Technical Report No. 281

A METHOD OF GENERATING A SINGLE R=2 FORTRAN CROSS REFERENCE MAP FOR A LARGE SIMCOMP SIMULATION

George W. Cole

Natural Resource Ecology Laboratory

Colorado State University

Fort Collins, Colorado

GRASSLAND BIOME

U.S. International Biological Program

April 1975

TABLE OF CONTENTS

ABSTRACT

A method is described whereby a single FORTRAN cross reference map can be generated for a multi-subroutive SIMCOMP simulation program. This map is essential for tracing through the FORTRAN coding when the simulation consists of a large number of statements.

INTRODUCTION

When making changes to or trying to comprehend a large multi-subroutine SIMCOMP simulation (Gustafson and Innis 1973), a FORTRAN cross reference map (Control Data Corporation 1973) is essential for tracing out where variables are utilized within the coding. Prior to the development of this series of three programs which generates one cross reference map, the user was required to accept a cross reference map for each routine. This order method generates excessive amounts of useless paper and is quite difficult to use for a large simulation with many subroutines. The difficulty involves searching each map for the variable of interest. For a small program, there is probably no advantage to utilizing the approach described herein; however, for medium-sized and certainly for a large program such as ELM, it is essential for sanity.

This method can also be used for a multi-subroutine FORTRAN program by starting with an UPDATE file of the FORTRAN program, called UPDATE A (see Task 2 below).

OBJECTIVES

The objective of this set of three programs, identified as Tasks 1, 2, and 3, is to take a SIMCOMP program, which has been successfully compiled and run, and produce a single R=2 FORTRAN cross reference map. This is accomplished by combining all subroutines into a single subroutine prior to compilation by the FORTRAN compiler.

Figures 1, 3, and 5 illustrate the sequence of steps in flow chart format with the attendant control cards. For convenience the total job has been subdivided into three convenient tasks, whose separation points coincide with those points in the procedure where manual intervention is

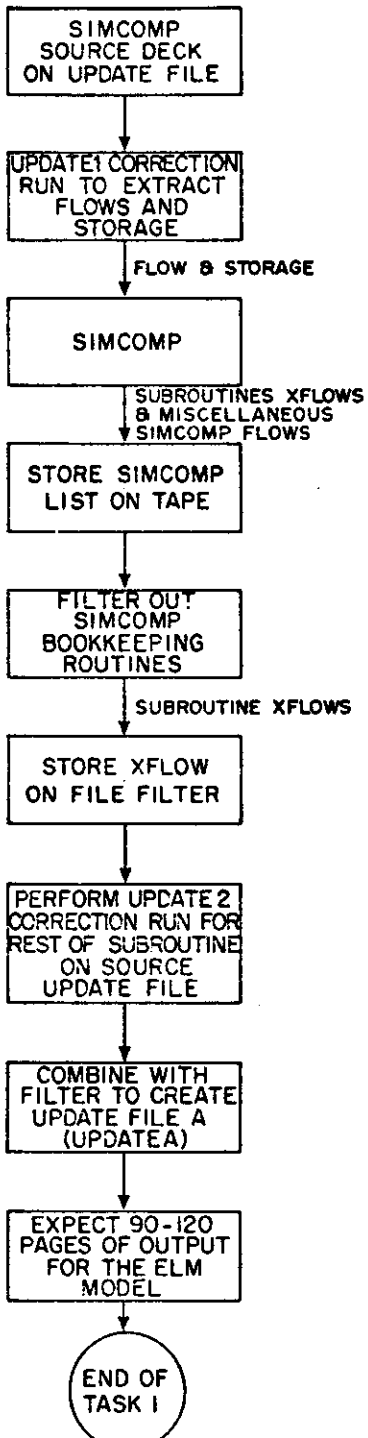
required. In the following task descriptions, references to Fig. 1-6 are suggested.

