AUTOMATIC GRADING PROGRAMS

by

George E. Forsythe and Niklaus Wirth

Abstract

The ALGOL grader programs are presented for the computer evaluation of student ALGOL programs. One is for a beginner's program; it furnishes random data and checks answers. The other provides a searching test of the reliability and efficiency of a rootfinding procedure. There is a statement of the essential properties of a computer system, in order that grader programs can be effectively used.

*/ Reproduction in Whole or in Part is permitted for any Purpose of the United States government. This report was supported in part by Office of Naval Research Contract Nonr-225(37) (NR 044-211) at Stanford University.

**/ Presented to the national meeting of the Association for Computing Machinery, Philadelphia, 27 August 1964, under the title "Automatic machine grading programs". The Stanford computers used for program tests are partially supported by the National Science Foundation under Grant No. GP-c-8.
Discussion

In connection with introductory programming and numerical analysis courses at Stanford University, grading programs have been used intermittently since 1961. Our programs furnish data, check student performance in various ways, sometimes keep track of running time, and keep a "grade book" for the problems.

The Stanford routines are written separately for each problem. The most flexible and useful system for elementary classes was used with the Burroughs 220 computer in the BALGOL language, a dialect of ALGOL 58, and will be described first. Each grader program was written as a BALGOL-language procedure. It was then compiled together with a procedure called BUTTERFLY, written by Roger Moore. The result was a relocatable machine-language procedure, with a mechanism for equating its variables to variables of any BALGOL program, in just the form of the BALGOL compiler's own machine-language library procedures (SIN, WRITE, READ, etc.). Finally, the grader program was added to the compiler library tape for the duration of its use.

The use of a powerful algebraic language and system made it easy for an instructor to write grader programs with sophistication appropriate to the problem. The student needed only furnish one or two procedure statements to call a grader; we often furnished him cards to decrease the chance of error. Since each grader was precompiled on the library tape, little time was lost in adding it to each student's program at compile time. A simple handcoded mechanism made it easy for the operator to rescue a program from a run-time loop and send it into the
next case without dismissing the program entirely. It was possible to have several different graders concurrently on the library tape, to take care of different classes.

This powerful grading mechanism was possible only because of the BALGOL compiler with its own compiler-with-library generator, and because BUTTERFLY could generate relocatable machine-language programs. Neither our IBM 7090 BALGOL system nor our Burroughs B5000 ALGOL system has been so well adapted to the grading problem, to our regret, and we have had to make do with less desirable expedients. What these newer systems lack is an easy means of producing a machine-language library procedure which is fully equivalent to a source-language procedure with several parameters.

We recommend grading programs to all who teach programming and numerical analysis to masses of students, but the prospective user should first carefully investigate the systems available to him.

We give below a typical grading program, GRADER2, which was used in connection with an early problem in a beginning programming course. GRADER2 is suitable for grading a student's program which is written as a block. It has been translated into ALGOL and is given as part of a block containing one or more student programs.

In ALGOL there is a practical difficulty in putting many beginner programs into the same block with GRADER2—a syntactical error in just one student program may upset the compiler and prevent the testing of any program. It would be better if we could have GRADER2 precompiled into a system library and called separately by each student.
The student's problem is this: Given the integer \( N \) (\( 0 < N < 10 \)) and the real array elements \( A[0], \ldots, A[N], B[0], \ldots, B[N] \), to write a program which makes \( \text{MAX} \) the maximum of \( A[0], \ldots, A[N] \) and which computes the numbers \( C[0], \ldots, C[2 \times N] \) defined by the polynomial multiplication

\[
\sum_{k=0}^{2N} C[k]t^k \equiv \left( \sum_{k=0}^{N} A[k]t^k \right) \cdot \left( \sum_{k=0}^{N} B[k]t^k \right).
\]

The student is told that he must arrange his program in the form:

```
begin
<all declarations>

GRADER2 (<student number>, 1, N, MAX, A, B, C, START, FIN);

START: <all statements of his solution>

GRADER2 (<student number>, 2, N, MAX, A, B, C, START, FIN);

FIN:

end;
```

Note that the subprocedure SET UP of GRADER2 goes to great trouble to be sure that no student will get the same data at different times, and that no two students will get the same data. This was intended to be sure that a student could not get correct answers from GRADER2 on one run, and use them for another run. We doubt the value of such precautions.

Observe that GRADER2 evaluates the correctness of the student answers, but in no way evaluates the running speed of the program nor the amount of storage used. This is appropriate for a beginning student of programming.
A more advanced student should have his performance examined more critically. As an example of this, we give a second grader program called Test, to be used in a numerical analysis class whose members can already program in ALGOL.

