stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE (DEPARTMENT OF NUMERICAL MATHEMATICS)

NN 17/78 DECEMBER

H.J. BOS & D.T. WINTER

AFLINK: A NEW ALGOL 68 - FORTRAN INTERFACE

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68-Ao5

AFLINK: A new ALGOL 68 - FORTRAN interface*)

by

H.J. Bos & D.T. Winter

ABSTRACT

This report contains an external description of AFLINK. AFLINK is a newly designed ALGOL 68 - FORTRAN interface, which can be used with the ALGOL 68 version 1.2.1 (78325) and FORTRAN version 4.6. level 460 compilers on the CDC CYBER machines. The new interface can handle all situations the standard CDC ALGOL 68 - FORTRAN interface can. Moreover, AFLINK is capable of passing a large variety of ALGOL 68 procedures as parameters to FORTRAN routines. Special interface routines are available for use with the IMSL and NAG libraries.

KEY WORDS & PHRASES: *ALGOL 68 - FORTRAN - Interface*

*) This report will be submitted for publication elsewhere.

CONTENTS

1. INTRODUCTION

This paper contains a description of AFLINK (an abbreviation of ALGOL 68 ([1], [2])-FORTRAN ([3])-LINK). Such a language link (or interface) is used to call FORTRAN subroutines from ALGOL 68 programs. In particular AFLINK has been designed to be able to pass ALGOL 68 procedures as parameters to the FORTRAN routines used.

Thus it provides the facility of calling an ALGOL 68 procedure from a FORTRAN routine. Moreover, in its turn the ALGOL 68 procedure called may call some (other) ALGOL 68 procedure or FORTRAN routine again (provided the latter is not already active), or it may be a FORTRAN routine, supplied with an ALGOL 68 heading.

In principle this may be repeated to any depth. However, a problem may occur when the ALGOL 68 program makes a heap request. This problem and how to avoid it will be discussed in section 3.3.

2. THE STRUCTURE OF THE INTERFACE

AFLINK is a library consisting of two ALGOL 68 preludes (called AFL and LAFL), two separately compiled ALGOL 68 procedures (called LINKA68FTN and UERTSTX) and four COMPASS modules (called LINK, A68IMSL, A68NAG and DECODE).

An ALGOL 68 source program that passes one or more ALGOL 68 procedures as parameters to one or more FORTRAN routines should be compiled (by means of the P-option) with the prelude AFL if no long modes are used, and with the prelude LAFL if long modes are used. Moreover, the ALGOL 68 source program should contain a declaration of the procedure LINKA68FTN for each ALGOL 68 procedure that has to be passed to some FORTRAN routine (see section 3.4.). Upon loading of the binary code an explicit call of the library by a LIBRARY, AFLINK control statement is not necessary, since the program has been compiled with the P-option, which already added AFLINK to the local library set.

3. HOW TO USE THE INTERFACE

3.1. The preludes AFL and LAFL

The preludes AFL and LAFL contain declarations of the mode link (which will be discussed in section 3.4) and the following TAG-symbols and bold-TAG-symbols:

ref
plain
double
array
onedim
twodim
threedim
param
actual

These symbols are used to describe the parameters of an ALGOL 68 procedure that will be passed to FORTRAN. Why and how this is done will also be discussed in section 3.4.

3.2. Passing parameters from ALGOL 68 to FORTRAN

Any FORTRAN routine to be called from an ALGOL 68 program should be declared in the ALGOL 68 source program. This should be done as follows:

```
proc name = (model par1,...,moden parn) result:
  pr xref a68ftn, name of the FORTRAN routine pr skip;
```

"name" may be any TAG-symbol.
"result" should be void if the FORTRAN routine is a subroutine. If the FORTRAN routine is a function, the correspondence between the type of the function and result is as follows:

type of FORTRAN function	result
REAL	real
INTEGER	int
LOGICAL	int (see remark at the end of this section)
COMPLEX	compl
DOUBLE PRECISION	long real

"name of the FORTRAN routine" should be the symbolic name of the FORTRAN routine.