Task 1 (See Fig. 1 and 2)

In this task the objective is to create a file of FORTRAN-compatible statements; that is, we must correct all unique SIMCOMP statements to their FORTRAN equivalent. This is required only for SIMCOMP FLOW & STORAGE statements, since the rest of the subroutines are already FORTRAN-compatible, with the exception of EVENTS which are handled in Task 2. From Fig. 1 it can be noted the flows and storage statements are processed first (UPDATE 1) and the rest of the subroutines are processed later (UPDATE 2). The details are as follows.

It is assumed that a SIMCOMP source deck is available on a previously created UPDATE file (Control Data Corporation 1973). With this file we perform UPDATE 1 to extract out the storage and flow statements which are to be processed through the SIMCOMP compiler. Only the flows and storage statements are used here, for if we fed the total simulation into SIMCOMP, we would get redundant common cards in each subroutine. These would eventually require removal, before the generation, in Task 3, of the cross reference map.

After storage on tape of the SIMCOMP listing of subroutine XFLOWS and the storage statements (for future print out with the cross reference map), a FORTRAN program (called SIFT) is compiled and executed. SIFT sifts or filters out some SIMCOMP-generated subroutines and leaves only subroutines XFLOW, START, CYCL1, CYCL2, and FINIS; the latter four subroutines contain nothing. The results of the sifting are stored temporarily on a file called FILTER. This file is used later with a file called COMP1 to create a new update file.




```
TA***,CM53000,T100,MT1,ANM*****. NAME. TASK 1 CREATE UPDATE FILE A
MAP(OFF)
COMMENT. THE FOLLOWING ATTACH SHOULD BE CHANGED TO ATTACH YOUR UPDATE FILE NAME
ATTACH(OLDPL,SELM74UPDATE102374,ID=NREL,MR=1)
UPDATE(L=A12,D)


ATTACH(SIMCOM,SIMCOM3,CY=1,MR=1,ID=NREL)
SIMCOM(COMPILE)



COMMENT. SIMCOMP LIST IS STORED ON TAPE
REQUEST(TAPE1,ID=A****,MY,WRITE)SWIFT
REWIND(TAPE1)
REWIND(OUTPUT)
SKIPF(TAPE1,2,17,C) THE EXACT POSITION DEPENDS ON YOUR TAPE
COPYCF(OUTPUT,TAPE1)
RETURN(TAPE1)

REWIND(SIMPRG)
FTN.
ATTACH(LIB,FTNLIB,ID=APPLE)
SELECT.
LGO(SIMPRG,,FILTER)



REWIND(FILTER)
REWIND(SIMPRG,COMP1)
UPDATE(C=COMP1,L=A12,D)



REQUEST(NEWW,*PF)
UPDATE(O,N=NEWW,L=A1234,D)
CATALOG(NEWW,UPDATEA,ID=NREL,RP=5)
6
```