The procedure Test listed below is designed to examine rootfinding procedures. The students are asked to write an ALGOL procedure which finds an approximation to a 'root' \( x \) (i.e., point of sign-change) of a (not necessarily continuous) function \( f \) in the interval \([a, b]\). To be precise, \( x \) is any number such that
\[
f(x_0) \leq 0 \equiv f(x_1) \geq 0 \quad \text{(in the sense of ALGOL 60)}
\]
and
\[
a \leq x_0 \leq x \leq x_1 \leq b
\]
and
\[
|x_0 - x_1| \leq \epsilon,
\]
where \( f, a, b, \) and \( \epsilon \) are given parameters. Such an \( x \) always exists, if \( f(a) \leq 0 \equiv f(b) \geq 0 \). Each student is asked to submit an ALGOL program containing his procedure declaration and a single statement of the form
\[
\text{Test (rootfinder, 'student name')}
\]
As with GRADER2, all submitted programs are then enclosed together in an outer block, whose head contains the declaration of the procedure Test. Thus no use of library tapes is required. The block structure of ALGOL plays the very important role that all identifiers used by the student, including the name of his rootfinder, are strictly local to his program. They can therefore be chosen freely and cannot interfere
with identifiers in any other program. Nor can an identifier of the
procedure Test interfere with any student's program. Of course this
technique requires that the contributions be syntactically correct, but
this is considered to be the minimum requirement for acceptance of a
program from students at this level.

In order to obtain an estimate of the quality of the programs,
one would like to know the accuracy of the answers, the number of
evaluations of the function \( f \) it took to find the root (with the
possibility of terminating the test, if a limit is exceeded), and per-
haps also the time it took to find the root. It is furthermore desirable
to check that the limits \( a \) and \( b \) and the tolerance \( \epsilon \) were not
changed during a test (this might occur, for example, if these para-
meters were called by name instead of by value), and whether the argument
of \( f \) always remained within the interval \([a, b]\).

The following description of the procedure Test explains the
methods of achieving these goals within the framework of an ALGOL program.

The declaration of Test contains the following variable-declarations:
grade represents the student's grade; it is cumulated during
execution of several partial tests of one student's
program.

\[ m \] denotes his number of successful tests.

\[ x \] is a real variable used as abscissa for the evaluation of \( f \).

\[ t \] records the time spent by the rootfinder.

A procedure \( P \) declared inside Test is the heart of the entire grader.
The body of Test contains a series of calls of \( P \); each call of \( P \)
contains as actual parameters the data for one test example. E.g., the
call

P(0,2,10^-5,1,1 - x, \text{true}, 20, 1)

would cause the testing of the student's program with the function \( f: f(x) = 1 - x \) in the interval \( a = 0, b = 2 \), with a desired accuracy of \( 10^{-5} \). The expected result \((=1)\) is the fourth parameter to \( P \). The sixth parameter indicates that a solution exists, the seventh is a limit for the number of evaluations of \( f \), and the eighth indicates the number of the test case.

The procedure \( P \) subsequently calls the rootfinder (which is a formal parameter to \( \text{Test} \)) with the given parameters as data. However, \( P \) does not furnish the function \( f \) directly to the rootfinder, but rather substitutes a function procedure \( Q \), which is declared in the head of \( P \). Each call of \( Q \) then serves to increment the counter of calls of \( f \) and is also used to examine whether the argument of \( f \) lies within the prescribed interval.

The grader program has been used on the 7090 computer. In order to measure the time spent by the rootfinder and to recover from a possible error in the logic of the student's program, two code procedures have been introduced which cannot be described in ALGOL:

\begin{verbatim}
procedure Setime (n,L); integer n; label L;
initializes the core-clock to trap after n msec and to transfer control to L. Also,

procedure Heset (t); integer t;

disables the trap and assigns to \( t \) the number of msec spent since initialization of the clock. These procedures protect the entire grading run from failure due to one particular examinee's inability to solve a certain test problem.
\end{verbatim}
The authors feel that particular emphasis should be put not only on the efficiency of the student's contribution, but also on its reliability. The choice of the test data reflects the possibilities of this grading method, since "wildly behaving" functions are used which are not likely to be foreseen by a careless programmer.

The program Test is believed to mark a further step in the automation of grading. Whereas GRADER2 bases its grade only on the binary answer "correct" or "wrong", Test also evaluates a program's quality i.e., reliability and effectiveness. It thus relieves the teacher from long and tedious grading work. Last, but not least, the machine may be more objective in grading than the human, because of its notable lack of prejudice and its inability to become bored.

