AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AN IBM 360 VERSION OF THE NEUTRON DIFFUSION CODE CRAM

by

B. McGREGOR

A. G. RICHARDS*

R. G. J. WOOD*

ABSTRACT

The A.A.E.C. 360 CRAM program, written in FORTRAN IV and ASSEMBLER, is designed to run under the 360 Operating System on a 256K model 50. The program can readily be changed to take advantage of additional storage that may be available on other 360 configurations. The code includes standard output routines as well as providing the user with the facility of including his own FORTRAN coded output routines. The code package includes sets of test problems for both one and two dimensional geometries.

* IBM Australia

CONTENTS

1. INTRODUCTION

The 360 CRAM program, used in neutron diffusion studies by the Australian Atomic Energy Commission, has been developed from the version of CRAM previously available for the 7040 system. The 7040 version was produced in 1964 by translating the 7090 CRAM code, originally written in FORTRAN II and FAP by Dr. A. Hassitt of the United Kingdom Atomic Energy Authority. The U.K.A.E.A. report on CRAM, TRG report 229(R), and the CRAM USERS GUIDE prepared by B. M. Segal of Atomic Power Development Associates Inc. are in general applicable to the 360 CRAM program. Parts which no longer apply are noted in the present report, as well as new features of the code. This report uses the terminology of the earlier CRAM reports.

The A.A.E.C. 360 CRAM program, written in FORTRAN IV and ASSEMBLER, is designed to run under the 360 Operating System on a 256K 360 model 50. The program can readily be changed to take advantage of additional storage that may be available on other 360 configurations.

2. CODE ALTERATIONS

The changes noted in this section were made over a period of time, whenever code usage showed that some new feature would improve the code. The six major alterations are as follows:

2.1 Timing

All references to sense switches have been deleted and a clock routine has been included in the code. Problems are terminated by forced convergence after running for a specified time, the time being given in minutes in Columns 70 to 72 of the title card. The time between successive entries into Segment 3 is also printed. If Columns 70 to 72 are left blank a time of 900 minutes is allowed.

2.2 Convergence

The code now requires that both $|MAX^n - 1.0|$ and $|MIN^n - 1.0|$ be less than Accuracy 3 before convergence is satisfied. Previously problems which had failed to satisfy the criterion $|MAX^n - MIN^n| <$ Accuracy 2 could accidentally converge with misleading results.

2.3 Size Searches for Criticality

It was found that many survey calculations included criticality searches on systems which were actually subcritical infinite systems. In these cases the

program had to be terminated by the computer operators and further useful problems were not attempted.. Subroutine CONTRL now includes a test on the magnitude of the control eigenvalue and a test for a negative mesh value. If the absolute value of the control eigenvalue exceeds 1000 or a control eigenvalue leads to a negative mesh, the search is terminated, mesh values are reset to their initial values, and the control eigenvalue is set to zero. Segment 3 is then entered as if convergence had been achieved.

2.4 Alter Accuracy

Accuracy numbers can be changed with the ALTER facility. All five numbers are required as input, e.g. ALTER ACCURACY .0005 .0001 .0005 .0005 .05 ENTER.

2.5 Storage

Storage not used (bytes) is now printed out.

2.6 Dump Unit

The standard dump unit is Data Set 4 (Tape 9 on 7090).

3. INPUT DATA PREPARATION

Data should be prepared in the manner described for the 7090 code. However, the following restrictions apply for the 360 version:

(a) The SYSTEM option is not available.

(b) The LIBRARY option is not available.

(c) Special characters in the data should be punched according to the EBCDIC code (029 keypunch).

(d) E format floating point data should have a decimal point, e.g. 1E9 should be 1.E9 for 360 CRAM.

For input the CRAM program makes use of the subroutine SCAN of Bennett and Pollard (1967) which is a free format input routine coded in ASSEMBLER. This input routine allows the following features which were unavailable in the 7090 code:

(i) A repeat convention. For example 20*5.0 would put 5.0 into the next 20 input locations. No imbedded blanks are allowed in this statement and floating point numbers must have a decimal point.

(ii) Except for the title card, no data item need start on a new card. The input routine is in the root segment.

4. ERROR PRINT OUT

When a problem fails for any reason, a message is printed giving the number of the overlay segment in which the problem failed. The correspondence between segment numbers and program functions is:

Segment 1 : the root segment (initiates execution)

Segment 2 : data input

Segment 3 : processes CRAM options

Segment 4 : problem solution

Segments 5, 6, 7 : standard output routines

Since the most likely causes of problem failure are data errors, the following variables are also printed:

NOON(13) : indicates the mode of the last data item read (0 if numeric, -1 if alphanumeric) ,

NOON(15) : indicates if a re-read function was being performed (0 for re-read) , and

NOON(16) : the pointer to the last column scanned by the input routine.

