DTIE

GEMP-644

RECEIVED BY DTIE NOV 27 1968

MASTER

RECEIVED BY DTIE NOV 6 1968

COMPUTER SIMULATION OF POINT DEFECT
ANNEALING IN METALS

October, 1967

Nuclear Systems Programs
Space Systems
GENERAL ELECTRIC
Missile and Space Division
Cincinnati, Ohio 45215

DISTRIBUTION

US Atomic Energy Commission

Washington Headquarters

J. M. Simmons
A. J. Van Echo

Oak Ridge Operations Office

C. L. Karl, CAO
J. F. Weissenberg, GE-NSP Site Rep.

Argonne National Laboratory

S. Harkness

Atomics International

H. Pearlman

Brookhaven National Laboratory

D. H. Gurinsky

DTIE

General Electric Company

APO - Sunnyvale, California

F. Comprelli

Naval Research Laboratory

L. E. Steele

North Carolina State University

J. R. Beeler (10)
R. E. Dahl

Oak Ridge National Laboratory

J. R. Weir
M. Wechsler

Pacific Northwest Laboratory

A. Bement
T. Bierlein
T. Claudson
A. DePino
J. Holmes
H. Kissinger
J. Ulseth
H. Yoshikawa

D. H. Ahmann
N. R. Baumgardt
D. G. Besco (10)
H. C. Brassfield
V. P. Calkins
C. G. Collins
P. K. Conn
J. B. Conway
W. E. Edwards

R. E. Fryxell
W. B. Henderson
J. L. Kamphouse
L. K. Keys
J. A. McGurty
J. Moteff (5)
J. F. Scherrer, Jr.
J. VanHoomissen, NTPO
J. F. White
Library (3)

LEGAL NOTICE

United States Atomic Energy Commission Contract No. AT (40-1)-2847

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, material, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, material, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission or his employment with such contractor.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

**COMPUTER SIMULATION OF POINT DEFECT
ANNEALING IN METALS**

David G. Besco

October, 1967

**General Electric Company
Nuclear Systems Programs
Cincinnati, Ohio 45215**

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

COMPUTER SIMULATION OF POINT DEFECT*
ANNEALING IN METALS

David G. Bescot†

General Electric Company
Nuclear Materials and Propulsion Operation
Cincinnati, Ohio 45215

ABSTRACT

A method is outlined for directly simulating on a digital computer the annealing of point defects produced by irradiation or quenching. The properties of a general annealing simulation model are defined, and a particular model is developed for the isothermal annealing of neutron-irradiated α -iron near room temperature, based on the results of Johnson, and of Erginsoy et al. A FORTRAN program which implements this model is described in detail.

*This paper originated from work sponsored by the Fuels and Materials Development Branch, U.S. Atomic Energy Commission, under Contract AT(40-1)-2847.

†Present address is General Electric Company, Computations Operation, Aircraft Engine Group, Cincinnati, Ohio 45215.

TABLE OF CONTENTS

	<u>PAGE</u>
I. INTRODUCTION.	6
II. GENERAL SIMULATION MODEL.	8
III. A PARTICULAR SIMULATION MODEL - DEFECT ANNEALING IN α - IRON AT 300° K.	13
IV. COMPUTER PROGRAM USAGE.	17
1. General Information	17
2. Input Data.	19
3. Error Messages.	22
4. Printed Output.	22
5. Punched Output.	24
6. Running Time.	24
7. Auxiliary Input-Output Files.	24
V. PROGRAMMING INFORMATION	25
1. Element Table	25
2. Subprograms	26
VI. REFERENCES.	31
VII. APPENDICES.	33
1. Flow Charts	33
2. Sample Input Data	36
3. Sample Printout	39
4. Random Number Subroutines	54
5. Source Program Listings	57

I. INTRODUCTION

During the last six years a program of computer experiments on the production and interaction of lattice defects has been in progress at GE-NMPO. The results of these investigations [1-6] have served to delineate the consequences of various assumptions concerning the nature of the radiation damage process, and the forces between atoms. A series of computer programs [7-9] has been developed to carry out these experiments.

One long-term objective of this research program has been to develop a method for the direct digital simulation of the annealing of radiation damage. This report describes such a method, and a computer program which implements it.

The traditional method of doing annealing calculations is by setting up and solving a system of chemical rate equations describing the interactions of different defect species (migration to sinks, clustering, dissociation). This method has been applied to a wide variety of problems by Damask and Dienes, among others, and they give an extensive review in their book [10]. When rate equations are employed, it is explicitly assumed that the distribution of defects is homogeneous; no information about their spatial deployment is either input or output. In the case of quenching experiments this assumption is thought to be appropriate, and the method is frequently used in aiding the interpretation of experimental data [10,11]. However, in the case of neutron irradiation no analysis of annealing kinetics can be truly complete without in some way accounting for the spatial distribution of defects in the damaged region.

Recently, insight has been gained into the primary damage state created in neutron-irradiated metals, both theoretically, by the computer experiment method [3,4], and experimentally, by the use of the field ion microscope [17]. These studies provide atomic-level detail regarding the nature of the damaged region. In the particular case of α -iron further fundamental data on the nature of individual point defects has become available [13, 14, 15]. This data has made possible the construction of a model for the direct simulation of annealing on a digital computer.

The initial conditions for simulation consist of the exact position in the crystal lattice of every vacancy and interstitial. Using straightforward Monte Carlo sampling techniques, the path of each point defect is traced through the lattice, jump by jump. The detailed changes of the defect distribution resulting from migration, annihilation and clustering may then be observed. Direct simulation was applied by Beeler [3] to short-range interstitial migration in α -iron. Subsequently, Beeler and Besco [6] used a more detailed simulation model, allowing for correlations and di-interstitial migration, in a study of the annealing of 5 keV displacement spikes in α -iron. That model is described in Section III of this report. The production of displacement spikes at 0°K is computed by program CASCADE, which also uses a direct simulation technique.

The scope of a simulation experiment is necessarily much more restricted than that of a rate equation calculation, because of the amount of atomic level detail involved. The largest system which can reasonably be simulated consists of a single displacement spike (several hundred defects) annealed for several hundred jump times. The value of the simulation lies in the detailed description of short-term changes in local damaged regions, and

in this connection should be useful in the design of further field-ion microscope experiments.

This report is organized in the following way. The requirements for a defect annealing simulation model are described in Section V in general mathematical terms. A particular simulation model for α - iron, based on existing theoretical and experimental data, is described in Section III. Computer program DAIQUIRI, (Defect Annealing in Quenched or Irradiated Specimens) is an implementation of this model. Detailed instructions for the use of DAIQUIRI are given in Section IV, and a discussion of the program itself in Section V. An appendix contains input and output for a sample case, along with source program listings.

The intent is to provide a complete definition of the mathematical model, complete instructions for running DAIQUIRI, and sufficient programming information to guide a user with knowledge of FORTRAN in implementing other simulation models within the DAIQUIRI framework.

II. GENERAL SIMULATION MODEL

Computer simulation has come into increasingly widespread use to study the behavior of systems whose state changes with time [12]. Simulation is frequently used in connection with systems for which direct experimentation on a prototype is impractical, direct mathematical analysis is inadequate, and experience and intuition are undependable. A solid containing point defects is such a system.

We assume the following:

The material is a monatomic single-grain solid whose atoms occupy the

equilibrium positions in an infinite lattice (body-centered cubic or face-centered cubic). The only imperfections in the solid are point defects (vacancies, interstitials, and impurities) and aggregates of point defects. An individual point defect of whatever type will be called an element of the system. To each element α corresponds a lattice vector $\bar{r}(\alpha)$ indicating its position and a type number $t(\alpha)$ indicating its nature ($t = 1, 2$, or 3 for a vacancy, interstitial, or impurity atom, respectively). The state of the system at a given time is defined by the set of position vectors and type numbers for all elements of the system. This set will be called S .

The set of all elements of a particular type may be further classified into subsets called clusters. By convention, each cluster contains one or more elements, and each element is a member of exactly one cluster. Every element α is assigned a class number $c(\alpha)$ which is equal to the number of elements in the cluster to which α belongs. (Examples: a trivacancy consists of three elements of type 1, class 3; a single interstitial consists of one element of type 2, class 1). The set of class numbers $\{c(\alpha)\}$ may be regarded as a function of S .

The annealing simulation is carried out according to a discrete-change model, using a Monte Carlo technique. A fundamental time-step τ is chosen, during which each position vector will transform (with some computable probability) into the position vector for an adjacent lattice site. Such a transformation is called a jump, and constitutes a change of S .

$\{c(\alpha)\}$ may or may not be affected by a jump. A jump may result in reduction of the total number of elements through mutual annihilation of

vacancies and interstitials. The successive transformation of S in discrete steps is meant to correspond to the isothermal annealing of a solid containing point defects produced by irradiation or by quenching. The initial condition S_0 for S corresponds to an assumed point defect configuration or "initial damage state." Initial damage states resulting from energetic primary knock on atoms produced by neutron irradiation have been calculated for several cubic metals using program CASCADE [8].

A wide variety of assumptions about the details of the annealing process may be accommodated within such a general framework. The computer program is organized in a modular fashion; each set of physical assumptions is defined in a specific sub-program, and these subprograms function independently of the parts of the program that are concerned with input-output, indexing, table modification, and similar chores. The intent is to permit investigation of the consequences of a variety of different annealing models. The amount of further programming involved for each model should be relatively small, primarily involving redefinition of the jump probability algorithm and certain logical functions.

The complete definition of an annealing model is divided into several parts, as enumerated below. An example of a particular annealing model and its implementation is given in Section III.

1. Clustering Criteria

A method is supplied for deciding whether two elements belong to the same cluster, based on $\bar{r}(\alpha)$, $\bar{r}(\beta)$, $t(\alpha)$, $t(\beta)$.

2. Annihilation Criterion

A method is supplied for deciding whether two elements will annihilate

(that is, whether a given Frenkel pair is unstable), based on $\bar{r}(\alpha)$, $\bar{r}(\beta)$. The annihilation criterion is in general temperature-dependent.

3. Correlation Criterion

An element α in a lattice of coordination number n , will jump in direction i with probability p_i , $i = 1, \dots, n$. If α is isolated, then $p_i = p_j$, $i, j = 1, \dots, n$. If a second element β is located so that its presence causes one or more of the p_i for α to be different than they would be for the isolated case, then β is said to be correlated with respect to α . The set of all $\bar{r}(\beta)$, $t(\beta)$, for β which are correlated with respect to α , given a system in state S , is called the environment of α and designated by $E(\alpha)$. Clearly, $E(\alpha)$ is a subset of S , and $E(\alpha)$ is null in the case that α is isolated (does not interact with other elements).

The correlation criterion then is an algorithm for determining $E(\alpha)$. On an element basis, it is simply a method for determining whether a given element β is correlated with respect to a given element α in state S . $c(\alpha)$ and $c(\beta)$ will ordinarily be important parameters in this determination.

4. Jump Probability Algorithm (JPA)

Finally, and most critically, an algorithm must be specified for determining the jump probabilities.

Given a fixed time interval τ , a system in state S , a jump candidate α , a proposed jump direction i , and environment $E(\alpha)$, the algorithm specifies the calculation of p_i .

It is the notions of correlation and environment which allow a wide degree of flexibility in specifying an annealing model. In the simplest possible model, $E(\alpha)$ would consist only of the elements in the same cluster as α , and the JPA would simply provide that all elements jump as if they were isolated, except for jump directions which might be prohibited by the existence of other elements of α 's cluster at sites neighboring α . In general, however, one can do better than that, based on the available information about the way in which point defects interact. The program permits an investigator to examine the consequences of whatever assumptions he cares to make about jump rates and the influence of the environment of defects on their migration characteristics, by supplying subprograms for the four items outlined above.

What follows next is a concise summary of the general simulation procedure, using the vocabulary and symbols defined in this section.

SUMMARY

Given an initial damage state S_0 , the simulation proceeds in discrete time intervals of length τ , during which S_0 is successively transformed by the jumps of individual elements. In a time interval, each element is selected exactly once for a possible jump, the order of selection being random. That is, elements are sampled at random from S without replacement. When an element has been selected, a potential jump direction is assigned, chosen at random from the available directions. A jump probability p_i is then assigned, based on τ , $c(\alpha)$, $t(\alpha)$, and $E(\alpha)$. The jump then occurs with probability p_i . The consequences of the jump (annihilation and clustering, if any) are recorded. This process continues until either S is null or a prescribed number of time intervals have been simulated.

III. A PARTICULAR SIMULATION MODEL - DEFECT ANNEALING IN α - IRON AT 300°K

The general ideas of Section II are illustrated in this section through a specific example. A model was developed for the annealing of α - iron at 300°K by the migration of single interstitials and di-interstitials, with impurity effects neglected. This model and its implementation within the general framework will be described in this section, which should serve by example as a guide for the implementation of other models.

The initial damage states (displacement spikes) which were of interest consisted of single vacancies, vacancy clusters of various sizes up to about 20, single interstitials, and a relatively small number of di-interstitials.

The rate of reaction for a single thermally activated process [13, p. 21] is proportional to $\exp(-E_m/kT)$, where E_m is the activation energy for migration (in eV) T is the temperature (in degrees Kelvin) and k is Boltzmann's constant ($k = 8.616 \times 10^{-5}$ eV/K°).

Johnson [14] has computed activation energies for the migration of simple defects in α - iron, as follows:

Single vacancy	$E_m = 0.68$ eV
Divacancy	$E_m = 0.66$ eV
Single interstitial	$E_m = 0.33$ eV
Di interstitial	$E_m = 0.18$ eV

Based on these results, the interstitial jump rate is faster than the vacancy jump rate by a factor of about 10^6 at 300°K. Vacancies may therefore be assumed to be immobile, since displacement spike annealing is clearly dominated by single interstitial and di-interstitial migration. Interstitial clusters of three or more are also taken to be immobile.

The predominant mobile defect present in the initial damage state is then the single interstitial, and it is reasonable to consider a time scale in terms of the mean time μ between single interstitial jumps. If τ is chosen to be a large fraction of μ , error results from allowing a maximum of one jump per time interval. On the other hand if τ is a small fraction of μ , computing efficiency is reduced. As a compromise between these two considerations, τ was taken equal to 0.1μ .

Di-interstitials jump faster than single interstitials by a factor of about 300 at 300°K . Di-interstitials are infrequent in S_0 , and their motion could reasonably be neglected were it not for the fact they will frequently form during annealing from two migrating singles. This is especially true because of the mutual attraction between singles.