The authors wish to acknowledge valuable suggestions made by A. J. Perlis and P. Naur in regard to grader programs.
An elementary grader program

begin

procedure GPADER2 (STUDENT, ENTRY, N, MAX, A, B, C, START, FIN);
value STUDENT, ENTRY; integer STUDENT, ENTRY, N; real MAX;
real array A, B, C; label START, FIN;
comment We assume the existence of a library real procedure
    TIME which produces the time of day as an integer in the
    interval [0, 2359];
begin

real procedure RANDOM;

begin

comment The value of RANDOM at each call is a different
    pseudo-random number from a flat distribution in the
    interval [0, 1]. The body is not written here ... ;
end RANDOM;

own real array CC[0 10];
real S; own real MMAX;
integer B3, B4, J, K;
own integer TALLY, B1, B2;
switch JUMP := L1, L2;
procedure SET UP (CASE, N);
value CASE; integer CASE, N;
comment SET UP furnishes the student's data, which depend on
    the student's number, the time of day, and a pseudo-random
    number. SET UP also solves the case for the use of EVALUATE;
begin  
  for K := 0 step 1 until N do 
begin  
    A[K] := RANDOM + (STUDENT + TIME) × 10⁻¹⁴;  
    B[K] := RANDOM × sign(RANDOM - 0.5) 
end;  
comment Now the student is messaged on the line-printer what 
data have been generated for him;  
outstring (1, 'FOR CASE'); outinteger (1, CASE);  
outstring (1, 'GRADER2 FURNISHES STUDENT'); outinteger(1, 
  STUDENT); outstring (1, 'THE FOLLOWING DATA: A IS');  
for K := 0 step 1 until N do outreal (1, A[K]);  
outstring (1, 'B IS');  
for K := 0 step 1 until N do outreal (1, B[K]);  
comment Now GRADER2 solves the student's case for itself.  
  GRADER2 does not use A[K] or B[K] for any values of K 
outside 0 ≤ K ≤ N;  
MMAX := A[0];  
for K := 1 step 1 until N do 
  if A[K] > MMAX then MMAX := A[K];  
for K := 0 step 1 until N do 
begin  
  S := 0;  
  CC[K] := S  
end;  
for K := N+1 step 1 until 2×N do
begin
S := 0;
CC[K] := S
end;

comment Now SET UP has solved the problem, and we exit to START, the entry to the student's solution. The next call of GRADER2 will bring us back to EVALUATE;
TALLY := TALLY + 1;
go to START;
end SET UP;

procedure EVALUATE (CASE, N);
value CASE; integer CASE, N;
begin
B3 := 1;

comment EVALUATE examines the student's answers, writing them and its own answers, with comments on the student's performance, all on the line-printer;
outstring (1, 'FOR CASE'); outinteger (1, CASE);
outstring (1, 'STUDENT'); outinteger (1, STUDENT);
outstring (1, 'COMPUTES C TO BE');
for J := 0 step 1 until 2 × N do outreal (1, C[J]);
outstring (1, 'GRADER2 COMPUTES C TO BE');
for J := 0 step 1 until 2 × N do outreal (1, CC[J]);
for K := 0 step 1 until 2 × N do
  if abs(C[K] - CC[K]) > 10^{-4} then B3 := 0;
if \( B3 = 1 \) then
\[ \text{outstring } (l, \text{'C } \Rightarrow \text{ IS } \Rightarrow \text{ ACCEPTABLE'}) \]
else
\[ \text{outstring } (l, \text{'C } \Rightarrow \text{ IS } \Rightarrow \text{ NOT } \Rightarrow \text{ ACCEPTABLE'}) \]
\[ \text{comment A large tolerance was allowed for possible differences in the solutions because of different rounding off;} \]
\[ Bh := 0; \]
if \( \text{MAX} = \text{MMAX} \) then
begin
\[ Qh := 1, \text{outstring } (l, \text{'MAX } \Rightarrow \text{ IS } \Rightarrow \text{ CORRECT'}) \]
end
else
\[ \text{outstring } (l, \text{'MAX } \Rightarrow \text{ IS } \Rightarrow \text{ INCORRECT'}) \]
end EVALUATE;