"a68ftn" is the interface routine through which the FORTRAN routine is entered. Three of these interface routines are available: A68FTN (a subroutine of LINK), A68IMSL and A68NAG. They will be discussed in section 3.3.

"par1" through "parn" are the ALGOL 68 parameters to be passed to the FORTRAN routine in the same order as they appear in the FORTRAN heading. The correspondence between the modes of these ALGOL 68 parameters ("model" through "moden") and the types of the FORTRAN parameters is as follows:

FORTRAN expects	ALGOL 68 passes	
REAL	ref real	
INTEGER	ref int	
LOGICAL	ref int	
COMPLEX	ref compl	
DOUBLE PRECISION	ref long real	
REAL ARRAY	ref real	ALGOL 68 should pass the first element only
INTEGER ARRAY	ref int	
LOGICAL ARRAY	ref int	
COMPLEX ARRAY	ref compl	
DOUBLE PRECISION ARRAY	ref long real	
SUBROUTINE	link	(the construction of an object of this mode will be discussed in section 3.4.)
TYPE FUNCTION	link	

(TYPE stands for REAL, INTEGER, LOGICAL, COMPLEX or DOUBLE PRECISION)

Two remarks should be made with relation to the correspondence between the ALGOL 68 and FORTRAN parameters:

1) If FORTRAN expects a LOGICAL, ALGOL 68 should pass a ref int, and if the FORTRAN routine is a LOGICAL FUNCTION, result should be int (as we saw). An integer value <0 corresponds with a LOGICAL value •TRUE•, a value ≥0 with a LOGICAL value •FALSE•.
2) FORTRAN expects all arrays stored columnwise and in contiguous memory locations, whereas ALGOL 68 stores its multiples rowwise. Moreover, ALGOL 68 multiples may contain "holes" due to e.g. slicing. Therefore it is advised to make a copy of the transposed multiple before it is passed to FORTRAN, unless one is absolutely sure that the multiple contains no holes and that it has to be used in its "transposed" form. Moreover, ALGOL 68 should pass a reference to the first element only instead of a reference to the whole multiple.

3.3. The interface routines A68FTN, A68IMSL & A68NAG

As already mentioned in the preceding section, there are three interface routines available: A68FTN (which is a subroutine of LINK), A68IMSL and A68NAG.

A68FTN is the standard interface routine through which a FORTRAN routine is entered. A68FTN should be used if the FORTRAN routine is not from the IMSL ([4]) or NAG ([5]) library. A68FTN performs the following actions:

1) Check if the FORTRAN routine to be entered is already active (FORTRAN allows no recursion). If so, the message: ATTEMPT TO CALL ALREADY ACTIVE FTN ROUTINE is printed, the program is terminated and a complete traceback is given.
2) Check if one or more ALGOL 68 procedures have to be passed. If not, the following two steps are skipped.
3) Check if at least 1000 words between the stack and the heap are available. If not, make a heap request to get them.
4) Take actions to make the heap completely static (so it can't be moved or changed). This is necessary because FORTRAN changes addresses in its code, so the FORTRAN routine to be entered wouldn't be able to find objects on the heap when an ALGOL 68 procedure entered from this FORTRAN routine

moves or changes the heap.

[Here the problem mentioned in the introduction may occur. Suppose the following situation arises:

A FORTRAN routine has been entered from the ALGOL 68 program, so the heap has been made completely static and at least 1000 words are available between the stack and the heap. This FORTRAN routine has called an ALGOL 68 procedure, which has put some things on the heap. If this ALGOL 68 procedure calls another FORTRAN routine, A68FTN will be entered, but now it could be impossible to get 1000 words between the stack and the heap, because perhaps less than 1000 words are available and the heap can't be moved. If this is the case, the program will be terminated. Of course an error message will be printed and a complete traceback will be given. This problem can be avoided if the program is executed with an EFL control statement (INTERCOM), a CM parameter on the job card (BATCH) or a RFL control statement (BATCH). In that case the top of the heap will be at the "top" of the specified fieldlength, so the program can use the complete specified fieldlength.]