Fig. 1. Flow chart and control cards for Task 1; the creation of UPDATE FILE A.

```
TA***,CM53000,T100,MT1,ANM*****. NAME/ TASK 1 CREATE UPDATE FILE A
MAP(OFF)
COMMENT. THE FOLLOWING ATTACH SHOULD BE CHANGED TO ATTACH YOUR UPDATE FILE NAME
ATTACH(OLDPL,SELM74UPDATE102374,ID=NREL,MR=1)
UPDATE(L=A12,D)
ATTACH(SIMCOM,SIMCOM3,CY=1,MR=1,ID=NREL)
SIMCOM(COMPILE)
COMMENT. SIMCOMP LIST IS STORED ON TAPE
REQUEST(TAPE1,ID-A****,HY,WRITE)SWIFT
REWIND(TAPE1)
REWIND(OUTPUT)
SKIPF(TAPE1,2,17,C) THE EXACT POSITION DEPENDS ON YOUR TAPE
COPYCF(OUTPUT,TAPE1)
RETURN(TAPE1)
REWIND(SIMPRG)
FTN.
ATTACH(LIB,FTNLIB,ID=APPLE)
SELECT.
LGO(SIMPRG,,FILTER)
REWIND(FILTER)
REWIND(SIMPRG,COMP1)
UPDATE(C=COMP1,L=A12,D)
REQUEST(NEWW,*PF)
UPDATE(Q,N=NEWW,L=A1234,D)
CATALOG(NEWW,UPDATEA,ID=NREL,RP=5)
&
*IDENT GO
*COMPILE ELM74
*D,ELM74.3
C      ALLOWS GENERATION OF SIMCOMP LISTING
*D,ELM74.137,1356
*D,ELM74.2378,4566
&
      PROGRAM SIFT(INPUT,OUTPUT,TAPE1,TAPE5=INPUT)
C
C.. THIS PGM SIFTS THROUGH THE OUTPUT OF A SIMCOMP COMPILATION AND WRITES
C..  SUBROUTINE XFLOWS AND ALL FOLLOWING SUBROUTINES TO TAPE1
C
      DIMENSION KARD(8),IVAR(2)
      IVAR(1)=10HSUBROUTINE    $  IVAR(2)=10H XFLOWS
   50 READ(5,100) KARD
  100 FORMAT(8A10)
      IF(CMPRF(IVAR,1,KARD,7,17))50,20,50
   20 WRITE(1,100)KARD
   30 READ(5,100) KARD
      IF(EOF(5)) 40,20
   40 CONTINUE
      END
&
*IDENT GO1
*COMPILE ELM74
*D,ELM74.2,136
*D,ELM74.1357,2377
&
*DECK DECK1
*READ FILTER
*READ COMP1
#
```

Fig. 2. Listing of typical control cards and source data decks for Task 1.

File COMP1 which brings on the rest of the subroutines for binding with subroutine XFLOWS is created after the second update of the original update file is performed.

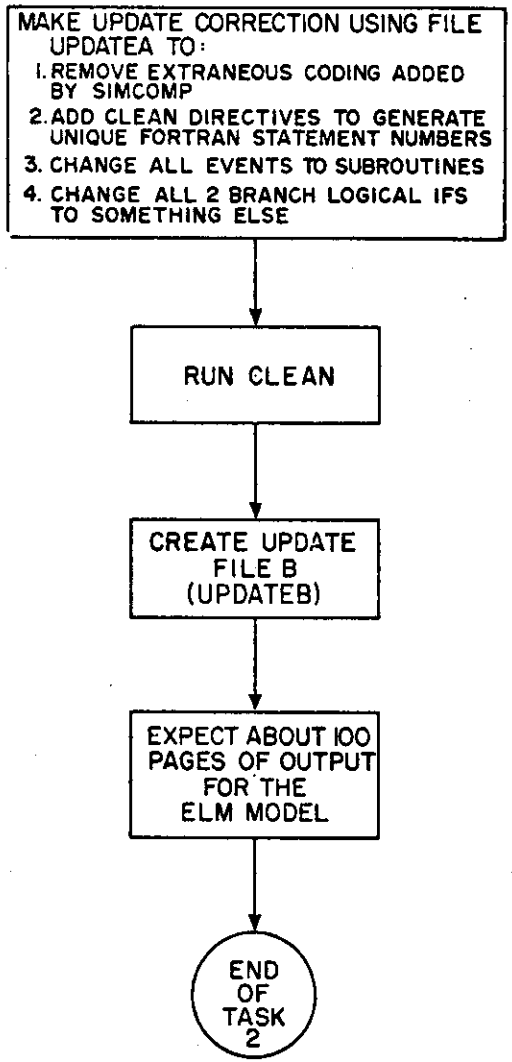
Finally a new update file is created by combining files FILTER (subroutine XFLOW, essentially) and COMP1 (the remainder of the subroutines) and storing on a file called UPDATE A.