comment Now come the statements of GRADER2 itself;
if \( \text{ENTRY} = 1 \) then \( \text{TALLY} := 0 \) else go to JUMP[\text{TALLY}];
\[ \text{comment On the first call of GRADER2 by each student, ENTRY is 1. On the later calls it is 2. This and TALLY provide the mechanism for permitting different entries to GRADER2 on different calls;} \]
\[ N := 5; \]
\[ \text{SET UP } (1, 5); \]
L1: EVALUATE (1, 5);
\[ B1 := B3; B2 := B4; \]
\[ N := 4; \]
\[ \text{SET UP } (2, 4); \]
L2: EVALUATE (2,4);  
   comment Case 2 is now complete, and GRADER2 punches a card for the "grade book";
   for K := STUDENT, 2, B1, B2, B3, B4 do outinteger (2, K);
   comment GRADER2 now summarizes the situation for the student's line-printer listing;
   outstring (1, 'STUDENT'); outinteger (1, STUDENT);
   outstring (1, 'SCORES'); outinteger (1, B1+B2+B3+B4);
   outstring (1, 'OUT OF 4 ON PROBLEM 2. IF SCORE IS LESS THAN 4, PLEASE SUBMIT REVISED PROGRAM LATER.');
   comment Now the program exits to the conclusion of the student's solution;
   go to FIN
end GRADER2;
comment Now follow the students' programs;
begin
   comment Here, for example, is the program for student number 515, with its calls on GRADER2;
   real array A, B, C [0:25];
   real S, MAX;
   integer J, K, N;
   GRADER2 (515, 1, N, MAX, A, B, C, START, FINISH);
            MAX := A[N];
            for K := N-1 step -1 until 0 do
               if MAX < A[K] then MAX := A[K];
-12-
for $K := 0$ step $1$ until $2 \times N$ do
begin
    $S := 0$;
    $C[K] := S$
end;

GRADER2 (515, 2, N, MAX, A, B, C, START, FINISH);

FINISH:
end program of student 515;
begin
    comment Program of another student ...
end
.....

end tests of all student programs
An advanced grader program

begin comment grader program for root-finding procedures;

procedure Test (Rootfinder, Name); procedure Rootfinder; string Name;
begin real x; integer m, grade, time;

procedure P (low, up, eps, root, f, is root, iteration limit, problem no);

value low, up, eps, root, is root, iteration limit, problem no;
real low, up, eps, root, f;
integer iteration limit, problem no;

Boolean is root;

begin real low l, up l, eps l, root l;

integer n, t; Boolean is root l;

procedure Setime (n, L); integer n; label L; code;
procedure Reset (t); integer t; code;
procedure error (text, charge);

value charge; integer charge; string text;

begin outstring (text); grade := grade - charge;

if grade ≤ 0 then
begin outstring (‘grade = 0’);

go to T exit

end

end error;

real procedure Q(y); value y; real y;

begin if y < low l V y > up l then

begin error (‘Argument outside interval’, 10);

Reset(t); go to P exit


-14-
end;
n := n + 1; if n > iteration limit then
begin error ("Convergence is too slow", 5);
    Reset(t); go to P exit
end;
x := y; Q := f
end Q;
low 1 := low; up 1 := up; eps 1 := eps; n := 0;
outstring ("problem no. ="); outreal (problem no);
Setime(100, fail);
Rootfinder (low 1, up 1, eps 1, Q, root 1, is root 1);
Reset(t); time := time + t;
if low 1 /= low V up 1 /= up then
    error ("boundary was altered", 3);
if eps 1 /= eps then error ("tolerance was altered", 5);
if 1 is root then
begin if is root 1 then
    error ("solution found where none exists", 5)
else begin outstring ("correct reaction for no root");
    m := m + 1
end
end
else
begin real tol; tol := abs (root - root 1);
    if tol > eps then error ("incorrect root", 5)
else begin outstring ("correct root found"); m := m+1
end;
-15-
outstring ('no. of iterations ='); outreal (n);
end;
go to P exit;
fail: error ('failure', 10);
P exit:
end P;

m := 0; grade := 100; time := 0; outstring (Name);

P(-2, 2, 10^-6, -1, x+1, true, 10, 1);
P(-1, 1, 10^-6, -1, x+1, true, 10, 2);
P(-1, 1, 10^-6, 1, x-1, true, 10, 3);
P(2, 5, 10^-6, 0, x-1, false, 10, 4);
P(-2, 3.5, 10^-5, 2, x^3 - x^3 - 2*10^-20, true, 30, 5);
P(10^-5, 99.9, 10^-5, 0.01, x + 1/x - 100.01, true, 50, 6);
P(-1, 2, 10^-5, 0, sign(x), true, 30, 7);
P(-3, 100, 10^-4, 0, exp(-x) -1, true, 50, 8);
P(0, 20, 10^-4, 0.95, (x + 0.05) - 0.1 - 1, true, 30, 9);
P(0, 100, 10^-5, 0, if x < 1 then x-1 else 10^-10, true, 30, 10);
P(-2.4, 4.2, 10^-4, 3, (((((x-3) x x+5) x x-15) x x+4) x x+1),
   true, 50, 11);
P(5*10^-3, 1, 5*10^-3, 0.0265, if x < 0.02122 then -1 else
   if x > 0.03183 then 1 else cos(1/x), true, 30, 12);
outstring ('end of test. No. of correct problems =');
outreal (m); outstring ('time ='); outreal (time);
outstring ('grade ='); outreal (grade);