A special procedure is therefore used for di-interstitial migration. When a di interstitial is selected, no random number is drawn for comparison with a jump probability. Instead, the di-interstitial is moved through a random sequence of 30 jumps (using Johnson's scheme for the individual jumps). Frequently the di will become a single (by annihilation of one member) or a tri (by encountering a single) within a few jumps. If this occurs before the 30th jump the sequence is ended. If not, the di is left in its position after 30 jumps, and will jump 30 more times at the next time interval.

The true kinetics would be better approximated by choosing τ equal to a fraction of the di-interstitial jump time, but such a procedure would be grossly inefficient for tracing single interstitials, which are the dominant defects.

The clustering criteria are chosen by appealing once again to the Johnson results. He finds that the binding energy between single vacancies is appreciable (greater than 0.05 eV) only for first neighbor and second neighbor separation. The second neighbor pair is the more tightly bound. The most stable di-interstitial consists of two split interstitials at nearest neighbor lattice sites. On this basis, vacancies are taken to be members of the same cluster at first or second neighbor separation, and interstitials at first neighbor separation only.

Single interstitials, however, are appreciably bound at distances greater than the nearest neighbor distance. This is accounted for in the model by assuming that two single interstitials α and β are correlated if and only if it is geometrically possible for them to become a di-interstitial in two jumps or less. Examination of the b.c.c. lattice shows that this can occur only when $d^2(\alpha, \beta) = [\bar{r}(\alpha) - \bar{r}(\beta)] \cdot [\bar{r}(\alpha) - \bar{r}(\beta)] = 4, 8, 11, 12, 19, \text{ or } 27$, in units of $(a/2)$, where a is the lattice constant. When $d^2(\alpha, \beta) = 27$, one must further require $[\bar{r}(\alpha) - \bar{r}(\beta)] = (\pm 3, \pm 3, \pm 3)$.

If β is correlated with α , then the effect of the correlation is to cause $p_i = 1.0$ for all jump directions i (for α) which would reduce $d^2(\alpha, \beta)$, and $p_i = 0.1$ for those which would increase it. A migrating di-interstitial is prohibited from making jumps which would increase its distance from a correlated single.

Erginsoy and Vineyard [15] calculated the region within which a Frenkel pair (vacancy-interstitial pair) is unstable in iron at absolute zero. Annihilation occurs at one, two, and three nearest-neighbor distances along close-packed lines, and at second neighbor separation. Since the details of this region for finite temperatures have not been calculated, the absolute zero

region is incorporated in the model.

This completes the definition of the annealing model. It is now restated symbolically using the symbols and terminology of Section II.

Let $n(\alpha, \beta)$ be the integer n such that $\bar{r}(\alpha)$ and $\bar{r}(\beta)$ are n th nearest neighbor lattice sites.

1. Clustering Criteria

α and β belong to the same cluster when:

$n(\alpha, \beta) = 1$ or 2 , if $t(\alpha) = t(\beta) = 1$

$n(\alpha, \beta) = 1$, if $t(\alpha) = t(\beta) = 2$

No clustering is defined when $t(\alpha) \neq t(\beta)$

2. Annihilation Criterion

α and β are an unstable Frenkel pair and will annihilate when:

$t(\alpha) \neq t(\beta)$ and

either $n(\alpha, \beta) = 1, 2$, or 5

or $\bar{r}(\alpha) - \bar{r}(\beta) = (\pm 3, \pm 3, \pm 3)$

3. Correlation Criteria

α and β are correlated when:

$t(\alpha) = t(\beta) = 2$ and

$n(\alpha, \beta) = 1, 2, 3, 4, 5$, or 7 , or

$\bar{r}(\alpha) - \bar{r}(\beta) = (\pm 3, \pm 3, \pm 3)$

If $t(\alpha) \neq t(\beta)$, correlation is accounted for by the annihilation criterion.

If $t(\alpha) = t(\beta) = 1$, correlation is irrelevant since type 1 elements are immobile.

4. Jump Probability Algorithm

Let $p_i(\alpha)$ be the probability for element α to jump in direction i .

If $t(\alpha) = 1$, then $p_i(\alpha) = 0$

If $t(\alpha) = 2$ and $c(\alpha) > 2$, then $p_i(\alpha) = 0$

If $t(\alpha) = 2$ and $c(\alpha) = 1$, and

(a) No elements are correlated with respect to α , then $p_i(\alpha) = 0.1$

(b) An element β is correlated with respect to α , then let $n_1 = n(\alpha, \beta)$ before a jump in direction i , and let $n_2 = n(\alpha, \beta)$ afterward. If $(n_1, n_2) = (2, 1), (3, 1), (5, 1), (4, 2), (4, 3), (7, 3), (7, 5)$ or $(10, 5)$,

then $p_i(\alpha) = 1.0$

Otherwise $p_i(\alpha) = 0.1$

If $t(\alpha) = 2$ and $c(\alpha) = 2$,

the pair (α, β) performs a random walk at the rate of 30 jumps per time interval according to the di-interstitial migration scheme of Johnson. According to this scheme, six jump vectors are available; the eight b.c.c. jump vectors $(\pm 1, \pm 1, \pm 1)$, excluding $\pm [\bar{r}(\alpha) - \bar{r}(\beta)]$.

IV. COMPUTER PROGRAM USAGE

1. General Information

The rest of this report describes the present status of a computer program which implements the particular model of Section III within the framework of the general model of Section II. The report is preliminary in the sense that the program will probably undergo considerable further evolution as other annealing models are developed and implemented. The program has been expressly

designed in order to permit this evolution. It is being documented at this stage of development in order to facilitate both application in its present form, and the programming of further extensions.

The symbolic name of the program is DAIQUIRI (Defect Annealing in Quenched or Irradiated Specimens). It is coded in FORTRAN IV with the exception of a random number subroutine coded in IBM 7090/94 assembly language (see appendix). The FORTRAN IV dialect conforms with ASA standard FORTRAN, with the exception of NAME LIST input. The program has been run to date only on the 7090/94, but conversion to other machines should be routine, requiring mainly the incorporation of an appropriate procedure for the generation of random numbers.

At present the program consists of a main program and 24 subprograms, comprising over 800 FORTRAN statements. When run under IBSYS, it occupies fewer than 24,000 locations, including system, library, and COMMON, and thus has considerable room left for growth within a 32K constraint.

Three input and output files are addressed in addition to the standard system I/O files (5 and 6).

BCD card images are written on NPUN (file 11) for off-line punching. The two "scratch" files are NAUX (file 1) and NAUX2 (file 2). There are no permanent data files to be readied prior to execution; all data is read from the system input.

The complete source program listings appear in an appendix. There is a glossary of symbols at the beginning of the listings.

2. Input Data

Any number of problems may be run in sequence. In each problem, a configuration of point defects (element table) is supplied as the initial conditions. The program simulates the annealing of this configuration according to the assumed model for as long a time period as desired.

The data for each problem is in four parts.

Part I - Problem Identification

Two cards containing arbitrary text in columns 1-72. The text is used as a printout heading. The cards must be included, even if blank.

Part II - NAMELIST Data

First card: ① \$ ANNEAL

Last card: ① \$

The NAMELIST items take on their initialized values (IV) until overridden by input data. There is no reinitialization between problems. The NAMELIST symbols, their meanings, initialized values if any, and rules for usage are given below:

NPROB - Problem number.

Each problem must have a unique number in order to properly identify the printed and punched output.

NIMIN - Initial time step.

For a new problem, NIMIN = 1

For a restart, NIMIN > 1

IV = 1

NIMAX - Final time step.

NPRI - Table listing interval.

Three values are entered:

NPRI = k_1 , k_2 , k_3

This causes the element table and cluster population to be listed every k_1 th time step up through step k_2 , and every k_3 th time step thereafter.

IV = 10, 500, 10

LAST - Last problem indicator.

The last problem in a sequence (even if it is also the first) must set LAST to a value of .TRUE. (LAST = T,). This causes the job terminating procedures to occur at the end of that problem.

BCC - Crystal structure.

BCC = .TRUE. selects the 300°K α - Fe model of Section III. BCC = .FALSE. selects a face-centered-cubic model which was implemented in order to checkout the f.c.c. "machinery" which will be required in the future. The latter model is trivial, consisting solely of jumping single interstitials with $p_i = 0.1$ and no correlations. All other defects are immobile.

IV = .TRUE.

LIST - Jump history.

LIST = .TRUE. signals that a complete jump history is to be tabulated and printed.

IV = .TRUE.

SAVE - Save input element table.

SAVE = TRUE. causes the input element table to be saved on NAUX2 for use

by a later problem in the same sequence. Restriction: only one table may be saved on NAUX2 at a time.

IV = .FALSE.

FETCH - Fetch input element table.

FETCH = .TRUE. signals that the input element table is to be read from NAUX2. This presupposes that it has been saved there by an earlier problem in the same sequence. When FETCH = .TRUE., part III of the data is omitted, and SAVE should have a value of .FALSE.
IV = .FALSE.

Part III - Element Table

Both the first and last cards of Part III have the number of elements punched in columns 6-10, and are blank elsewhere.

Between the first and last cards, there is one card for each element, with the following format:

Col. 1-5: element number.

These must run sequentially, starting with 1.

Col. 6-10, 11-15, 16-20: components of the position vector.

Col. 21-25: type number.

Col. 26-30: class number.

Col. 31-35: adjoin number (ADJ)

If the element is of class 1, ADJ = 0. Otherwise ADJ is the number of an element in the same cluster. The numbering must be "cyclical." For example, if I, J, and K constitute a cluster of 3, then ADJ (I) = J, ADJ (J) = K, ADJ (K) = I.

Restriction: the maximum number of elements is 500.

Part IV - Random Number Generator

Part IV consists of a single card, which contains a generator in the form of an octal integer punched in columns 1-12.

3. Error Messages

In addition to the checking done by NAMELIST, the program performs a number of further checks on the input data before beginning the simulation. If an error or inconsistency is found, a message is printed and execution terminated. The present version finds only one error per run. The meanings of the error codes follow:

1	The number of elements is outside the legal range 1 to 500.
2	$NTMAX \leq 0$, $NTMIN \leq 0$, or $NTMIN > NTMAX$
1000 + I	(where I is the number of the erroneous element) A vacancy has a position which is not a lattice site. Interstitials may legally have non-lattice positions, but if so they will be moved to lattice sites before simulation begins.
2000 + I	$CLASS \leq 0$
3000 + I	$CLASS > 1$ and $ADJ = 0$
4000 + I	$CLASS = 1$ and $ADJ = 0$
5000 + I	$CLASS > 50$
6000 + I	Position vector component ≤ 0
7000 + I	ADJ is different type.
8000 + I	ADJ numbering is non-cyclical.

4. Printed Output

(a) Title page.

The title page identifies the version of the program and briefly describes the annealing model.

(b) Listing of input parameters.

(c) Element table (input).

The table is listed, together with a count of the number of each kind of cluster, under the heading "Defect Population."

(d) Element table (initial conditions for simulation)

Same as (c), except now the interstitials have been moved to lattice sites where necessary.

(e) Element table (during simulation)

The table is listed at the end of the timesteps specified by NPRI, and at the end of the last timestep.

(f) Initially unstable pairs

If annihilations occurred as an immediate result of moving interstitials to lattice sites, such annihilations are counted and listed.

(g) Frenkel pairs annihilated

A listing of all annihilations which occurred during the simulation, and the time step for each.

(h) Annihilation count

T = time step

FPA (T) = number of annihilations during step T.

CUM = cumulative number of annihilations up to and including step T.

(i) Jump history

If LIST = .TRUE. a complete jump history is printed.

T = time step

J = jumping element

V = jump vector. The numbering of the jump vectors is given on the title page.

5. Punched Output

If NTMIN = 1, the initial condition element table is listed on BCD file NPUN for offline punching. The element cards have the word "START" punched in columns 71-75, and the problem number ending in column 80.

For every problem, the element table at the end of the simulation is punched similarly. The cards have "FINISH" in 71-76, and the problem number in 80.

Either of these sets of cards is acceptable input data for part III in a restart or subsequent problem. They are also acceptable to program DSMAP (NMPO Program No. 864) [16], which automatically draws pictures of displacement spikes using the CALCOMP plotter.

6. Running Time

Running time for a specific problem is difficult to estimate because it depends in a complicated way upon the size of the initial table and upon the rates of Frenkel pair annihilation and cluster formation. The time increases quite rapidly with the table size. Typical 7090 running times are eight minutes for the first 500 time steps with a 100-element table, and 15 minutes for the first 100 time steps with a 400 element table.

7. Auxiliary Input-Output Files

(a) NAUX

NAUX (file 1) is used to store the jump history during the simulation. NAUX output is buffered in logical records 750 words in length.

(b) NAUX2

NAUX2 (file 2) is used to store the input element table when the same table is to be used for a sequence of problems.

(c) NPUN

NPUN (file 11) is the BCD punch file. Data is accumulated on NPUN until the end of a problem sequence, at which time three EOF's are written.

When run under the Evendale IBSYS system, file 1 is tape A4, file 2 is tape A5, and file 11 is tape B2.

V. PROGRAMMING INFORMATION

1. Element Table

The program operates on a set of variables called the element table. This table is initialized by the input data, and the annealing simulation consists of repeatedly modifying it. The table completely defines the system at any point in time.

At most 500 elements are permitted, each of which has seven tabular entries. The program symbols and their meanings follow:

NORIG (I): the sequential number of the element in the input table. A particular element will not have the same sequential position in the table throughout the calculation if elements with a lower number are annihilated. Its original position number is retained in NORIG, however, for identification purposes.

For example, suppose elements 2 and 4 in a 6-element input table are later annihilated. It then becomes a 4-element table with:

NORIG (1) = 1

NORIG (2) = 3

NORIG (3) = 5

NORIG (4) = 6

R (I, J), J = 1, 2, 3: the X, Y, and Z coordinates of the element's position vector. The coordinates are in units of half lattice constants. By convention (0, 0, 0) is a lattice site. This implies that b.c.c. coordinates are either all odd or all even. F.c.c. coordinates have an even sum. On the assumption that the most stable interstitial is split symmetrically about a lattice site, interstitials are assigned lattice site coordinates. A single interstitial jump consists of transforming the coordinates from one lattice site to an adjacent one.

TYPE (I): 1 for vacancy, 2 for interstitial. Impurity atoms will be type 3 in a later version.