Task 2 (see Fig. 3 and 4)

This task is required to (1) remove the extraneous subroutines (START, CYCL1, CYCL2, and FINIS) added by the SIMCOMP, (2) change all SIMCOMP EVENTS to subroutines, since the FORTRAN compiler will not recognize these statements, and (3) resequence all statement numbers. If the EVENTS are only called externally to the program, they will not be present in the final map as "externals." Consequently, if the cross reference map is used to generate a glossary, these names will be missing. They must be accounted for manually. The resequencing is required because the FORTRAN compiler will issue fatal errors when the subroutine boundaries are removed, if there is an overlap in statement numbers between subroutines.

The resequencing of statement numbers is accomplished by using the CSU Computer Center program CLEAN (CSU Computer Center Consultants). Certain CLEAN directives are required and are shown in Fig. 4. The basic directives are:

(1)	/BASE = XXXX	Sets lower index on statement number. One required for each subroutines being resequenced.
(2)	/LAST /STOP	Lets CLEAN know that run is ended.
(3)	/NO COLLECT	Prevents relocating format statements after RETURN and before END cards. If not done, this complicates subroutines boundary removal in Task 3.




```
TA*!* CM70000,T225,ANMR*****. NAME.  TASK 2 CREATE UPDATE FILE B
MAP.OFF)
ATTACH (OLDPL,UPDATEA,ID=NREL,MR=1)
UPDATE (L=A12,D)

ATTACH(LIB,APPLIB,ID=APPLE)
SELECT(P=CLEAN)
LGO(OUT),,COMPILE)
REWIND(OUTPUT)

REWIND(OUT1)
REQUEST(NEW1,*PF)
UPDATE(Q,N=NEW1,L=A1234,D)
CATALOG(NEW1,UPDATEB,ID=NREL,RP=5)
&
```

Fig. 3. Flow chart and control cards for Task 2; creation of UPDATE FILE B.

```
TA***,CM70000,T225,ANMR*****. NAME/. TASK 2 CREATE UPDATE FILE B
MAP(OFF)
ATTACH(OLDPL,UPDATEA,ID=NREL,MR=1)
UPDATE(L=A12,D)
ATTACH(LIB,APPLIB,ID=APPLE)
SELECT(P=CLEAN)
LGO(OUT1,,COMPILE)
REWIND(OUTPUT)
REWIND(OUT1)
REQUEST(NEW1,*PF)
UPDATE(Q,N=NEW1,L=A1234,D)
CATALOG(NEW1,UPDATEB,ID=NREL,RP=5)
&
*IDENT G02
*COMPILE DECK1
*D,DECK1.1008,1009
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(SBF-2.0)8050,8020,8020
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
 8020 IF(SBF-SPBF(K))8050,8030,8030
*D,DECK1.1011
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(SBF-TEM1)8050,8040,8040
*D,DECK1.1168
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(STBLS-0.0)9000,9000,9001
*D,DECK1.1335
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(FRPCN.GE.0.0005)GO TO 9008
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(STBLS.GT.0.0)GO TO 9008
*D,DECK1.1339
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(SPHEN-5.2)9010,9010,9009
*D,DECK1.1362
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(FRPCN.GE.0.0005)GO TO 9018
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(STBCR.GT.0.0)GO TO 9018
*D,DECK1.1366
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(SPHEN-5.2)9020,9020,9019
*D,DECK1.1385,1386
C..    THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C      LOGICAL IF
       IF(SKEY(1)-1.)8305,8300
```