T exit:
end T;
comment Subsequently follow the students' programs, each containing
a procedure declaration and a call of Test enclosed in a block;

begin

procedure Bisect (x0, xl, tol, func, result, is result);
real x0, xl, tol, result;
real procedure func;
Boolean is result;
begin ............
 ............
end Bisect;
Test (Bisect, 'Tom Jones')
end;
............

comment further students' programs follow here;
............

end Grader program
<table>
<thead>
<tr>
<th>Stanford University</th>
<th>Technical Reports Distribution List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract No. 225/67</td>
<td>(NR 044-211)</td>
</tr>
</tbody>
</table>

**Armed Services Technical Information Agency**  
Arlington Hall Station, Arlington, Virginia

**Chief of Naval Research**  
Code 432, Office of Naval Research, Washington, D.C.

**Commanding Officer**  
Office of Naval Research, Branch Office, San Francisco, California

**Commanding Officer**  
Office of Naval Research, Branch Office, Boston, Massachusetts

**Commanding Officer**  
Office of Naval Research, Branch Office, New York, New York

**Commanding Officer**  
Office of Naval Research, Branch Office, Pasadena, California

**Argonne National Laboratory**  
Technical Information Division, 9700 South Cass Ave., Argonne, Illinois

**Commanding Officer**  
U.S. Naval Electronics Lab., Attn. R.F. Arenz, San Diego, California

**Computation Center**  
University of Texas, Austin, Texas

**Computer Reviews**  
Room 5-C, 211 E. 43rd Street, New York, New York

**Department of Mathematics**  
Harvard University, Cambridge, Massachusetts

**Department of Mathematics**  
Massachusetts Institute of Technology, Cambridge, Massachusetts

**Space Technology Labs.**  
One Space Park, Attn. Irene Mathews, Redondo Beach, California

**Institute for Mathematical Sciences**  
New York University, New York, New York

**Library**  
Digital Computer Laboratory, University of Illinois, Urbana, Illinois

**Librarian, Communications Research Division**  
Institute for Defense Analyses, Von Neumann Hall, Princeton, New Jersey

**National Aeronautics and Space Administration**  
Lewis Research Center, 21000 Brookpark Road, Attn. Library, Cleveland, Ohio

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

**Document Center**  
Stanford Research Institute, 333 Ravenswood Ave., Menlo Park, California

**Technical Information Officer**  
Naval Research Laboratory, Washington, D.C.

**The MITRE Corp.**  
Attn. Supervisor Library Services, Bedford, Massachusetts

**University of California**  
Attn. Library, Numerical Analysis Res., Los Angeles 24, California

**University of Miami**  
Scientific Computing Center, P.O. Box 8011, Coral Gables 46, Florida

**U.S. Naval Ordnance Laboratory**  
White Oak, Attn. Mathematics Department, Silver Spring, Maryland

**U.S. Naval Ordnance Laboratory**  
White Oak, Attn. Library, Silver Spring, Maryland

**U.S. Navy Management Office**  
Department of the Navy, Washington 25, D.C.

**U.S. Naval Weapons Laboratory**  
Dahlgren, Virginia

**Professor W. F. Atchison**  
Rich Electronics Computer Center, Georgia Institute of Technology, Atlanta, Georgia

**Dr. Erwin H. Bareiss**  
Applied Mathematics Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, Illinois

**Professor Robert C.F. Bartels**  
Computing Center, University of Michigan, Ann Arbor, Michigan

**Dr. Richard Bellman**  
The Rand Corporation, 1700 Main St., Santa Monica, California

**Professor Garrett Birkhoff**  
Mathematics Department, Harvard University, Cambridge 38, Massachusetts

**Bernard H. Bissinger**  
Department of Mathematics, Lehigh Valley College, Annville, Pennsylvania

**Mr. Truman Botts**  
Department of Mathematics, University of Virginia, Charlottesville, Virginia

**Professor James Bramble**  
Institute for Fluid Dynamics and Applied Mathematics, University of Maryland, College Park, Maryland