CLASS (I): an integer from 1 to 50, denoting the class number.

NBR (I): NBR is the program mnemonic for the adjoin number, referred to in the input description as ADJ.

All element table variables are declared to be of type INTEGER.

2. Subprograms

The purpose of each individual subprogram is described in the following, with references to Section IV of this report. The logical flow between subprograms is diagrammed in the accompanying flowcharts (see appendix).

INFO prints the title page (IV. 4.a)

INPUT reads the input data (IV. 2)

INPR provides a listing of the input parameters (IV. 4.b)

INCH checks the input data for errors (IV. 3)

PANIC is called whenever an error condition is encountered. Its argument list is of the form (N, P, Q), where N is an integer variable and P

and Q are logical variables. Their meanings are:

N = error code to be printed

P = .TRUE.; return to calling program

.FALSE.; terminate execution

Q = .TRUE., dump memory

.FALSE., do not dump memory

OCTIN processes input interstitials, if any, which have integer non-lattice coordinates. Such defects are permitted in the CASCADE computational model. They are called octahedral interstitials (b.c.c.) or body-centered interstitials (f.c.c.). OCTIN moves them at random to one of the nearest lattice sites.

CENSUS enumerates the defect population (IV. 4.c).

TPR provides a listing of the current element table and the defect population.

PRMUTE provides a randomly selected permutation of the integers from 1 to N, so that the elements may be selected for jump consideration in a random order.

DINTY is a special subroutine to handle di-interstitial migration in α - Fe at 300°K. Since di-interstitial migration involves the simultaneous jumping of two elements, it requires special procedures. According to Johnson [13], the most stable di-interstitial consists of two $\langle 110 \rangle$ split interstitials parallel to one another at nearest neighbor lattice sites, with their axes perpendicular to the line joining their centers. A di-interstitial "jump" is a two-step process during which the two split interstitials partially dissociate. The net result is that each element is translated through a b.c.c. jump vector. Six of the eight b.c.c. jump vectors are permitted, the two parallel to the line of centers being

excluded. (See Johnson's Figure 5). DINTY causes the di-interstitial to jump until either (a) it becomes bound to another interstitial or interstitial cluster, (b) one element annihilates a vacancy, or (c) 30 jumps have been completed. Binding between interstitials is accounted for by disallowing jumps which would increase the distance between the di-interstitial and its nearest correlated interstitial.

SJD selects a jump vector at random from the set of available jump vectors. A jump vector is "available" to an element if the lattice site resulting from translation through that vector is unoccupied. The maximum number of available jump vectors is the coordination number of the lattice (8 for b.c.c., 12 for f.c.c.).

JPR/B is the jump probability algorithm. In general, a different JPT/B is used for each annealing model.

JUMP translates the coordinates of a jumping element J through the pre-selected jump vector KDIR. It checks whether the jump results in annihilation, and calls UPDATE. If an annihilation occurs, it also calls CL/SE.

UPDATE makes the changes in the element table which are required by element J having jumped. The clustering criteria are applied to J, and to all elements which are in the same cluster as J, before and/or after the jump. The necessary changes to ADJ and CLASS are then made.

CL/SE is called after an annihilation. The annihilating elements are removed from the table and the table is closed up so as to remove vacant spots. By keeping the table as compact as possible, search time is minimized.

CLEAR clears the element table between problems.

LIST 1 records the time, element, and vector for every jump and stores them in a buffer of length 750 words.

LIST 2 transfers the contents of NAUX to NOUT after the completion of each problem, if a jump history (IV. 4.i) is requested.

CONNECT is a logical function defining the clustering criteria. CONNECT (J, K) is .TRUE. if elements J and K are correlated and .FALSE. otherwise.

DSQ is an integer function giving the distance squared between lattice sites.

FPSTAT prints statistics on Frenkel pair annihilation. The number of annihilations during each timestep is printed under the heading FPA (T), and the cumulative total under the heading CUM.

FRENKL defines the annihilation criterion. FRENKL (J, K) is .TRUE. if J and K are an unstable Frenkel pair, and .FALSE. otherwise.

RNUM is a FAP-coded subroutine which produces a floating-point random number drawn from a uniform distribution on the interval 0 to 1. Refer to the subroutine writeup in the appendix.

The program contains one region of labeled COMMON. The label of the region is BLOCK, and it contains a single three-dimensional array called MOVE. MOVE (I, J, K) is the Jth component of jump vector number I, in crystal K, where K is 1 for b.c.c. and 2 for f.c.c. MOVE is defined by a BLOCK DATA subprogram. It is defined as a one-dimensional array of length 72, rather than in the straightforward way as a three-dimensional

array dimensioned $12 \times 3 \times 2$. The latter was attempted, but resulted in an incorrect compilation due to an error existing in the IBM 7094 FORTRAN IV compiler at that time.

VI. REFERENCES

1. J. R. Beeler, Jr., and D. G. Besco, "Range and Damage Effects of Tunnel Trajectories in a Wurtzite Structure," J. Appl. Phys. 34, 2873 (1963).
2. J. R. Beeler, Jr., and D. G. Besco, "Effects of Channeling on Damage Production in α - Iron," Phys. Rev. 134, A530 (1964).
3. J. R. Beeler, Jr., "Vacancy and Interstitial Production in Neutron-Irradiated α - Iron," J. Appl. Phys. 37, 3000 (1966).
4. J. R. Beeler, Jr., "Displacement Spikes in Cubic Metals," Phys. Rev. 150, 470 (1966).
5. J. R. Beeler, Jr. and R. A. Johnson, "Vacancy Clusters in α - Iron," Phys. Rev. 156, 677 (1967).
6. J. R. Beeler, Jr., and D. G. Besco, "Annealing of 5-keV Displacement Spikes in α - Iron," Bull. Am. Phys. Soc. 12, 391 (1967).
7. D. G. Besco and J. R. Beeler, Jr., "Computer Programs Describing Collision Cascades in Binary Materials."
I - Square Planar Lattice GEMP-192 (1963)
II - Wurtzite Structure GEMP-200 (1963)
III - Body-centered Cubic and Face-centered Cubic Structures GEMP-243
(1963)
8. D. G. Besco and N. R. Baumgardt, "CASCADE and CLUSTER - Computer Programs to Simulate Radiation Damage Processes in Metals and Analyze the Distribution of Defects," GEMP-356 (1965).

9. N. R. Baumgardt, DEFECT program, report to be issued (1967).
10. A. C. Damask and G. J. Dienes, Point Defects in Metals, Gordon and Breach (1963).
11. J. R. Beeler Jr., and D. G. Besco, "Interactions of Lattice Defects in Alloys," GE-NMP 67-523 (1967).
12. D. Teichroew and J. F. Lubin, "Computer Simulation - Discussion of the Technique and Comparison of Languages," Comm. Association for Computing Machinery 9, 723 (1966).
13. J. Burke, The Kinetics of Phase Transformations in Metals. Pergamon (1965).
14. R. A. Johnson, "Interstitials and Vacancies in α - Iron," Phys. Rev. 134, A1329 (1964).
15. C. Erginsoy, G. H. Vineyard, and A. Englert, "Dynamics of Radiation Damage in a Body-Centered Cubic Lattice," Phys. Rev. 133, A1329 (1964).
16. D. G. Besco, "Automatic Plotting of Displacement Spikes," - GE report to be issued.
17. M. Attardo and J. M. Galligan, "A Field Ion Microscope Study of Neutron Irradiated Tungsten," Phys. Stat. Sol. 16, 449 (1966).

VII. APPENDIX 1 - FLOW CHARTS

APPENDIX 2 - SAMPLE INPUT DATA

\$DATA
 2.5 KEV FE %G11□
 SAMPLE RUN
 \$ANNEAL
 NPROB#999,
 NTMAX#20,NPRI#10,20,10,
 LAST#T,
 \$

-37-

	63					
1	105	105	91	1	1	0
2	113	101	101	1	1	0
3	107	103	109	1	1	0
4	108	112	92	1	1	0
5	114	110	94	1	1	0
6	114	110	100	1	1	0
7	116	100	102	1	1	0
8	110	104	102	1	1	0
9	108	106	108	1	1	0
10	110	106	110	1	1	0
11	106	98	112	1	1	0
12	106	94	114	1	1	0
13	109	109	93	1	2	14
14	110	108	94	1	2	13
15	103	101	101	1	2	16
16	104	102	102	1	2	15
17	109	105	105	1	2	18
18	108	104	106	1	2	17
19	109	103	99	1	3	20
20	111	103	99	1	3	21
21	108	102	100	1	3	19
22	105	103	105	1	3	23
23	107	101	107	1	3	24
24	106	102	106	1	3	22
25	116	106	100	1	3	26
26	118	106	100	1	3	27
27	118	108	100	1	3	25
28	111	103	95	1	4	29

29	109	105	95	1	4	30
30	112	102	94	1	4	31
31	110	104	96	1	4	28
32	102	105	86	2	1	0
33	102	105	90	2	1	0
34	111	100	92	2	1	0
35	110	113	94	2	1	0
36	115	114	96	2	1	0
37	118	103	98	2	1	0
38	109	112	98	2	1	0
39	110	98	101	2	1	0
40	122	110	101	2	1	0
41	107	107	102	2	1	0
42	119	102	104	2	1	0
43	110	101	106	2	1	0
44	107	102	112	2	1	0
45	104	92	113	2	1	0
46	114	108	113	2	1	0
47	110	105	89	2	1	0
48	109	112	89	2	1	0
49	115	102	95	2	1	0
50	118	113	95	2	1	0
51	113	105	98	2	1	0
52	111	100	99	2	1	0
53	116	113	99	2	1	0
54	119	101	100	2	1	0
55	111	109	100	2	1	0
56	110	107	101	2	1	0
57	118	111	101	2	1	0
58	106	106	104	2	1	0
59	117	100	105	2	1	0
60	105	109	108	2	1	0
61	108	98	115	2	1	0
62	106	95	117	2	1	0
63	113	106	119	2	1	0

-38-

63

114417737400 A

APPENDIX 3 - SAMPLE PRINTOUT

PROGRAM DAIQUIRI-

SIMULATION OF DEFECT ANNEALING IN QUENCHED OR IRRADIATED SPECIMENS

NMPO PROGRAM NO. 859

VERSION FOR PURE ALPHA-IRON (300 DEGREES KELVIN)

FCC GEOMETRY IS ALLOWED, BUT FCC ANNEALING MODEL
IS A TRIVIAL ONE FOR CHECKOUT PURPOSES ONLY

TYPE 1 ELEMENTS ARE VACANCIES
TYPE 2 ELEMENTS ARE INTERSTITIALS

ASSUMPTIONS ON JUMP PROBABILITIES FOLLOW

VACANCIES	P= 0.0	(EA=0.68 EV)
SINGLE INTERSTITIAL (UNCORRELATED JUMP)	P= 0.1	(EA=0.33 EV)
SINGLE INTERSTITIAL (CORRELATED JUMP)	P= 1.0	(EA=0.27 EV)
DI-INTERSTITIAL	30 JUMPS/TIME STEP	(EA=0.18 EV)
LARGER INTERSTITIAL CLUSTERS	P= 0.0	

SINGLE INTERSTITIALS ARE CORRELATED FOR NEIGHBORS 1,2,3,4,5,7, AND 10 (333 ONLY)

THE LAST TWO JUMPS IN DI-INTERSTITIAL FORMATION ARE CORRELATED. THESE ARE OF THE FORM
(2,1) (3,1) (5,1) (4,2) (4,3) (7,3) (7,5) (10,5)

INPUT/OUTPUT FILES

SYSTEM INPUT	NIN = 5
SYSTEM OUTPUT	NOUT= 6
FIRST AUXILIARY	NAUX= 1
SECOND AUXILIARY	NAUX2= 2
BCD PUNCHED OUTPUT	NPUN= 11

BCC JUMP VECTORS

1	(+1,+1,+1)
2	(+1,+1,-1)
3	(+1,-1,+1)
4	(+1,-1,-1)
5	(-1,+1,+1)
6	(-1,+1,-1)
7	(-1,-1,+1)
8	(-1,-1,-1)

LISTING OF INPUT PARAMETERS

CRYSTAL STRUCTURE	BCC	=	T
PROBLEM NUMBER	NPROB	=	999
FIRST TIME STEP	NTMIN	=	1
LAST TIME STEP	NTMAX	=	20
NUMBER OF ELEMENTS	NE	=	63
TABLE LISTING PARAMETERS	NPR	=	10 20 10
JUMP HISTORY REQUEST	LIST	=	T
CHECKOUT MODE	CHECK	=	1
RANDOM NUMBER GENERATOR	G	=	114417737400

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

ELEMENT TABLES (INPUT DATA)

I	J	POSITION			TYPE	CLASS	ADJ
1	1	105	105	91	1	1	0
2	2	113	101	101	1	1	0
3	3	107	103	109	1	1	0
4	4	108	112	92	1	1	0
5	5	114	110	94	1	1	0
6	6	114	110	100	1	1	0
7	7	116	100	102	1	1	0
8	8	110	104	102	1	1	0
9	9	108	106	108	1	1	0
10	10	110	106	110	1	1	0
11	11	106	98	112	1	1	0
12	12	106	94	114	1	1	0
13	13	109	109	93	1	2	14
14	14	110	108	94	1	2	13
15	15	103	101	101	1	2	16
16	16	104	102	102	1	2	15
17	17	109	105	105	1	2	18
18	18	108	104	106	1	2	17
19	19	109	103	99	1	3	20
20	20	111	103	99	1	3	21
21	21	108	102	100	1	3	19
22	22	105	103	105	1	3	23
23	23	107	101	107	1	3	24
24	24	106	102	106	1	3	22
25	25	116	106	100	1	3	26
26	26	118	106	100	1	3	27
27	27	118	108	100	1	3	25
28	28	111	103	95	1	4	29
29	29	109	105	95	1	4	30
30	30	112	102	94	1	4	31
31	31	110	104	96	1	4	28
32	32	102	105	86	2	1	0
33	33	102	105	90	2	1	0
34	34	111	100	92	2	1	0
35	35	110	113	94	2	1	0
36	36	115	114	96	2	1	0
37	37	118	103	98	2	1	0
38	38	109	112	98	2	1	0
39	39	110	98	101	2	1	0
40	40	122	110	101	2	1	0
41	41	107	107	102	2	1	0
42	42	119	102	104	2	1	0
43	43	110	101	106	2	1	0
44	44	107	102	112	2	1	0
45	45	104	92	113	2	1	0
46	46	114	108	113	2	1	0
47	47	110	105	89	2	1	0
48	48	109	112	89	2	1	0
49	49	115	102	95	2	1	0
50	50	118	113	95	2	1	0