When the FORTRAN routine is left, A68FTN performs the following action:
- Check if some FORTRAN routine is still active. If not, take actions to achieve that the heap is no longer static.

A68IMSL should be used if the FORTRAN routine is taken from the IMSL library. A68IMSL performs the following actions:
- Add IMSL to the local library set (so a LIBRARY, IMSL statement will be redundant for the current load).
- Instead of the IMSL error handling routine UERTST, load the following three routines from AFLINK:
  1) a new error handling routine UERTST (this is a subroutine of A68IMSL)
  2) a separately compiled ALGOL 68 procedure UERTSTX (which helps UERTST to print error messages)
  3) a COMPASS module DECODE (which helps UERTSTX to convert the error messages from display code to ASCII code).

These three routines take over the task of the original error handling routine.
- Enter the IMSL routine through A68FTN.

A68NAG should be used if the FORTRAN routine is taken from the NAG library. A68NAG performs the following actions:
- Add NAG to the local library set (so a LIBRARY, NAG statement will be redundant for the current load)
- Instead of the NAG error trapping routine P01AAF, load a new routine P01AAF (this is a subroutine of A68NAG). This routine takes over the task of the original error trapping routine.
- Enter the NAG routine through A68FTN.

3.4. Passing Parameters from FORTRAN to ALGOL 68: The Procedure LINKA68FTN.

If the FORTRAN routine, which is called from an ALGOL 68 program, expects a FORTRAN function or subroutine as a parameter, the ALGOL 68 program does not directly pass a corresponding ALGOL 68 procedure, but instead, it passes an object of the mode link. This object of the mode link is delivered by the ALGOL 68 procedure LINKA68FTN, which is a separately compiled procedure in the library AFLINK. For each creation of an object of the mode link the ALGOL 68 source program should contain a declaration of LINKA68FTN, (see also the note at the end of this section). The declaration of LINKA68FTN should read as follows:

```
proc linka68ftn = (procmode proc, ref [ ] bits pars)link:
  pr xref xlink pr skip;
```

where "procmode" is the mode of the ALGOL 68 procedure that will be passed to FORTRAN.

The actual parameters of LINKA68FTN should be as follows:

The first parameter of LINKA68FTN (procmode proc) should be the ALGOL 68 procedure to be passed. This procedure may have the mode

```
proc(amode1,...,amoden)bmode
```

where "amodei" (i = 1,...,n) may be any of the following modes:

cmode
ref cmode
ref [ ] cmode
ref [,] cmode
ref [,,] cmode

where cmode may be int, real, long real or compl and bmode may be a cmode or void. If FORTRAN expects a subroutine as a parameter, bmode should be void. If FORTRAN expects a function, the correspondence between the type of the function and bmode should be as follows:

type of FORTRAN function	bmode
REAL	real
INTEGER	int
LOGICAL	int (see remark at the end of section 3.2)
COMPLEX	compl
DOUBLE PRECISION	long real

The second parameter of LINKA68FTN should be a reference to a row of bits. This row of bits is used to identify the modes of the parameters of the procmode proc at run time. This is necessary because otherwise the modes of the parameters would be unknown at run time, and FORTRAN passes only addresses of parameters. So, to distinguish between plain values, references to values and references to rows of values, some appropriate action has to be taken at runtime when parameters are passed from FORTRAN to ALGOL 68. The elements of the row of bits (the "descriptors") tell exactly what action should be taken with relation to each parameter: there should be a one to one correspondence between the parameters and the descriptors. The descriptors (and their constitutive parts) have been declared in the preludes, and they all yield a value of mode bits. The correspondence between the ALGOL 68 parameters, the FORTRAN parameters and the descriptors is as follows:

ALGOL 68 expects (parameters of procmode proc)	Descriptor	FTN passes
int	plain	LOGICAL (see remark at end of section 3.2)
ref int	ref	
ref [ ] int	array onedim (op n_1)	
ref [,] int	array twodim (op n_1, op n_2)	
ref [,,] int	array threedim (op n_1, op n_2, op n_3)	
int	plain	INTEGER
ref int	ref	
ref [ ] int	array onedim (op n_1)	
ref [,] int	array twodim (op n_1, op n_2)	
ref [,,] int	array threedim (op n_1, op n_2, op n_3)	
real	plain	REAL
ref real	ref	
ref [ ] real	array onedim (op n_1)	
ref [,] real	array twodim (op n_1, op n_2)	
ref [,,] real	array threedim (op n_1, op n_2, op n_3)	
long real	double plain	DOUBLE PRECISION
ref long real	double ref	
ref [ ] long real	double array onedim (op n_1)	
ref [,] long real	double array twodim (op n_1, op n_2)	
ref [,,] long real	double array threedim (op n_1, op n_2, op n_3)	
compl	double plain	COMPLEX
ref compl	double ref	
ref [ ] compl	double array onedim (op n_1)	
ref [,] compl	double array twodim (op n_1, op n_2)	
ref [,,] compl	double array threedim (op n_1, op n_2, op n_3)	

where op stands for param or actual.

If the ALGOL 68 procedure expects a row, the following two rules are obeyed:

1) All lowerbounds of rows passed to it are equal to 1.
2) The i-th upperbound
   a) can be found in the n_i-th parameter of procmode proc (i.e. also the n_i-th parameter in the FORTRAN routine that has been replaced by procmode proc) if op is param
   b) is equal to n_i if op is actual (the operand n_i should yield an integer value)

As stated earlier in this section, LINKA68FTN delivers an object of mode link. This object should be passed to the FORTRAN routine instead of the procmode proc. If the FORTRAN routine calls the procmode proc, a jump is made to the address specified by the object of the mode link. Next, a return jump to FTNA68 (a subroutine of LINK through which an ALGOL 68 procedure is entered) is made. FTNA68 knows, by means of the object of the mode link, where the procedure word of the ALGOL 68 procedure and the ref [ ] bits can be found. FTNA68 modifies the parameters according to their descriptors and finally the ALGOL 68 procedure is entered via G;CALL just like any other ALGOL 68 procedure.

Note: When the modes of two or more ALGOL 68 procedures that are passed to one or more FORTRAN routines are identical, the same declaration of LINKA68FTN may be used to create more than one object of the mode link.

E.g., after the declarations

```
mode fun = proc(real)real;
fun f1 = (real x)real: x * x,
    f2 = (real x)real: exp(x);
```

one may write

```
proc linka68ftn = (fun f, ref [ ] bits pars)link:
  pr xref xlink pr skip;
link link1 = linka68ftn(f1, heap [1:1] bits := (plain)),
     link2 = linka68ftn(f2, heap [1:1] bits := (plain));
```

instead of

```
link link1 =
   (proc linka68ftn = (fun f, ref [ ] bits pars)link:
      pr xref xlink pr skip;
   linka68ftn(f1, heap [1:1] bits := (plain))),
     link2 =
   (proc linka68ftn = (fun f, ref [ ] bits pars)link:
      pr xref xlink pr skip;
   linka68ftn(f2, heap [1:1] bits := (plain)));
```