Fig. 4. Listing of typical control cards and source decks for Task 2.

```
C..   THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C     LOGICAL IF
 8300 IF(TIME-TSTRT)8305,8301
*D,DECK1.1404
C..   THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C     LOGICAL IF
      IF(SKEY(2)-1.0)8350,8340
*D,DECK1.2601
C..   THE FOLLOWING IF STATEMENT IN THE ORIGINAL CODING IS A 2 BRANCH
C     LOGICAL IF
      IF(FRPCN-0.001)9010,9020,9020
*I,DECK1.1
/BASE=1000
*I,DECK1.1417,1428
/BASE=2000
*I,DECK1.1786
/BASE=3000
*I,DECK1.2649
/BASE=4000
*I,DECK1.2694
/BASE=5000
*I,DECK1.2710
/BASE=6000
*I,DECK1.2727
/BASE=7000
*I,DECK1.2743
/BASE=8000
*I,DECK1.2765
/BASE=9000
*I,DECK1.2790
/BASE=10000
*I,DECK1.2823
/BASE=11000
*I,DECK1.2970
/BASE=12000
*I,DECK1.2975
/BASE=13000
*I,DECK1.2988
/BASE=14000
*I,DECK1.3003
/BASE=15000
*I,DECK1.3125
/BASE=16000
*I,DECK1.3257
/BASE=17000
*I,DECK1.3358
/BASE=18000
*I,DECK1.3376
/BASE=19000
*I,DECK1.3525
/BASE=20000
*I,DECK1.3553
/BASE=21000
*I,DECK1.3595
/BASE=22000
*I,DECK1.3695
```

Fig. 4. continued

```
/BASE=23000
*I,DECK1.3714
/BASE=24000
*I,DECK1.3729
/BASE=25000
*I,DECK1.3784
/BASE=26000
*I,DECK1.3985
/BASE=27000
*I,DECK1.3998
/BASE=28000
*D,DECK1.4024
/BASE=29000
      SUBROUTINE COUNT                                          2938
*D,DECK1.4115
/BASE=30000
      SUBROUTINE COWON                                          3025
*D,DECK1.4152
      SUBROUTINE COWOF                                          3056
/BASE=31000
*I,DECK1.4182
/BASE=32000
*I,DECK1.4208
/BASE=33000
*I,DECK1.4255
/BASE=34000
*I,DECK1.4347
/BASE=35000
*I,DECK1.4356
/BASE=36000
*I,DECK1.4363
/BASE=37000
*I,DECK1.4368
/BASE=38000
*I,DECK1.4378
/BASE=39000
*I,DECK1.4387
/BASE=40000
*I,DECK1.4671
/BASE=41000
*D,DECK1.4685
/BASE=42000
      SUBROUTINE          S A C F                               3430
*I,DECK1.4697
/BASE=43000
*I,DECK1.4715
/BASE=44000
*I,DECK1.4724
/BASE=45000
*I,DECK1.4729
/BASE=46000
*D,DECK1.4736
/BASE=47000
      SUBROUTINE          S F E R T                             3463
```

Fig. 4. continued

```
*I,DECK1.4762
/BASE=48000
*I,DECK1.4768
/BASE=49000
*I,DECK1.4802
/BASE=50000
*I,DECK1.4809
/BASE=51000
*D,DECK1.4817
/BASE=52000
      SUBROUTINE              S R S E T                                  3506
*I,DECK1.4828
/BASE=53000
*I,DECK1.4837
/LAST
/STOP
&
*DECK DECK2
*READ OUT1
#
```

Fig. 4. continued

CLEAN has a problem in that it cannot handle two Branch logical IF statements. These statements must be changed to absolute transfers or three Branch arithmetic IF statements.

Should the resequencing of statement numbers not be required, Task 2 can be eliminated with the exception that the EVENTS must still be changed to subroutines; however, this could be moved to Task 3. Also, the input file for Task 3 would now be UPDATE A rather than the present UPDATE B.

Task 3 (see Fig. 5 and 6)

Task 3 is basically the combining of many subroutines into one and compiling the FORTRAN coding to get a single cross reference map. The map is also stored on tape for future generation of extra copies. The three essential tasks are outlined in Fig. 5.