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

ELEMENT TABLES (INPUT DATA)

I	J	POSITION			TYPE	CLASS	ADJ
51	51	113	105	98	2	1	0
52	52	111	100	99	2	1	0
53	53	116	113	99	2	1	0
54	54	119	101	100	2	1	0
55	55	111	109	100	2	1	0
56	56	110	107	101	2	1	0
57	57	118	111	101	2	1	0
58	58	106	106	104	2	1	0
59	59	117	100	105	2	1	0
60	60	105	109	108	2	1	0
61	61	108	98	115	2	1	0
62	62	106	95	117	2	1	0
63	63	113	106	119	2	1	0

DEFECT POPULATION

TYPE CLASS NUMBER

1	1	12
1	2	3
1	3	3
1	4	1
2	1	32

31 TYPE 1

32 TYPE 2

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

ELEMENT TABLES (INITIAL CONDITIONS FOR SIMULATION)

I	J	POSITION			TYPE	CLASS	ADJ
1	1	105	105	91	1	1	0
2	5	114	110	94	1	1	0
3	6	114	110	100	1	1	0
4	8	110	104	102	1	1	0
5	9	108	106	108	1	1	0
6	10	110	106	110	1	1	0
7	11	106	98	112	1	1	0
8	12	106	94	114	1	1	0
9	13	109	109	93	1	1	0
10	15	103	101	101	1	2	11

11	16	104	102	102	1	2	10
12	17	109	105	105	1	2	13
13	18	108	104	106	1	2	12
14	19	109	103	99	1	3	15
15	20	111	103	99	1	3	16
16	21	108	102	100	1	3	14
17	22	105	103	105	1	3	18
18	23	107	101	107	1	3	19
19	24	106	102	106	1	3	17
20	26	118	106	100	1	2	21
21	27	118	108	100	1	2	20
22	28	111	103	95	1	4	23
23	29	109	105	95	1	4	24
24	30	112	102	94	1	4	25
25	31	110	104	96	1	4	22
26	32	102	104	86	2	1	0
27	33	102	106	90	2	1	0
28	34	112	100	92	2	1	0
29	36	116	114	96	2	2	40
30	38	108	112	98	2	1	0
31	39	110	98	100	2	1	0
32	40	122	110	100	2	1	0
33	41	107	107	101	2	1	0
34	44	108	102	112	2	1	0
35	45	104	92	114	2	1	0
36	46	114	108	114	2	1	0
37	47	109	105	89	2	1	0
38	48	109	113	89	2	1	0
39	49	115	101	95	2	1	0
40	50	117	113	95	2	2	29
41	53	117	113	99	2	1	0
42	54	119	101	99	2	1	0
43	55	111	109	99	2	1	0
44	56	111	107	101	2	1	0
45	57	119	111	101	2	1	0
46	58	106	106	104	2	1	0
47	59	117	101	105	2	1	0
48	60	105	109	109	2	1	0
49	61	108	98	116	2	1	0
50	62	105	95	117	2	1	0

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

ELEMENT TABLES (INITIAL CONDITIONS FOR SIMULATION)

I	J	POSITION			TYPE	CLASS	ADJ
51	63	113	105	119	2	1	0

DEFECT POPULATION

TYPE CLASS NUMBER

1	1	9
1	2	3
1	3	2
1	4	1
2	1	24
2	2	1

25 TYPE 1

26 TYPE 2

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

TIME STEP 10

I	J	POSITION			TYPE	CLASS	ADJ
1	1	105	105	91	1	1	0
2	6	114	110	100	1	1	0
3	9	108	106	108	1	1	0
4	11	106	98	112	1	1	0
5	13	109	109	93	1	1	0
6	15	103	101	101	1	2	7
7	16	104	102	102	1	2	6
8	17	109	105	105	1	2	9
9	18	108	104	106	1	2	8
10	19	109	103	99	1	3	11
11	20	111	103	99	1	3	12
12	21	108	102	100	1	3	10
13	23	107	101	107	1	2	14
14	24	106	102	106	1	2	13
15	26	118	106	100	1	2	16
16	27	118	108	100	1	2	15
17	28	111	103	95	1	4	18
18	29	109	105	95	1	4	19
19	30	112	102	94	1	4	20
20	31	110	104	96	1	4	17
21	32	104	104	84	2	1	0
22	33	100	106	90	2	1	0
23	34	112	100	92	2	1	0
24	38	108	112	98	2	3	36
25	39	110	98	100	2	1	0
26	40	122	110	100	2	4	32
27	45	104	92	114	2	1	0
28	46	110	108	118	2	1	0
29	47	109	105	89	2	1	0
30	48	109	113	89	2	1	0
31	49	116	102	94	2	1	0
32	50	119	111	97	2	4	33

33	53	120	112	98	2	4	37
34	54	120	102	100	2	1	0
35	55	109	111	99	2	3	24
36	56	110	110	100	2	3	35
37	57	121	111	99	2	4	26
38	59	118	100	104	2	1	0
39	60	104	106	108	2	1	0
40	61	107	97	115	2	1	0
41	63	112	106	118	2	1	0

DEFECT POPULATION

TYPE CLASS NUMBER

1	1	5
1	2	4
1	3	1
1	4	1
2	1	14
2	3	1
2	4	1

20 TYPE 1

21 TYPE 2

2.5 KEV FE (G11)

SAMPLE RUN

PROBLEM NO. 999

TIME STEP 20

I	J	POSITION			TYPE	CLASS	ADJ
1	1	105	105	91	1	1	0
2	6	114	110	100	1	1	0
3	9	108	106	108	1	1	0
4	11	106	98	112	1	1	0
5	13	109	109	93	1	1	0
6	15	103	101	101	1	2	7
7	16	104	102	102	1	2	6
8	17	109	105	105	1	2	9
9	18	108	104	106	1	2	8
10	19	109	103	99	1	3	11
11	20	111	103	99	1	3	12
12	21	108	102	100	1	3	10
13	23	107	101	107	1	1	0
14	26	118	106	100	1	2	15
15	27	118	108	100	1	2	14
16	28	111	103	95	1	4	17
17	29	109	105	95	1	4	18
18	30	112	102	94	1	4	19
19	31	110	104	96	1	4	16
20	32	102	104	86	2	1	0
21	33	99	107	89	2	1	0
22	34	112	100	92	2	1	0
23	38	108	112	98	2	3	35

24	39	110	98	100	2	1	0
25	40	122	110	100	2	4	31
26	45	104	92	114	2	1	0
27	46	105	97	117	2	3	38
28	47	110	104	88	2	1	0
29	48	111	111	89	2	1	0
30	49	116	104	92	2	1	0
31	50	119	111	97	2	4	32
32	53	120	112	98	2	4	36
33	54	122	100	102	2	1	0
34	55	109	111	99	2	3	23
35	56	110	110	100	2	3	34
36	57	121	111	99	2	4	25
37	59	118	100	104	2	1	0
38	61	107	97	115	2	3	39
39	63	106	96	116	2	3	27

DEFECT POPULATION

TYPE CLASS NUMBER

1	1	6
1	2	3
1	3	1
1	4	1
2	1	10
2	3	2
2	4	1

19 TYPE 1

20 TYPE 2

6 INITIALLY UNSTABLE PAIRS

35	4
37	25
42	7
43	3
51	14
52	2

FRENKEL PAIRS ANNIHILATED

TIME	PAIR	
1	62	12
1	36	5
2	58	22
2	41	8
4	44	10
15	60	24

T	FPA(T)	CUM
1	2	2
2	2	4
4	1	5
15	1	6

JUMP HISTORY

T=TIME
J=JUMPING ELEMENT
V=JUMP VECTOR

T	J	V	T	J	V	T	J	V	T	J	V	T	J	V
1	53	3	1	62	4	1	36	3	1	50	3	1	46	5
1	55	5	1	32	2	1	34	3	1	57	2	1	53	2
1	57	4	1	53	4	1	49	2	1	56	6	2	63	6
2	58	4	2	41	4	2	55	8	2	34	6	2	56	5
2	55	5	2	56	2	2	55	2	2	54	1	2	32	4
3	33	5	4	44	5	4	33	8	5	46	7	5	60	8
7	60	8	7	46	5	9	50	3	9	46	7	10	59	4
10	60	3	10	61	8	11	49	2	11	33	8	13	63	1
13	59	8	13	32	5	14	32	7	14	46	3	14	33	1
14	39	5	14	63	8	15	63	7	15	46	7	15	63	7
15	46	7	15	63	8	15	46	8	15	63	7	15	46	7
15	63	1	15	46	1	15	63	2	15	46	2	15	63	8
15	46	8	15	63	8	15	46	8	15	63	7	15	46	7
15	63	1	15	46	1	15	63	7	15	46	7	15	63	8
15	46	8	15	63	3	15	46	3	15	63	3	15	46	3
15	63	2	15	46	2	15	63	1	15	46	1	15	63	2
15	46	2	15	63	2	15	46	2	15	63	2	15	46	2
15	63	7	15	46	7	15	63	6	15	46	6	15	63	2
15	46	2	15	63	6	15	46	6	15	63	3	15	46	3
15	63	7	15	46	7	15	63	7	15	46	7	15	63	3
15	46	3	15	63	3	15	46	3	15	63	7	15	46	7
15	63	8	15	46	8	15	60	7	15	48	4	15	39	4
16	46	3	16	63	3	16	46	3	16	63	3	16	46	3
16	63	3	16	46	8	16	63	8	16	46	6	16	63	6
16	46	2	16	63	2	16	46	8	16	63	8	16	46	3
16	63	3	16	46	6	16	63	6	16	46	7	16	63	7
16	46	1	16	63	1	16	46	1	16	63	1	16	46	8
16	63	8	16	46	8	16	63	8	16	46	8	16	63	8
16	46	8	16	63	8	16	46	7	16	63	7	16	46	8
16	63	8	16	46	1	16	63	1	16	46	6	16	63	6
16	46	6	16	63	6	16	46	6	16	63	6	16	46	2
16	63	2	16	46	1	16	63	1	16	46	8	16	63	8
16	46	2	16	63	2	18	54	3	18	59	1	19	48	3
19	54	3	20	47	4	20	49	6	20	33	6			

APPENDIX 4 - RANDOM NUMBER SUBROUTINE

SUBROUTINE

IDENTIFICATION:

Title: Random Number Subroutine

Source Language: MAP

Application: FORTRAN IV - IBM 7090/94

Author: David G. Besco

Component: NMPO

Installation: General Electric-Evendale

Date: January 12, 1965

PURPOSE:

To supply a floating point random number X , $0 < X < 1$, from a uniform distribution.

USAGE:

Before any random numbers are produced, the subroutine must be initialized by the statement:

CALL RSTART (G, N)

G is an initial generator, or starting number, which is arbitrary. Different generators produce different (and independent) sequences of random numbers.

N is the number of bits desired in the random numbers (before conversion to floating point). If N is greater than 33, 33 is used.

Subsequent to initialization, the statement

CALL RNUM (X)

causes the real variable X to be set equal to a random number, $0 < X < 1$, chosen from a uniform distribution.

METHOD:

The routine produces N-bit fixed point random numbers, which are then converted to floating point numbers between 0 and 1.

The fixed point numbers are computed from the recursion formula

$$R_{i+1} = F(M[R_i + 2^{-(N+1)}]) - 2^{-(N+1)}$$

where $F(x)$ is the fractional part of x , M is an integer $N + 3$ bits in length of the form $8k + 5$, and $R_0 = G$.

This procedure will produce all possible N-bit numbers before repeating, that is,

$$R_i + 2^n = R_i$$

TIMING:

On the 7094, a random number is generated in 23 machine cycles (46 microseconds).

STORAGE REQUIREMENT:

42_{10} locations

HISTORY:

The method used is taken from a 704 subroutine by D.L. Shell of G.E. (1956). The present version is the latest of several adaptations of that subroutine to various coding systems and special purposes.

APPENDIX 5 - SOURCE PROGRAM LISTINGS

\$IBFTC DAIQUI NOLIST,NOREF,DECK,M90,XR3,DD

CDAIQUIRI

C

C DEFECT ANNEALING IN QUENCHED OR IRRADIATED SPECIMENS

C

C DAVID G. BESCO

C GENERAL ELECTRIC -- NMPO

C DECEMBER,1966

C NMPO PROGRAM NO. 859

C

C CODED IN ASA STANDARD FORTRAN EXCEPT FOR NAMELIST INPUT

C

C BCC AND FCC

C

C GLOSSARY OF SYMBOLS

C

C SYMBOL MEANING

C

C AVDIR(I) AVAILABLE JUMP VECTORS

C BCC TRUE WHEN LATTICE IS BCC

C BUFF(I,K) JUMP HISTORY BUFFER

C CH(I) LIST OF ELEMENTS WHOSE CLASS NUMBER MAY NEED TO BE
C UPDATED

C CHECK MODE OF OPERATION

C CHT(I) TENTATIVE LIST FOR CLASS NUMBER CHECKING

C CLASS(J) CLASS OF ELEMENT J(SIZE OF CLUSTER TO WHICH IT BELONGS)

C CLK CLASS OF ELEMENT K

C DRAWN(I) FALSE WHEN ELEMENT I NEEDS TO BE CHECKED FOR
C CORRELATION. TRUE OTHERWISE

C FCC TRUE WHEN LATTICE IS FCC

C FETCH FETCH TABLE FROM NAUX2

C FPA(I,K) ITH FRENKEL PAIR TO BE ANNIHILATED

C K=1 IS TIME STEP, K=2,3 ARE ELEMENT NUMBERS(ORIGINAL)

C G RANDOM NUMBER GENERATOR

C GR(I) ELEMENTS GROUPED IN CLUSTER

C ID1(I) FIRST LINE OF IDENTIFICATION AND PAGE HEADING

C ID2(I) SECOND LINE OF IDENTIFICATION AND PAGE HEADING

C IDC(I,K) LIST OF ELEMENTS OF TYPE K WHICH ARE CORRELATED WITH
C ELEMENT J

C IDR(I) STORAGE FOR PERMUTED LIST OF FIRST N INTEGERS

C J ELEMENT CURRENTLY BEING EXAMINED

C JCH(I) LIST OF ELEMENTS TO JUMP (CHECKOUT MODE)

