compounds by Countercurrent Dialysis: A Stochastic Process," *Biometrika* **47**, 69–77 (1960).

Kastenbaum, M. A., "A Note on the Additive Partitioning of Chi-Square in Contingency Tables," *Biometrics* **16**, 416–22 (1960).

Schopf, A. H.,* "On the Kantorovich Inequality," *Numerische Math.* **2**, 344–46 (1960).

---

*Deceased.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ORNL-3082
UC-32 – Mathematics and Computers
TID-4500 (16th ed.)

INTERNAL DISTRIBUTION

1. C. E. Center
2. Biology Library
3. Health Physics Library
4. Reactor Division Library
5–7. Central Research Library
8–14. Laboratory Records Department
15. Laboratory Records, ORNL RC
16. A. M. Weinberg
17. J. P. Murray (K-25)
18. R. G. Jordan (Y-12)
19. J. A. Swartout
20. R. S. Cockreham
21. E. D. Shipley
22. M. L. Nelson
23–24. A. S. Householder
25. A. H. Snell
26. E. H. Taylor
27. A. Hollaender
28. W. H. Jordan
29. H. E. Seagren
30. M. T. Kelley
31. K. Z. Morgan
32. D. Phillips
33. C. E. Winters
34. E. O. Wollan
35. E. P. Blizard
36. R. S. Livingston
37. C. P. Keim
38. M. M. Goff
39. C. D. Susano
40. G. C. Williams
41. R. A. Charpie
42. M. B. Emmett
43. N. M. Dismuke
44. A. H. Culkowski
45. M. T. Harkrider
46. M. R. Arnette
47. P. M. Reyling
48. A. C. Downing
49. G. J. Atta
50. J. A. Lane
51. R. W. Johnson
52. M. J. Skinner
53. J. L. Gabbard
54. E. C. Long
55. M. A. Kastenbaum
56. J. P. Kelly (K-25)
57. S. E. Atta
58. D. A. Gardiner
59. C. A. Preskitt
60. F. L. Culler
61. W. S. Snyder
62. J. L. Fowler
63. C. J. Borkowski
64. G. E. Boyd
65. W. R. Grimes
66. D. S. Billington
67. J. H. Frye
68. R. B. Briggs
69. A. Simon
70. H. Bottenbruch
71. E. L. Cooper
72. M. Feliciano
73. W. Gautschi
74. A. A. Grau
75. N. B. Alexander
76. D. E. Arnurius
77. N. A. Betz
78. L. L. Bumgarner
79. M. J. Mader
80. H. P. Carter
81. J. J. Rayburn
82. J. E. Parham
83. D. J. Wehe
84. B. A. Flores
85. J. G. Latorre
86. M. P. Lietzke
87. ORNL – Y-12 Technical Library, Document Reference Section

EXTERNAL DISTRIBUTION

88. Division of Research and Development, AEC, ORO
89–702. Given distribution as shown in TID-4500 (16th ed.) under Mathematics and Computers (75 copies – OTS)

Mathematics Panel progress reports previously issued in this series are as follows:

ORNL-345	December, January, February, 1948–1949
ORNL-408	Period Ending July 31, 1949
ORNL-516	Period Ending October 31, 1949
ORNL-634	Period Ending January 31, 1950
ORNL-726	Period Ending April 30, 1950
ORNL-818	Period Ending July 31, 1950
ORNL-888	Period Ending October 31, 1950
ORNL-979	Period Ending January 31, 1951
ORNL-1029	Period Ending April 30, 1951
ORNL-1091	Period Ending July 31, 1951
ORNL-1151	Period Ending October 31, 1951
ORNL-1232	Period Ending January 31, 1952
ORNL-1290	Period Ending April 30, 1952
ORNL-1360	Period Ending July 31, 1952
ORNL-1435	Period Ending October 31, 1952
ORNL-1498	Period Ending January 31, 1953
ORNL-1588	Period Ending June 30, 1953
ORNL-1662	Period Ending December 31, 1953
ORNL-1751	Period Ending June 30, 1954
ORNL-1842	Period Ending December 31, 1954
ORNL-1928	Period Ending June 30, 1955
ORNL-2037	Period Ending December 31, 1955
ORNL-2134	Period Ending June 30, 1956
ORNL-2283	Period Ending February 28, 1957
ORNL-2652	Period Ending August 31, 1958
ORNL-2915	For Period September 1, 1958, to December 31, 1959