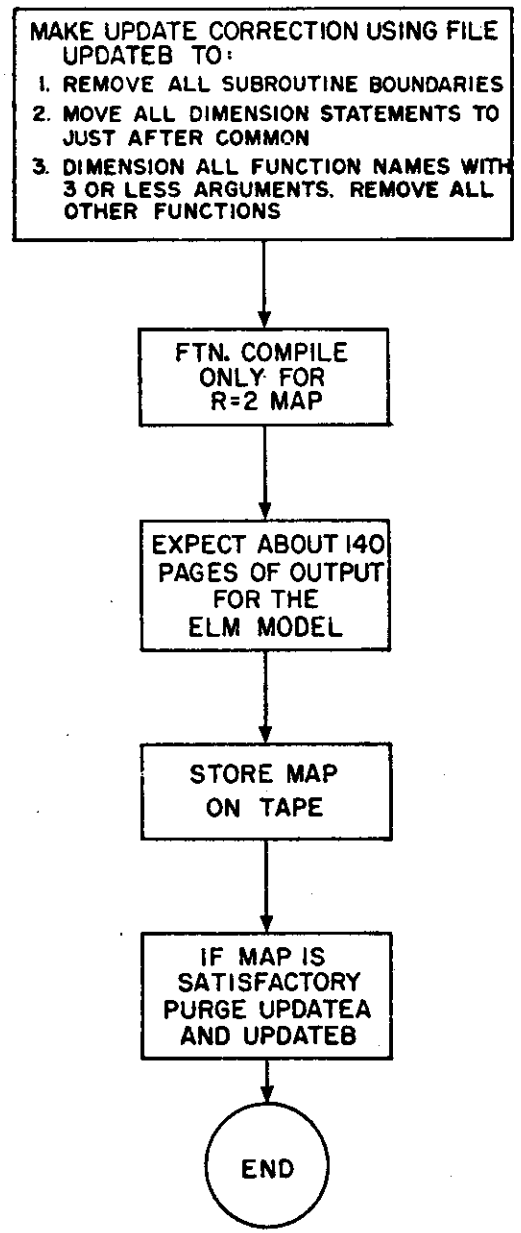
Removal of subroutine boundaries is accomplied by using the update directives in conjunction with update file UPDATE B. We remove END cards except that of the last subroutine and all RETURN cards except those that allow for the program to flow beyond the RETURN statement. All DIMENSION statements which were local to a given subroutine (i.e., not in a STORAGE. statement) must be moved forward to just after the COMMON. statements in subroutine XFLOWS.

All FORTRAN FUNCTIONS must be accounted for in either of two ways, or the FORTRAN compiler will scream:

(1) For those functions with three or less arguments, include the name of the function as a dimensioned variable.

(2) With greater than three arguments, delete the function or rewrite as a subroutine if it is essential that it be included in the map.

Incidently, for all of these cases, sprinkle heavily with comment cards. See examples in Fig. 2, 4, and 6.




```
TA***,MT1,CM140000,T125,ANM*****. NAME. TASK 3 CREATE AND STORE R=2 MAP
RFL(53000)
ATTACH(OLDPL,UPDATEB,ID=NKEL,MR=1)
UPDATE(.=A12,D)
```

```
RFL(140000)
FTN(I=COMPILE,R=2)
```

```
RFL(10000)
REWIND(OUTPUT)
COPYCF(OUTPUT,B)
REQUEST(TAPE1,ID=A****,MY,WRITE)SWIFT
REWIND(TAPE1)
REWIND(B)
SKIPF(TAPE1,3,17,C) THE EXACT POSITION DEPENDS ON TAPE USED
COPYCF(B,TAPE1)
REWIND(TAPE1)
&
```