C JSITE(I) POSITION VECTOR OF JUMP SITE

C JSJ,JSK DI-INTERSTITIAL JUMP SITES

C JT TYPE TRIGGER

C KCR 1 FOR BCC, 2 FOR FCC

C KDCH(I) LIST OF JUMP DIRECTIONS (CHECKOUT MODE)

C KDIR SELECTED JUMP VECTOR

C KODE ERROR CODE

C LAST LAST PROBLEM IN RUN

C LIST JUMP HISTORY REQUEST

C LISTA(I) DISTANCE SQUARED FOR CORRELATED INTERSTITIALS

C LISTB(I) CODE NUMBERS FOR CORRELATED INTERSTITIAL JUMPS

C M250 NUMBER OF JUMPS (MOD 250)

C MCL(I) MAXIMUM CLASS NUMBER FOR ELEMENTS OF TYPE I

C MOVE(I,K,KCR) JUMP VECTOR FOR DIRECTION I IN LATTICE KCR

C MOVSC(I,K) JUMP VECTOR FOR DIRECTION I IN SIMPLE CUBIC LATTICE

C	N(I)	DISTANCE SQUARED FOR UNSTABLE FRENKEL PAIRS
C	NAUX	FIRST AUXILIARY FILE
C	NAUX2	SECOND AUXILIARY FILE
C	NAVDIR	NUMBER OF AVAILABLE JUMP VECTORS
C	NBR(J)	ELEMENT ADJOINING J
C	NC(I)	NUMBER OF ELEMENTS OF TYPE I CORRELATED WITH J
C	NCD	COORDINATION NUMBER
C	NCT(I,K)	NUMBER OF ELEMENTS OF CLASS I AND TYPE K
C	NE	TOTAL NUMBER OF ELEMENTS
C	NEL	NE AT THE BEGINNING OF CURRENT TIME STEP
C	NFPA	NUMBER OF FRENKEL PAIRS ANNIHILATED
C	NIN	SYSTEM INPUT FILE
C	NJCH	NUMBER OF ELEMENTS TO JUMP (CHECKOUT MODE)
C	NJM	NUMBER OF JUMPS
C	NORIG(I)	ORIGINAL NUMBER OF ELEMENT I
C	NOUT	SYSTEM OUTPUT FILE
C	NPR	TABLES ARE LISTED EVERY NPR TIME INTERVALS
C	NPROB	PROBLEM NUMBER
C	NPUN	BCD PUNCH FILE
C	NREC	NUMBER OF RECORDS ON NAUX
C	NTIME	CURRENT TIME STEP
C	NTMIN	FIRST TIME STEP
C	NTMAX	LAST TIME STEP
C	NZA	NUMBER OF INITIALLY UNSTABLE FRENKEL PAIRS
C	P	JUMP PROBABILITY
C	PAR(J)	PARITY VECTOR
C	R(J,K)	POSITION VECTOR OF ELEMENT J
C	RCD	COORDINATION NUMBER
C	RN	RANDOM NUMBER FROM UNIFORM DISTRIBUTION
C	SAVE	SAVE TABLE ON NAUX2
C	TYPE(J)	TYPE OF ELEMENT J -- 1 FOR VACANCY, 2 FOR INTERSTITIAL

```

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC,NPRI(3)

```

```

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

```

```

LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC

```

```

LOGICAL CORL
NPUN=11
REWIND NPUN
NPRI(1)=10
NPRI(2)=500
NPRI(3)=10
NTMIN=1
LAST=.FALSE.
LIST=.TRUE.
SAVE=.FALSE.
FETCH=.FALSE.
BCC=.TRUE.
CHECK=1
CALL INFO

```

```

1 CALL INPUT
  DO 190 I=1,NE
190 NORIG(I)=I
  FCC=.NOT.BCC
  CALL INPR
  CALL INCH
C
C   INITIALIZE RANDOM NUMBER SUBROUTINE
C
  CALL RSTART(G,33)
C

  NTIME=0
  NJM=0
  NPR=NPRI(1)
  IF (BCC) GO TO 340
  NCD=12
  KCR=2
  GO TO 350
340 NCD=8
  KCR=1
350 RCD=NCD
  IF (LIST) REWIND NAUX
  NUMCH=0
  NFPA=0
  IF (NTMIN.GT.1) GO TO 330
  CALL OCTIN
C
C   PRINT INPUT DATA
C
  CALL CENSUS
  CALL TPR
  WRITE (NPUN,1004) NPROB,NE
  WRITE (NPUN,1004) (I,(R(I,J),J=1,3),TYPE(I),CLASS(I),NBR(I),NPROB,
1 I=1,NE)
  WRITE (NPUN,1004) NPROB,NE
330 NTIME=NTMIN
C
C   REORDER THE INTEGERS FROM 1 TO N AT RANDOM
C   IN THE ARRAY IDR
C
120 CALL PRMUTE(NE,IDR)
C
C   LOOP 10 IS EXECUTED ONCE PER TIME INTERVAL.
C   EACH ELEMENT IS GIVEN A CHANCE TO MOVE.
C   NE IS THE NUMBER OF ELEMENTS.
C
310 NEL=NE
  DO 10 I=1,NEL
  IF (CHECK.LT.3) GO TO 160
  NUMCH=NUMCH+1
  IF (NUMCH.GT.NJCH) GO TO 170
  J=JCH(NUMCH)
  GO TO 70

```

```

C
C   IDR(I) = 0 IS SIGNAL TO SKIP
C   IDR(I) CAN BE ZERO FOR ONE OF TWO REASONS
C   THE ELEMENT WHOSE NUMBER WAS STORED IN IDR(I) WAS ANNIHILATED
C   EARLIER IN THE TIMESTEP, OR ELSE IS THE SECOND MEMBER OF A
C   DI-INTERSTITIAL WHICH MOVED EARLIER IN THE TIMESTEP
C
160 IF (IDR(I).EQ.0) GO TO 10
    J=IDR(I)
    IDR(I)=0
C
C   IN THIS VERSION, ONLY SINGLE INTERSTITIALS AND DI-INTERSTITIALS
C   ARE MOBILE
C
    IF ((TYPE(J).EQ.1).OR.(CLASS(J).GT.2).OR.(FCC.AND.(CLASS(J).EQ.2)))
1 ) GO TO 10
C
C   IS J PART OF A DI-INTERSTITIAL
C
    IF (CLASS(J).EQ.1) GO TO 70
C
C   YES, DON-T DRAW ITS PARTNER AT THIS TIME STEP
C
    DO 260 N=1,NEL
    IF (IDR(N).NE.NBR(J)) GO TO 260
    IDR(N)=0
    GO TO 270
260 CONTINUE
C
270 CALL DINTY
    GO TO 10
C
C   ELEMENT J WILL BE ALLOWED TO TRY TO MOVE.
C   FIRST INFORMATION ABOUT ITS ENVIRONMENT MUST
C   BE OBTAINED IN ORDER TO GET A JUMP PROBABILITY
C
C   DRAWN IS INITIALIZED FALSE. ELEMENTS WHICH
C   NEED NOT BE EXAMINED WITH RESPECT TO J HAVE
C   THE APPROPRIATE LOCATION IN DRAWN SET TO .TRUE.
C
70 DO 20 M=1,NE
20 DRAWN(M)=.FALSE.
   DRAWN(J)=.TRUE.
C
C   CLEAR NUMBER CORRELATED ELEMENTS
C
    NC(1)=0
    NC(2)=0
C
C   LOOP 30 CHECKS EACH ELEMENT FOR CORRELATION WITH J
C
    DO 30 K=1,NE
    IF (DRAWN(K)) GO TO 30

```



```

C
C   CORL=.TRUE.  -- J AND K ARE CORRELATED
C   CORL=.FALSE. -- J AND K ARE INDEPENDENT
C   (CORL IS A LOGICAL FUNCTION)
C
C   IF (.NOT.CORL(J,K)) GO TO 30
C   KT=TYPE(K)
C   NC(KT)=NC(KT)+1
C   NCS=NC(KT)
C   CLK=CLASS(K)
C   IDC(NCS,KT)=K
C   KX=K
90  IF (CLK.EQ.1) GO TO 30
C
C   IF K IS PART OF A CLUSTER, THE REMAINING
C   ELEMENTS OF THE CLUSTER NEED NOT BE CHECKED
C   FOR CORRELATION
C
C   NB=NBR(KX)
C   DRAWN(NB)=.TRUE.
C   CLK=CLK-1
C   KX=NB
C   GO TO 90
30  CONTINUE
C
C   KDIR IS THE POSSIBLE JUMP DIRECTION
C   IF J IS OF CLASS 1, KDIR IS AN INTEGER FROM 1 TO NCD WITH
C   EQUAL PROBABILITY
C   IF NOT, SOME DIRECTIONS MAY BE UNAVAILABLE, AND SUBROUTINE SJD
C   SETS KDIR TO ONE OF THE AVAILABLE DIRECTIONS WITH EQUAL PROB.
C   IF NO DIRECTION IS AVAILABLE, KDIR=0
C
C
320 IF (CHECK.EQ.3) GO TO 150
C
C   IF (CLASS(J).NE.1) GO TO 200
C   CALL RNUM(RN)
C   KDIR=1.0+RCD*RN
C   GO TO 210
200 CALL SJD
290 IF (KDIR.EQ.0) GO TO 10
C
C   DETERMINE JUMP PROBABILITY
C
210 CALL JPROB(P).
C
C   DOES JUMP OCCUR
C
300 IF (P.EQ.0.0) GO TO 10
C   CALL RNUM(RN)
280 IF (RN-P) 140,140,10
150 KDIR=KDCH(NUMCH)
C
C   WRITE JUMP PARAMETERS ON NAUX EVERY 250TH JUMP
C

```

```

140 NJM=NJM+1
    IF (LIST) CALL LIST1
    CALL JUMP(J,KDIR)
    IF (CHECK.EQ.1) GO TO 10
    CALL CENSUS
    CALL TPR
10 CONTINUE
C
    IF (NTIME.LT.NTMAX) GO TO 100
C
C    PRINT FINAL OUTPUT
C
170 CALL CENSUS
    CALL TPR
    IF (NFPA.GT.0) CALL FPSTAT
220 IF (LIST) CALL LIST2
    WRITE (NOUT,1002)
C
C    WRITE TABLES ON NPUN FOR OFFLINE PUNCHING
C
    IF (NE.EQ.0) GO TO 230
    WRITE (NPUN,1003) NPROB,NE
    WRITE (NPUN,1003) (I,(R(I,J),J=1,3),TYPE(I),CLASS(I),NBR(I),NPROB,
1 I=1,NE)
    WRITE (NPUN,1003) NPROB,NE
    IF (.NOT.LAST) GO TO 230
    DO 250 I=1,3
250 END FILE NPUN
    REWIND NPUN
    CALL EXIT
C
C    CLEAR TABLES FOR NEXT CASE
C
230 CALL CLEAR
    GO TO 1
C
100 IF (MOD(NTIME,NPR).NE.0) GO TO 110
C
C    PRINT INTERMEDIATE OUTPUT
C
    CALL CENSUS
    CALL TPR
C
110 IF (NE.EQ.0) GO TO 130
    NTIME=NTIME+1
    IF (NTIME.GT.NPRI(2)) NPR=NPRI(3)
    GO TO 120
130 WRITE (NOUT,1000) NTIME
    GO TO 170
1000 FORMAT(47H1      EXECUTION TERMINATED DURING TIME STEP NO.15//38H
1      ALL DEFECTS HAVE BEEN ANNIHILATED)
1002 FORMAT (1H1)
1003 FORMAT(7I5,35X,6HFINISHI4)
1004 FORMAT(7I5,35X,5HSTARTI5)
    END

```

```

$IBFTC BLDATA LIST,NOREF,DECK,M90,XR3,DD
CBLDATA

```

```

BLOCK DATA
COMMON /BLOCK/MOVE(72)
DATA MOVE/4*1,4*-1,4*0,2*1,2*-1,2*1,2*-1,4*0,1,-1,1,-1,1,-1,1,-1,
1 4*0,2*1,2*-1,2*1,2*-1,4*0,1,-1,1,-1,4*0,2*1,2*-1,4*0,
2 1,-1,1,-1,1,-1,1,-1/
END

```

```

$IBFTC CENSUS  NOLIST,NOREF,DECK,M90,XR3,DD
CCENSUS

```

```

C
C      SUBROUTINE CENSUS
C
C      NCT(I,J) IS SET EQUAL TO THE NUMBER OF DEFECTS (NOT THE NUMBER OF
C      ELEMENTS) OF CLASS I AND TYPE J
C
C      MAXIMUM CLASS NUMBER IS 50
C
C      COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1  DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2  JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3  JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4  ,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
C
C      INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C
C      DO 10 I=1,50
C      DO 10 J=1,2
10  NCT(I,J)=0
C
C      DO 20 K=1,NE
C      I=CLASS(K)
C      J=TYPE(K)
20  NCT(I,J)=NCT(I,J)+1
C      DO 30 I=2,50
C      DO 30 J=1,2
30  NCT(I,J)=NCT(I,J)/I
C
C      FIND HIGHEST CLASS NUMBER FOR EACH TYPE
C
C      O 70 J=1,2
C
C      DO 60 II=1,50
C      I=51-II
C      IF (NCT(I,J).NE.0) GO TO 80
60  CONTINUE
C      MCL(J)=0
C      GO TO 70
80  MCL(J)=I
70  CONTINUE
C      RETURN
C      END

```

\$IBFTC CLEAR NOLIST,NOREF,DECK,M90,XR3,DD

CCLEAR

C

C CLEAR OUT TABLES FOR NEXT CASE

C

SUBROUTINE CLEAR

C

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC

C

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

C

LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC

C

DO 10 I=1,500
DO 20 J=1,3
20 R(I,J)=0
TYPE(I)=0
CLASS(I)=0
NORIG(I)=0
10 NBR(I)=0
RETURN
END

CCLOSE

SUBROUTINE CLOSE(J,K)