**Professor A. T. Brauer**  
Department of Mathematics, University of North Carolina, Chapel Hill, North Carolina

**Professor George W. Brown**  
Western Data Processing Center, University of California, Los Angeles 24, California

**Professor John W. Carr, III**  
Moore School of Ele. Engr., University of Pennsylvania, Philadelphia 4, Pennsylvania

**Mr. Robert L. Causey**  
Lockheed Aircraft Corp., Missiles and Space Division, 3201 Hanover St., Palo Alto, California

**Professor Abraham Charnes**  
The Technological Institute, Northwestern University, Evanston, Illinois

**Professor B. A. Chartres**  
Division of Applied Mathematics, Brown University, Providence 12, Rhode Island

**Professor Harvey Cohn**  
Department of Mathematics, University of Arizona, Tucson 25, Arizona

**Professor Stephen Crandall**  
Department of Mathematics, Mass. Inst. of Technology, Cambridge 39, Massachusetts

**Professor Philip Davis**  
Department of Mathematics, Brown University, Providence 12, Rhode Island

**Professor David B. Dekker**  
Director, Research Computer Lab., University of Washington, Seattle 5, Washington

**Professor R. J. Devogelaere**  
Department of Mathematics, University of California, Berkeley 4, California

**Professor C. L. Dolph**  
311 Park Avenue, Ann Arbor, Michigan

**Professor Jim Douglas, Jr.**  
Department of Mathematics, New York University, Houston 1, Texas

**Professor R. J. Duffin**  
Mathematics Department, Carnegie Institute of Technology, Pittsburgh 13, Pennsylvania

**Dr. Robert Dickson**  
Lockheed Aircraft Corporation, Missiles and Space Division, 3251 Hanover Street, Palo Alto, California

**Professor Paul Dryer**  
403 Lenawee Drive, Ann Arbor, Michigan

**Mr. Thomas A. Dryer**  
Director, Computer Science Department, University of Dayton, Dayton 9, Ohio

**Mr. Edgar L. Eichert**  
Burrroughs Corporation, Electradata Facility, 460 Sierra Moree Villa, Pasadena, California

**Mr. Frank Engel, Jr.**  
Analytical Department 4-L-39, Westinghouse Electric Corp., 700 Bradloock Avenue, East Pittsburgh, Pennsylvania
OR. WILLIAM HOOKER
OAK RIDGE, TENNESSEE
P. 0. BOX X
OAK RIDGE NATIONAL LAB.
P. O. BOX 281
MOUNTAIN VIEW, CALIF.

OR. A. S. HOUSEHOLDER
3875 FABIAN WAY
OHIO STATE UNIVERSITY
PROF. T. W. HILDEBRANDT
ENGINEERING EXPERIMENT STATION
DIRECTOR, UNIVERSITY OF WISCONSIN

OR. W. PARLETT
INST. FOR MATHEMATICAL SCIENCES
4 WASHINGTON PLACE
NEW YORK 3, NEW YORK

DR. SEYMOUR V. PATERA
DEPARTMENT OF MATHEMATICS
UNIVERSITY OF WISCONSIN

DR. J. J. STOKER
CHAIRMAN, MATHEMATICS DEPARTMENT
STATE COLLEGE, PENNSYLVANIA

PROF. J. J. STOKER
PROF. M. L. STEIN
DIRECTOR, DEPARTMENT OF MATHEMATICS
INSTITUTE OF TECHNOLOGY
UNIVERSITY OF MINNESOTA
MINNEAPOLIS 4, MINNESOTA

MR. WERNER FRANK
NEW YORK UNIVERSITY
DEPARTMENT OF MATHEMATICS
PROF. EUGENE ISAACSON
AEC COMPUTING CENTER
NEW YORK 3, NEW YORK

PROF. R. D. RICHTMYER
AEC COMPUTING CENTER
UNIVERSITY OF MICHIGAN

PROF. LOWELL SCHOEFELD
UNIVERSITY OF PENNSYLVANIA
MATHEMATICS DEPARTMENT
PROF. B. E. HUBBARD
INSTITUTE FOR APPLIED MATHEMATICS
UNIVERSITY OF MICHIGAN
COLLEGE PARK, MARYLAND

DR. M. A. HYMAN
IBM SYSTEMS DEVELOPMENT CENTER
7220 ICLAS AVENUE
BETHESDA, MARYLAND

DR. J. P. NASH
LOCKHEED MISSILE AND SPACE CORP.
SUNNYVALE, CALIF.