Fig. 5. Flow chart and control cards for Task 3; the creation and storage of the FORTRAN R=2 Cross Reference Map.

```
TA***,MT1,CM140000,T125,ANM*****. NAME. TASK 3 CREATE AND STORE R=2 MAP
RFL(53000)
ATTACH(OLDPL,UPDATEB,ID=NREL,MR=1)
UPDATE(L=A12,D)
RFL(140000)
FTN(I=COMPILE,R=2)
RFL(10000)
REWIND(OUTPUT)
COPYCF(OUTPUT,B)
REQUEST(TAPE1,ID-A****,HY,WRITE)SWIFT
REWIND(TAPE1)
REWIND(B)
SKIPF(TAPE1,3,17,C) THE EXACT POSITION DEPENDS ON TAPE USED
COPYCF(B,TAPE1)
REWIND(TAPE1)
&
*IDENT G03
*COMPILE DECK2
*I,DECK2.102
C      THE FOLLOWING 5 CARDS OF DIMENSION STATEMENTS WERE REQUIRED
C      TO FOOL THE FORTRAN COMPILER. THEY ARE REALLY THE NAMES OF FUNCTIONS
       DIMENSION ACLAS (10),ABSEV(10),APXT(10),ARAND(1,1)
       DIMENSION ATENS(1,1),ATREV(10),CFBI(10),        DEFT(10),DLCH(10)
       DIMENSION SFA(1,1),SFB(1,1),SFSTC(1)
       DIMENSION DMERQ(10),FS(10),PINTR(10),SAVG(1,1),SCACC(1,1)
        DIMENSION SFSW1(1),SFSW2(1),STANX(1,1,1)
C      THE FOLLOWING 2 CARDS IN THE ORIGINAL LIST ARE IN SUBROUTINE ASTCH
*COPY DECK2,DECK2.3054,DECK2.3055
C... THE FOLLOWING CARD IN THE ORIGINAL LIST IS IN FUNCTION SAVG
*COPY DECK2,DECK2.5322
C... THE FOLLOWING 3 CARDS IN THE ORIGINAL LIST ARE IN SUBROUTINE CYCL1
*COPY DECK2,DECK2.1793,DECK2.1795
C      THE FOLLOWING 2 CARDS IN THE ORIGINAL LIST ARE IN SUBROUTINE ASTCH
*COPY DECK2,DECK2.3056,DECK2.3057
*D,DECK2.1349
*D,DECK2.1355,1374
C      SUBROUTINE START
*D,DECK2.1773
*D,DECK2.1789,1795
C      SUBROUTINE CYCL1
*D,DECK2.2829
*D,DECK2.2833,2837
C      SUBROUTINE CYCL2
*D,DECK2.2887,2894
C      SUBROUTINE FINIS
*D,DECK2.2908
*D,DECK2.2912,2915
C      FUNCTION ACLAS(I)
*D,DECK2.2931,2936
C      FUNCTION ABSEV(TEM1)
*D,DECK2.2951,2956
C      SUBROUTINE APENN
*D,DECK2.2977,2982
C      FUNCTION APXT(I)
*D,DECK2.3006,3011
```

Fig. 6. Listing of typical control cards and source decks for Task 3.

```
C      FUNCTION ARAND(TEM1,TEM2)
*D,DECK2.3043,3046
C      SUBROUTINE ASTCH
*D,DECK2.3054,3057
*D,DECK2.3223
*D,DECK2.3228,3238
C      FUNCTION ATANF(Z,A,B,C,D)
C      THIS FUNCTION WAS DELETED FROM THIS COMPILATION  DUE TO EXCESS
C      ARGUMENTS. THE COMPILER COULD NOT BE FOOLED INTO THINKING THAT
C      IT WAS A VECTOR.
C      FUNCTION ATENS(K,TEM1)
*D,DECK2.3250,3255
C      FUNCTION ATREV(TEM1)
*D,DECK2.3269,3275
C      SUBROUTINE BDULT
*D,DECK2.3413
*D,DECK2.3418,3422
C      SUBROUTINE BYMPH
*D,DECK2.3572,3578
C      SUBROUTINE BEGG
*D,DECK2.3680
*D,DECK2.3684,3688
C      SUBROUTINE BDEAD
*D,DECK2.3704,3709
C      SUBROUTINE CANON
*D,DECK2.3861
*D,DECK2.3865,3869
C      SUBROUTINE CBIND(I,J)
*D,DECK2.3898
*D,DECK2.3902,3905
C      SUBROUTINE CBIRT(I,J)
*D,DECK2.3942
*D,DECK2.3948,3951
C      SUBROUTINE CDETH(I,J)
*D,DECK2.4065,4070
C      SUBROUTINE CDIG(NFC,I,J)
*D,DECK2.4095,4100
C      FUNCTION CFBI(L)
*D,DECK2.4122,4127
C      SUBROUTINE CFOFL(I)
*D,DECK2.4183
*D,DECK2.4189,4192
C      SUBROUTINE CGROW(I,J)
*D,DECK2.4430,4450
C      FUNCTION CLI(AA,BB,CC,DD,EE)
C      THIS FUNCTION WAS DELETED FROM THIS COMPILATION  DUE TO EXCESS
C      ARGUMENTS. THE COMPILER COULD NOT BE FOOLED INTO THINKING THAT
C      IT WAS A VECTOR.
C      SUBROUTINE CMISC
*D,DECK2.4465,4470
C      SUBROUTINE COUNT
*D,DECK2.4559
*D,DECK2.4566,4569
C      SUBROUTINE COWON
*D,DECK2.4600,4605
C      SUBROUTINE COWOF
*D,DECK2.4631,4636
```