C

C

DAIQUIRI

C

C

C CLOSES UP THE TABLES AFTER THE FRENKEL PAIR (J,K) HAS COMBINED
C RECOMPUTES NEIGHBOR NUMBERS WHERE NECESSARY

C

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC

C

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

C

LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC

C

NEM2=NE-2

C

C

M = SMALLER NUMBER TO BE SCRATCHED FROM TABLE, N = LARGER

C

IF (J.GT.K) GO TO 110

M=J

N=K

GO TO 120

110 M=K

N=J

120 NM1=N-1

IF (N.EQ.(M+1)) GO TO 10

NM2=N-2


```

C
C   ENTRIES BELOW M UNCHANGED
C   ENTRIES BETWEEN M AND N MOVED DOWN ONE NOTCH
C
      DO 20 I=M,NM2
      DO 15 L=1,3
15  R(I,L)=R(I+1,L)
      TYPE(I)=TYPE(I+1)
      CLASS(I)=CLASS(I+1)
      NORIG(I)=NORIG(I+1)
20  NBR(I)=NBR(I+1)
C
10  IF (NM1.GT.NEM2) GO TO 60
C
C   ENTRIES ABOVE N MOVED DOWN TWO NOTCHES
C
      DO 30 I=NM1,NEM2
      DO 25 L=1,3
25  R(I,L)=R(I+2,L)
      TYPE(I)=TYPE(I+2)
      CLASS(I)=CLASS(I+2)
      NORIG(I)=NORIG(I+2)
30  NBR(I)=NBR(I+2)
C
60  NE=NE-2
      IF (NE.EQ.0) RETURN
C
C   RECOMPUTE NUMBERS OF ADJOINING ELEMENTS IN THE SAME WAY
C
      DO 40 I=1,NE
      IF (NBR(I).LT.M) GO TO 40
      IF (NBR(I).LT.N) GO TO 50
      NBR(I)=NBR(I)-2
      GO TO 40
50  NBR(I)=NBR(I)-1
40  CONTINUE
C
C   SAME FOR IDR TABLE
C
      DO 80 I=1,NEL
      IF (IDR(I).LT.M) GO TO 80
      IF ((IDR(I).EQ.M).OR.(IDR(I).EQ.N)) GO TO 90
      IF (IDR(I).LT.N) GO TO 100
      IDR(I)=IDR(I)-2
      GO TO 80
90  IDR(I)=0
      GO TO 80
100 IDR(I)=IDR(I)-1
80  CONTINUE
      RETURN
      END

```

\$IBFTC CONECT NOLIST,NOREF,DECK,M90,XR3,DD
 CCONECT
 C

LOGICAL FUNCTION CONECT(J,K)

THE CONECT FUNCTION DETERMINES CLUSTERING

FOR TYPE 1, CONECT=.TRUE. FOR FIRST AND SECOND NEIGHBORS (BCC)
 FOR TYPE 1, CONECT=.TRUE. FOR FIRST NEIGHBORS ONLY (FCC)
 FOR TYPE 2, CONECT=.TRUE. FOR FIRST NEIGHBORS ONLY

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
 1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
 2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
 3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
 4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
 5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC

IF (FCC) GO TO 50
 IF (TYPE(J).EQ.2) GO TO 30
 JKDEL=0
 DO 10 I=1,3
 JKDEL=JKDEL+IABS(R(J,I)-R(K,I))
 IF (JKDEL.GT.3) GO TO 20
 10 CONTINUE
 15 CONECT=.TRUE.
 RETURN
 20 CONECT=.FALSE.
 RETURN
 30 DO 40 I=1,3
 IF (IABS(R(J,I)-R(K,I)).NE.1) GO TO 20
 40 CONTINUE
 GO TO 15
 50 JKDEL=0
 DO 60 I=1,3
 JKDEL=JKDEL+(R(J,I)-R(K,I))**2
 IF (JKDEL.GT.2) GO TO 20
 60 CONTINUE
 GO TO 15
 END

\$IBFTC CORL NOLIST,NOREF,DECK,M90,XR3,DD
 CCORL

LOGICAL FUNCTION CORL(J,K)

DETERMINES WHETHER THE MOTION OF J AND K IS CORRELATED
 OR INDEPENDENT

IN THIS VERSION 1-1 PAIRS AND 1-2 PAIRS ARE UNCORRELATED (.FALSE.)

2-2 PAIRS ARE CORRELATED FOR NEIGHBOR SHELLS 1 THRU 5, FOR 7TH
NEIGHBORS, AND FOR (333) 10TH NEIGHBORS. THESE PAIRS CAN
BECOME CONNECTED IN TWO OR FEWER JUMPS

THIS VERSION RECOGNIZES NO FCC CORRELATIONS
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
INTEGER D2,DSQ
LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
DIMENSION LISTA(6)

LISTA CONTAINS DISTANCE SQUARED FOR NEIGHBORS 4,7,3,1,5,2
IN THAT ORDER

DATA LISTA/11,19,8,3,12,4/

IF (FCC.OR.(TYPE(J).EQ.1).OR.(TYPE(K).EQ.1)) GO TO 1
D2=DSQ(J,K)
DO 2 I=1,6
IF (D2.EQ.LISTA(I)) GO TO 3
2 CONTINUE
DO 4 I=1,3
IF (IABS(R(J,I)-R(K,I)).NE.3) GO TO 1
4 CONTINUE
3 CORL=.TRUE.
RETURN
1 CORL=.FALSE.
RETURN
END

\$IBFTC DINTY NOLIST,NOREF,DECK,M90,XR3,DD
CDINTY

SUBROUTINE DINTY

DINTY IS A SPECIAL ROUTINE TO HANDLE DI-INTERSTITIAL MIGRATION.
DI-INTERSTITIAL MIGRATION OCCURS 300 TIMES FASTER THAN SINGLE
INTERSTITIAL MIGRATION IN IRON AT 300 K

EACH DI DOES A 30 JUMP RANDOM WALK AT EACH TIME STEP

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500),
4 MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
COMMON /BLOCK/MOVE(12,3,2)

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC

```

C      INTEGER DSQ1,DSQMIN,DJSKDI,DISTF
C      LOGICAL CORL
C      INTEGER DSQ
C      DIMENSION JSJ(3),JSK(3)

C
C      DISTF (K1,K2,K3,K4,K5,K6 )=(K1-K4)**2+(K2-K5)**2+(K3-K6)**2
C      JDI=J
C      KDI=NBR(J)
C      NJ1=NORIG(J)

C      DO 1 IDUM=1,30
C
C      DSQMIN=999

C      LOOP 2 LOCATES THE CLOSEST CORRELATED INTERSTITIAL (NTGT) TO THE
C      MIGRATING DI (JDI,KDI)
C
C      DO 2 N=1,NE
C      IF ((TYPE(N).EQ.1).OR.(N.EQ.JDI).OR.(N.EQ.KDI).OR.((.NOT.CORL(JDI,
1 N)).AND.(.NOT.CORL(KDI,N)))) GO TO 2
C      DSQ1=DSQ(JDI,N)+DSQ(KDI,N)
C      IF (DSQ1.GE.DSQMIN) GO TO 2
C      DSQMIN=DSQ1
C      NTGT=N
2 CONTINUE

C      3 CALL RNUM(RN)
C      KDIR=1.0+RCD*RN

C      THERE ARE SIX PERMITTED JUMPS FOR THE DI.
C      THE EIGHT BCC 1ST NBR JUMPS, EXCEPT FOR THE TWO JUMPS ALONG THE
C      AXIS OF THE DI. SEE JOHNSON, PHYS REV 134, A1334, (1964)
C
C      DO 4 I=1,3
C      JSK(I)=R(KDI,I)+MOVE(KDIR,I,KCR)
4 JSJ(I)=R(JDI,I)+MOVE(KDIR,I,KCR)
C      DJSKDI=DISTF(JSJ(1),JSJ(2),JSJ(3),R(KDI,1),R(KDI,2),R(KDI,3))
C      IF ((DJSKDI.EQ.12).OR.(DJSKDI.EQ.0)) GO TO 3

C      IF THE DI IS UNCORRELATED,MAKE THE INDICATED JUMP
C
C      IF (DSQMIN.GE.999) GO TO 5

C      IF CORRELATED,THEN THE JUMP IS PROHIBITED IF IT TAKES THE DI
C      FARTHER AWAY FROM THE NEAREST CORRELATED INTERSTITIAL
C
C      DSQ1=DISTF(JSJ(1),JSJ(2),JSJ(3),R(NTGT,1),R(NTGT,2),R(NTGT,3))+
1 DISTF(JSK(1),JSK(2),JSK(3),R(NTGT,1),R(NTGT,2),R(NTGT,3))
C      IF (DSQ1.GT.DSQMIN) GO TO 3

C      5 NJM=NJM+1
C      IF (.NOT.LIST) GO TO 7
C      CALL LIST1
C      NJM=NJM+1

```



```

JTEMP=J
J=KDI
CALL LIST1
J=JTEMP
NOKDI=NORIG(KDI)
7 CALL JUMP(JDI,KDIR)
DO 8 I=1,NE
IF (NORIG(I).EQ.NOKDI) GO TO 9
8 CONTINUE
9 KDI=I
CALL JUMP(KDI,KDIR)
C
C IF THE DI SURVIVES THE JUMP AS A DI, CONTINUE LOOP 1
C
IF (CHECK.EQ.1) GO TO 6
CALL CENSUS
CALL TPR
C
6 NJ2=NORIG(J)
IF ((NJ1.NE.NJ2).OR.(CLASS(J).NE.2).OR.(J.GT.NE)) RETURN
1 CONTINUE
C
RETURN
END

```

```

$IBFTC DSQ      NOLIST,NOREF,DECK,M90,XR3,DD
CDSQ

```

```

C      INTEGER FUNCTION DSQ(J,K)

```

```

C      DISTANCE SQUARED BETWEEN LATTICE SITES

```

```

C      COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN

```

```

C      INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

```

```

C      LOGICAL CMBINE,DRAWN,LIST

```

```

C      DSQ=(R(J,1)-R(K,1))**2+(R(J,2)-R(K,2))**2
1 +(R(J,3)-R(K,3))**2
RETURN
END

```

```

$IBFTC FPSTAT  NOLIST,NOREF,DECK,M90,XR3,DD
CFPSTAT

```

```

C      SUBROUTINE FPSTAT

```

```

C
C PRINT FRENKEL PAIR ANNIHILATION STATISTICS
C
C FPA(I,J) CONTAINS THE TIME AND ELEMENT NUMBERS FOR THE ITH
C ANNIHILATION
C
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
C
C INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C NZA=0
C DO 1 I=1,NFPA
C IF (FPA(I,1).NE.0) GO TO 2
1 NZA=NZA+1
C
C PRINT INITIALLY UNSTABLE PAIRS (T=0) SEPARATELY
C
2 IF (NZA.GT.0) WRITE (NOUT,1000) NZA,(FPA(I,2),FPA(I,3),I=1,NZA)
C
3 IF (NZA.EQ.NFPA) RETURN
C NZA1=NZA+1
C WRITE (NOUT,1001) ((FPA(I,J),J=1,3),I=NZA1,NFPA)
C NAT=0
C NAC=0
C WRITE (NOUT,1002)
C
C PRINT NUMBER ANNIHILATED AT EACH TIME STEP, AND CUMULATIVE NUMBER
C
C
C DO 4 I=NZA1,NFPA
C NAC=NAC+1
C NAT=NAT+1
C IF ((FPA(I,1).EQ.FPA(I+1,1)).AND.(I.LT.NFPA))GO TO 4
C WRITE (NOUT,1003) FPA(I,1),NAT,NAC
C NAT=0
4 CONTINUE
C
C RETURN
C
1000 FORMAT (1H15X,I3,25H INITIALLY UNSTABLE PAIRS// (I12,I5))
1001 FORMAT (34H1 FRENKEL PAIRS ANNIHILATED//25H TIME
1 PAIR// (I15,I8,I5))
1002 FORMAT (23H1 T FPA(T) CUM)
1003 FORMAT (I8,I7,I8)
END

$IBFTC FRENKL NOLIST,NOREF,DECK,M90,XR3,DD
CFRENKL
C
C DAIQUIRI -- BCC
C
C LOGICAL FUNCTION FRENKL(J,K)

```

J AND K ARE A FRENKEL PAIR.
 FRENKL = .TRUE. SIGNIFIES THE PAIR IS UNSTABLE
 AT ABSOLUTE ZERO

SEE GEMP-356, PAGES 8-9

```
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC,NPRI(3)
```

```
INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
```

```
LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
```

```
DIMENSION NBCC(4),NFCC(4)
DATA (NBCC(I),I=1,4),(NFCC(I),I=1,4)/27,12,4,3,18,8,4,2/
INTEGER D2,DSQ
D2=DSQ(J,K)
IF (D2.GT.27) GO TO 2
IF (FCC) GO TO 3
DO 4 I=1,4
IF (D2.EQ.NBCC(I)) GO TO 5
4 CONTINUE
GO TO 2
5 IF ((I.EQ.1).AND.(IABS(R(J,1)-R(K,1)).NE.3)) GO TO 2
GO TO 1

3 IF (D2.GT.18) GO TO 2
DO 6 I=1,4
IF (D2.EQ.NFCC(I)) GO TO 7
6 CONTINUE
GO TO 2
7 IF ((I.EQ.1).AND.(R(J,1).NE.R(K,1)).AND.(R(J,2).NE.R(K,2)).AND.
1 (R(J,3).NE.R(K,3))) GO TO 2

1 FRENKL=.TRUE.
RETURN
2 FRENKL=.FALSE.
RETURN
END
```

\$IBFTC INCH NOLIST,NOREF,DECK,M90,XR3,DD
 CINCH

SUBROUTINE INCH

SEARCHES INPUT DATA FOR ERRORS

```
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
```

LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
LOGICAL CE1

```
C
C      1 -- NUMBER OF ELEMENTS NOT IN LEGAL RANGE
C      2 -- MAXIMUM TIME STEP NON-POSITIVE
C      4 -- ILLEGAL CHECKOUT MODE
C      5 -- CHECKOUT MODE 3 WITH NO FORCED JUMPS
C 1000+I -- TYPE 1 ELEMENT NOT ON LATTICE SITE
C 2000+I -- CLASS NON-POSITIVE
C 3000+I -- CLASS GREATER THAN ONE, BUT NBR=0
C 4000+I -- CLASS=1, AND NBR NON-ZERO
C 5000+I -- CLASS GREATER THAN 50
C 6000+I -- NEGATIVE COORDINATE
C 7000+I -- OPPOSITE TYPES IN CLUSTER
C 8000+I -- INCONSISTENT NBR CYCLE
C
1 IF ((NE.GT.0).AND.(NE.LE.500)) GO TO 2
  KODE=1
  GO TO 100
C
2 IF ((NTMAX.GT.0).AND.(NTMIN.GT.0).AND.(NTMAX.GE.NTMIN)) GO TO 4
  KODE=2
  GO TO 100
C
4 IF (NPROB.LT.0) NPROB=0
  IF ((CHECK.GE.1).AND.(CHECK.LE.3)) GO TO 5
  KODE=4
  GO TO 100
C
5 IF ((CHECK.NE.3).OR.(NJCH.NE.0)) GO TO 6
  KODE=5
  GO TO 100
C
6 DO 50 I=1,NE
  CE1=CLASS(I).EQ.1
  IF (FCC) GO TO 14
  IF ((TYPE(I).NE.1).OR.((MOD(R(I,1),2).EQ.MOD(R(I,2),2)).AND.
1 (MOD(R(I,1),2).EQ.MOD(R(I,3),2)))) GO TO 7
  KODE=1000+I
  GO TO 100
14 LSUM=0
  DO 15 K=1,3
15 LSUM=LSUM+R(I,K)
  IF (MOD(LSUM,2).NE.0) GO TO 7
  KODE=1000+I
  GO TO 100
C
7 IF (CLASS(I).GT.0) GO TO 8
  KODE=2000+I
  GO TO 100
C
8 IF (CE1.OR.(NBR(I).GT.0)) GO TO 9
  KODE=3000+I
  GO TO 100
```



```

C
9 IF ((.NOT.CE1).OR.(NBR(I).EQ.0)) GO TO 10
  KODE=4000+I
  GO TO 100
C
10 IF (CLASS(I).LE.50) GO TO 11
  KODE=5000+I
  GO TO 100
C
11 IF ((R(I,1).GT.0).AND.(R(I,2).GT.0).AND.(R(I,3).GT.0)) GO TO 12
  KODE=6000+I
  GO TO 100
C
12 IF (CE1) GO TO 50
  NS=NBR(I)
  IF (TYPE(NS).EQ.TYPE(I)) GO TO 13
  KODE=7000+I
  GO TO 100
C
13 CLM1=CLASS(I)-1
  DO 25 K=1,CLM1
25  NS=NBR(NS)
  IF (NS.EQ.I) GO TO 50
  KODE=8000+I
  GO TO 100
50 CONTINUE
  RETURN
C
100 CALL PANIC(KODE, .FALSE., .FALSE.)
  RETURN

  END

```

\$IBFTC INFO NOLIST,NOREF,DECK,M90,XR3,DD
CINFO

```

C
  SUBROUTINE INFO
C
C  PRINTS TITLE PAGE,INCLUDING INFORMATION ABOUT
C  THIS VERSION OF THE PROGRAM
C
  COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1  DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2  JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3  JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4  ,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5  ,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
C
  INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
  LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
C
  NOUT=6
  WRITE (NOUT,1000)
  WRITE (NOUT,1001)
  RETURN

```

```

1000 FORMAT(1H121H      PROGRAM DAIQUIRI//
1 72H      SIMULATION OF DEFECT ANNEALING IN QUENCHED OR IRRADIATED
2 SPECIMENS//
3 26H      NMPO PROGRAM NO. 859//
4 54H      VERSION FOR PURE ALPHA-IRON (300 DEGREES KELVIN)//
4 54H      FCC GEOMETRY IS ALLOWED, BUT FCC ANNEALING MODEL/
4 49H      IS A TRIVIAL ONE FOR CHECKOUT PURPOSES ONLY//
5 35H      TYPE 1 ELEMENTS ARE VACANCIES/
6 39H      TYPE 2 ELEMENTS ARE INTERSTITIALS//
7 46H      ASSUMPTIONS ON JUMP PROBABILITIES FOLLOW//
8 70H      VACANCIES                                P= 0.0 (EA=
90.68 EV)//
1 70H      SINGLE INTERSTITIAL (UNCORRELATED JUMP) P= 0.1 (EA=
20.33 EV)//
3 70H      SINGLE INTERSTITIAL (CORRELATED JUMP)   P= 1.0 (EA=
40.27 EV)//

5 70H      DI-INTERSTITIAL                        30 JUMPS/TIME STEP (EA=
60.18 EV)//
7 56H      LARGER INTERSTITIAL CLUSTERS            P= 0.0//)
1001 FORMAT (87H      SINGLE INTERSTITIALS ARE CORRELATED FOR NEIGHBORS
1 1,2,3,4,5,7, AND 10 (333 ONLY)//
2 92H      THE LAST TWO JUMPS IN DI-INTERSTITIAL FORMATION ARE CORR
3ELATED.  THESE ARE OF THE FORM/
4 57H      (2,1) (3,1) (5,1) (4,2) (4,3) (7,3) (7,5) (10,5)//
5 25H      INPUT/OUTPUT FILES/
6 38H      SYSTEM INPUT                NIN = 5/
7 38H      SYSTEM OUTPUT              NOUT= 6/
8 38H      FIRST AUXILIARY            NAUX= 1/
8 38H      SECOND AUXILIARY           NAUX2= 2/
9 38H      BCD PUNCHED OUTPUT         NPUN= 11//
1 23H      BCC JUMP VECTORS/
2 23H      1 (+1,+1,+1)//
3 23H      2 (+1,+1,-1)//
4 23H      3 (+1,-1,+1)//
5 23H      4 (+1,-1,-1)//
6 23H      5 (-1,+1,+1)//
7 23H      6 (-1,+1,-1)//
8 23H      7 (-1,-1,+1)//
9 23H      8 (-1,-1,-1)//
END

```

```

$IBFTC INPR      NOLIST,NOREF,DECK,M90,XR3,DD
CINPR
C

```

```

SUBROUTINE INPR
C

```

```

PRINT INPUT DATA
C

```

```

COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC,NPRI(3)
C

```

```

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C

```

```

C      LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
C
WRITE (NOUT,1000) BCC,NPROB,NTMIN,NTMAX,NE,(NPRI(I),I=1,3),LIST,
1 CHECK,G
CALL CENSUS
NTIME=-1
CALL TPR
RETURN
1000 FORMAT(33H1      LISTING OF INPUT PARAMETERS///
1 41H      CRYSTAL STRUCTURE      BCC      =L4/
1 41H      PROBLEM NUMBER      NPROB =I4/
2 41H      FIRST TIME STEP      NTMIN =I4/
2 41H      LAST TIME STEP      NTMAX =I4/
3 41H      NUMBER OF ELEMENTS      NE      =I4/
4 41H      TABLE LISTING PARAMETERS NPR      =3I4/
5 41H      JUMP HISTORY REQUEST      LIST =L4/
6 41H      CHECKOUT MODE      CHECK =I4/
7 41H      RANDOM NUMBER GENERATOR G      =0I2)
END
$IBFTC INPUT      NOLIST,NOREF,DECK,M90,XR3,DD
CINPUT
C
C      INPUT ROUTINE FOR PROGRAM DAIQURI
C
C      SUBROUTINE INPUT
C
C
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC,NPRI(3)
C
C      INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C      LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
C
C      NAMELIST/ANNEAL/CHECK,JCH,KDCH,LIST,NJCH,NPR,NTMIN,NTMAX,TAU,NPROB
1,BCC,FETCH,LAST,SAVE,NPRI
C
NIN=5
NOUT=6
NAUX=1
NAUX2=2
NPUN=11
READ (NIN,1000)(ID1(I),I=1,12)
READ (NIN,1000)(ID2(I),I=1,12)
READ (NIN,ANNEAL)
IF (FETCH) GO TO 1
READ (NIN,1003) NE
READ (NIN,1001) (R(I,1),R(I,2),R(I,3),TYPE(I),CLASS(I),NBR(I),I=1,
1 NE)
READ (NIN,1003) DUMMY
GO TO 2

```

```

1 REWIND NAUX2
  READ (NAUX2) NE,((R(I,J),J=1,3),TYPE(I),CLASS(I),NBR(I),I=1,NE)
  REWIND NAUX2
2 READ (NIN,1002) G
  IF (.NOT.SAVE) RETURN
  REWIND NAUX2
  WRITE (NAUX2) NE,((R(I,J),J=1,3),TYPE(I),CLASS(I),NBR(I),I=1,NE)
  REWIND NAUX2
  RETURN

```

```

C
1000 FORMAT (12A6)
1001 FORMAT (5X,6I5)

```

```

1002 FORMAT (012)
1003 FORMAT (5X,I5)

```

```

C
  END

```

```

$IBFTC JPROB  NOLIST,NOREF,DECK,M90,XR3,DD
CJPROB

```

```

C
  SUBROUTINE JPROB(P)

```

```

C
  P=0 FOR VACANCIES,VACANCY CLUSTERS, AND INTERSTITIAL CLUSTERS
C
  P= 0.10 FOR UNCORRELATED SINGLE INTERSTITIALS
C
  P= 1.00 FOR A SINGLE INTERSTITIAL IN ITS LAST TWO JUMPS
C
  APPROACHING ANOTHER INTERSTITIAL
C

```

```

  COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1  DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2  JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3  JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4  ,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5  ,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
  COMMON /BLOCK/MOVE(12,3,2)
  INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
  INTEGER D2B,D2A,DSQ
  LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
  DIMENSION LISTB(8),JS(3)

```

```

C
  LISTB CONTAINS CODE NUMBERS OF CORRELATED JUMPS
C

```

```

  DATA LISTB/403,803,1203,1104,1108,1908,1912,2712/

```

```

C
  IF (FCC) GO TO 10
  IF ((TYPE(J).EQ.1).OR.(CLASS(J).GT.1)) GO TO 1
  IF (NC(2).GT.0) GO TO 2

```

```

C
3 P=0.10
  RETURN

```

```

C
1 P=0.00
  RETURN

```



```

C
C      JS IS POSITION AFTER PROSPECTIVE MOVE
C
2 DO 4 I=1,3
4 JS(I)=R(J,I)+MOVE(KDIR,I,KCR)
C
      NC2=NC(2)
C
C      LOOP 5 TESTS WHETHER THE PROSPECTIVE JUMP IS CORRELATED WITH
C      RESPECT TO ANY OF THE CORRELATED ELEMENTS
C
      DO 5 I1=1,NC2
      ITEST=IDC(I1,2)
      KL=CLASS(ITEST)-1
9 D2B=DSQ(J,ITEST)
      D2A=(JS(1)-R(ITEST,1))**2+(JS(2)-R(ITEST,2))**2+(JS(3)-R(ITEST,3))
      1**2
C
C      JCODE IS A CODE NUMBER DESCRIBING THE JUMP OF J RELATIVE TO ITEST
C      IN TERMS OF DISTANCE SQUARED BEFORE AND AFTER THE JUMP
C
      JCODE=100*D2B+D2A
      DO 8 I2=1,8
      IF (JCODE.EQ.LISTB(I2)) GO TO 6
8 CONTINUE
      IF (KL.EQ.0) GO TO 5
      KL=KL-1
      ITEST=NBR(ITEST)
      GO TO 9
5 CONTINUE
      GO TO 3
C
6 P=1.00
      RETURN
10 IF((TYPE(J).EQ.1).OR.(CLASS(J).GT.1)) GO TO 11
      P=0.10
      RETURN
11 P=0.00
      RETURN
      END

```

```

$IBFTC JUMP      NOLIST,NOREF,DECK,M90,XR3,DD
CJUMP
C
C

```

```

      SUBROUTINE JUMP(J,KDIR)

```

```

      JUMPS ELEMENT J IN DIRECTION KDIR

```

```

      COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
      COMMON /BLOCK/MOVE(12,3,2)

```

C INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE

C LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
C LOGICAL FRENKL

C
C DO 10 I=1,3
10 R(J,I)=R(J,I)+MOVE(KDIR,I,KCR)

C
C DOES THIS JUMP RESULT IN ANNIHILATION

C
C DO 20 K=1,NE
C IF (TYPE(J).EQ.TYPE(K)) GO TO 20
C IF (FRENKL(J,K)) GO TO 30
20 CONTINUE
GO TO 40

C
C YES

C
C 30 TYPE(J)=0
C TYPE(K)=0
C NFPA=NFPA+1
C FPA(NFPA,1)=NTIME
C FPA(NFPA,2)=NORIG(J)
C FPA(NFPA,3)=NORIG(K)
C CALL UPDATE(J)
C CALL UPDATE(K)
C CALL CLOSE(J,K)
C RETURN

C
C NO

C
C 40 CALL UPDATE(J)
C RETURN
C END

\$IBFTC LIST1 NOLIST,NOREF,DECK,M90,XR3,DD
CLIST1

C
C SUBROUTINE LIST1

C
C TRANSFERS JUMP PARAMETERS TO BUFFER AFTER EVERY JUMP
C DUMPS BUFFER ON NAUX FILE WHEN FULL (250 JUMPS)

C
C COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
C INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C LOGICAL CMBINE,DRAWN,LIST

```

C
M250=MOD(NJM,250)
IF (M250.EQ.0) M250=250
BUFF(M250,1)=NTIME
BUFF(M250,2)=NORIG(J)
BUFF(M250,3)=KDIR
IF (M250.NE.250) RETURN
WRITE (NAUX) ((BUFF(I1,I2),I2=1,3),I1=1,250)
RETURN
END

```

```

$IBFTC LIST2  NOLIST,NOREF,DECK,M90,XR3,DD
CLIST2

```

```

C
SUBROUTINE LIST2
C
C DUMPS PARTLY FILLED BUFFER AS LAST RECORD ON NAUX
C TRANSFERS NAUX CONTENTS TO NOUT
C
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4 MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
C
INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
LOGICAL IEQN,EVEN
C
EVEN=M250.EQ.250
IF (.NOT.EVEN) WRITE (NAUX) ((BUFF(I1,I2),I2=1,3),I1=1,M250)
1 END FILE NAUX
REWIND NAUX
C
NREC = NUMBER OF LOGICAL RECORDS ON NAUX
C ALL RECORDS EXCEPT PROBABLY THE LAST ARE 750 WORDS
C
IF (EVEN) NREC=NJM/250
IF (.NOT.EVEN) NREC=NJM/250+1
3 DO 4 I=1,NREC
IEQN=I.EQ.NREC
IF (IEQN) IMAX=M250
IF (.NOT.IEQN) IMAX=250
6 READ (NAUX) ((BUFF(I1,I2),I2=1,3),I1=1,IMAX)
4 WRITE (NOUT,1000) ((BUFF(I1,I2),I2=1,3),I1=1,IMAX)
C
REWIND NAUX
RETURN
1000 FORMAT(16H1 JUMP HISTORY//
1 12H T=TIME/
2 23H J=JUMPING ELEMENT/
3 19H V=JUMP VECTOR//
470H T J V T J V T J V T J V T
5 J V// (5(I7,I4,I3)))
END