PROF. C. D. OLDZ
P. O. BOX 462
LOS ALTOS, CALIFORNIA

DR. J. J. STOKER
INSTITUTE OF MATHEMATICAL SCIENCES
25 Waverly Place
NEW YORK 3, NEW YORK

MR. H. C. THACHER, JR.
ARGONNE NAD OF THE UNITED STATES
7900 SOUTH CAZ AVENUE
ARGONNE, ILLINOIS

PROF. B. E. HUBBARD
INSTITUTE FLUID DYNAMICS AND
APPLIED MATHEMATICS
UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND

DR. J. P. NASH
LOCKHEED MISSILE AND SPACE CORP.
SUNNYVALE, CALIF.

PROF. B. E. HUBBARD
INSTITUTE FLUID DYNAMICS AND
APPLIED MATHEMATICS
UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND

DR. J. P. NASH
LOCKHEED MISSILE AND SPACE CORP.
SUNNYVALE, CALIF.

PROF. B. E. HUBBARD
INSTITUTE FLUID DYNAMICS AND
APPLIED MATHEMATICS
UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
MR. ROLAND THOMPSON
BUNKERHILL CORPORATION
6071 2ND AVE.
DREIDT 32, MICHIGAN

DR. GENE THOMPSON
UNITED TECHNOLOGY CORP.
1030 E. ARKES AVE.
SYNNVILLA, CALIF.

PROF. JOHN TODD
MATHEMATICS DEPARTMENT
CALIF. INST. OF TECHNOLOGY
PASADENA 4, CALIFORNIA

PROF. RICHARD S. VANGA
DEPARTMENT OF MATHEMATICS
CASE INSTITUTE OF TECHNOLOGY
UNIVERSITY CIRCLE
CLEVELAND 6, OHIO

PROF. WOLFGANG WASOW
DEPARTMENT OF MATHEMATICS
UNIVERSITY OF WISCONSIN
MADISON 6, WISCONSIN

PROF. M. G. WURTLE
DEPARTMENT OF METEOROLOGY
UNIVERSITY OF CALIFORNIA
LOS ANGELES 24, CALIF.

PROF. DAVID M. YOUNG, JR.
DEPARTMENT OF MATHEMATICS
UNIVERSITY OF TEXAS
AUSTIN 17, TEXAS

DISTRIBUTION VIA OMR LONDON

F. C. ADENY F. L., CHIEF LIBRARIAN
LEEDS TECH COLLEGE OF ART AND TECH.
THE NEWKIRK S, LEICESTER ENGLAND

MATHEMATISCHES CENTRUM
26 GOLDSCHAFFENSTR. 49
AMSTERDAM 21, NETHERLANDS

PROF. A. C. AIKEN
MATHEMATICAL INSTITUTE
UNIVERSITY OF EDINBURGH
J0 DIAMOND STREET
EDINBURGH 1, SCOTLAND

DR. ENZO APPARO
ISTITUTO NAZIONALE PER LE APPLICAZIONI DEL CALCOLO
PIAZZALE DELLE SCUOLE 7
ROMA, ITALY

PROF. DR. CHARLES BLANC
CENTRE NATIONAL DE CALCUL ELECTRONIQUE
AVENUE DE LA CONCORDE 33
LAUSANNE, SWITZERLAND

JEAN A. FABOT
COMPUTING CENTRE
DEPARTMENT OF MATHEMATICS
UNIVERSITÉ DE MONTREAL
MONTREAL, CANADA

PROF. DR. F. L. BAUEER
MATHEMATISCHES INSTITUT, DER TECHNISCHEN HOCHSCHULE
BRAUNSHAW 2
ARISGSTASSE 21
GERMANY

MR. ERIC J. EVANS
RENECENTRALEN
GR. CARLBERGVEJ 2
VALBY, DENMARK

J. C. BYRNE
ENGINEERING SCHOOL
TRINITY COLLEGE
DUBLIN 2, IRELAND

DR. SIG. COLO
INTERNATIONAL COMPUTATION CENTRE
PALAZZO DEGLI UFFICI
ZONA DELLE U. R.
ROMA, ITALY

PROF. DR. L. COLLATZ
INSTITUT FÜR ANGEWANDTE MATHEMATIK
UNIVERSITÄT HAMBURG
2014 3, HAMBURG 13
ROTHENBAUMCHAUSSEE 67/69
GERMANY