4. EXAMPLES OF USE

```
     1.            ( #  EXAMPLE  1  #
     2.
     3.               'MODE' 'ROUT' = 'PROC'('REF' [] 'REAL', 'REAL',
     4.                                      'REF' [] 'REAL')'VOID';
     5.
     6.
     7.               'PROC' CHRISTIANSEN = ('ROUT' F, 'REAL' START, END,
     8.                                      'REF' 'REAL' STEPSIZE,
     9.                                      'REF' [] 'REAL' X)'VOID':
    10.
    11.                   # THIS PROCEDURE SOLVES A SYSTEM OF FIRST ORDER ORDINARY        #
    12.                   # DIFFERENTIAL EQUATIONS. FOR FURTHER INFORMATION SEE          #
    13.                   # DESCRIPTION OF SUBROUTINE DASCRU FROM THE IMSL LIBRARY.      #
    14.
    15.                  'BEGIN' 'INT' N := 'UPB' X; [1:4*N] 'REAL' WK;
    16.                    'MODE' 'ROUT1' = 'PROC'('REF' [] 'REAL', 'REAL', 'INT',
    17.                                           'REF' [] 'REAL')'VOID';
    18.                    'ROUT1' FF = ('REF' [] 'REAL' X0, 'REAL' T, 'INT' N,
    19.                                  'REF' [] 'REAL' XP)'VOID': F(X0, T, XP);
    20.                    'PROC' LINKA68FTN = ('ROUT1' PROC, 'REF' [] 'BITS' PARS)'LINK':
    21.                       'PR' XREF XLINK 'PR' 'SKIP';
    22.                    'PROC' DASCRU = ('LINK' L, 'REF' 'REAL' A, B, H, 'REF' 'INT' N,
    23.                                     'REF' 'REAL' X0, WK, 'REF' 'INT' IER)'VOID':
    24.                       'PR' XREF A68IMSL,DASCRU 'PR' 'SKIP';
    25.                     DASCRU(LINKA68FTN(FF, 'HEAP' [1:4] 'BITS' :=
    26.                                       ('ARRAY''ONEDIM' ('PARAM' 3), PLAIN, PLAIN,
    27.                                        'ARRAY''ONEDIM' ('PARAM' 3))),
    28.                            'LOC' 'REAL' := START, 'LOC' 'REAL' := END,
    29.                            STEPSIZE, N, X[1], WK[1], 'LOC' 'INT')
    30.                  'END'; # CHRISTIANSEN #
    31.
    32.
    33.               'REAL' STEPSIZE := 1.0E-4, [1:2]'REAL' X := (1.0, 0.0);
    34.
    35.
    36.               'ROUT' F = ('REF' [] 'REAL' X, 'REAL' T,
    37.                           'REF' [] 'REAL' DXDT)'VOID':
    38.                  ( DXDT[1] := X[2]; DXDT[2] := X[1] + T );
    39.
    40.
    41.               #  SYSTEM  :  X'' - X = T  ;  X(0) = 1  ;  X'(0) = 0  #
    42.
    43.
    44.                CHRISTIANSEN(F, 0.0, 2.0, STEPSIZE, X);
    45.
    46.
    47.                PRINT(("COMPUTED SOLUTION  : X(2.0) = ", X[1], NEWLINE,
    48.                       "EXACT    SOLUTION  : X(2.0) = ", EXP(2.0) - 2.0))
    49.
    50.
    51.            )
PROGRAM LENGTH     000521B WORDS
REQUIRED  CM 047600. CP  2.434 SEC.
SPECIFIED OPTIONS         PDS

 COMPUTED SOLUTION  : X(2.0) =  +5.389025825869879E  +0
 EXACT    SOLUTION  : X(2.0) =  +5.389056098930610E  +0
```

```
      1.            ( #  EXAMPLE  2  #
      2.
      3.              'MODE' 'ROUT1' = 'PROC'('REAL')'LONG' 'REAL',
      4.                     'ROUT2' = 'PROC'('LONG' 'REAL', 'REAL', 'INT')'LONG' 'REAL';
      5.
      6.
      7.              'PROC' SMALL = ('REF' 'REAL' X, 'LINK' F, G)'LONG' 'REAL':
      8.                'PR' XREF A68FTN,FTNSM 'PR' 'SKIP';
      9.
     10.
     11.              'ROUT1' LEXP = ('REAL' X)'LONG' 'REAL': LONGEXP('LENG' X);
     12.
     13.