When a problem fails because an unexpected data item appeared in the input stream, the message 'WORD NOT RECOGNISED' is printed, followed by the word in both A format and F format.

5. THE COMPILE OPTION

The 7090 CRAM code used FAP coded routines to compile and execute output programs which were entered as part of the CRAM data. Two facilities have been provided to replace this feature. The standard program (Segments 5, 6, 7) allows one or more of six output routines to be performed within a CRAM run. If a different routine is required a job is constructed consisting of the FORTRAN compilation of the routine and the link editing of CRAM from a library of object decks stored on disk.

The current CRAM program has the following standard routines, the underlined paragraph headings being the required inputs.

ROUTINE 1 A dummy routine

ROUTINE 2 IOUT

IOUT : Ignored if negative, but if positive is the unit on which a copy of the printer output is produced.

This routine calculates group-dependent volume-integrated fluxes and currents for each material zone in the problem. Besides providing useful output information, it gives an estimate of the multiplication factor, which is usually more accurate than the code value if the problem has not converged.

ROUTINE 3 with additional data

Program to compute microscopic reaction rates for isotopes as loaded in the I-data. For each isotope, data consists of a list of cross section numbers and output options. The program reads pairs of numbers A and B, loaded as data after the words ROUTINE 3. A specifies cross section type:

1. TRANSPORT
2. ABSORPTION
3. NU*FISSION
4. CAPTURE
5. FISSION
6. POWER
7. TOTAL SCATTERING

B specifies the type of print chosen from the following items:

ITEM 1 Final zone cross sections including any composition changes.

ITEM 2 Flux times cross section for all points and all groups.

ITEM 3 Item 2 summed over all groups.

ITEM 4 Data for each zone in the reactor as follows:

(a) Volume of zone.
(b) Cross section times flux times volume integrated over zone, for each group and for sum over groups.
(c) Flux times volume integrated for all groups and for sum over groups.
(d) Item (b) divided by Item (c).

Print options are determined by the value of B.

B = 1 gives Items 1, 2, 3 and 4.
B = 2 gives Items 1, 3 and 4.
B = 3 gives Items 1 and 4.

The program goes to the next isotope in order if A = - 1. If no data is entered Item 1 is still produced.

In the example

ROUTINE 3 1 3 5 3 -1 -1 2 1 -1 -1 -1

there were five isotopes. For Isotope 1, the transport reaction rate is calculated and printed with output Option 3, then the fission reaction rate is printed again with output Option 3. There is no output for Isotope 2, while for Isotope 3 the absorption rate is printed with output Option 1. No output is required for Isotopes 4 and 5.

ROUTINE 4

Program to compute power density at each point and power integrals over zones; all are normalised to a total reactor power of 500 MW. This routine assumes that the I data has three extra cross sections, the last being the power cross section.

ROUTINE 5 POWER, NAZ

POWER : Required power

NAZ : The power is calculated in the first NAZ zones. If NAZ = 0, all zones are included.

Similar to ROUTINE 4 except that only the capture and fission cross sections need to be included as part of the I data.

ROUTINE 6

Prepares flux dump cards for input to TDC.

Routines 3, 4 and 5 are based on compiler routines supplied with the original code package distributed by the Argonne Code Centre.

The control card with the keyword COMPILE (preceded by DUMP), is still used in the same place as with 7090 CRAM.

Following the COMPILE card, a card containing ROUTINE 5 ROUTINE 3 ROUTINE 4... is punched. This will cause the specified routines to be executed in the order given. If, however, one of these routines requires input data from cards, then this data must immediately follow the ROUTINE name which calls for it. Further ROUTINE requests may then be executed by placing their names in the input stream.

The last ROUTINE executed (or its data) must be followed by FINISH punched into a card. This causes return to the program, which then analyses any further control cards, for example,

CARD 1 COMPILE

CARD 2 ROUTINE 6 ROUTINE 3

CARD 3 Input data for Routine 3
etc.

CARD X ROUTINE 4 ROUTINE 2 -1 FINISH

Other Routines

With the use of a library of object decks new output routines can be written, and changed CRAM decks compiled and tested. Segment 5 includes a set of subroutines which are used by the output routines. The COMMON block (Set B) should be copied and included in any FORTRAN output routine that is written. The deck for a run of this type is included in the code package.