PROF. GERHARD G. DHALQUIST
ROYAL INST. OF TECHNOLOGY
STOCKHOLM DIV. OF APPLIED MATHEMATICS
STOCKHOLM 70, SWEDEN

DR. LESLIE FOX
OXFORD UNIVERSITY
OXFORD, ENGLAND

DR. CARL-FRIK FROBERG
ADV. FUR NUMERISCHES ANALYSIS
LUNDS UNIVERSITET
LUND, SWEDEN

MR. RICHARD GOODMAN
DIRECTOR, AUTOMATIC PROGRAMMING
INFORMATION CENTER
DEPARTMENT OF MATHEMATICS
THE HAMMERSLEY COLLEGE
BRIGHTON, ENGLAND

DR. J. M. HAMILTONS
INSTITUTE OF STATISTICS
ST. CROSS ROAD
OXFORD, ENGLAND

PROF. DR. PETER HENRICH
LEHRSTUHL FUR HOHERE MATHEMATIK
EIGENNACHSCHTE TECHNISCHE HOCHSHULE
ZURICH, SWITZERLAND

DR. JOSEPH HERSCH
E. T. H. ZINNM 200
ZURICH 6, SWITZERLAND

INTERNATIONAL COMPUTATION CENTER
PALAZZO DEGLI UFFICI
ZONA DELLE U. R.
ROMA, ITALY

PROF. J. KONTZMANN
LABORATOIRE DE CALCUL DU CENTRE NATIONAL DE RECHERCES
44-46 AVENUE FELIX-VIALLET
GRENOBLE, ISERE
FRANCE

PROF. PENTTI LAASONEN
FINLAND INSTITUTE OF TECHNOLOGY
HELSINKI, FINLAND

PROF. C. LANCZOS
INSTITUTE FOR ADVANCED STUDIES
64-65 MERRION SQUARE
DUBLIN, IRELAND

PROF. J. J. LIU
DEPARTMENT OF MATHEMATIQUES
UNIVERSITE DE PARIS
11 RUE PIERRE CURIE
PARIS 5E, FRANCE

PROF. W. MUSKUS
THE UNIVERSITY
SHEFFIELD, ENGLAND

DR. A. R. MITCHELL
MATHEMATICS DEPARTMENT
ST. SALVATORS COLLEGE
UNIVERSITY OF ST. ANDREWS
ST. ANDREWS, FIFE, SCOTLAND

MR. JEAN A. MUSARD
SECRETARIAT A L'AFAC
48 RUE DES PONTIERS
MONTBELIARD, DOUBS
FRANCE

PROF. DR. H. W. WIELANDT
MATHEMATISCHES INSTITUT
UNIVERSITÄT TUBINGEN
TUBINGEN 148
WILHELMSSTRASSE 7, GERMANY

MR. V. WILKES
UNIVERSITY MATHEMATICAL LAB.
CORR EXCHANGE STREET
CAMBRIDGE, ENGLAND

DR. J. H. WILKINSON
MATHEMATICAL DIVISION
NATIONAL PHYSICAL LAB.
TEEDINGTOWN, MIDDLESEX
ENGLAND

HONORARY LIBRARIAN, BRITISH COMPUTER
SOCIETY,
FINCHBURY COURT, FINCHBURY PAVEMENT
LONDON E. C. 2, ENGLAND

THE LIBRARY, BUILDING 445
ATOMIC ENERGY RES. ESTABLISHMENT
HARWELL, DIDCOT
BERKSHIRE, ENGLAND

JEAN A. BAUDOT
COMPUTING CENTER
DEPARTMENT OF MATHEMATICS
UNIVERSITÉ DE MONTREAL
MONTRÉAL, CANADA

OTHER FOREIGN ADDRESSES

DR. JOHN M. BENNETT
ADOLPH BASSER COMPUTER LAB.
UNIVERSITY OF SYDNEY
SYDNEY, N.S.W.
AUSTRALIA

W. A. CONWAY
LIBRARY
THE UNIVERSITY
ADELAIDE
SOUTH AUSTRALIA

PROF. A. GONZALEZ DOMINGUEZ
THE INSTITUTE OF CALCULUS
UNIVERSITY OF BUENOS AIRES
AVENIDA DE MAYO 760, 20 PISO
BUENOS AIRES, ARGENTINA
The ALGOL grader programs are presented for the computer evaluation of student ALGOL programs. One is for a beginner's program; it furnishes random data and checks answers. The other provides a searching test of the reliability and efficiency of a rootfinding procedure. There is a statement of the essential properties of a computer system, in order that grader programs can be effectively used.
1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (corporate author) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system number, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBERS: If the report has been assigned any other report numbers (either by the originator or by the sponsor), also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U.S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through"

(4) "U.S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through"

(5) "All distribution of this report is controlled. Qualified DDC users shall request through"

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (paying for) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report. It may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TM), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.