     14.              'ROUT2' LTERM = ('LONG' 'REAL' LAST, 'REAL' X, 'INT' I)'LONG' 'REAL':
     15.                ( LAST * 'LENG' X / 'LENG' 'REAL' (I) );
     16.
     17.
     18.              'LINK' LINKLEXP =
     19.                ('PROC' LINKA68FTN = ('ROUT1' PROC, 'REF' [] 'BITS' PARS)'LINK':
     20.                   'PR' XREF XLINK 'PR' 'SKIP';
     21.                  LINKA68FTN(LEXP, 'HEAP'[1:1]'BITS' := (PLAIN))),
     22.
     23.                      LINKLTERM =
     24.                ('PROC' LINKA68FTN = ('ROUT2' PROC, 'REF' [] 'BITS' PARS)'LINK':
     25.                   'PR' XREF XLINK 'PR' 'SKIP';
     26.                  LINKA68FTN(LTERM, 'HEAP'[1:3]'BITS' :=
     27.                                    ('DOUBLE' PLAIN, PLAIN, PLAIN)));
     28.
     29.
     30.              'FOR' I 'TO' 5
     31.              'DO' PRINT((NEWLINE,
     32.                     FIXED(SMALL('LOC' 'REAL' := 'REAL' (I), LINKLEXP, LINKLTERM),
     33.                           35, 29)))
     34.              'OD'
     35.
     36.
     37.            )
PROGRAM LENGTH      000271B WORDS
REQUIRED  CM 052600. CP  1.798 SEC.
SPECIFIED OPTIONS          PDS
```

```
      1                DOUBLE PRECISION FUNCTION FTNSM(X, F, G)
                       EXTERNAL F, G
                       DOUBLE PRECISION F, G, RESULT, LAST
                       RESULT = F(X)
      5                LAST = 1
                       RESULT = RESULT - LAST
                       DO 100 J = 1, 50
                       LAST = G(LAST, X, J)
                  100  RESULT = RESULT - LAST
     10                FTNSM = RESULT
                       RETURN
                       END
```

```
       SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    5  FTNSM

VARIABLES     SN  TYPE            RELOCATION                          54  J          INTEGER
   46  FTNSM     DOUBLE                                                50  RESULT     DOUBLE
   52  LAST      DOUBLE
    0  X         REAL                      F.P.

EXTERNALS          TYPE   ARGS
       F           DOUBLE    1   F.P.                                     G          DOUBLE     3   F.P.

STATEMENT LABELS
    0  100

LOOPS  LABEL    INDEX      FROM-TO     LENGTH     PROPERTIES
   22  100     * J           7 9         13B                EXT REFS

STATISTICS
  PROGRAM LENGTH                  61B       49
          52000B CM USED
```

```
      -0.00000000000000000000000000006
      -0.00000000000000000000000000001
      -0.00000000000000000000000000016
      -0.00000000000000000000000000047
      +0.00000000000000000000000000137
```

```
( #  EXAMPLE  3  #

   'MODE' 'FUN' = 'PROC'('REAL')'REAL';


   'PROC' ROMBERG = ('FUN' FUN, 'REAL' X0, XE, RELACC,
                     'INT' MAX, 'REF' 'INT' EVAL)'REAL':

       # THIS PROCEDURE INTEGRATES THE FUNCTION FUN. FOR FURTHER INFOR-#
       # MATION SEE DESCRIPTION OF SUBROUTINE D01ABF FROM THE NAG      #
       # LIBRARY.                                                      #

      'BEGIN' 'REAL' ANS;
         'PROC' LINKA68FTN = ('FUN' PROC,
                              'REF' [] 'BITS' PARS)'LINK':
            'PR' XREF XLINK 'PR' 'SKIP';
         'PROC' D01ABF = ('REF' 'REAL' A, B, 'LINK' F, 'REF' 'REAL' ACC,
                          'REF' 'INT' NMAX, N, 'REF' 'REAL' ANS,
                          'REF' 'INT' IFAIL)'VOID':
            'PR' XREF A68NAG,D01ABF 'PR' 'SKIP';