The standard parameter list for the output routines is the following:

SUBROUTINE ROUTX(I,J,K,A,B,C,D,X,NOX)

Parameters I,J,K refer to integer arrays each of dimension 100

Parameters A,B,C,D refer to real arrays each of dimension 1000

Parameter X refers to a real array of dimension NOX

The following variables are available to users' FORTRAN routines via the COMMON block:

IMAX	=	Number of points in R direction
JMAX	=	Number of points in Z direction
NG	=	Number of groups
ICON	=	Control number
NEI	=	Number of isotopes
NEM	=	Number of materials
NEZ	=	Number of zones
NDSCAT	=	Number of down scatter groups
NOON(5)	=	Number of dump data set
NT2	=	Number of final flux data set
NT3	=	Number of penultimate flux data set
BSQ	=	Buckling
CONNU	=	1.0/k
NAMGM	=	Geometry code

1 : RZ
2 : ZR
3 : XY
4 : R-THETA
5 : Cylindrical
6 : Slab
7 : Spherical

ADJSW	=	Normal/adjoint (+1/-1)
CHAN	=	Eigenvalue for control types 2,3,4, or 7
R	=	R mesh
Z	=	Z mesh
EAMDA	=	BC
EAMI	=	IBC
SPECT	=	Spectrum
CHANGE	=	Change vector
TITLE	=	Title, format 18A4

All of these arrays are of dimension 100

The following subroutines are available to users' FORTRAN routines:

FRI(L,M)	:	Read M integers from cards into vector L.
FRF(R,M)	:	Read M floating point numbers from cards into vector R.
FFMAX(R,L,M,N)	:	Find the largest number in array R between R(L) and R(M) inclusive and place its subscript in N.
FFMIN(R,L,M,N)	:	Similar to FFMAX but finding the smallest number.
FPA(R,L,M,N)	:	Print the array R comprising L columns and M rows, restricting a line of print to a maximum of N numbers. e.g. If array R contains 18 elements - 6 columns and 3 rows - and print is required to give a maximum of four elements per row, then the calling sequence will be CALL FPA(R,6,3,4) and print will occur as follows:

R(1) R(2) R(3) R(4)
R(7) R(8) R(9) R(10)
R(13) R(14) R(15) R(16)
R(5) R(6)
R(11) R(12)
R(17) R(18)

Note: Since FORTRAN stores matrices with the first index varying most rapidly,

for use with FPA the first index should give the column and the second one indicate the row. This non-standard choice is dictated by the way information is stored by the compiler routines.

FXZ(R,L)	:	Set R(1), R(2),... etc. equal to cross sections for zone L. The complete cross section block is stored.
FXI(R,L)	:	Set R(1), R(2),... etc. equal to cross sections for isotope L.
FXIZ(R,L,M)	:	Set R(1), R(2), etc. equal to volume fraction times density of isotope L in zone M.
FSELT(L)	:	Rewind and select data set L.
FTAPER(M,R,N)	:	Read from the data set previously selected, data relating to channel M. Set N(1) onwards equal to the zone numbers and R(1) onwards equal to flux in group 1 point 1, flux in group 1 point 2,....., flux in group 1 point JMAX, flux in group 2 point 1, flux in group 2 point 2, etc. Note: This process is most efficient if channels are taken in increasing order.
FTRA(R,S,L)	:	Transfer L numbers from vector R to vector S, i.e. S(1) = R(1), S(2) = R(2).... S(L) = R(L).
FVOL(R)	:	Set R(1), R(2), etc. equal to the volumes of regions 1, 2, etc. of the current channel.
FZERO(R,L)	:	Set L numbers in vector R to zero.
FDIV(R,L,S)	:	Divide L numbers in vector R by the corresponding L numbers in vector S. e.g. R(1) = R(1)/S(1) etc.

Note that all of the above routines are called by a normal FORTRAN CALL statement, e.g. CALL FTAPER(M,R,N).

Also available is the function LOC. For example, K = LOC(X(213)) causes K to be set to the value of the absolute byte address of the element X(213).

6. STORAGE ALLOCATION FOR 360 CRAM

CRAM is designed to make use of all core storage available for finite difference coefficients before using an F.D. coefficient data set as an extension of core. Data sets are used to pass fluxes between successive iterations.

The A.A.E.C. CRAM program has thus far been run only under the O/S Primary Control Program.

Blank COMMON can be defined so as to fill completely the available storage after space has been allocated for the program and I/O buffer areas. In the present version of the CRAM program (August 1967), any change in the amount of storage available for blank COMMON can be reflected only by reassembly of the following subroutines:

BLKCOM	:	the assembly language routine that defines the number of 4-byte words in COMMON
MAIN 1 SETUP	:	comments in the FORTRAN source program indicate constants that must be changed.

Present size is 40,000 single precision words.

It is hoped that a subsequent improvement to the program will provide automatic storage allocation by using the GETMAIN macro.

The program uses an overlay structure which is described in detail by comment cards that have been included in the FORTRAN source coding for the CRAM main program. The CRAM data sets and their D.D. names are also given as comments in the main program.

7. TEST PROBLEMS

The code package includes sets of test problems for both one and two dimensional geometries. These test most of the CRAM options and provide a useful guide to input data preparation.

8. REFERENCES

Hassitt, A. (1962). - A computer program to solve the multigroup diffusion equations. TRG-229(R).

Bennett, N. W. and Pollard, J. P. (1967). - SCAN - A free input routine for the IBM 360. AAEC/TM399.