Fig. 6. continued

```
C      SUBROUTINE CPRIN(I,J)
*D,DECK2.4651
*D,DECK2.4669,4673
C      SUBROUTINE CRANK(NFC,I,J)
*D,DECK2.4712
*D,DECK2.4717,4721
C      SUBROUTINE DECOM
*D,DECK2.4825
*D,DECK2.4829,4832
C      FUNCTION DEFT(DTEMP)
*D,DECK2.4840,4845
C      FUNCTION DLCH(DTEMP)
*D,DECK2.4849,4854
C      FUNCTION DMERG(DTEMP)
*D,DECK2.4856,4861
C      FUNCTION FS(Y)
*D,DECK2.4869,4874
C      FUNCTION PINTR(A)
*D,DECK2.4879,4884
C      SUBROUTINE PLANT
*D,DECK2.5279,5298
C      FUNCTION PLI(A,B,C,D,E)
C      THIS FUNCTION WAS DELETED FROM THIS COMPILATION  DUE TO EXCESS
C      ARGUMENTS. THE COMPILER COULD NOT BE FOOLED INTO THINKING THAT
C      IT WAS A VECTOR.
C      SUBROUTINE SACF
*D,DECK2.5311,5316
C      FUNCTION SAVG(ARRAY,KK)
*D,DECK2.5322
*D,DECK2.5335,5340
C      FUNCTION SCACC(I,J)
*D,DECK2.5350,5353
C      FUNCTION SFA(STMP,STENN)
*D,DECK2.5358,5363
C      FUNCTION SFB(STMP,STENN)
*D,DECK2.5368,5373
C      SUBROUTINE SFERT
*D,DECK2.5400,5403
C      FUNCTION SFSTC(STMP)
*D,DECK2.5409,5414
C      SUBROUTINE SMASS
*D,DECK2.5461,5466
C      FUNCTION SFSW1(STENN)
*D,DECK2.5472,5477
C      FUNCTION SFSW2(STENN)
*D,DECK2.5483,5488
C      SUBROUTINE SRSET
*D,DECK2.5500,5505
C      FUNCTION STANX(X1,X2,X3)
#
```

Fig. 6. continued

LITERATURE CITED

Control Data Corporation. 1972. SCOPE reference manual, 6000 version 3.0, Chap. 10 Library preparation and maintenance, Revision F, Publication #60305200. Control Data Corporation, Sunnyvale, Calif.

Control Data Corporation. 1973. FORTRAN extended reference manual, 6000 version 3.0, Appendix C Cross reference map, Revision D, Publication #60329100. Control Data Corporation, Sunnyvale, Calif.

CSU Computer Center Consultants. CLEAN documentation. Colorado State Univ,, Fort Collins.

Gustafson, J. D., and G. S. Innis. 1973. SIMCOMP version 3.0 user's manual. US/IBP Grassland Biome Tech. Rep. No. 218. Colorado State Univ., Fort Collins. 149 p.