```

\$IBFTC OCTIN NOLIST,NOREF,DECK,M90,XR3,DD
COCTIN

SUBROUTINE OCTIN

CHECKS WHETHER INPUT INTERSTITIALS ARE AT LATTICE SITES

OCTAHEDRALS (BCC) ARE MOVED TO ONE OF THE TWO ADJOINING LATTICE SITES. (FCC,SIX)

PAR(J)=PARITY OF JTH COORDINATE, ZERO FOR EVEN, ONE FOR ODD

```
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,  
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),  
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,  
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)  
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN  
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
```

DIMENSION MOVSC(6,3)

```

INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
LOGICAL FRENKL,P12,P13,PLUS
INTEGER PAR(3)

```

```
DATA ((MOVSC(1,J),J=1,3),I=1,6)/
1 +1, 0, 0, -1, 0, 0, 0,+1, 0, 0,-1, 0, 0, 0,+1, 0, 0,-1/
```

```
DO 90 I=1,NE
90 IDR(I)=I
```

```
NEL=NE
DO 10 M=1,NEL
IF (IDR(M).EQ.0) GO TO 10
I=IDR(M)
IF (TYPE(I).EQ.1) GO TO 10
```

```

IF (FCC) GO TO 30
DO 20 J=1,3
PAR(J)=MOD(R(I,J),2)
P12=PAR(1).EQ.PAR(2)
P13=PAR(1).EQ.PAR(3)
IF (P12.AND.P13) GO TO 10
IF (P12.AND.(.NOT.P13)) JODD=3
IF ((.NOT.P12).AND.P13) JODD=2
IF ((.NOT.P12).AND.(.NOT.P13)) JODD=1

```

```

60 CALL RNUM(RN)
   IF (RN.GT.0.5) GO TO 50
   R(I,JODD)=R(I,JODD)-1
   GO TO 100
50 R(I,JODD)=R(I,JODD)+1
   GO TO 100

```



```

30 IF (MOD(R(I,1)+R(I,2)+R(I,3),2).NE.0) GO TO 10
   CALL RNUM(RN)
   KSCL=1.0+RN*6.0
   DO 40 K=1,3
40 R(I,K)=R(I,K)+MOVSCS(KSCL,K)
C
100 DO 70 K=1,NE
   IF (TYPE(K).EQ.2) GO TO 70
   IF (FRENKL(I,K)) GO TO 80
70 CONTINUE
   CALL UPDATE(I)
   GO TO 10
C
C   ANNIHILATION RESULTS FROM MOVE TO LATTICE SITE
C
80 TYPE(I)=0
   TYPE(K)=0
   NFPA=NFPA+1
   FPA(NFPA,1)=NTIME
   FPA(NFPA,2)=NORIG(I)
   FPA(NFPA,3)=NORIG(K)
   CALL UPDATE(I)
   CALL UPDATE(K)
   CALL CLOSE(I,K)
C
10 CONTINUE
   RETURN
   END

```

```

$IBFTC PANIC   NOLIST,NOREF,DECK,M90,XR3,DD
CPANIC

```

```

C   SUBROUTINE PANIC(CODE,GOON,DMP)
C
C   GENERAL EXECUTION DIAGNOSTIC PRINTOUT
C
   INTEGER CODE
   LOGICAL GOON,DMP
   WRITE (6,1000) CODE
   IF (.NOT.DMP) GO TO 10
   CALL PDUMP
10 IF (GOON) RETURN
   CALL EXIT
   STOP
1000 FORMAT(29H1          *****/29H          *
1      */21H          * ERROR CODE16,2H */29H          *
2      */29H          *****/29H          *)
   END

```

```

$IBFTC PRMUTE   NOLIST,NOREF,DECK,M90,XR3,DD
CPRMUTE
C

```

```

   SUBROUTINE PRMUTE(N,LIST)

```

```

C
C THIS SUBROUTINE PRODUCES A RANDOMLY SELECTED PERMUTATION OF THE
C INTEGERS FROM 1 TO N AND STORES THE PERMUTED SET IN LIST
C
C RNUM IS INITIALIZED EXTERNALLY
C
C DIMENSION LIST(500),INTEGR(500)
C
C DO 10 I=1,N
10 INTEGR(I)=I
C
C NN=N
C DO 20 I=1,N
C FNN=NN
C NNMI=NN-1
C CALL RNUM(R)
C K=FNN*R+1.0
C LIST(I)=INTEGR(K)
C IF (K.EQ.NN) GO TO 20
C DO 40 L=K,NNMI
40 INTEGR(L)=INTEGR(L+1)
20 NN=NN-1
C
C RETURN
C END

```

\$IBFTC SJD NOLIST,NOREF,DECK,M90,XR3,DD

CSJD

```

C
C SUBROUTINE SJD
C
C COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
5,NTMIN,LAST,NAUX2,SAVE,FETCH,BCC,NCD,RCD,KCR,FCC
C COMMON /BLOCK/MOVE(12,3,2)
C
C INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C LOGICAL BCC,DRAWN,FETCH,LAST,LIST,SAVE,FCC
C
C SET JUMP DIRECTION AT RANDOM FROM AVAILABLE DIRECTIONS
C DIMENSION JS(12,3)
C INTEGER AVDIR(12)
C NAVDIR=0
C DO 1 I=1,NCD
C DO 1 K=1,3
1 JS(I,K)=R(J,K)+MOVE(I,K,KCR)
C
C CLM1=CLASS(J)-1

```

```

C      DO 2 I=1,NGD
      NB=NBR(J)
      DO 3 L=1,CLM1
      DO 4 K=1,3
      IF (R(NB,K).NE.JS(I,K)) GO TO 5
4 CONTINUE

C      IF LOOP 4 IS COMPLETED, DIRECTION I IS NOT AVAILABLE
C
C      GO TO 2

C      5 NB=NBR(NB)
      3 CONTINUE

C      IF LOOP 3 IS COMPLETED, DIRECTION I IS AVAILABLE
C
C      NAVDIR=NAVDIR+1
      AVDIR(NAVDIR)=I
      2 CONTINUE

C      WHEN LOOP 2 IS COMPLETE, THE NUMBER OF AVAILABLE DIRECTIONS IS
C      NAVDIR,AND THEY ARE TABULATED IN AVDIR
C
      IF (NAVDIR.EQ.0) GO TO 6
      FNA=NAVDIR
      FNAI=1.0/FNA
      CALL RNUM(RN)
      KDSUB=FNA*(RN+FNAI)
      KDIR=AVDIR(KDSUB)
      RETURN

C      6 KDIR=0
      RETURN
      END

```

```

$IBFTC TPR      NOLIST,NOREF,DECK,M90,XR3,DD
CTPR

```

```

C      SUBROUTINE TPR
C
C      PRINT COMPLETE TABLES
C
      COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1 DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2 JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3 JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4,MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN

C      INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C      LOGICAL CMBINE,DRAWN,LIST
C
      DIMENSION NTOT(2),ALPH1(9),ALPH2(9)
      DATA ALPH1/51HELEMENT TABLES (INPUT DATA) /
      DATA ALPH2/51HELEMENT TABLES (INITIAL CONDITIONS FOR SIMULATION) /

C      PRINT 50 ELEMENTS PER PAGE
C

```

```

NPAGES=1+(NE-1)/50
MNE50=MOD(NE,50)
DO 1 I=1,NPAGES
WRITE (NOUT,1000) (ID1(J),J=1,12),(ID2(J),J=1,12),NPROB
IF (NTIME) 2,3,4

```

```

C
C
C   NTIME = -1 WHEN LISTING INPUT DATA

```

```

2 WRITE (NOUT,1001) (ALPH1(I),I=1,9)
GO TO 5
3 WRITE (NOUT,1001) (ALPH2(I),I=1,9)
GO TO 5
4 WRITE (NOUT,1002) NTIME

```

```

C
5 WRITE (NOUT,1008)
I50=50*I
JFIRST=I50-49
IF (I.EQ.NPAGES) GO TO 6
JLAST=I50
GO TO 1
6 IF (MNE50.EQ.0) MNE50=50
JLAST=JFIRST+MNE50-1

```

```

C
1 WRITE (NOUT,1003) (J,NORIG(J),(R(J,K),K=1,3),TYPE(J),CLASS(J),
1 NBR(J),J=JFIRST,JLAST)

```

```

C
C   PRINT CENSUS RESULTS
C

```

```

IF (MNE50.GT.35) WRITE (NOUT,1004)
WRITE (NOUT,1005)
DO 7 J=1,2
MCS=MCL(J)
NTOT(J)=0
IF (MCS.EQ.0) GO TO 7
DO 8 I=1,MCS
IF (NCT(I,J).EQ.0) GO TO 8
WRITE (NOUT,1006) J,I,NCT(I,J)
NTOT(J)=NTOT(J)+1*NCT(I,J)
8 CONTINUE
7 CONTINUE
WRITE (NOUT,1007) NTOT(1),NTOT(2)
RETURN

```

```

C
1000 FORMAT (1H112A6//1X,12A6//16H      PROBLEM NO.14/)
1001 FORMAT (/5X,12A6//)
1002 FORMAT (14H0      TIME STEPI6/)
1003 FORMAT (2I4,18,2I5,2I6,I7)
1004 FORMAT (1H1)
1005 FORMAT (/22H      DEFECT POPULATION//22H      TYPE CLASS NUMBER)
1006 FORMAT (I8,I5,I7)
1007 FORMAT (/17,7H TYPE 1/17,7H TYPE 2)
1008 FORMAT (46H0 I J      POSITION      TYPE CLASS ADJ)
END

```



```

$IBFTC UPDATE  NOLIST,NOREF,DECK,M90,XR3,DD
CUPDATE
C
C   DAIQUIRI
C
C   SUBROUTINE TO UPDATE THE TABLES AFTER A JUMP BY ELEMENT J
C
C   SUBROUTINE UPDATE(J)
C
C   CLASS AND NBR MUST BE UPDATED FOR J AND FOR ALL ITS NEIGHBORS
C   CH = ELEMENTS TO CHECK
C   NCH = NUMBER OF ELEMENTS TO CHECK
C
COMMON CH(50),CHECK,CHK,CHT(50),CL(50,2),CLASS(500),CLK,CLM1,G,
1  DRAWN(500),GR(50),ID1(12),ID2(12),IDC(50,2),IDR(500),J,JSITE(3),
2  JT,KDIR,NBR(500),NC(2),NCT(50,2),NE,NIN,NOUT,NPR,NTIME,NTMAX,
3  JCH(100),KDCH(100),NJCH,NORIG(500),NPROB,R(500,3),TAU,TYPE(500)
4  MCL(2),NEL,FPA(250,3),NFPA,BUFF(250,3),LIST,M250,NAUX,NJM,NPUN
C
C   INTEGER BUFF,CH,CHECK,CHK,CHT,CL,CLASS,CLK,CLM1,FPA,G,GR,R,TYPE
C
C   LOGICAL CMBINE,DRAWN,LIST
C   LOGICAL CONECT
C
C   NCH=0
C   F (TYPE(J).EQ.0) GO TO 20
C   NCH=NCH+1
C   CH(NCH)=J
10  IF (CLASS(J).EQ.1) GO TO 10
C   NB=NBR(J)
C   CLM1=CLASS(J)-1
C   DO 30 IDUM=1,CLM1
C   NCH=NCH+1
C   CH(NCH)=NB
30  NB=NBR(NB)
C
C   CHECK FOR POSSIBLE NEW NEIGHBORING ELEMENTS FOR J
C
C   10  IF (TYPE(J).EQ.0) GO TO 150
C   NT=0
C   DO 50 K=1,NE
C   IF ((TYPE(J).NE.TYPE(K)).OR.(J.EQ.K).OR.(.NOT.CONECT(J,K))) GOTO50
C   NT=NT+1
C
C   CHT IS THE LIST OF ELEMENTS TENTATIVELY TO BE ADDED TO THE CH LIST
C
C   CHT(NT)=K
C   IF (CLASS(K).EQ.1) GO TO 50
C   CLM1=CLASS(K)-1
C   NB=NBR(K)
C   DO 60 IDUM=1,CLM1
C   NT=NT+1
C   CHT(NT)=NB
60  NB=NBR(NB)
50  CONTINUE

```

```

C
C      ADD TO CH ALL THE ELEMENTS OF CHT THAT ARE NOT DUPLICATES
C
      IF (NT.EQ.0) GO TO 150
      IF (NCH.EQ.0) GO TO 190
      DO 70 I=1,NT
      DO 80 K=1,NCH
      IF (CHT(I).EQ.CH(K)) GO TO 70
      80 CONTINUE
      160 NCH=NCH+1
      CH(NCH)=CHT(I)
      70 CONTINUE
      GO TO 210
C
      190 DO 200 I=1,NT
      200 CH(I)=CHT(I)
      NCH=NT
      GO TO 210
C
C      CH LIST IS NOW COMPLETE AND READY FOR SORTING
C
      150 IF (NCH.EQ.0) RETURN
      210 DO 90 I=1,NCH
      NGR=0
      IF (CH(I).EQ.0) GO TO 90
      NGR=NGR+1
      GR(NGR)=CH(I)
      CH(I)=0
      IF (I.GE.NCH) GO TO 170
      IP1=I+1
      220 NGRB=NGR
      DO 100 K=IP1,NCH
      IF (CH(K).EQ.0) GO TO 100
      CHK=CH(K)
      DO 110 L=1,NGR
      IF (.NOT.CONECT(CHK,GR(L))) GO TO 110
      GO TO 120
      110 CONTINUE
      GO TO 100
      120 NGR=NGR+1
      GR(NGR)=CHK
      CH(K)=0
      100 CONTINUE
      IF (NGR.GT.NGRB) GO TO 220
      170 IF (NGR.GT.1) GO TO 180
      NG=GR(1)
      NBR(NG)=0
      CLASS(NG)=1
      GO TO 90
      180 DO 130 L=1,NGR
      NG=GR(L)
      CLASS(NG)=NGR
      IF (L.EQ.NGR) GO TO 140
      NBR(NG)=GR(L+1)
      GO TO 130
      140 NBR(NG)=GR(1)
      130 CONTINUE
      90 CONTINUE
C
      40 RETURN
      END

```