         D01ABF('LOC' 'REAL' := X0, 'LOC' 'REAL' := XE,
                LINKA68FTN(FUN, 'HEAP' [1:1] 'BITS' := (PLAIN)),
                'LOC' 'REAL' := RELACC, 'LOC' 'INT' := MAX, EVAL,
                ANS, 'LOC' 'INT' := 0);
         ANS
      'END'; # ROMBERG #


   'FUN' F = ('REAL' XF)'REAL':

       # THIS FUNCTION WILL BE INTEGRATED, SO IT WILL BE PASSED TO AND #
       # CALLED FROM THE NAG ROUTINE D01ABF. BUT NOTE THAT THIS FUNC-  #
       # TION CALLS ANOTHER FORTRAN ROUTINE (SUBROUT), WHICH, IN TURN, #
       # CALLS THE ALGOL68 PROCEDURE MESSAGE, WHICH PRINTS THE ARGUMENT#
       # OF AND THE VALUE DELIVERED BY THE FUNCTION.                   #

      'BEGIN'
         'MODE' 'MESSPROC' = 'PROC'('REAL','REAL')'VOID';
         'REAL' YF := XF * EXP(XF);
         'PROC' SUBROUT = ('LINK' F, 'REF' 'REAL' XS, YS)'VOID':
            'PR' XREF A68FTN,SUBROUT 'PR' 'SKIP';
         'MESSPROC' MESSAGE = ('REAL' XM, YM)'VOID':
             PRINT((NEWLINE, XM, YM));
         'PROC' LINKA68FTN = ( MESSPROC' PROC,
                               REF'[]'BITS' PARS)'LINK':
            'PR' XREF XLINK 'PR' 'SKIP';
         SUBROUT(LINKA68FTN(MESSAGE, 'HEAP'[1:2]'BITS' :=
                            (PLAIN, PLAIN)),
                 'LOC' 'REAL' := XF, YF);
         YF
      'END'; # F #


   'INT' EVAL;
   'REAL' RES = ROMBERG(F, 0.0, 2.0, 1.0E-3, 1024, EVAL);
```

```
     56.
     57.
     58.                PRINT((NEWLINE, NEWLINE, "COMPUTED SOLUTION :", RES,
     59.                      "         NUMBER OF FUNCTION EVALUATIONS = ", WHOLE(EVAL, -5),
     60.                      NEWLINE, "EXACT    SOLUTION :", EXP(2.0) + 1.0))
     61.
     62.
     63.           )
PROGRAM LENGTH      000567B WORDS
REQUIRED  CM 050000. CP  2.587 SEC.
SPECIFIED OPTIONS          PDS
```

```
      1                SUBROUTINE SUBROUT(SUBRT, XX, YY)
                       EXTERNAL SUBRT
                       CALL SUBRT(XX, YY)
                       RETURN
      5                END
```

```
      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  SUBROUT

VARIABLES     SN  TYPE            RELOCATION
    0  XX         REAL                 F.P.          0  YY        REAL                 F.P.

EXTERNALS           TYPE    ARGS
       SUBRT                  2   F.P.

STATISTICS
  PROGRAM LENGTH                  21B       17
         52000B CM USED
```

```
 +0.000000000000000E  +0 +0.000000000000000E  +0
 +2.000000000000000E  +0 +1.477811219786122E  +1
 +1.000000000000000E  +0 +2.718281828459041E  +0
 +5.000000000000000E  -1 +8.243606353500645E  -1
 +1.500000000000000E  +0 +6.722533605507095E  +0
 +2.500000000000000E  -1 +3.210063541719332E  -1
 +7.500000000000000E  -1 +1.587750012459495E  +0
 +1.250000000000000E  +0 +4.362928696827254E  +0
 +1.750000000000000E  +0 +1.007055468300996E  +1

 COMPUTED SOLUTION : +8.389058729209694E  +0        NUMBER OF FUNCTION EVALUATIONS =      9
 EXACT    SOLUTION : +8.389056098930610E  +0
```

REFERENCES

[1] WIJNGAARDEN, A. VAN et al. (eds.) [1976], *Revised report on the algorithmic language ALGOL 68*, MC Tract 50, Mathematisch Centrum, Amsterdam.

[2] ALGOL 68 VERSION 1 REFERENCE MANUAL

[3] FORTRAN EXTENDED VERSION 4 REFERENCE MANUAL

[4] IMSL, *International Mathematical and Statistical Libraries*, Library 3, Reference manual.

[5] NAG, *Numerical Algorithms Group*, Mark 6, Reference manual.